Отчет по лабораторной работе №4

Егор Витальевич Кузьмин

Содержание

1 Цель работы **6**

2 Постановка задачи **7**

3 Теоретические сведения **8**
3.1 Агентное моделирование 8
3.2 Модель SIR 8

4 Выполнение работы **9**
4.1 Инициализация проекта 9
4.2 Базовый скрипт модели SIR 10
4.3 Параметризованная версия базового скрипта 19
4.4 Скрипт параметрического исследования коэффициента заражения . 29
4.5 Параметризованная версия второго скрипта 45
4.6 Скрипт исследования миграции 55
4.7 Параметризованная версия скрипта миграции 70
4.8 Скрипт оптимизации параметров модели 82
4.9 Параметризованная версия скрипта оптимизации 92
4.10 Скрипт комплексной визуализации результатов 99

5 Дополнительные задания **107**
5.1 Дополнительные задания 1–3 107
5.2 Дополнительные задания 4–5 120
5.3 Дополнительное задание 6 134

6 Итоговые выводы **143**

7 Список литературы **144**

Список иллюстраций

4.1 Проверка установленных пакетов Julia и создание первого файла . . . 9
4.2 Выполнение первого скрипта, создание literate-версии и генерация необходимых файлов . . . 13
4.3 Notebook literate-версии первого скрипта . . . 13
4.4 График динамики базовой модели SIR . . . 14
4.5 Notebook параметризованной literate-версии первого скрипта . . . 20
4.6 Сравнение сценариев по доле инфицированных . . . 21
4.7 Сравнение сценариев по доле выздоровевших . . . 22
4.8 Выполнение второго скрипта в обычном формате . . . 35
4.9 Notebook literate-версии второго скрипта . . . 35
4.10 Таблица результатов параметрического исследования . . . 36
4.11 Зависимость характеристик эпидемии от коэффициента заражения . 37
4.12 Создание параметрической версии второго скрипта и генерация файлов . . . 45
4.13 Notebook параметризованной версии второго скрипта . . . 46
4.14 Сценарное исследование смертности . . . 46
4.15 Сценарное исследование пика эпидемии . . . 47
4.16 Создание и выполнение третьего скрипта, подготовка literate-версии 60
4.17 Notebook literate-версии третьего скрипта . . . 61
4.18 Влияние интенсивности миграции на динамику эпидемии . . . 62
4.19 Создание параметрической версии третьего скрипта . . . 70
4.20 Notebook параметрической версии третьего скрипта . . . 71
4.21 Сценарное исследование времени до пика при миграции . . . 72
4.22 Сценарное исследование пикового уровня инфицирования при миграции . . . 73
4.23 Создание и выполнение четвёртого скрипта, подготовка literate-версии . . . 85
4.24 Notebook literate-версии четвёртого скрипта . . . 86
4.25 Генерация файлов и notebook для параметризованной версии четвёртого скрипта . . . 92
4.26 Выполненная параметризованная версия четвёртого скрипта . . . 92
4.27 Создание и выполнение пятого скрипта, генерация literate-версии и notebook . . . 101
4.28 Комплексный анализ результатов моделирования . . . 102

5.1 Создание и выполнение скрипта дополнительных заданий 1–3 . . . 116

5.2 Базовая динамика SIR для дополнительных заданий 117
5.3 Исследование порога эпидемии 118
5.4 Динамика инфицированных по городам при гетерогенных параметрах 119
5.5 Общая динамика при гетерогенных параметрах 120
5.6 Создание и выполнение скрипта дополнительных заданий 4–5 . . . 132
5.7 Сравнение времени достижения пика с карантином и без карантина 132
5.8 Сравнение величины пика с карантином и без карантина 133
5.9 Проверка оптимального решения для дополнительного задания 6 . . 142

Список таблиц

1 Цель работы

Целью лабораторной работы является изучение реализации эпидемиологической модели SIR в агентном подходе, исследование влияния параметров на динамику распространения инфекции, освоение литературного программирования с использованием пакета Literate.jl, а также выполнение дополнительных вычислительных экспериментов, связанных с порогом эпидемии, гетерогенностью, миграцией, карантинными мерами и оптимизацией параметров модели.

2 Постановка задачи

В ходе выполнения лабораторной работы требовалось:

1. реализовать базовую модель SIR в агентном подходе;
2. выполнить запуск скриптов в обычном формате;
3. оформить решения в literate-стиле;
4. сгенерировать notebook и документацию Quarto;
5. выполнить параметрические исследования;
6. проанализировать влияние миграции;
7. исследовать эффекты карантинных мер;
8. решить задачу оптимизации параметров;
9. оформить результаты в виде отчёта.

3 Теоретические сведения

3.1 Агентное моделирование

Агентное моделирование представляет систему как совокупность взаимодействующих агентов. Каждый агент описывается собственным состоянием и набором правил поведения. Глобальная динамика системы формируется как результат большого числа локальных взаимодействий.

3.2 Модель SIR

Модель SIR описывает распространение инфекционного заболевания через три состояния агента:

- S — восприимчивый;
- I — инфицированный;
- R — выздоровевший.

В агентной постановке каждый человек задаётся отдельным агентом. В рассматриваемой лабораторной работе учитываются заражение, выздоровление, повторное заражение, смертность и перемещение агентов между городами.

4 Выполнение работы

4.1 Инициализация проекта

На первом этапе была выполнена активация проекта, проверка установленных библиотек Julia и подготовка первого рабочего файла.

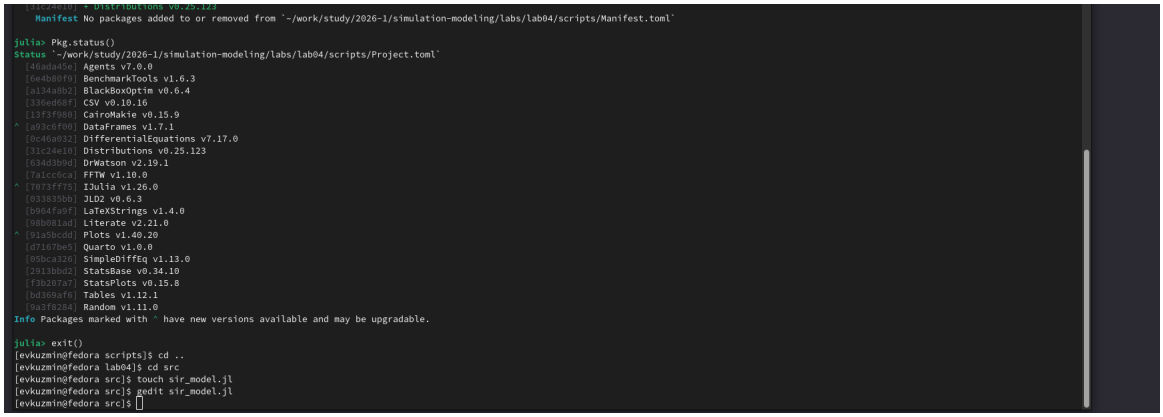


Рисунок 4.1: Проверка установленных пакетов Julia и создание первого файла

На рисунке показан этап инициализации проекта. Видно, что окружение Julia активировано корректно и доступны необходимые библиотеки для выполнения лабораторной работы.

4.2 Базовый скрипт модели SIR

4.2.1 Обычная версия

Для начала была подготовлена и выполнена обычная версия первого скрипта, реализующего базовую динамику модели SIR.

Листинг 4.1. Обычная версия базового скрипта sir_run_basic.jl

```
using DrWatson
@quickactivate "project"

using Agents, DataFrames, Plots
using JLD2

include(srcdir("sir_model.jl"))

# □□□□□□□□□ □□□□□□□□□□□□
params = Dict(
    :Ns => [1000, 1000, 1000],
    :□_und => [0.5, 0.5, 0.5],
    :□_det => [0.05, 0.05, 0.05],
    :infection_period => 14,
    :detection_time => 7,
    :death_rate => 0.02,
    :reinfection_probability => 0.1,
    :Is => [0, 0, 1],
    :seed => 42,
    :n_steps => 100,
)
```

```julia
# □□□□□□□□□□□□□ □□□□□□
model = initialize_sir(; params...)

# □□□□□□□□□□ □□□□□□□□ □□□ □□□□□□□□ □□□□□□
times = Int[]
S_vals = Int[]
I_vals = Int[]
R_vals = Int[]
total_vals = Int[]

# □□□□□□ □□□□□□□□□ □□□□□□□
for step = 1:params[:n_steps]
    Agents.step!(model, 1)
    push!(times, step)
    push!(S_vals, susceptible_count(model))
    push!(I_vals, infected_count(model))
    push!(R_vals, recovered_count(model))
    push!(total_vals, total_count(model))
end

# □□□□□□□ DataFrame □□□ □□□□□□□□ (□□□□□□□□□□□)
agent_df = DataFrame(
    time = times,
    susceptible = S_vals,
    infected = I_vals,
    recovered = R_vals,
)
```

```
model_df = DataFrame(
    time = times,
    total = total_vals,
)

# □□□□□□□□□□□□
plot(
    agent_df.time,
    agent_df.susceptible,
    label = "□□□□□□□□□□□□□",
    xlabel = "□□□",
    ylabel = "□□□□□□□□□□",
)
plot!(agent_df.time, agent_df.infected, label = "□□□□□□□□□□□□□□")
plot!(agent_df.time, agent_df.recovered, label = "□□□□□□□□□□□□□")
plot!(agent_df.time, model_df.total, label = "□□□□□ (□□□□□□□ □□□□□□□)",

savefig(plotsdir("sir_basic_dynamics.png"))

# □□□□□□□□□□ □□□□□□
@save datadir("sir_basic_agent.jld2") agent_df
@save datadir("sir_basic_model.jld2") model_df
```

4.2.2 Выполнение и генерация literate-версии

После запуска обычной версии была сформирована literate-версия и сгенерированы производные файлы: чистый код, notebook и Quarto-документация.

Рисунок 4.2: Выполнение первого скрипта, создание literate-версии и генерация необходимых файлов

На рисунке показан этап выполнения первого скрипта и генерации literate-артефактов. Также видна проверка того, что необходимые файлы были созданы.

4.2.3 Notebook literate-версии первого скрипта

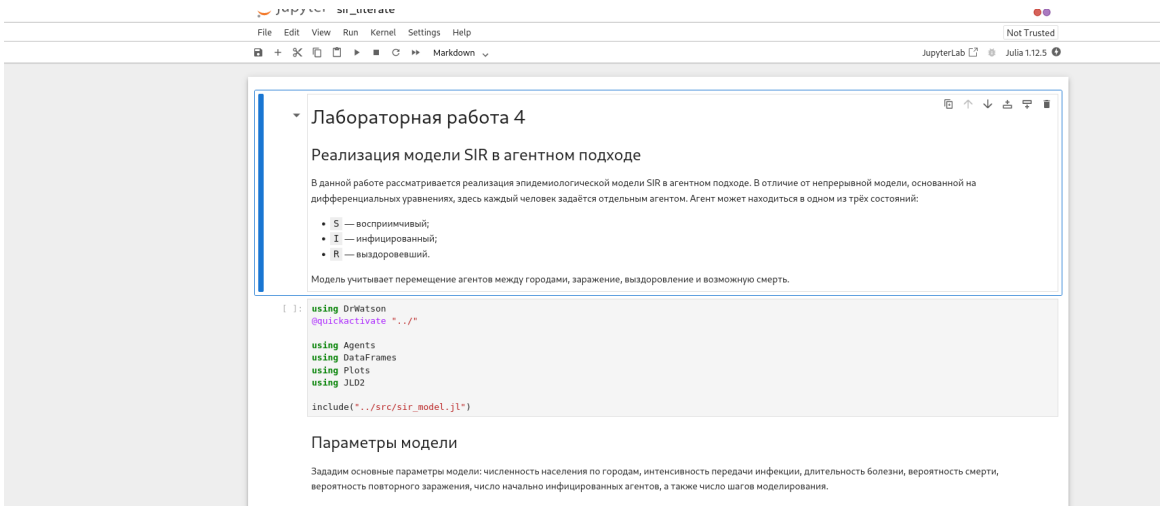


Рисунок 4.3: Notebook literate-версии первого скрипта

На рисунке представлен notebook, полученный из literate-версии первого скрипта. Это подтверждает корректность преобразования исходного файла в

формат документации.

4.2.4 Результат выполнения базовой модели

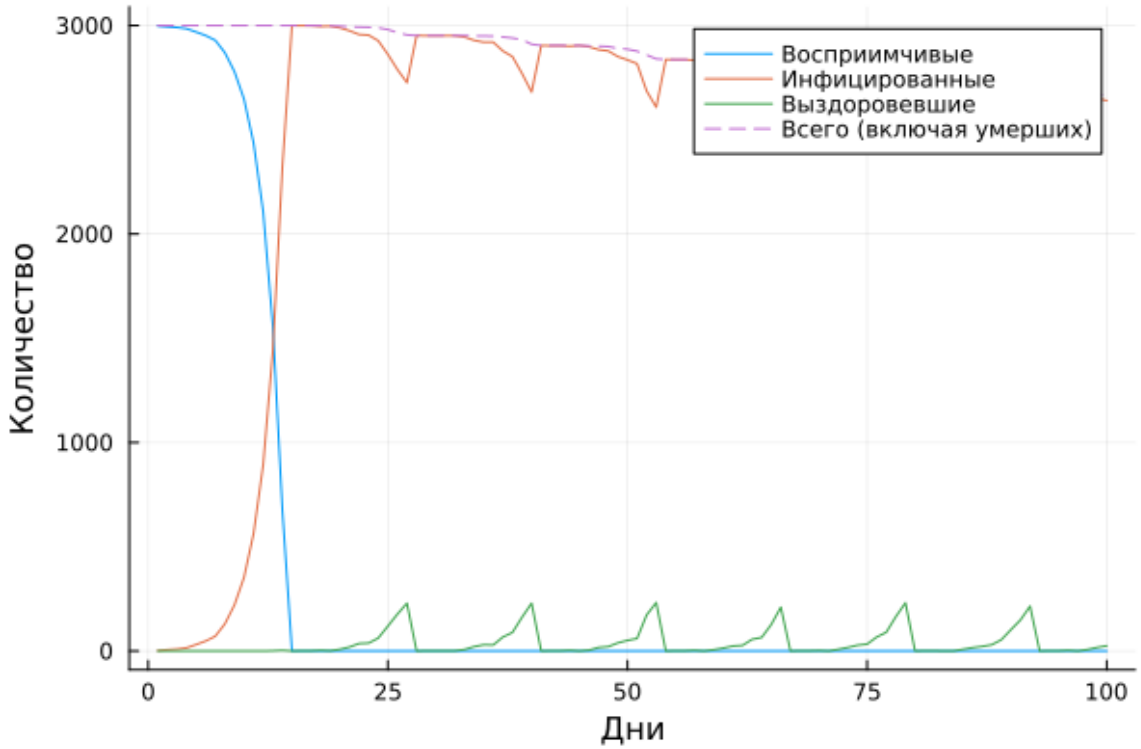


Рисунок 4.4: График динамики базовой модели SIR

На рисунке представлена динамика числа восприимчивых, инфицированных и выздоровевших агентов в базовой модели. В начале моделирования число инфицированных растёт, затем после достижения максимума начинает уменьшаться. Одновременно сокращается число восприимчивых агентов и увеличивается число выздоровевших, что соответствует классической интерпретации модели SIR.

4.2.5 Literate-версия базового скрипта

Листинг 4.2. Literate-версия базового скрипта sir_literate.jl

```
# # [illegible] 4
# ## [illegible] SIR [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] SIR
# [illegible] [illegible]. [illegible] [illegible] [illegible], [illegible] [illegible]
# [illegible] [illegible], [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]:
#
# - `S` — [illegible];
# - `I` — [illegible];
# - `R` — [illegible].
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible], [illegible],
# [illegible] [illegible] [illegible].

using DrWatson
@quickactivate "../"

using Agents
using DataFrames
using Plots
using JLD2

include("../src/sir_model.jl")
```

```
# ## [redacted]
#
# [redacted]: [redacted] [redacted] [redacted],
# [redacted] [redacted], [redacted], [redacted], [redacted],
# [redacted] [redacted] [redacted], [redacted] [redacted] [redacted],
# [redacted] [redacted] [redacted] [redacted].

params = Dict(
    :Ns => [1000, 1000, 1000],
    :[redacted]_und => [0.5, 0.5, 0.5],
    :[redacted]_det => [0.05, 0.05, 0.05],
    :infection_period => 14,
    :detection_time => 7,
    :death_rate => 0.02,
    :reinfection_probability => 0.1,
    :Is => [0, 0, 1],
    :seed => 42,
    :n_steps => 100,
)

# ## [redacted]
#
# [redacted] [redacted] [redacted] [redacted] [redacted].

model = initialize_sir(; params...)

# ## [redacted] [redacted] [redacted] [redacted]
#
```

```
# □□ □□□□□□ □□□□ □□□□□□□□□□□□□ □□□□□ □□□□□□□□□:
#
# - □□□□□ □□□□;
# - □□□□□ □□□□□□□□□□□□□;
# - □□□□□ □□□□□□□□□□□□□□;
# - □□□□□ □□□□□□□□□□□□□;
# - □□□□□ □□□□□ □□□□□□□.

times = Int[]
S_vals = Int[]
I_vals = Int[]
R_vals = Int[]
total_vals = Int[]

# ## □□□□□□□□□□ □□□□□□□□□□□□□
#
# □□□□□□□□ □□□□□□ □□ □□□□□□□□ □□□□□ □□□□□ □ □□ □□□□□□ □□□□ □□□□□□□□ □□□□□□□
# □□□□□□□□ □□□□□□□□ □□□□□□□□□□□□□.

for step = 1:params[:n_steps]
    Agents.step!(model, 1)
    push!(times, step)
    push!(S_vals, susceptible_count(model))
    push!(I_vals, infected_count(model))
    push!(R_vals, recovered_count(model))
    push!(total_vals, total_count(model))
end
```

```
# ## □□□□□□□□□□□□ □□□□□□ □□□□□□□□□□□
#
# □□□ □□□□□□□□ □□□□□□□□□□□ □□□□□□□ □□□□□□□ □□□□□□□□□□□ □□□□□□ □ □□□□ □□□□□□.

agent_df = DataFrame(
    time = times,
    susceptible = S_vals,
    infected = I_vals,
    recovered = R_vals,
)

model_df = DataFrame(
    time = times,
    total = total_vals,
)

# ## □□□□□□□□□□□□ □□□□□□□□□□□
#
# □□□□□□□□ □□□□□□ □□□□□□□□□ □□□□□□□□□□□ □□□□□ □□ □□□□□□□.

plot(
    agent_df.time,
    agent_df.susceptible,
    label = "□□□□□□□□□□□□□□",
    xlabel = "□□□",
    ylabel = "□□□□□□□□□□",
)
plot!(agent_df.time, agent_df.infected, label = "□□□□□□□□□□□□□□□")
```

```
plot!(agent_df.time, agent_df.recovered, label = "[redacted]")
plot!(agent_df.time, model_df.total, label = "[redacted] ([redacted] [redacted])",

savefig(plotsdir("sir_basic_dynamics_literate.png"))

# ## [redacted] [redacted]
#
# [redacted] [redacted] [redacted] [redacted] [redacted] JLD2.

@save datadir("sir_basic_agent_literate.jld2") agent_df
@save datadir("sir_basic_model_literate.jld2") model_df

# ## [redacted]
#
# [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted]
# [redacted], [redacted] [redacted] [redacted], [redacted] [redacted]
# [redacted] [redacted] [redacted] [redacted].
```

4.3 Параметризованная версия базового скрипта

Следующим этапом была подготовлена параметризованная literate-версия первого скрипта.

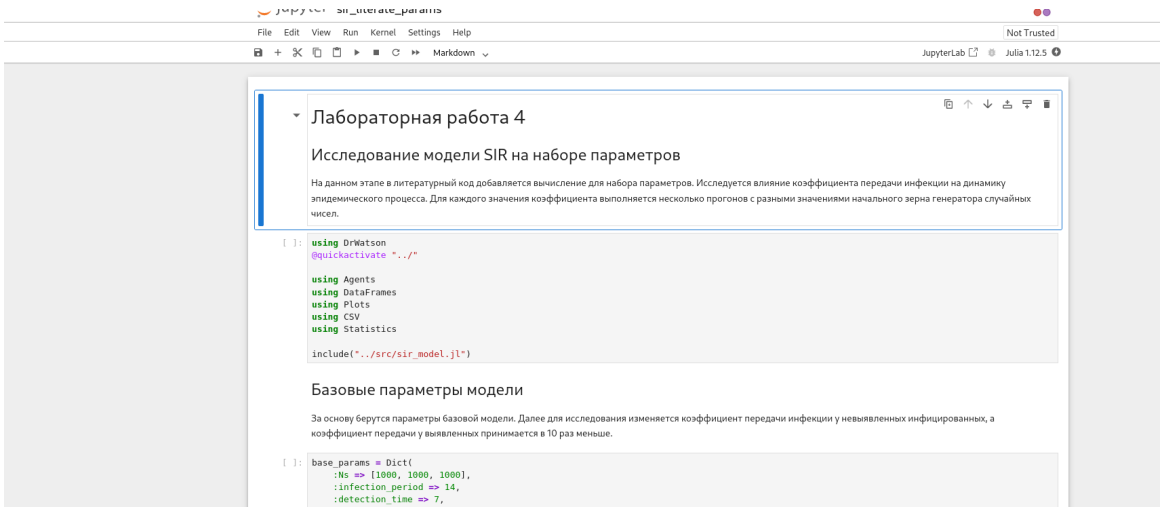


Рисунок 4.5: Notebook параметризованной literate-версии первого скрипта

На рисунке показан notebook параметризованной версии первого скрипта.

4.3.1 Результаты параметризованной версии

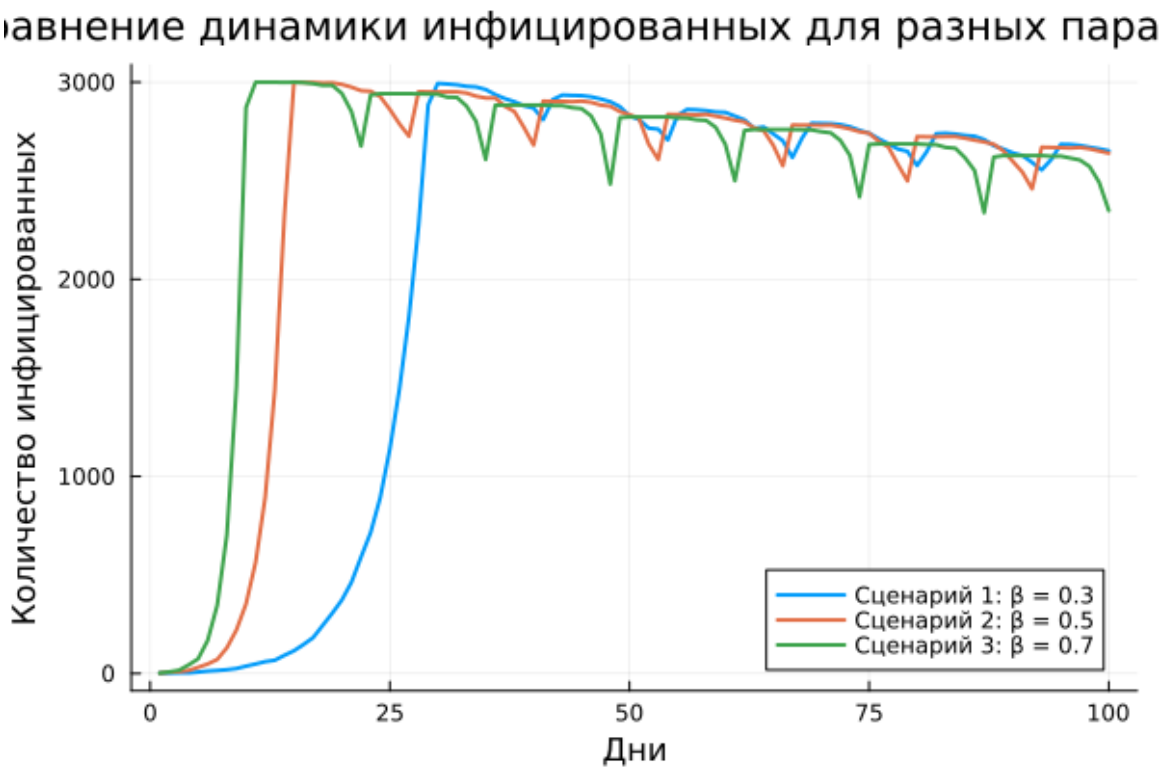


Рисунок 4.6: Сравнение сценариев по доле инфицированных

На рисунке сравниваются сценарии по доле инфицированных. Видно, что более высокий коэффициент заражения ускоряет развитие эпидемии и увеличивает максимум числа инфицированных.

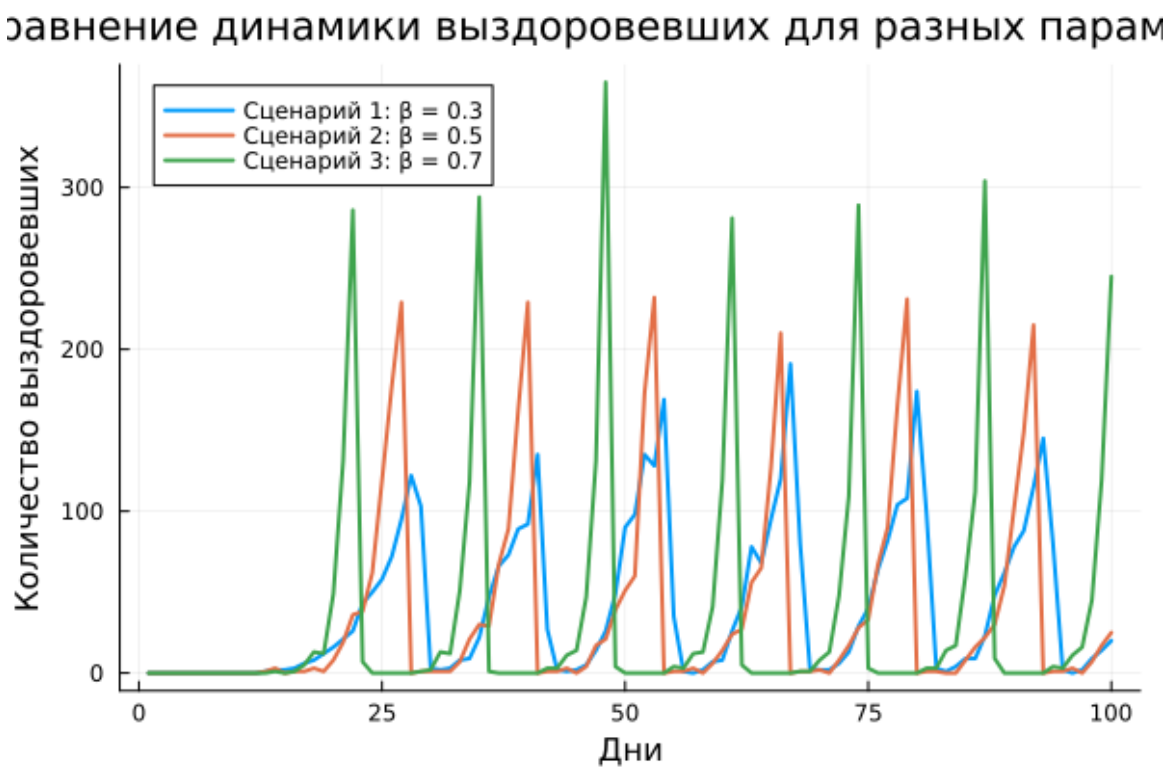


Рисунок 4.7: Сравнение сценариев по доле выздоровевших

На рисунке показана динамика выздоровевших для различных параметров. При более интенсивном распространении инфекции число выздоровевших также растёт быстрее.

Листинг 4.3. Параметризованная literate-версия базового скрипта sir_literate_params.jl

```
# # [illegible] 4
# ## [illegible] SIR [illegible] [illegible]
# ## [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible], [illegible]
# [illegible] [illegible] [illegible] [illegible] [illegible].
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
```

```julia
# □□□ □□□□□□ □□□□□□□□ □□□□□□□□□□□□ □□□□□□□□ □□□□□□□□.

using DrWatson
@quickactivate "../"

using Agents
using DataFrames
using Plots
using JLD2

include("../src/sir_model.jl")

# ## □□□□□□□ □□□□□□□□□ □□□□□□
#
# □□□□□□□ □□□□□□□□□, □□□□□□□ □□□□□ □□□□□□□□□□□□ □□□ □□□□ □□□□□□□□□.

base_params = Dict(
    :Ns => [1000, 1000, 1000],
    :infection_period => 14,
    :detection_time => 7,
    :death_rate => 0.02,
    :reinfection_probability => 0.1,
    :Is => [0, 0, 1],
    :seed => 42,
    :n_steps => 100,
)

# ## □□□□□ □□□□□□□□□□
```

```
#
# □□□□□□□□□□ □□□□□□□□□ □□□□□□□□□ □ □□□□□□□ □□□□□□□□□□ □□□□□□□□□□□□
# □□□□□□□□ □□□□□□□□ □ □□□□□□□□□□□□ □□□□□□□□□□□□□□. □□□ □□□□□□□□□□□
# □□□□□□□□□□□□□□ □□□□□□□□□□□ □□□□□□□□ □□□□□□ □ 10 □□□ □□□□□□.

param_sets = [
    (name = "□□□□□□□□ 1: □ = 0.3", □_und = [0.3, 0.3, 0.3], □_det = [0
    (name = "□□□□□□□□ 2: □ = 0.5", □_und = [0.5, 0.5, 0.5], □_det = [0
    (name = "□□□□□□□□ 3: □ = 0.7", □_und = [0.7, 0.7, 0.7], □_det = [0
]

# ## □□□□□□□ □□□□□□□ □□□□□□ □□□□□□□□
#
# □□□ □□□□□□□ □□□□□□ □□□□□□□□□□ □□□□□□□□□□ □□□□□□□□□ □□□□□□□□□□□□□
# □ □□□□□□□□□□□ □□□□□□□□□ □□□□ □□□□□□□□ □□□□□□□□□□□□□.

function run_scenario(base_params, □_und, □_det, scenario_name)
    params = Dict(
        :Ns => base_params[:Ns],
        :□_und => □_und,
        :□_det => □_det,
        :infection_period => base_params[:infection_period],
        :detection_time => base_params[:detection_time],
        :death_rate => base_params[:death_rate],
        :reinfection_probability => base_params[:reinfection_probab
        :Is => base_params[:Is],
        :seed => base_params[:seed],
        :n_steps => base_params[:n_steps],
```

```
    )

    model = initialize_sir(; params...)

    times = Int[]
    S_vals = Int[]
    I_vals = Int[]
    R_vals = Int[]
    total_vals = Int[]

    for step = 1:params[:n_steps]
        Agents.step!(model, 1)
        push!(times, step)
        push!(S_vals, susceptible_count(model))
        push!(I_vals, infected_count(model))
        push!(R_vals, recovered_count(model))
        push!(total_vals, total_count(model))
    end

    agent_df = DataFrame(
        time = times,
        susceptible = S_vals,
        infected = I_vals,
        recovered = R_vals,
        scenario = fill(scenario_name, length(times)),
    )

    model_df = DataFrame(
```

```
        time = times,
        total = total_vals,
        scenario = fill(scenario_name, length(times)),
    )

    return agent_df, model_df
end

# ## □□□□□□□□□□ □□□□□□□□□□□□□ □□□ □□□□ □□□□□□□ □□□□□□□□□□
#
# □□□□□□□□ □□□□□□ □□□ □□□□□□□ □□□□□□□□ □ □□□□□□□□□ □□□□□□□□□□
# □ □□□□□ □□□□□□□.

all_agent_df = DataFrame()
all_model_df = DataFrame()

for p in param_sets
    agent_df, model_df = run_scenario(base_params, p.□_und, p.□_det,
    append!(all_agent_df, agent_df)
    append!(all_model_df, model_df)
end

# ## □□□□□□□□ □□□□□□□□□□□
#
# □□□□□□□ □□□□□□ □□□□□□ □□□□□□□□□□□□ □□□□□□□.

first(all_agent_df, 10)
```

```
# ## □□□□□□□□□□□□ □□□□□□□□□□□□
#
# □□□□□□□□ □□□□□□□□□□□□□ □□□□□□ □□□□□ □□□□□□□□□□□□□□ □□□ □□□□ □□□□□□□□□.

plot(
    xlabel = "□□□□",
    ylabel = "□□□□□□□□□□□ □□□□□□□□□□□□□□□",
    title = "□□□□□□□□□□ □□□□□□□□□ □□□□□□□□□□□□□□□ □□□□ □□□□□□ □□□□□□□□□□□",
)

for p in param_sets
    subset_df = all_agent_df[all_agent_df.scenario .== p.name, :]
    plot!(
        subset_df.time,
        subset_df.infected,
        label = p.name,
        lw = 2,
    )
end

savefig(plotsdir("sir_literate_params_infected.png"))

# ## □□□□□□□□□□□□□□□ □□□□□□□
#
# □□□□□□□□□ □□□□□□ □□□□□□□□□□□□□□ □□□□□□□ □□□□□□ □□□□□□□□□□□□□□□.

plot(
    xlabel = "□□□□",
```

```
    ylabel = "□□□□□□□□□□ □□□□□□□□□□□□□",
    title = "□□□□□□□□□ □□□□□□□□ □□□□□□□□□□□□□ □□□ □□□□□□ □□□□□□□□□□",
)

for p in param_sets
    subset_df = all_agent_df[all_agent_df.scenario .== p.name, :]
    plot!(
        subset_df.time,
        subset_df.recovered,
        label = p.name,
        lw = 2,
    )
end

savefig(plotsdir("sir_literate_params_recovered.png"))

# ## □□□□□□□□□□ □□□□□□□□□□□
#
# □□□□□□□□ □□□□□□□□□□□□ □□□□□□□□□□ □ □□□□□ □□□□□□□ JLD2.

@save datadir("sir_literate_params_agent.jld2") all_agent_df
@save datadir("sir_literate_params_model.jld2") all_model_df

# ## □□□□□
#
# □ □□□□□□□□□□ □□□□ □□□□□□□□□ □□□□□□□□□ □□□□□□□□ □□□□□□ SIR
# □□□ □□□□□□□□□ □□□□□□□□ □□□□□□□□□□□□ □□□□□□□□ □□□□□□□□.
# □□□ □□□□□□□□□ □□□□□□□□ □□□□□□□□ □□□□□ □□□□□□□□□□□□□□ □ □□□□□□□□□□□□□
```

```
# □□□□□□□ □□ □□□□□□□ □□□ □□□□□□ □□□□□□□□□ □□□□□□□□ □□□□□□□□.
```

4.4 Скрипт параметрического исследования коэффициента заражения

Во втором скрипте реализуется перебор значений коэффициента заражения β и вычисление ключевых характеристик эпидемического процесса.

4.4.1 Обычная версия

Листинг 4.4. Обычная версия скрипта параметрического исследования sir_scan_beta.jl

```
using DrWatson
@quickactivate "project"

using Agents, DataFrames, Plots, CSV, Random
using Statistics

include(srcdir("sir_model.jl"))

# □□□□□□□ □□□□□□□ □□□□□□ □□□□□□□□□□□□ □ □□□□□□□□ □□□□□□
function run_experiment(p)
# □□□□□□□ □_und □ □_det □□ □□□□□□ □□□□□□□□□□ beta
beta = p[:beta]
□_und = fill(beta, 3)
□_det = fill(beta / 10, 3)
```

```
# □□□□□□□□□□□□□ □□□□□□
model = initialize_sir(;
Ns = p[:Ns],
□_und = □_und,
□_det = □_det,
infection_period = p[:infection_period],
detection_time = p[:detection_time],
death_rate = p[:death_rate],
reinfection_probability = p[:reinfection_probability],
Is = p[:Is],
seed = p[:seed],
n_steps = p[:n_steps],
)

# □□□□ □□□□□□□□□□□□□ □ □□□□□□□ □□□□□□
infected_fraction(model) =
count(a.status == :I for a in allagents(model)) / nagents(model)

peak_infected = 0.0

# □□□□□□□□ □□□□ □□□□□□□□□□□□□
for step = 1:p[:n_steps]
# □□□□□□ □□□□□□□□□□ □□□ □□ □□□□□□□
agent_ids = collect(allids(model))
for id in agent_ids
agent = try
model[id]
catch
```

```
nothing
end

if agent !== nothing
sir_agent_step!(agent, model)
end
end

frac = infected_fraction(model)
if frac > peak_infected
peak_infected = frac
end
end

# ⌧⌧⌧⌧⌧⌧⌧⌧ ⌧⌧⌧⌧⌧⌧⌧
final_infected = infected_fraction(model)
final_recovered = count(a.status == :R for a in allagents(model)) /
total_deaths = sum(p[:Ns]) - nagents(model)

return (
peak = peak_infected,
final_inf = final_infected,
final_rec = final_recovered,
deaths = total_deaths,
)
end

# ⌧⌧⌧⌧⌧⌧⌧⌧ ⌧⌧⌧⌧⌧⌧⌧⌧ beta
```

```
beta_range = 0.1:0.1:1.0

# □□□□□ □□□□□□□□ seed □□□ □□□□□□□□□ □□□□□□□□
seeds = [42, 43, 44]

# □□□□□□□□□□□□ □□□□□□ □□□□□□□□□□□
params_list = []

for b in beta_range
for s in seeds
push!(
params_list,
Dict(
:beta => b,
:Ns => [1000, 1000, 1000],
:infection_period => 14,
:detection_time => 7,
:death_rate => 0.02,
:reinfection_probability => 0.1,
:Is => [0, 0, 1],
:seed => s,
:n_steps => 100,
),
)
end
end

# □□□□□□□ □□□□□ □□□□□□□□□□□□□□
```

```
results = []

for params in params_list
data = run_experiment(params)
push!(results, merge(params, Dict(pairs(data))))
println("□□□□□□□□ □□□□□□□□□□□ □ beta = $(params[:beta]), seed = $(params
end

# □□□□□□□□□ □□□□□□□□□□ □□□□ □□□□□□□□
df = DataFrame(results)
CSV.write(datadir("beta_scan_all.csv"), df)

# □□□□□□□□□□ □□□□□□□□□□ □□ □□□□□□□□□□ □□□□□□□□□ beta
grouped = combine(
groupby(df, [:beta]),
:peak => mean => :mean_peak,
:final_inf => mean => :mean_final_inf,
:deaths => mean => :mean_deaths,
)

# □□□□□□□□□□ □□□□□□□□□ □□□□□□□
plot(
grouped.beta,
grouped.mean_peak,
label = "□□□ □□□□□□□□",
xlabel = "□□□□□□□□□□□ □□□□□□□□□□ □",
ylabel = "□□□□ □□□□□□□□□□□□□□",
marker = :circle,
```

```
linewidth = 2,
)

plot!(
grouped.beta,
grouped.mean_final_inf,
label = "□□□□□□□□ □□□□ □□□□□□□□□□□□□□",
marker = :square,
)

plot!(
grouped.beta,
grouped.mean_deaths ./ 3000,
label = "□□□□ □□□□□□□",
marker = :diamond,
)

savefig(plotsdir("beta_scan.png"))

println("□□□□□□□□□□ □□□□□□□□□ □ data/beta_scan_all.csv □ plots/beta_scan
```

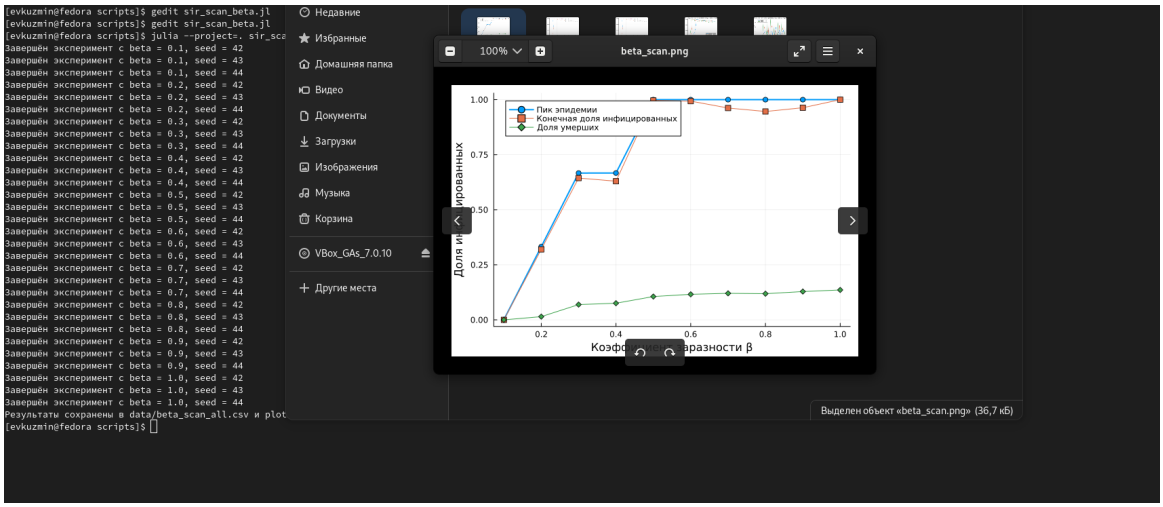


Рисунок 4.8: Выполнение второго скрипта в обычном формате

На рисунке показан запуск второго скрипта в обычной версии.

4.4.2 Notebook literate-версии

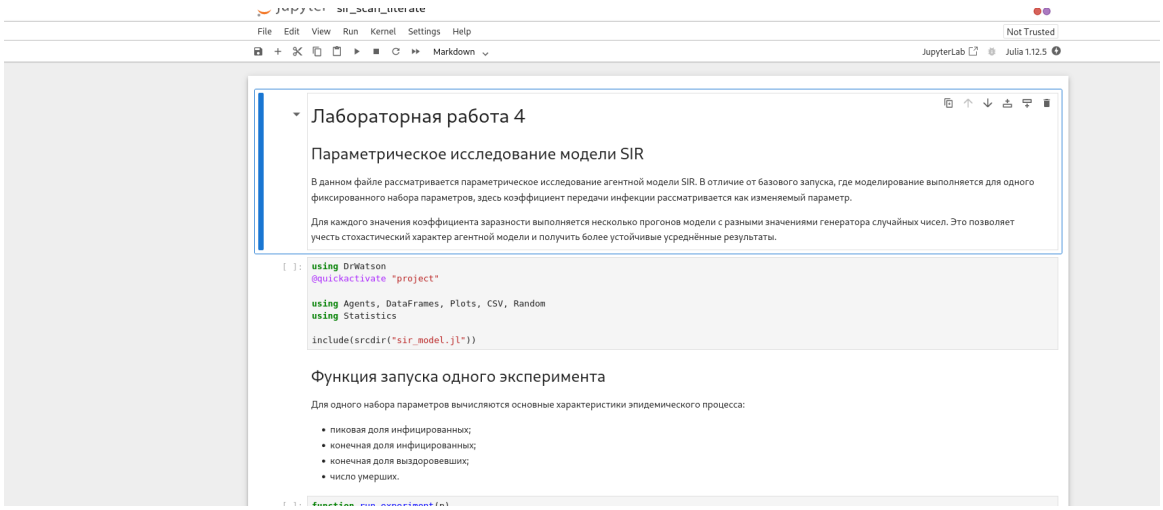
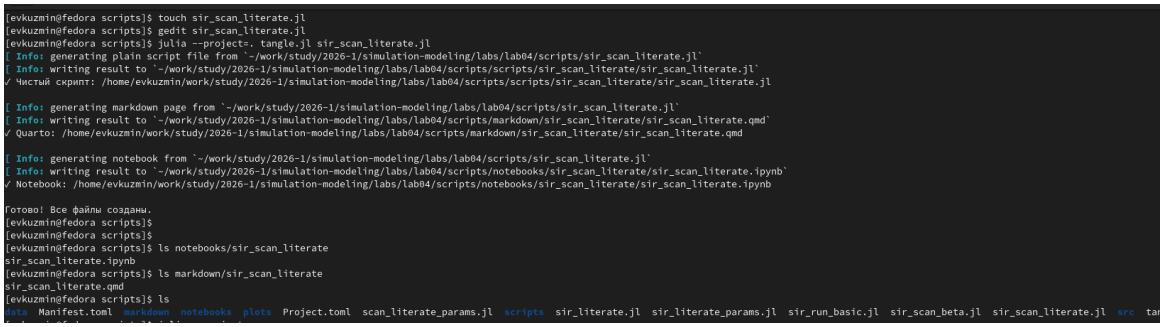


Рисунок 4.9: Notebook literate-версии второго скрипта

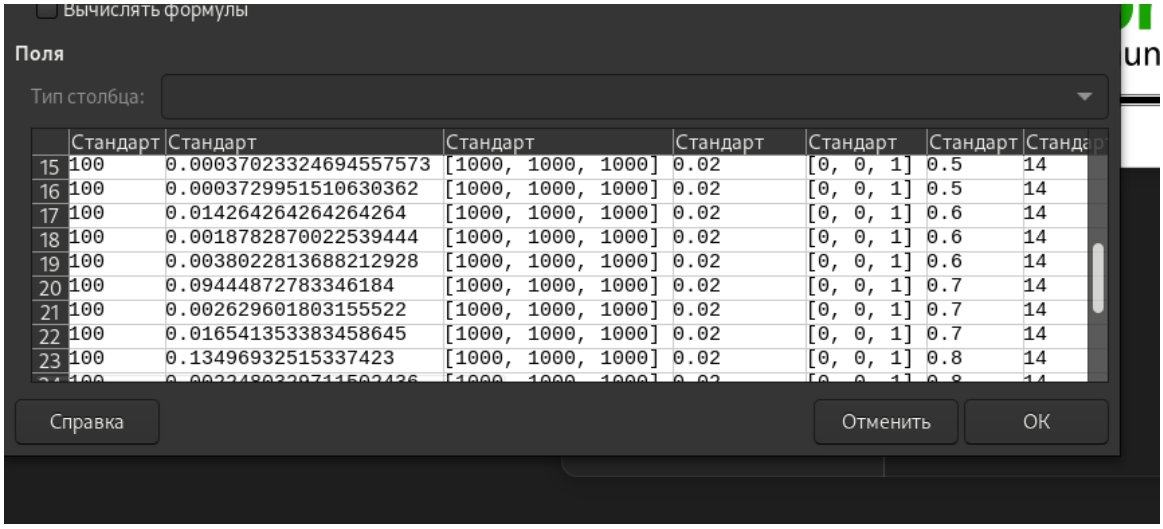
На рисунке представлен notebook literate-версии второго скрипта.

4.4.3 Генерация notebook и документации



На рисунке показан этап генерации notebook и Quarto-документации для literate-версии второго скрипта.

4.4.4 Таблица результатов



Вычислять формулы

Поля

Тип столбца:

	Стандарт	Стандарт	Стандарт	Стандарт	Стандарт	Стандарт	Станда
15	100	0.00037023324694557573	[1000, 1000, 1000]	0.02	[0, 0, 1]	0.5	14
16	100	0.0003729951510630362	[1000, 1000, 1000]	0.02	[0, 0, 1]	0.5	14
17	100	0.014264264264264264	[1000, 1000, 1000]	0.02	[0, 0, 1]	0.6	14
18	100	0.0018782870022539444	[1000, 1000, 1000]	0.02	[0, 0, 1]	0.6	14
19	100	0.0038022813688212928	[1000, 1000, 1000]	0.02	[0, 0, 1]	0.6	14
20	100	0.09444872783346184	[1000, 1000, 1000]	0.02	[0, 0, 1]	0.7	14
21	100	0.002629601803155522	[1000, 1000, 1000]	0.02	[0, 0, 1]	0.7	14
22	100	0.016541353383458645	[1000, 1000, 1000]	0.02	[0, 0, 1]	0.7	14
23	100	0.13496932515337423	[1000, 1000, 1000]	0.02	[0, 0, 1]	0.8	14

Справка Отменить ОК

Рисунок 4.10: Таблица результатов параметрического исследования

На рисунке представлена таблица результатов параметрического исследования. Она содержит значения β и соответствующие показатели: пик эпидемии, конечную долю инфицированных, долю выздоровевших и число умерших.

4.4.5 Результат параметрического исследования

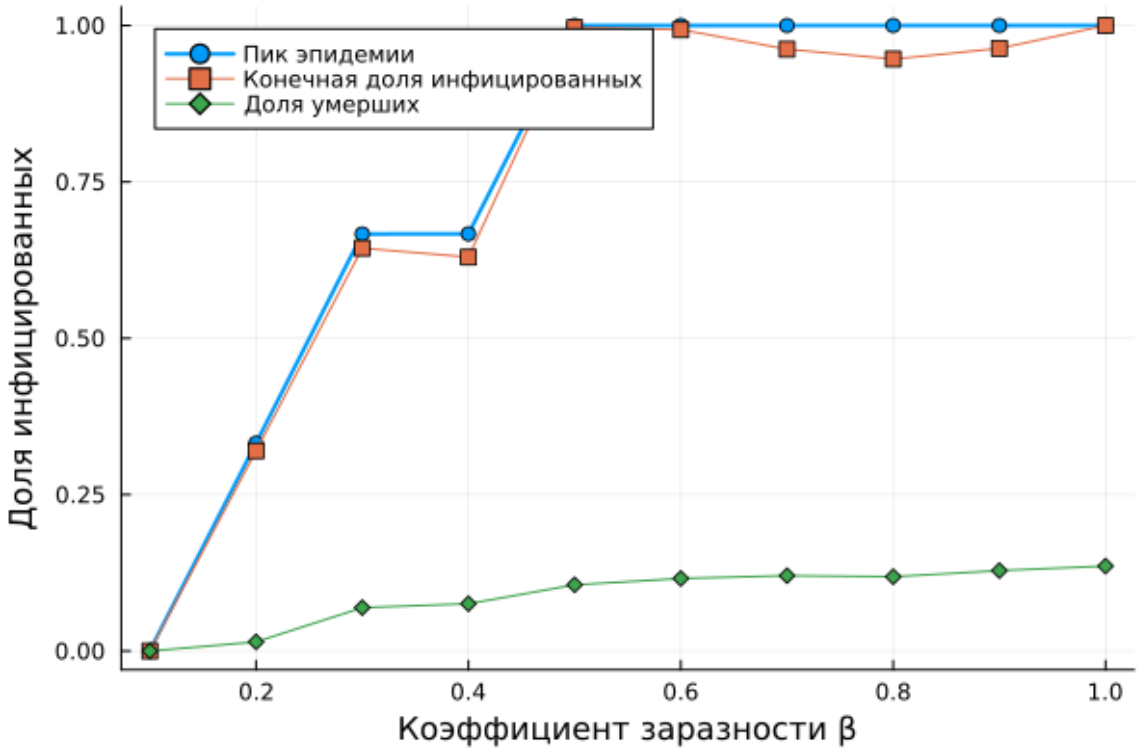


Рисунок 4.11: Зависимость характеристик эпидемии от коэффициента заражения

На рисунке показано влияние коэффициента заражения на основные характеристики эпидемии. С ростом β увеличиваются пик эпидемии, конечная доля инфицированных и смертность. Это соответствует ожидаемой интерпретации: более высокая заразность приводит к более интенсивному развитию эпидемического процесса.

4.4.6 Literate-версия второго скрипта

Листинг 4.5. Literate-версия скрипта параметрического исследования sir_scan_literate.jl

```
# # [illegible] 4
# ## [illegible] SIR
#
# [illegible]
# [illegible] SIR. [illegible], [illegible]
# [illegible]
# [illegible]
#
# [illegible]
# [illegible] [illegible]. [illegible]
# [illegible]
# [illegible].

using DrWatson
@quickactivate "project"

using Agents, DataFrames, Plots, CSV, Random
using Statistics

include(srcdir("sir_model.jl"))

# ## [illegible]
#
# [illegible]
# [illegible]:
#
# - [illegible];
# - [illegible];
```

```
# - [illegible] [illegible] [illegible];
# - [illegible] [illegible].

function run_experiment(p)
# [illegible] β_und β β_det [illegible] [illegible] [illegible] beta
beta = p[:beta]
β_und = fill(beta, 3)
β_det = fill(beta / 10, 3)

# [illegible] [illegible]
model = initialize_sir(;
Ns = p[:Ns],
β_und = β_und,
β_det = β_det,
infection_period = p[:infection_period],
detection_time = p[:detection_time],
death_rate = p[:death_rate],
reinfection_probability = p[:reinfection_probability],
Is = p[:Is],
seed = p[:seed],
n_steps = p[:n_steps],
)

# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
infected_fraction(model) =
count(a.status == :I for a in allagents(model)) / nagents(model)

peak_infected = 0.0
```

```
# □□□□□□□□ □□□□ □□□□□□□□□□□□□
for step = 1:p[:n_steps]
# □□□□□□□□□ □□□□□□□□□□ □□□□□□ □□□ □□ □□□□ □□□□□□□
agent_ids = collect(allids(model))
for id in agent_ids
agent = try
model[id]
catch
nothing
end

if agent !== nothing
sir_agent_step!(agent, model)
end
end

frac = infected_fraction(model)
if frac > peak_infected
peak_infected = frac
end
end

# □□□□□□□□ □□□□□□□□□□□□□□
final_infected = infected_fraction(model)
final_recovered = count(a.status == :R for a in allagents(model)) /
total_deaths = sum(p[:Ns]) - nagents(model)

return (
```

```
peak = peak_infected,
final_inf = final_infected,
final_rec = final_recovered,
deaths = total_deaths,
)
end

# ## □□□□□□□□ □□□□□□□□□□
#
# □□□□□□□□□□□□□□□ □□□□□□□□ □□□□□□□□□□□□ □□□□□□□□ □□□□□□□□ □□ 0.1 □□ 1.0
# □ □□□□□ 0.1. □□□ □□□□□□□ □□□□□□□□ □□□□□□□□□□□ □□□ □□□□□□□ □□□□□□
# □ □□□□□□□□□□ □□□□□□□□□□ seed.

beta_range = 0.1:0.1:1.0
seeds = [42, 43, 44]

# ## □□□□□□□□□□□□ □□□□□□ □□□□□□□□□□
#
# □□□ □□□□□□□ □□□□□□□□ beta □ □□□□□□□ □□□□□□□□ seed □□□□□□□□□ □□□□□□□□□
# □□□□□ □□□□□□□□□□.

params_list = []

for b in beta_range
for s in seeds
push!(
params_list,
Dict(
```

```
:beta => b,
:Ns => [1000, 1000, 1000],
:infection_period => 14,
:detection_time => 7,
:death_rate => 0.02,
:reinfection_probability => 0.1,
:Is => [0, 0, 1],
:seed => s,
:n_steps => 100,
),
)
end
end

# ## □□□□□□ □□□□□□□□□□□□□
#
# □□□ □□□□□□□ □□□□□□ □□□□□□□□□□ □□□□□□□□□□□ □□□□□□□□□□□□□, □ □□□□□
# □□□□□□□□□□ □□□□□□□□□□□ □ □□□□□ □□□□□□.

results = []

for params in params_list
data = run_experiment(params)
push!(results, merge(params, Dict(pairs(data))))
println("□□□□□□□□ □□□□□□□□□□□ □ beta = $(params[:beta]), seed = $(params
end

# ## □□□□□□□□□□ □□□□□□□□□□□
```

```
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] CSV-[illegible].

df = DataFrame(results)
CSV.write(datadir("beta_scan_all.csv"), df)

# ## [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] beta [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible].

grouped = combine(
groupby(df, [:beta]),
:peak => mean => :mean_peak,
:final_inf => mean => :mean_final_inf,
:deaths => mean => :mean_deaths,
)

# ## [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible] beta.

plot(
grouped.beta,
grouped.mean_peak,
label = "[illegible] [illegible]",
```

```
xlabel = "□□□□□□□□□□□ □□□□□□□□□□□ □",
ylabel = "□□□□ □□□□□□□□□□□□□□□",
marker = :circle,
linewidth = 2,
)

plot!(
grouped.beta,
grouped.mean_final_inf,
label = "□□□□□□□□ □□□□ □□□□□□□□□□□□□□□",
marker = :square,
)

plot!(
grouped.beta,
grouped.mean_deaths ./ 3000,
label = "□□□□ □□□□□□□□",
marker = :diamond,
)

savefig(plotsdir("beta_scan.png"))

println("□□□□□□□□□□□ □□□□□□□□□□□ □ data/beta_scan_all.csv □ plots/beta_scan

# ## □□□□□
#
# □ □□□□□□□□□□□ □□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□ □□□□□ □□□□□□□□□ □□□□□□□□□□□□□
# □□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□ □□□□□□□□□ □□□ □□□□□□□□□□ □□□□□□□□□
```

```
# [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible]
# beta [illegible] [illegible] [illegible], [illegible] [illegible]
# [illegible].
```

4.5 Параметризованная версия второго скрипта

На следующем этапе была подготовлена параметрическая версия второго скрипта, а также сгенерированы notebook и qmd-файлы.

```
[evkuzmin@fedora scripts]$ gedit sir_scan_literate.jl
[evkuzmin@fedora scripts]$ touch scan_literate_params.jl
[evkuzmin@fedora scripts]$ gedit scan_literate_params.jl
[evkuzmin@fedora scripts]$ julia --project=. tangle.jl scan_literate_params.jl
[ Info: generating plain script file from `~/work/study/2026-1/simulation-modeling/labs/lab04/scripts/scan_literate_params.jl`
[ Info: writing result to `~/work/study/2026-1/simulation-modeling/labs/lab04/scripts/scripts/scan_literate_params/scan_literate_params.jl`
✓ Чистый скрипт: /home/evkuzmin/work/study/2026-1/simulation-modeling/labs/lab04/scripts/scripts/scan_literate_params/scan_literate_params.jl

[ Info: generating markdown page from `~/work/study/2026-1/simulation-modeling/labs/lab04/scripts/scan_literate_params.jl`
[ Info: writing result to `~/work/study/2026-1/simulation-modeling/labs/lab04/scripts/markdown/scan_literate_params/scan_literate_params.qmd`
✓ Quarto: /home/evkuzmin/work/study/2026-1/simulation-modeling/labs/lab04/scripts/markdown/scan_literate_params/scan_literate_params.qmd

[ Info: generating notebook from `~/work/study/2026-1/simulation-modeling/labs/lab04/scripts/scan_literate_params.jl`
[ Info: writing result to `~/work/study/2026-1/simulation-modeling/labs/lab04/scripts/notebooks/scan_literate_params/scan_literate_params.ipynb`
✓ Notebook: /home/evkuzmin/work/study/2026-1/simulation-modeling/labs/lab04/scripts/notebooks/scan_literate_params/scan_literate_params.ipynb

Готово! Все файлы созданы.
[evkuzmin@fedora scripts]$
[evkuzmin@fedora scripts]$
[evkuzmin@fedora scripts]$ ls notebooks/scan_literate_params.jl
ls: невозможно получить доступ к 'notebooks/scan_literate_params.jl': Нет такого файла или каталога
[evkuzmin@fedora scripts]$ ls notebooks/scan_literate_params
scan_literate_params.ipynb
[evkuzmin@fedora scripts]$ ls markdown/scan_literate_params
scan_literate_params.qmd
[evkuzmin@fedora scripts]$ julia --project

               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.12.5 (2026-02-09)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org release
|__/                   |

julia> using Pkg

julia> using IJulia

julia> notebook()
notebook = `/usr/bin/jupyter notebook`
[ Info: running `/usr/bin/jupyter notebook`
```

Рисунок 4.12: Создание параметрической версии второго скрипта и генерация файлов

На рисунке показан этап подготовки параметризованной версии второго скрипта.

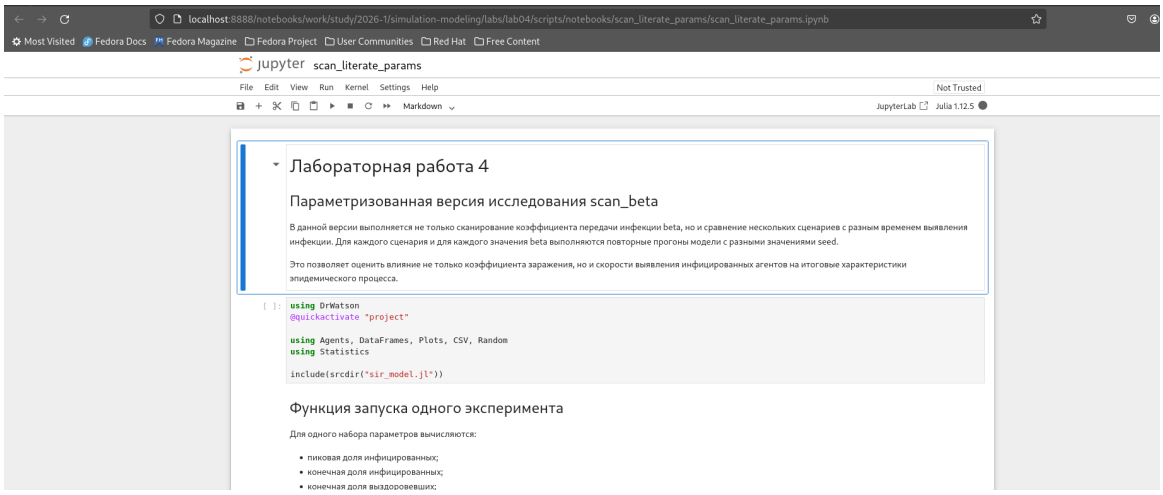


Рисунок 4.13: Notebook параметризованной версии второго скрипта

На рисунке представлен notebook параметризованной версии второго скрипта.

4.5.1 Результаты параметризованного исследования

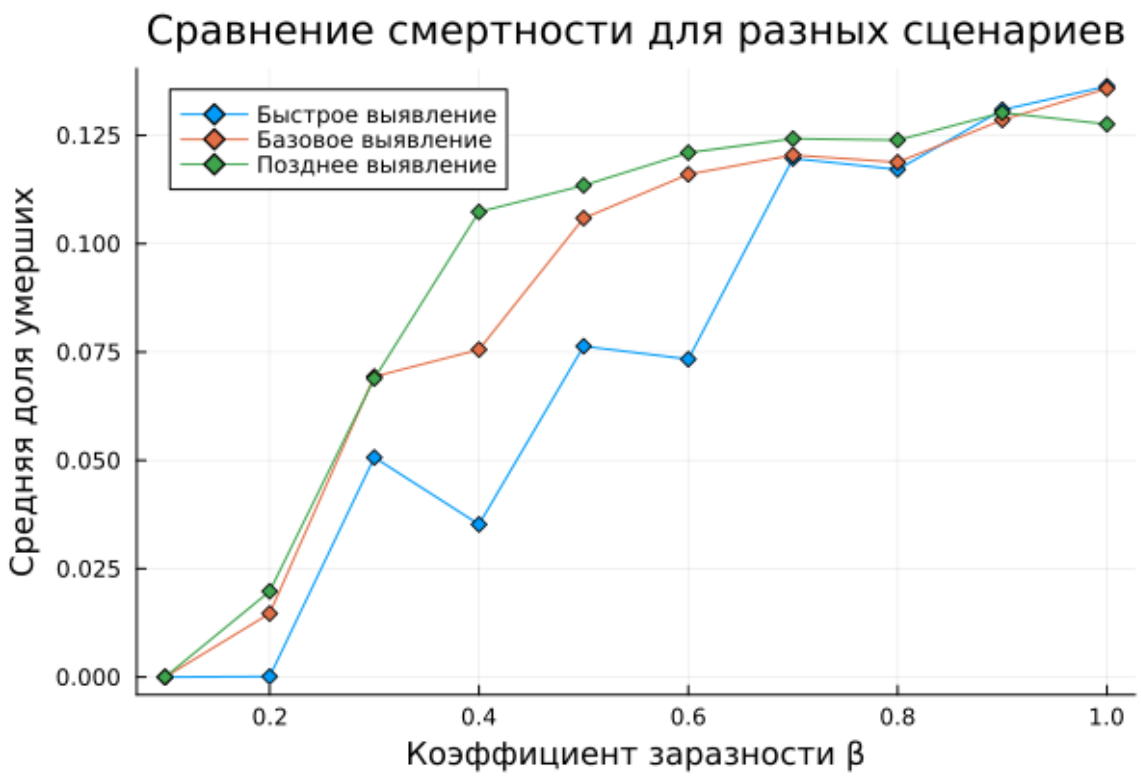


Рисунок 4.14: Сценарное исследование смертности

На рисунке показана зависимость средней доли умерших от коэффициента заражения для нескольких сценариев. Видно, что различия между сценариями усиливаются при росте заразности.

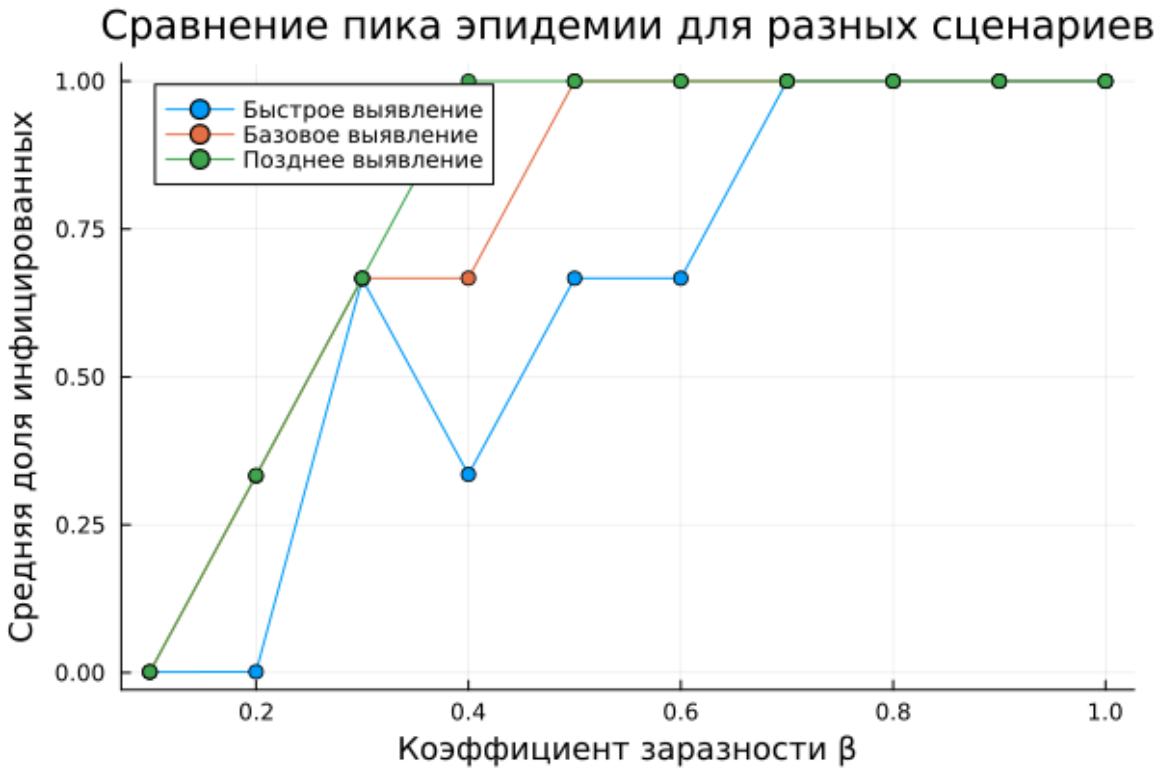


Рисунок 4.15: Сценарное исследование пика эпидемии

На рисунке сравниваются значения средней пиковой доли инфицированных в различных сценариях. Это позволяет оценить влияние дополнительных условий моделирования на развитие эпидемии.

Листинг 4.6. Параметризованная версия скрипта параметрического исследования scan_literate_params.jl

```
# # [redacted] 4
# ## [redacted] scan_beta
#
# [redacted]
```

```
# [illegible] beta, [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] beta
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] seed.
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible], [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible].

using DrWatson
@quickactivate "project"

using Agents, DataFrames, Plots, CSV, Random
using Statistics

include(srcdir("sir_model.jl"))

# ## [illegible] [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible]:
#
# - [illegible] [illegible] [illegible];
# - [illegible] [illegible] [illegible];
# - [illegible] [illegible] [illegible];
# - [illegible] [illegible].

function run_experiment(p)
beta = p[:beta]
[illegible]_und = fill(beta, 3)
```

```
β_det = fill(beta / 10, 3)

model = initialize_sir(;
Ns = p[:Ns],
β_und = β_und,
β_det = β_det,
infection_period = p[:infection_period],
detection_time = p[:detection_time],
death_rate = p[:death_rate],
reinfection_probability = p[:reinfection_probability],
Is = p[:Is],
seed = p[:seed],
n_steps = p[:n_steps],
)

infected_fraction(model) =
count(a.status == :I for a in allagents(model)) / nagents(model)

peak_infected = 0.0

for step = 1:p[:n_steps]
agent_ids = collect(allids(model))
for id in agent_ids
agent = try
model[id]
catch
nothing
end
```

```
if agent !== nothing
sir_agent_step!(agent, model)
end
end

frac = infected_fraction(model)
if frac > peak_infected
peak_infected = frac
end
end

final_infected = infected_fraction(model)
final_recovered = count(a.status == :R for a in allagents(model)) /
total_deaths = sum(p[:Ns]) - nagents(model)

return (
peak = peak_infected,
final_inf = final_infected,
final_rec = final_recovered,
deaths = total_deaths,
)
end

# ## □□□□□ □□□□□□□□□
#
# □□□□□□□□□□□□□□□ □□□ □□□□□□□□:
#
# - □□□□□□□ □□□□□□□□□;
```

```
# - □□□□□□□ □□□□□□□□□;
# - □□□□□□□ □□□□□□□□□.

scenario_sets = [
(name = "□□□□□□□ □□□□□□□□□", detection_time = 5),
(name = "□□□□□□□ □□□□□□□□□", detection_time = 7),
(name = "□□□□□□□ □□□□□□□□□", detection_time = 10),
]

# ## □□□□□□□□ □□□□□□□□ beta □ □□□□□ □□□□□□□□ seed

beta_range = 0.1:0.1:1.0
seeds = [42, 43, 44]

# ## □□□□□□□□□□□□ □□□□□□ □□□□□□□□□□

params_list = []

for scenario in scenario_sets
for b in beta_range
for s in seeds
push!(
params_list,
Dict(
:scenario => scenario.name,
:beta => b,
:Ns => [1000, 1000, 1000],
:infection_period => 14,
```

```
:detection_time => scenario.detection_time,
:death_rate => 0.02,
:reinfection_probability => 0.1,
:Is => [0, 0, 1],
:seed => s,
:n_steps => 100,
),
)
end


# ## □□□□□□ □□□□ □□□□□□□□□□□□□

results = []

for params in params_list
data = run_experiment(params)
push!(results, merge(params, Dict(pairs(data))))
println("□□□□□□□□ = $(params[:scenario]), beta = $(params[:beta]), se
end

# ## □□□□□□□□□□ □□□□ □□□□□□□□

df = DataFrame(results)
CSV.write(datadir("beta_scan_scenarios_all.csv"), df)

# ## □□□□□□□□□□ □□□□□□□□□□□
#
```

```
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] beta.

grouped = combine(
groupby(df, [:scenario, :beta]),
:peak => mean => :mean_peak,
:final_inf => mean => :mean_final_inf,
:final_rec => mean => :mean_final_rec,
:deaths => mean => :mean_deaths,
)

CSV.write(datadir("beta_scan_scenarios_grouped.csv"), grouped)

# ## [illegible] [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

plot(
xlabel = "[illegible] [illegible] [illegible]",
ylabel = "[illegible] [illegible] [illegible]",
title = "[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]",
linewidth = 2,
)

for scenario in scenario_sets
subset_df = grouped[grouped.scenario .== scenario.name, :]
plot!(
subset_df.beta,
subset_df.mean_peak,
```

```
label = scenario.name,
marker = :circle,
)
end

savefig(plotsdir("beta_scan_scenarios_peak.png"))

# ## [illegible]
#
# [illegible] beta.

plot(
xlabel = "[illegible]",
ylabel = "[illegible]",
title = "[illegible]",
linewidth = 2,
)

for scenario in scenario_sets
subset_df = grouped[grouped.scenario .== scenario.name, :]
plot!(
subset_df.beta,
subset_df.mean_deaths ./ 3000,
label = scenario.name,
marker = :diamond,
)
end
```

```
savefig(plotsdir("beta_scan_scenarios_deaths.png"))

println("□□□□□□□□□□ □□□□□□□□□ □ data/beta_scan_scenarios_all.csv")
println("□□□□□□□□□□□ □□□□□□□□□□ □□□□□□□□□ □ data/beta_scan_scenarios_group
println("□□□□□□□ □□□□□□□□□□ □ plots/beta_scan_scenarios_peak.png □ plots

# ## □□□□□
#
# □ □□□□□□□□□□ □□□□ □□□□□□□□ □ □□□□□□□□□ □□□□□□ □□□ □□□□□□□□□□ □□□□□□□□□
# □ □□□□□□ □□□□□□□□ □□□□□□□□□ □□□□□□□□. □□□ □□□□□□□□□ □□□□□□□□ □□□□□□□
# □□□□□□□□□ detection_time □□ □□□ □□□□□□□□ □ □□□□□□□□□□ □□□ □□□□□□□□□
# □□□□□□□□□ □□□□□□□□□□□□ □□□□□□□□□ beta.
```

4.6 Скрипт исследования миграции

Третий скрипт посвящён исследованию влияния интенсивности миграции на распространение инфекции между городами.

4.6.1 Обычная версия

Листинг 4.7. Обычная версия скрипта исследования миграции sir_migration_effect.jl

```
using DrWatson
@quickactivate "project"

using Agents, DataFrames, Plots, CSV, Random
using Statistics
```

```
include(srcdir("sir_model.jl"))

# □□□□□□□ □□□ □□□□□□□□ □□□□□□□ □□□□□□□□
function create_migration_matrix(C, intensity)
M = ones(C, C) .* intensity / (C - 1)
for i in 1:C
M[i, i] = 1 - intensity
end
return M
end

# □□□□□□□ □□□ □□□□□□□□ □□□□□□□ □□□□□□□□□□ □□□□
function peak_time(p)
migration_rates = create_migration_matrix(p[:C], p[:migration_intens

model = initialize_sir(;
Ns = p[:Ns],
□_und = p[:□_und],
□_det = p[:□_det],
infection_period = p[:infection_period],
detection_time = p[:detection_time],
death_rate = p[:death_rate],
reinfection_probability = p[:reinfection_probability],
Is = p[:Is],
seed = p[:seed],
migration_rates = migration_rates,
)
```

```
infected_frac(model) =
count(a.status == :I for a in allagents(model)) / nagents(model)

peak = 0.0
peak_step = 0

for step in 1:p[:n_steps]
agent_ids = collect(allids(model))
for id in agent_ids
agent = try
model[id]
catch
nothing
end

if agent !== nothing
sir_agent_step!(agent, model)
end
end

frac = infected_frac(model)
if frac > peak
peak = frac
peak_step = step
end
end

return (peak_time = peak_step, peak_value = peak)
```

```
end

# □□□□□□□□ □□□□□□□□□□□□□ □□□□□□□□
migration_intensities = 0.0:0.1:0.5
seeds = [42, 43, 44]

# □□□□□□□□□□□□□ □□□□□□□□□□□
params_list = []

for mig in migration_intensities
for s in seeds
push!(
params_list,
Dict(
:migration_intensity => mig,
:C => 3,
:Ns => [1000, 1000, 1000],
:□_und => [0.5, 0.5, 0.5],
:□_det => [0.05, 0.05, 0.05],
:infection_period => 14,
:detection_time => 7,
:death_rate => 0.02,
:reinfection_probability => 0.1,
:Is => [1, 0, 0],
:seed => s,
:n_steps => 150,
),
)
```

```
end
end

# □□□□□□ □□□□□□□□□□□□□
results = []

for params in params_list
data = peak_time(params)
push!(results, merge(params, Dict(pairs(data))))
println("□□□□□□□□ □□□□□□□□□□□ □ migration_intensity = $(params[:migratio
end

# □□□□□□□□□□ □□□□ □□□□□□□□
df = DataFrame(results)
CSV.write(datadir("migration_scan_all.csv"), df)

# □□□□□□□□□□
grouped = combine(
groupby(df, [:migration_intensity]),
:peak_time => mean => :mean_peak_time,
:peak_value => mean => :mean_peak_value,
)

# □□□□□□□□□□□□
plot(
grouped.migration_intensity,
grouped.mean_peak_time,
marker = :circle,
```

```
xlabel = "□□□□□□□□□□□□□ □□□□□□□□",
ylabel = "□□□□□ □□ □□□□ (□□□)",
label = "□□□□□ □□□□",
)

plot!(
grouped.migration_intensity,
grouped.mean_peak_value .* 3000,
marker = :square,
label = "□□□□□□□ □□□□□□□□□□□□□□□",
)

savefig(plotsdir("migration_effect.png"))

println("□□□□□□□□□□□ □□□□□□□□□□ □ data/migration_scan_all.csv □ plots/migr
```

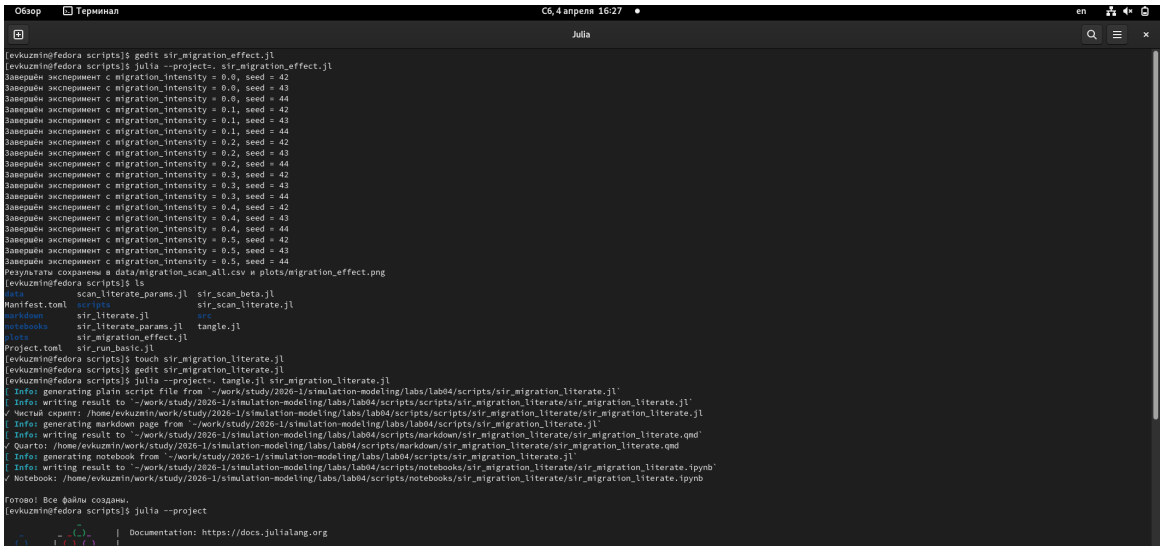


Рисунок 4.16: Создание и выполнение третьего скрипта, подготовка literate-версии

На рисунке показан запуск третьего скрипта и подготовка его literate-версии.

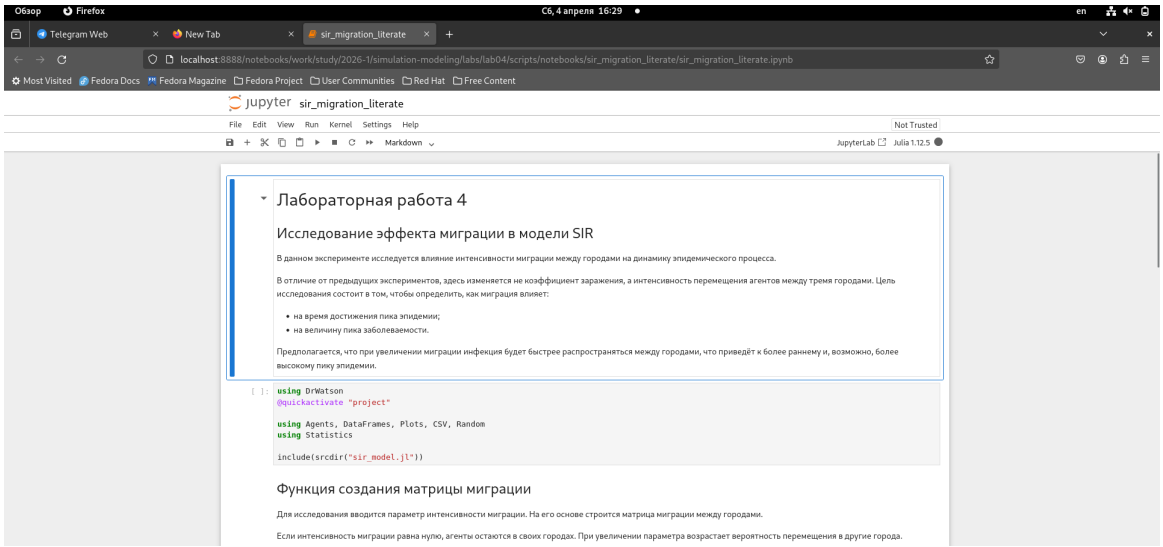


Рисунок 4.17: Notebook literate-версии третьего скрипта

На рисунке представлен notebook literate-версии третьего скрипта.

4.6.2 Результат исследования миграции

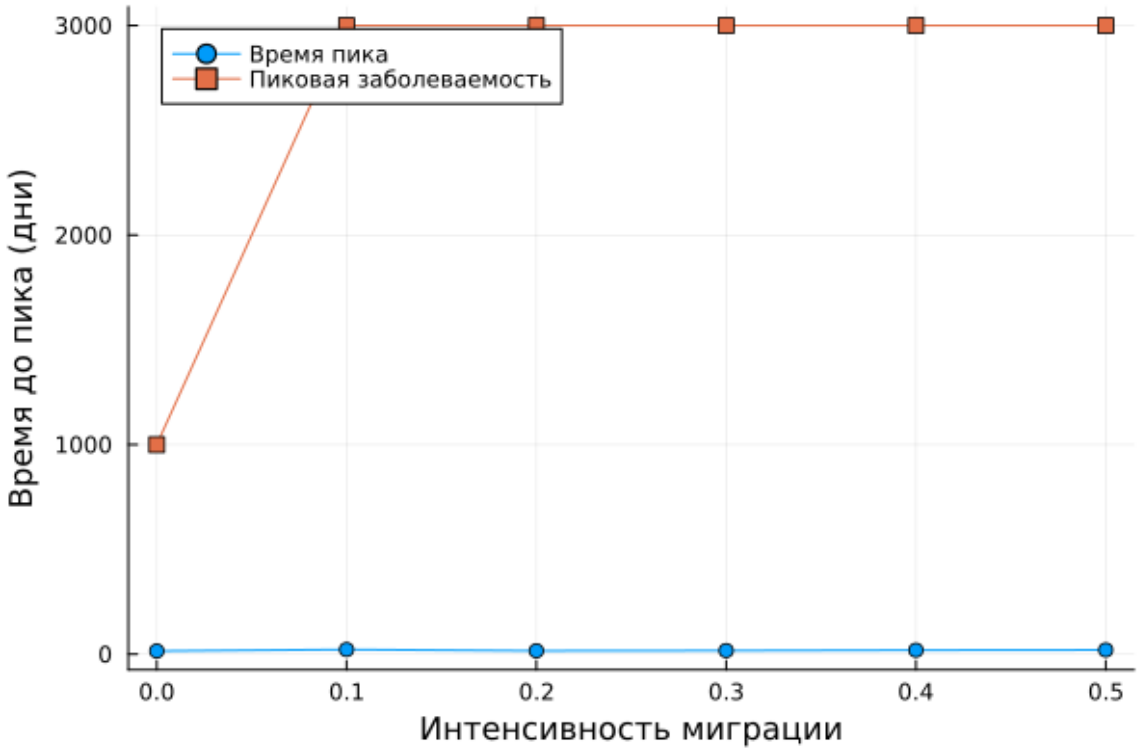


Рисунок 4.18: Влияние интенсивности миграции на динамику эпидемии

На рисунке показано влияние миграции на время достижения пика и на величину пика. При увеличении миграции инфекция распространяется между городами быстрее, что приводит к ускорению эпидемического процесса.

4.6.3 Literate-версия

Листинг 4.8. Literate-версия скрипта исследования миграции sir_migration_literate.jl

```
# # □□□□□□□□□□□□ □□□□□□ 4
# ## □□□□□□□□□□□□ □□□□□□□ □□□□□□□□ □ □□□□□□ SIR
#
```

```
# [illegible]
# [illegible].
#
# [illegible], [illegible] [illegible], [illegible] [illegible]
# [illegible], [illegible] [illegible] [illegible] [illegible] [illegible].
# [illegible] [illegible] [illegible] [illegible], [illegible] [illegible], [illegible] [illegible] [illegible]:
#
# - [illegible] [illegible] [illegible] [illegible];
# - [illegible] [illegible] [illegible] [illegible].
#
# [illegible], [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible], [illegible] [illegible] [illegible] [illegible],
# [illegible], [illegible] [illegible] [illegible] [illegible].

using DrWatson
@quickactivate "project"

using Agents, DataFrames, Plots, CSV, Random
using Statistics

include(srcdir("sir_model.jl"))

# ## [illegible] [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].
#
# [illegible] [illegible] [illegible] [illegible] [illegible], [illegible] [illegible] [illegible] [illegible].
```

```
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

function create_migration_matrix(C, intensity)
M = ones(C, C) .* intensity / (C - 1)
for i in 1:C
M[i, i] = 1 - intensity
end
return M
end

# ## [illegible] [illegible] [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible], [illegible] [illegible]
# [illegible]:
#
# - [illegible] [illegible], [illegible] [illegible] [illegible] [illegible] [illegible];
# - [illegible] [illegible] [illegible].

function peak_time(p)
migration_rates = create_migration_matrix(p[:C], p[:migration_intens

model = initialize_sir(;
Ns = p[:Ns],
[illegible]_und = p[:[illegible]_und],
[illegible]_det = p[:[illegible]_det],
infection_period = p[:infection_period],
detection_time = p[:detection_time],
death_rate = p[:death_rate],
```

```
reinfection_probability = p[:reinfection_probability],
Is = p[:Is],
seed = p[:seed],
migration_rates = migration_rates,
)

infected_frac(model) =
count(a.status == :I for a in allagents(model)) / nagents(model)

peak = 0.0
peak_step = 0

for step in 1:p[:n_steps]
agent_ids = collect(allids(model))
for id in agent_ids
agent = try
model[id]
catch
nothing
end

if agent !== nothing
sir_agent_step!(agent, model)
end
end

frac = infected_frac(model)
if frac > peak
```

```
peak = frac
peak_step = step
end
end

return (peak_time = peak_step, peak_value = peak)
end

# ## [illegible] [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] 0.0 [illegible] 0.5
# [illegible] 0.1. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible] [illegible].

migration_intensities = 0.0:0.1:0.5
seeds = [42, 43, 44]

# ## [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] seed
# [illegible] [illegible] [illegible] [illegible].

params_list = []

for mig in migration_intensities
for s in seeds
push!(
params_list,
```

```
Dict(
:migration_intensity => mig,
:C => 3,
:Ns => [1000, 1000, 1000],
:β_und => [0.5, 0.5, 0.5],
:β_det => [0.05, 0.05, 0.05],
:infection_period => 14,
:detection_time => 7,
:death_rate => 0.02,
:reinfection_probability => 0.1,
:Is => [1, 0, 0],
:seed => s,
:n_steps => 150,
),
)
end
end

# ## □□□□□□ □□□□□□□□□□□□□
#
# □□□ □□□□□□□ □□□□□□ □□□□□□□□□□ □□□□□□□□□□□ □□□□□□□□□ □□□□□□□□□□□□□,
# □ □□□□□□□□□□ □□□□□□□□□□□ □ □□□□□ □□□□□□.

results = []

for params in params_list
data = peak_time(params)
push!(results, merge(params, Dict(pairs(data))))
```

```
println("□□□□□□□□ □□□□□□□□□□□ □ migration_intensity = $(params[:migratio
end

# ## □□□□□□□□□□ □□□□□□□□□□□
#
# □□□□□□□□□□ □□□□ □□□□□□□□ □□□□□□□□□□□ □ CSV-□□□□ □□□ □□□□□□□□□□□□ □□□□□□□.

df = DataFrame(results)
CSV.write(datadir("migration_scan_all.csv"), df)

# ## □□□□□□□□□□ □□□□□□□□□□□
#
# □□□ □□□□□□□□ □□□□□□□□ □□□□□□□□□□□□□ □□□□□□□□ □□□□□□□□□□□ □□□□□□□ □□□□□□□□:
#
# - □□□□□□□ □□□□□□□□□□ □□□□;
# - □□□□□□□□ □□□□ □□□□□□□□□□□□□□.

grouped = combine(
groupby(df, [:migration_intensity]),
:peak_time => mean => :mean_peak_time,
:peak_value => mean => :mean_peak_value,
)

# ## □□□□□□□□□□□□ □□□□□□□□□□□
#
# □□□□□□□□□ □□□□□□ □□□□□□□□□□□ □□□□□□□□ □□ □□□□ □ □□□□□□□□□ □□□□
# □□ □□□□□□□□□□□□□ □□□□□□□□□.
```

```
plot(
grouped.migration_intensity,
grouped.mean_peak_time,
marker = :circle,
xlabel = "□□□□□□□□□□□□□ □□□□□□□□",
ylabel = "□□□□□□□□ □□□□□□□□□□",
label = "□□□□□ □□□□",
)

plot!(
grouped.migration_intensity,
grouped.mean_peak_value .* 3000,
marker = :square,
label = "□□□□□□□ □□□□□□□□□□□□□□",
)

savefig(plotsdir("migration_effect.png"))

println("□□□□□□□□□□ □□□□□□□□□ □ data/migration_scan_all.csv □ plots/migr

# ## □□□□□
#
# □ □□□□□□□□□□ □□□□□□□□□□□□□ □□□□ □□□□□□□□□□ □□□□□□□□□□ □□□□□□□
# □□□□□□□□□□ □□□□ □□□□□□□□ □ □□□□□□□□ □□□□ □□□□□□□□□□□□□□ □□ □□□□□□□□□□□□□
# □□□□□□□□ □□□□□ □□□□□□□□.
#
# □□□□□□□□□□ □□□□□□□□□□□□□ □□□□□□□□ □□□□□□□□ □ □□□□□ □□□□□□□□ □□□□□□□□□□□□□□□
# □□□□□□□□ □□□□□ □□□□□□□□, □□□, □□□ □□□□□□□, □□□□□□□□□ □□□□□ □□ □□□□□□□□□□
```

```
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].
```

4.7 Параметризованная версия скрипта миграции

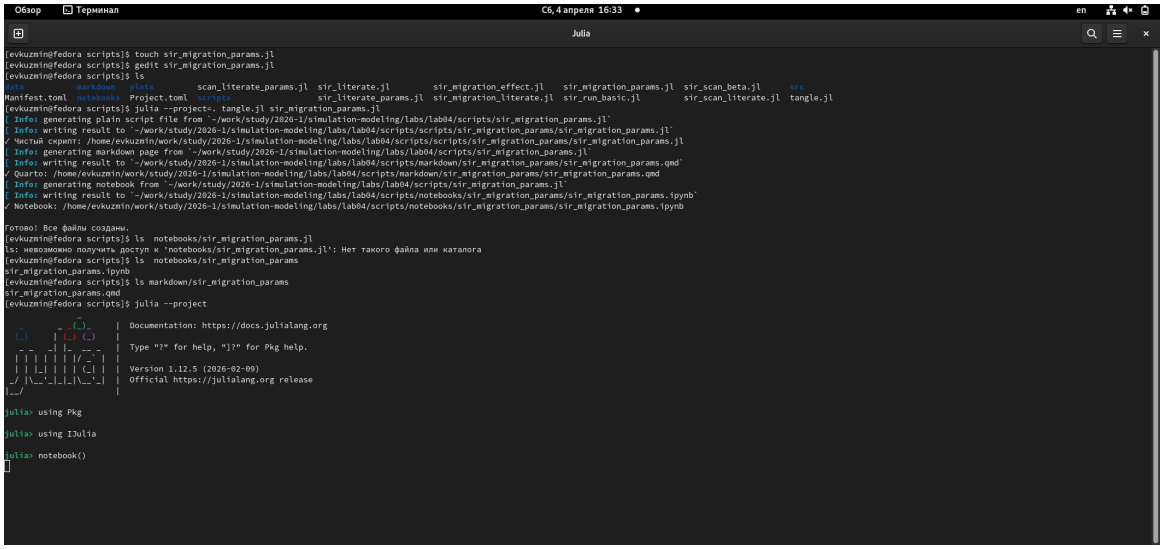


Рисунок 4.19: Создание параметрической версии третьего скрипта

На рисунке показан этап создания параметрической версии третьего скрипта.

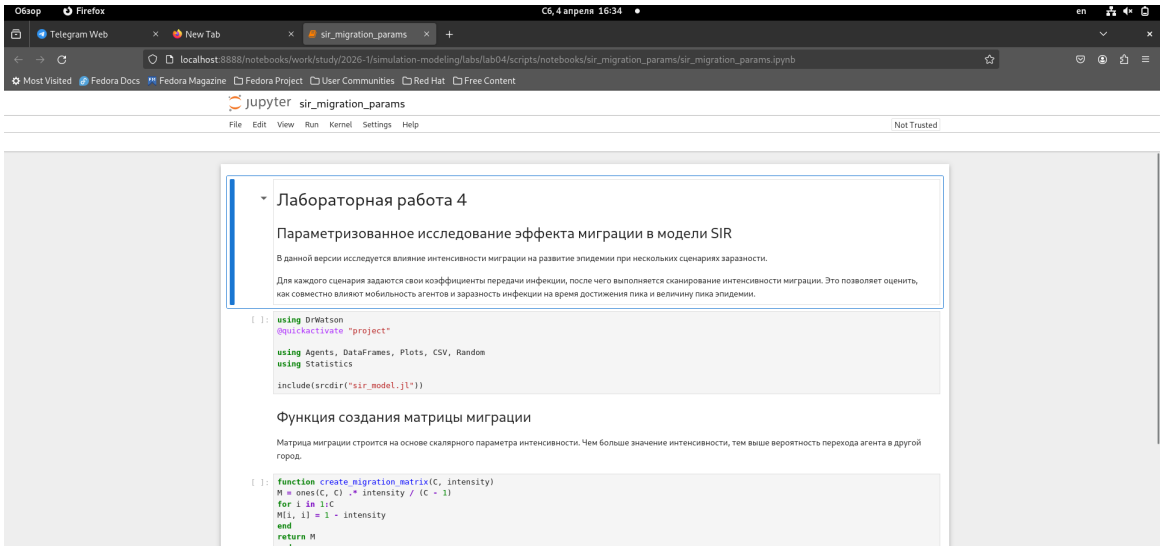


Рисунок 4.20: Notebook параметрической версии третьего скрипта

На рисунке показан notebook параметрической версии третьего скрипта.

4.7.1 Результаты параметризованного исследования миграции

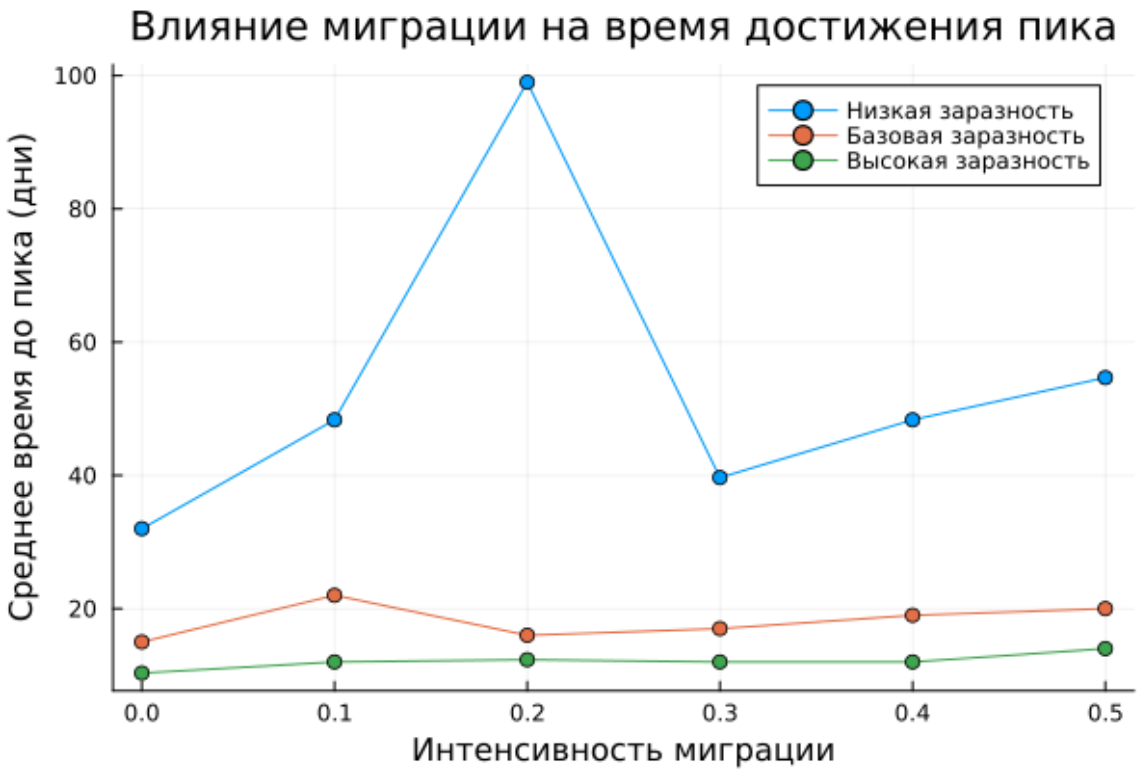


Рисунок 4.21: Сценарное исследование времени до пика при миграции

На рисунке представлена зависимость времени достижения пика от интенсивности миграции в разных сценариях. Это позволяет сравнить скорость распространения инфекции при различных условиях.

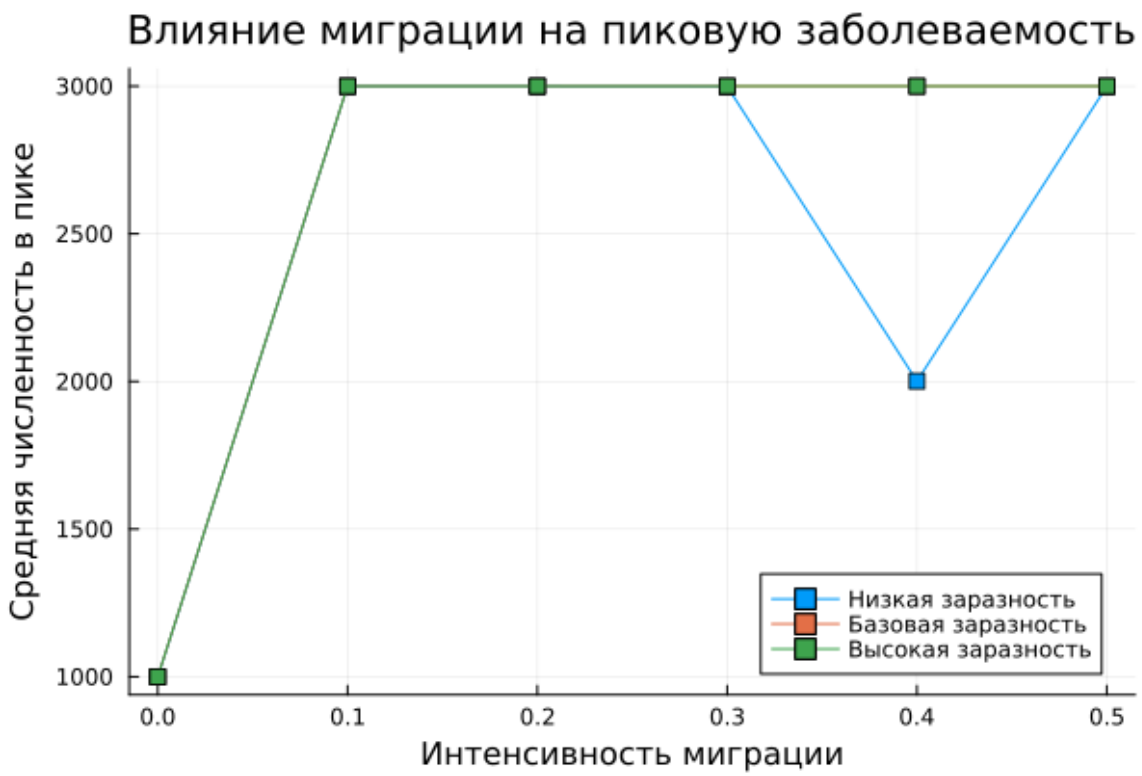


Рисунок 4.22: Сценарное исследование пикового уровня инфицирования при миграции

На рисунке показано, как в разных сценариях меняется величина пика эпидемии. Рост миграции, как правило, способствует более быстрому перемешиванию популяции и усилению распространения инфекции.

Листинг 4.9. Параметризованная версия скрипта исследования миграции sir_migration_params.jl

```
# # [illegible] 4
# ## [illegible] [illegible] [illegible] [illegible] [illegible] SIR
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible] [illegible].
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible],
```

```
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible]
# [illegible], [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

using DrWatson
@quickactivate "project"

using Agents, DataFrames, Plots, CSV, Random
using Statistics

include(srcdir("sir_model.jl"))

# ## [illegible] [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].
# [illegible] [illegible] [illegible] [illegible], [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible].

function create_migration_matrix(C, intensity)
M = ones(C, C) .* intensity / (C - 1)
for i in 1:C
M[i, i] = 1 - intensity
end
return M
end

# ## [illegible] [illegible] [illegible] [illegible]
#
```

```
# □□□ □□□□□□□□□ □□□□□□ □□□□□□□□□□ □□□□□□□□□□□:
#
# - □□□□□ □□□□□□□□□□ □□□□ □□□□□□□□;
# - □□□□□□□□ □□□□ □□□□□□□□.

function peak_time(p)
migration_rates = create_migration_matrix(p[:C], p[:migration_intens

model = initialize_sir(;
Ns = p[:Ns],
□_und = p[:□_und],
□_det = p[:□_det],
infection_period = p[:infection_period],
detection_time = p[:detection_time],
death_rate = p[:death_rate],
reinfection_probability = p[:reinfection_probability],
Is = p[:Is],
seed = p[:seed],
migration_rates = migration_rates,
)

infected_frac(model) =
count(a.status == :I for a in allagents(model)) / nagents(model)

peak = 0.0
peak_step = 0

for step in 1:p[:n_steps]
```

```
agent_ids = collect(allids(model))
for id in agent_ids
agent = try
model[id]
catch
nothing
end

if agent !== nothing
sir_agent_step!(agent, model)
end
end

frac = infected_frac(model)
if frac > peak
peak = frac
peak_step = step
end
end

return (peak_time = peak_step, peak_value = peak)
end

# ## □□□□□ □□□□□□□□□
#
# □□□□□□□□□□□□□□□ □□□ □□□□□□□□:
#
# - □□□□□□ □□□□□□□□□□;
```

```
# - [illegible];
# - [illegible].
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] 10 [illegible], [illegible] [illegible].

scenario_sets = [
(name = "[illegible]", [illegible]_und = [0.3, 0.3, 0.3], [illegible]_det = [0.03, 0.
(name = "[illegible]", [illegible]_und = [0.5, 0.5, 0.5], [illegible]_det = [0.05, 0
(name = "[illegible]", [illegible]_und = [0.7, 0.7, 0.7], [illegible]_det = [0.07, 0
]

# ## [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] 0.0 [illegible] 0.5 [illegible] [illegible] 0.1.

migration_intensities = 0.0:0.1:0.5
seeds = [42, 43, 44]

# ## [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible], [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] seed
# [illegible] [illegible] [illegible] [illegible].

params_list = []

for scenario in scenario_sets
```

```
for mig in migration_intensities
for s in seeds
push!(
params_list,
Dict(
:scenario => scenario.name,
:migration_intensity => mig,
:C => 3,
:Ns => [1000, 1000, 1000],
:□_und => scenario.□_und,
:□_det => scenario.□_det,
:infection_period => 14,
:detection_time => 7,
:death_rate => 0.02,
:reinfection_probability => 0.1,
:Is => [1, 0, 0],
:seed => s,
:n_steps => 150,
),
)
end
end
end

# ## □□□□□□ □□□□□□□□□□□□□
#
# □□□ □□□□ □□□□□□□ □□□□□□□□□□ □□□□□□□□□□□ □□□□□□□□□□□□□.
```

```
results = []

for params in params_list
data = peak_time(params)
push!(results, merge(params, Dict(pairs(data))))
println("□□□□□□□□ = $(params[:scenario]), migration = $(params[:migra
end

# ## □□□□□□□□□□ □□□□□□□□□□□
#
# □□□□□□□□ □□□□□□ □□□□□□□□□□ □□□□ □□□□□□□□.

df = DataFrame(results)
CSV.write(datadir("migration_scenarios_all.csv"), df)

# ## □□□□□□□□□□ □□□□□□□□□□□
#
# □□□□□□□□ □□□□□□□□□□ □□ □□□□□□□□□□ □□□□□□□□□ □□□□□□□□ □ □□□□□□□□□□□□□ □□□□□□□□.

grouped = combine(
groupby(df, [:scenario, :migration_intensity]),
:peak_time => mean => :mean_peak_time,
:peak_value => mean => :mean_peak_value,
)

CSV.write(datadir("migration_scenarios_grouped.csv"), grouped)

# ## □□□□□□□□□□ □□□□□□□□ □□□□□□□□ □□ □□□□
```

```
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible].

plot(
xlabel = "[illegible] [illegible]",
ylabel = "[illegible] [illegible] [illegible] [illegible] ([illegible])",
title = "[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]",
linewidth = 2,
)

for scenario in scenario_sets
subset_df = grouped[grouped.scenario .== scenario.name, :]
plot!(
subset_df.migration_intensity,
subset_df.mean_peak_time,
label = scenario.name,
marker = :circle,
)
end

savefig(plotsdir("migration_scenarios_peak_time.png"))

# ## [illegible] [illegible] [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

plot(
```

```
xlabel = "□□□□□□□□□□□□□ □□□□□□□□",
ylabel = "□□□□□□□ □□□□□□□□□□□ □ □□□□",
title = "□□□□□□□ □□□□□□□□ □□ □□□□□□□ □□□□□□□□□□□□□□",
linewidth = 2,
)

for scenario in scenario_sets
subset_df = grouped[grouped.scenario .== scenario.name, :]
plot!(
subset_df.migration_intensity,
subset_df.mean_peak_value .* 3000,
label = scenario.name,
marker = :square,
)
end

savefig(plotsdir("migration_scenarios_peak_value.png"))

println("□□□□□□□□□□ □□□□□□□□□ □ data/migration_scenarios_all.csv")
println("□□□□□□□□□□□ □□□□□□□□□□ □□□□□□□□□ □ data/migration_scenarios_group
println("□□□□□□□ □□□□□□□□□□ □ plots/migration_scenarios_peak_time.png □ p

# ## □□□□□
#
# □ □□□□□□□□□□ □□□□ □□□□□□□□□□□ □□□□□□□□□ □□□□□□□□□ □ □□□□□□□□□ □□□□□□□□□□□
# □□□□□□□□□ □□□ □□□□□□ □□□□□□□□□ □□□□□□□□□□□□□ □□□□□□□□. □□□ □□□□□□□□□ □□□□□□□,
# □□□ □□□□□□□□□□ □□□□□□□□□ □□□□□□□□□□□ □□□□□□□ □ □□□□□□□□□□□□ □□□□□□□□ □□□□□□□□
# □□□□□□□ □□ □□□□□□□□ □□□□□□□□ □□□□□□□□ □ □□□□□□□□ □□ □□□□.
```

4.8 Скрипт оптимизации параметров модели

Четвёртый скрипт решает задачу оптимизации параметров модели SIR.

4.8.1 Обычная версия

Листинг 4.10. Обычная версия скрипта оптимизации sir_optimize_parameters.jl

```
using DrWatson
@quickactivate "project"

using BlackBoxOptim, Random, Statistics
using Agents
using JLD2

include(srcdir("sir_model.jl"))

function cost_fast(x)
infected_frac(model) =
count(a.status == :I for a in allagents(model)) / nagents(model)

dead_count(model) = 1500 - nagents(model)

peak_vals = Float64[]
dead_vals = Float64[]

for rep in 1:2
```

```
model = initialize_sir(;
Ns = [500, 500, 500],
β_und = fill(x[1], 3),
β_det = fill(x[1] / 10, 3),
infection_period = 10,
detection_time = round(Int, x[2]),
death_rate = x[3],
reinfection_probability = 0.1,
Is = [0, 0, 1],
seed = 42 + rep,
n_steps = 50,
)

peak_infected = 0.0

for step in 1:50
Agents.step!(model, 1)
frac = infected_frac(model)
if frac > peak_infected
peak_infected = frac
end
end

push!(peak_vals, peak_infected)
push!(dead_vals, dead_count(model) / 1500)
end

mean_peak = mean(peak_vals)
```

```
mean_dead = mean(dead_vals)

return mean_peak + mean_dead
end

result = bboptimize(
cost_fast;
Method = :random_search,
SearchRange = [
(0.1, 1.0),
(3.0, 14.0),
(0.01, 0.1),
],
NumDimensions = 3,
MaxTime = 20,
TraceMode = :silent,
)

best = best_candidate(result)
fitness = best_fitness(result)

println("□□□□□□□□□□□ □□□□□□□□□:")
println("□_und = $(best[1])")
println("□□□□□ □□□□□□□□□ = $(round(Int, best[2])) □□□□")
println("□□□□□□□□□□ = $(best[3])")

println("□□□□□□□□ □□□□□□□ □□□□□□□: $(fitness)")
```

```
save(
datadir("optimization_fast_result.jld2"),
Dict(
"best" => best,
"fitness" => fitness,
),
)
```

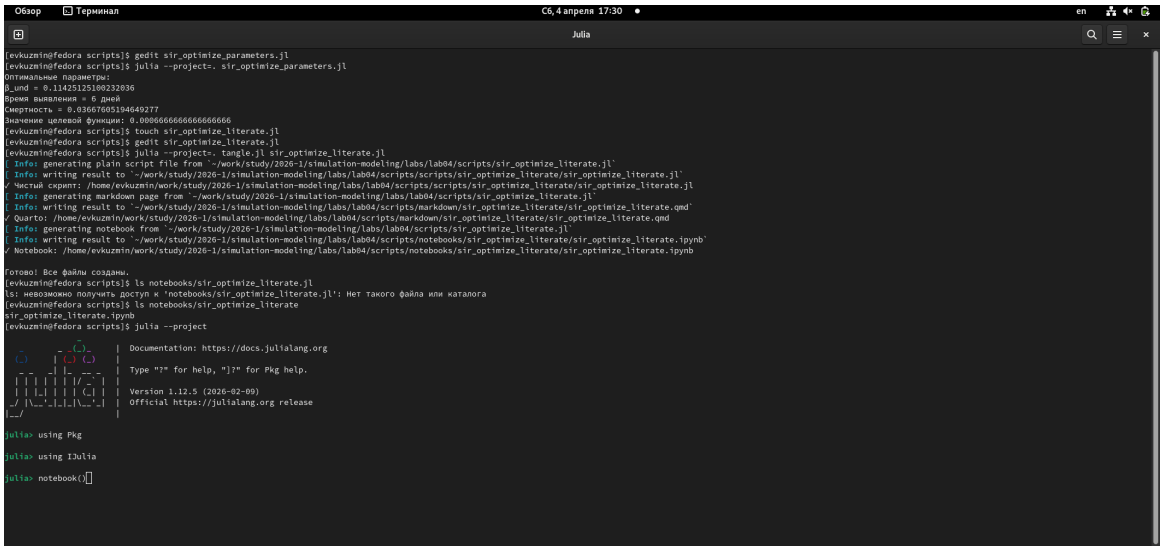


Рисунок 4.23: Создание и выполнение четвёртого скрипта, подготовка literate-версии

На рисунке показан запуск четвёртого скрипта и создание его literate-версии.

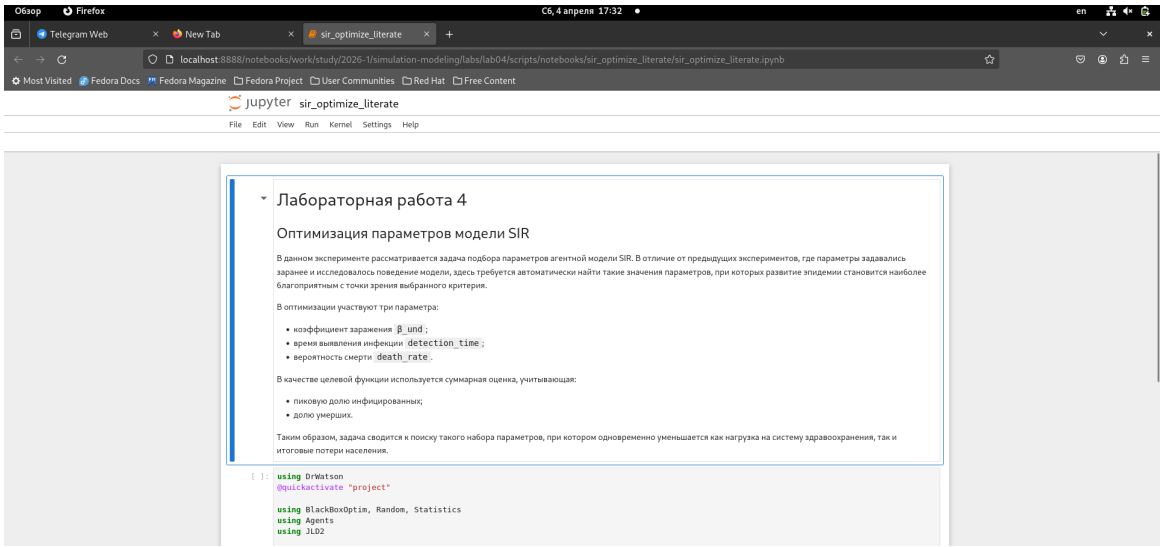


Рисунок 4.24: Notebook literate-версии четвёртого скрипта

На рисунке представлен notebook literate-версии четвёртого скрипта.

4.8.2 Literate-версия

Листинг 4.11. Literate-версия скрипта оптимизации sir_optimize_literate.jl

```
# # [redacted] 4
# ## [redacted] [redacted] [redacted] SIR
#
# [redacted] [redacted] [redacted] [redacted] [redacted] [redacted]
# [redacted] [redacted] SIR. [redacted] [redacted] [redacted] [redacted], [redacted]
# [redacted] [redacted] [redacted] [redacted] [redacted] [redacted],
# [redacted] [redacted] [redacted] [redacted] [redacted] [redacted],
# [redacted] [redacted] [redacted] [redacted] [redacted] [redacted]
# [redacted] [redacted] [redacted] [redacted].
#
# [redacted] [redacted] [redacted] [redacted]:
```

```
#
# - [illegible] [illegible] `[illegible]_und`;
# - [illegible] [illegible] [illegible] `detection_time`;
# - [illegible] [illegible] `death_rate`.
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible],
# [illegible]:
#
# - [illegible] [illegible] [illegible];
# - [illegible] [illegible].
#
# [illegible] [illegible], [illegible] [illegible] [illegible] [illegible] [illegible] [illegible],
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible], [illegible] [illegible] [illegible] [illegible] [illegible].

using DrWatson
@quickactivate "project"

using BlackBoxOptim, Random, Statistics
using Agents
using JLD2

include(srcdir("sir_model.jl"))

# ## [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] `x`, [illegible]:
#
```

```
# - `x[1]` — □□□□□□□□□□□ □□□□□□□□□ `□_und`;
# - `x[2]` — □□□□□ □□□□□□□□□ □□□□□□□□;
# - `x[3]` — □□□□□□□□□□□ □□□□□□.
#
# □□□ □□□□□□□□□□ □□□□□□□ □□□□□□□□□□□ □□□□□□□□□ □□□□□□□□□□□
# □□ □□□□□□□□□□ □□□□□□□□ □□□□□□.

function cost_fast(x)
infected_frac(model) =
count(a.status == :I for a in allagents(model)) / nagents(model)

dead_count(model) = 1500 - nagents(model)

peak_vals = Float64[]
dead_vals = Float64[]

for rep in 1:2
model = initialize_sir(;
Ns = [500, 500, 500],
□_und = fill(x[1], 3),
□_det = fill(x[1] / 10, 3),
infection_period = 10,
detection_time = round(Int, x[2]),
death_rate = x[3],
reinfection_probability = 0.1,
Is = [0, 0, 1],
seed = 42 + rep,
n_steps = 50,
```

```
)

peak_infected = 0.0

for step in 1:50
Agents.step!(model, 1)
frac = infected_frac(model)
if frac > peak_infected
peak_infected = frac
end
end

push!(peak_vals, peak_infected)
push!(dead_vals, dead_count(model) / 1500)
end

mean_peak = mean(peak_vals)
mean_dead = mean(dead_vals)

return mean_peak + mean_dead
end

# ## □□□□□□ □□□□□□□□□□□
#
# □□□ □□□□□□ □□□□□□□□□□□ □□□□□□□□ □□□□□□□□□□□ □□□□□□□□ □□□□□□□□□□ □□□□□□.
# □□ □□□□□□□□ □□□□□□□ □□□□□ □□□□□□□ □□□□□□□□□□□□□□□□□□ □□□□□□□ □ □□□□□□□□
# □□□ □□□□□□□□□□□ □□□□□□□□□□□□□□□□ □□□□□□ □□□□□□.
```

```
result = bboptimize(
cost_fast;
Method = :random_search,
SearchRange = [
(0.1, 1.0),
(3.0, 14.0),
(0.01, 0.1),
],
NumDimensions = 3,
MaxTime = 20,
TraceMode = :silent,
)

# ## □□□□□□□□□□ □□□□□□□□□□□
#
# □□□□□ □□□□□□□□□□ □□□□□□□□□□□ □□□□□□□□□□□ □□□□□□□□□ □□□□□□□□□
# □ □□□□□□□□ □□□□□□□ □□□□□□□.

best = best_candidate(result)
fitness = best_fitness(result)

println("□□□□□□□□□□□ □□□□□□□□□:")
println("□_und = $(best[1])")
println("□□□□□ □□□□□□□□□ = $(round(Int, best[2])) □□□□")
println("□□□□□□□□□□ = $(best[3])")

println("□□□□□□□□ □□□□□□□ □□□□□□□: $(fitness)")
```

```
# ## [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible].

save(
datadir("optimization_fast_result.jld2"),
Dict(
"best" => best,
"fitness" => fitness,
),
)

# ## [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible],
# [illegible] [illegible] [illegible] [illegible].
# [illegible], [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible], [illegible] [illegible] [illegible] [illegible],
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible].
```

4.9 Параметризованная версия скрипта оптимизации

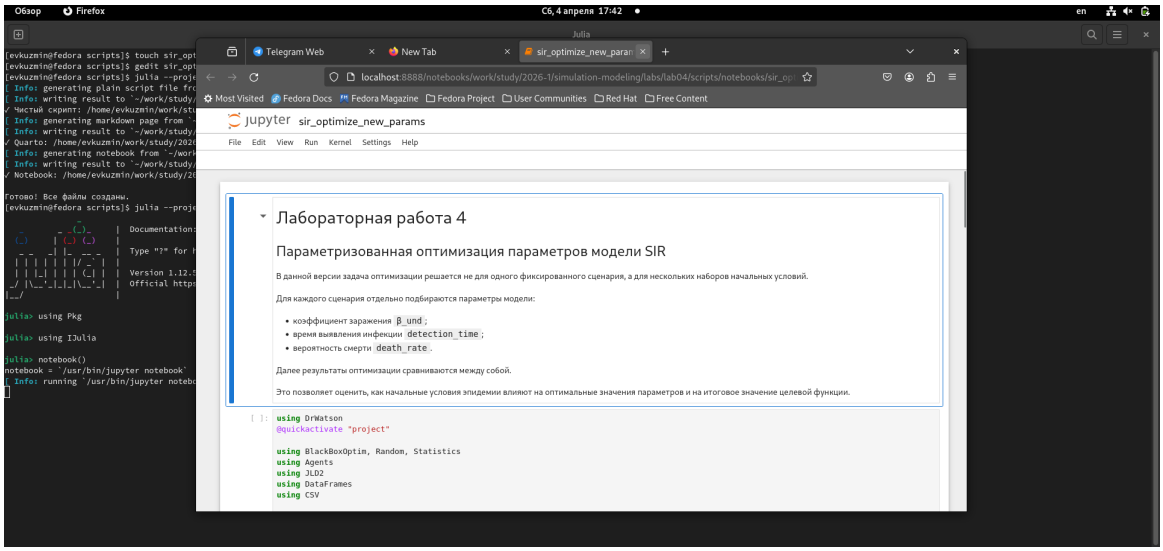


Рисунок 4.25: Генерация файлов и notebook для параметризованной версии четвёртого скрипта

На рисунке показан этап генерации артефактов для параметризованной версии четвёртого скрипта.

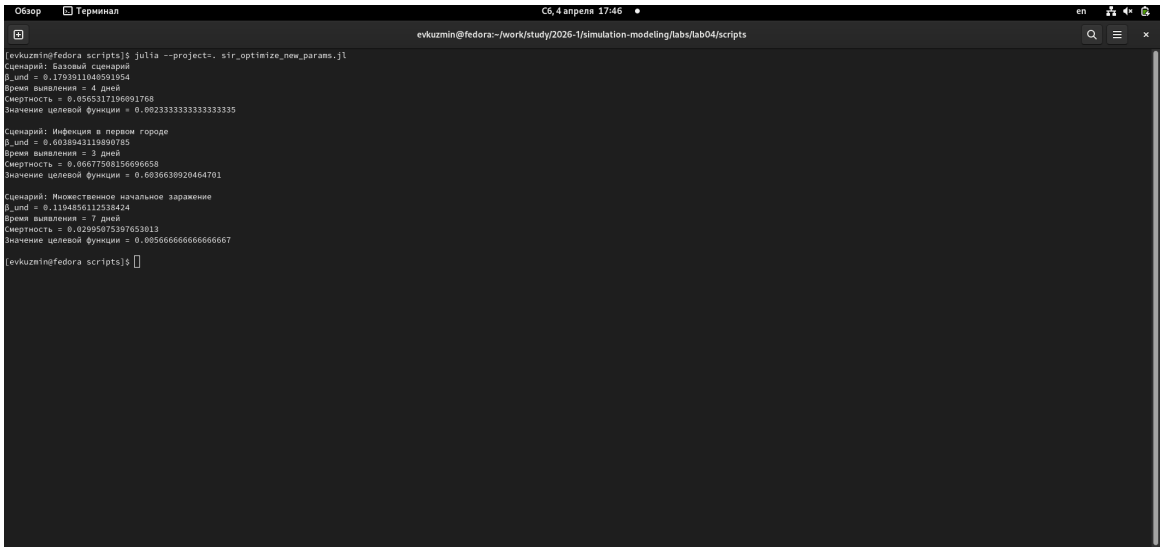


Рисунок 4.26: Выполненная параметризованная версия четвёртого скрипта

На рисунке показан запуск параметризованной версии четвёртого скрипта.

4.9.1 Параметризованная версия

Листинг 4.12. Параметризованная версия скрипта оптимизации sir_optimize_new_params.jl

```
# # [illegible] 4
# ## [illegible] SIR
#
# [illegible]
# [illegible], [illegible].
#
# [illegible]:
#
# - [illegible] `[illegible]_und`;
# - [illegible] `detection_time`;
# - [illegible] `death_rate`.
#
# [illegible].
#
# [illegible], [illegible]
# [illegible] [illegible].

using DrWatson
@quickactivate "project"

using BlackBoxOptim, Random, Statistics
using Agents
using JLD2
```

```
using DataFrames
using CSV

include(srcdir("sir_model.jl"))

# ## [illegible] [illegible]
#
# [illegible] [illegible] [illegible]:
#
# - [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible];
# - [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible];
# - [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

scenario_sets = [
(name = "[illegible] [illegible]", Is = [0, 0, 1], Ns = [500, 500, 500]),
(name = "[illegible] [illegible] [illegible]", Is = [1, 0, 0], Ns = [500, 500, 500]
(name = "[illegible] [illegible] [illegible]", Is = [5, 0, 0], Ns = [500, 50
]

# ## [illegible] [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible], [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

function make_cost_function(scenario)
    function cost_fast(x)
        infected_frac(model) =
            count(a.status == :I for a in allagents(model)) / nagen
```

```
        total_population = sum(scenario.Ns)
        dead_count(model) = total_population - nagents(model)

        peak_vals = Float64[]
        dead_vals = Float64[]

        for rep in 1:2
            model = initialize_sir(;
                Ns = scenario.Ns,
                β_und = fill(x[1], 3),
                β_det = fill(x[1] / 10, 3),
                infection_period = 10,
                detection_time = round(Int, x[2]),
                death_rate = x[3],
                reinfection_probability = 0.1,
                Is = scenario.Is,
                seed = 42 + rep,
                n_steps = 50,
            )

            peak_infected = 0.0

            for step in 1:50
                Agents.step!(model, 1)
                frac = infected_frac(model)
                if frac > peak_infected
                    peak_infected = frac
                end
```

```
            end

            push!(peak_vals, peak_infected)
            push!(dead_vals, dead_count(model) / total_population)
        end

        mean_peak = mean(peak_vals)
        mean_dead = mean(dead_vals)

        return mean_peak + mean_dead
    end

    return cost_fast
end

# ## □□□□□□□□□□ □□□□□□□□□□□ □□□ □□□□ □□□□□□□□□
#
# □□□ □□□□□□□ □□□□□□□□ □□□□□□□□ □□□□□□□□□□□ □□□□□□□□□□□ □ □□□□□□□□□□□
# □□□□□□□□□ □□□□□□□□□.

results = DataFrame(
    scenario = String[],
    beta_und = Float64[],
    detection_time = Int[],
    death_rate = Float64[],
    fitness = Float64[],
)
```

```
for scenario in scenario_sets
    cost_fun = make_cost_function(scenario)

    result = bboptimize(
        cost_fun;
        Method = :random_search,
        SearchRange = [
            (0.1, 1.0),
            (3.0, 14.0),
            (0.01, 0.1),
        ],
        NumDimensions = 3,
        MaxTime = 20,
        TraceMode = :silent,
    )

    best = best_candidate(result)
    fitness = best_fitness(result)

    push!(
        results,
        (
            scenario.name,
            best[1],
            round(Int, best[2]),
            best[3],
            fitness,
        ),
```

```
    )

    println("□□□□□□□□: $(scenario.name)")
    println("□_und = $(best[1])")
    println("□□□□□ □□□□□□□□□ = $(round(Int, best[2])) □□□□")
    println("□□□□□□□□□□ = $(best[3])")
    println("□□□□□□□□ □□□□□□□ □□□□□□□ = $(fitness)")
    println()
end

# ## □□□□□□□□ □□□□□□□□ □□□□□□□

results

# ## □□□□□□□□□□ □□□□□□□□□□□
#
# □□□□□□□□ □□□□□□□ □□ □□□□□ □□□□□□□□□□.

CSV.write(datadir("optimization_scenarios.csv"), results)

save(
    datadir("optimization_scenarios.jld2"),
    Dict("results" => results),
)

# ## □□□□□
#
# □ □□□□□□□□□□ □□□□ □□□□□□□□ □□□□□□□□□□□ □□□□□□□□ □□□□□□ □□□ □□□□□□□□□□
```

```
# [illegible] [illegible] [illegible]. [illegible] [illegible], [illegible]
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible].
```

4.10 Скрипт комплексной визуализации результатов

Пятый скрипт используется для построения составного рисунка по уже рассчитанным данным.

4.10.1 Обычная версия и literate-версия

Листинг 4.13. Скрипт комплексной визуализации sir_visualize_dynamics.jl

```
using DrWatson
@quickactivate "project"

using Agents, DataFrames, Plots, CSV

include(srcdir("sir_model.jl"))

# [illegible] [illegible] [illegible]
df = CSV.read(datadir("beta_scan_all.csv"), DataFrame)

# [illegible] [illegible]: [illegible] [illegible] [illegible] [illegible]
p1 = plot(
```

```
    df.beta,
    df.peak,
    label = "□□□",
    xlabel = "□",
    ylabel = "□□□□ □□□□□□□□□□□□□□",
)

plot!(
    p1,
    df.beta,
    df.final_inf,
    label = "□□□□□□□□",
)

# □□□□□□ □□□□□□: □□□□□ □□□□□□□
p2 = plot(
    df.beta,
    df.deaths,
    xlabel = "□",
    ylabel = "□□□□□ □□□□□□□",
    label = "□□□□□□□",
)

# □□□□□□ □□□□□□: □□□□ □□□□□□□□□□□□□
p3 = plot(
    df.beta,
    df.final_rec,
    xlabel = "□",
```

```
    ylabel = "□□□□ □□□□□□□□□□□□□",
    label = "□□□□□□□□□□□□□□",
)

# □□□□□□□□□□ □□□□□□□
plot(
    p1,
    p2,
    p3,
    layout = (3, 1),
    size = (800, 900),
)

savefig(plotsdir("comprehensive_analysis.png"))
```

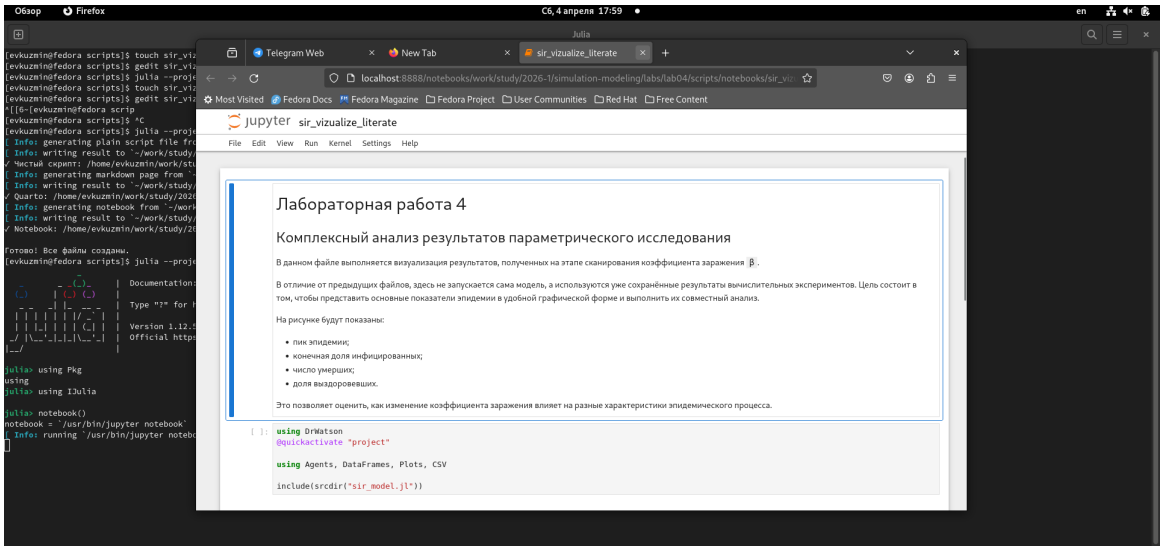


Рисунок 4.27: Создание и выполнение пятого скрипта, генерация literate-версии и notebook

На рисунке показан этап выполнения пятого скрипта и генерации для него notebook и literate-версии.

4.10.2 Результат комплексного анализа

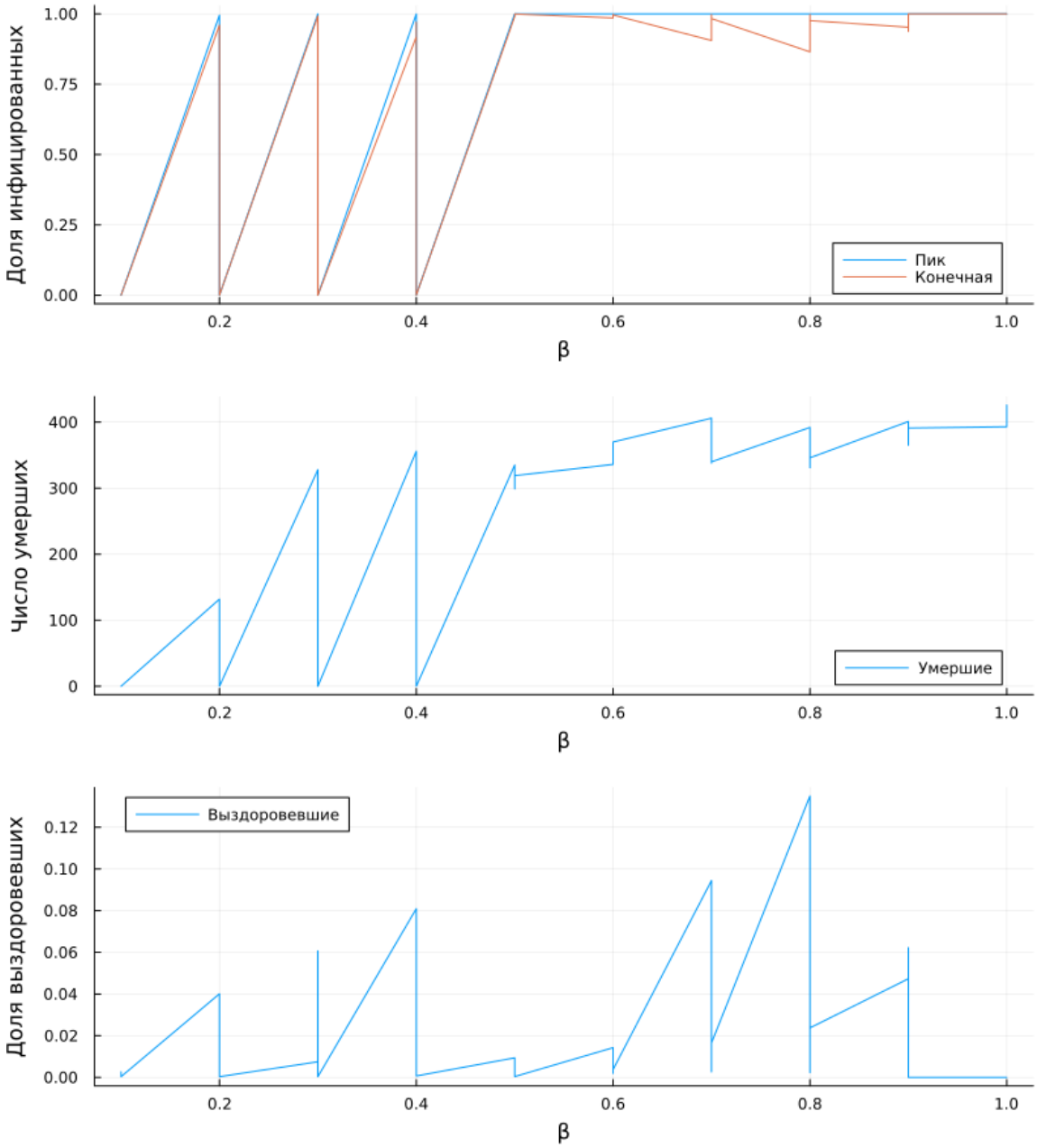


Рисунок 4.28: Комплексный анализ результатов моделирования

На рисунке объединены несколько итоговых зависимостей: пик эпидемии, конечная доля инфицированных, число умерших и доля выздоровевших. Такой

составной график позволяет одновременно анализировать несколько ключевых характеристик модели.

Листинг 4.14. Literate-версия скрипта комплексной визуализации sir_visualize_literate.jl

```
# # [illegible] 4
# ## [illegible]
#
# [illegible], [illegible]
# [illegible] `[illegible]`.
#
# [illegible], [illegible],
# [illegible].
# [illegible], [illegible]
# [illegible].
#
# [illegible]:
#
# - [illegible];
# - [illegible];
# - [illegible];
# - [illegible].
#
# [illegible], [illegible]
# [illegible].

using DrWatson
@quickactivate "project"
```

```julia
using Agents, DataFrames, Plots, CSV

include(srcdir("sir_model.jl"))

# ## [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] CSV-[illegible].

df = CSV.read(datadir("beta_scan_all.csv"), DataFrame)

# ## [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible]:
#
# - [illegible] [illegible];
# - [illegible] [illegible] [illegible].
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible] [illegible].

p1 = plot(
    df.beta,
    df.peak,
    label = "[illegible]",
    xlabel = "[illegible]",
    ylabel = "[illegible] [illegible]",
)
```

```
plot!(
    p1,
    df.beta,
    df.final_inf,
    label = "□□□□□□□□",
)

# ## □□□□□□□□□□ □□□□□□□ □□□□□□□
#
# □□ □□□□□□ □□□□□□□ □□□□□□□□□□□□ □□□□□ □□□□□□□
# □ □□□□□□□□□□□ □□ □□□□□□□□□□□□ □□□□□□□□□.

p2 = plot(
    df.beta,
    df.deaths,
    xlabel = "□",
    ylabel = "□□□□□ □□□□□□□",
    label = "□□□□□□□",
)

# ## □□□□□□□□□□ □□□□□□□□ □□□□□□□
#
# □□ □□□□□□□ □□□□□□□ □□□□□□□□□□□□ □□□□ □□□□□□□□□□□□□
# □ □□□□□□□□□□□ □□ □□□□□□□□□□□□ □□□□□□□□□.

p3 = plot(
    df.beta,
    df.final_rec,
```

```
    xlabel = "□",
    ylabel = "□□□□ □□□□□□□□□□□□□",
    label = "□□□□□□□□□□□□□",
)

# ## □□□□□□□□□□□□ □□□□□□□□□□ □□□□□□□
#
# □□□□□□□□□ □□□ □□□□□□□ □ □□□□ □□□□□□□□□ □□□□□□□
# □□□ □□□□□□□□□□□ □□□□□□□ □□□□□□□□□□□.

plot(
    p1,
    p2,
    p3,
    layout = (3, 1),
    size = (800, 900),
)

savefig(plotsdir("comprehensive_analysis.png"))

# ## □□□□□
#
# □ □□□□□□□□□□ □□□ □□□□□□□□ □□□□□□□□□ □□□□□□□, □□□□□□□□□□ □□□□□□□
# □□□□□□□□□□□□□ □□□□□□□□□ `□` □□ □□□□□□□□ □□□□□□□□□□□□□□□ □□□□□□□□.
# □□□□□□□□□□ □□□□□□ □□□□ □□□□□□□□ □□□□□□□□□ □□□□□□□ □□ □□□□□□
# □□□□□□□□□□□□□□ □□□□□□□, □□ □ □□ □□□□□□□□□□□□ □ □□□□□ □□□□□□
# □□□□□□□□□□ □ □□□□□□□□□□□□□□.
```

5 Дополнительные задания

5.1 Дополнительные задания 1–3

Дополнительные задания 1–3 объединены в один скрипт, так как они связаны с базовой динамикой модели, исследованием порога эпидемии и анализом гетерогенности коэффициента заражения.

Листинг 5.1. Скрипт дополнительных заданий 1–3

extra_1_2_3.jl

```
# # [redacted] 1–3
# ## [redacted], [redacted] [redacted] [redacted] [redacted]
#
# [redacted] [redacted] [redacted] [redacted] [redacted]:
#
# - [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted];
# - [redacted] [redacted];
# - [redacted] [redacted] [redacted] [redacted] [redacted].
#
# [redacted] [redacted] [redacted], [redacted] [redacted] [redacted] [redacted]
# [redacted] [redacted] [redacted] `[redacted]` [redacted] [redacted] [redacted] [redacted] [redacted] SIR.

using DrWatson
@quickactivate "project"
```

```
using Agents, DataFrames, Plots, CSV, Statistics
using JLD2

include(srcdir("sir_model.jl"))

function main()

# ## 1. □□□□□□□ □□□□□□□
#
# □□□□□□□ □□□□□□□□ □□□□□□ □□□□□□ □ □□□□□□□□□□□ □□ □□□□□□□□□□,
# □□□□□□□□ □□□□□□ □□□□□□□□ □□□□□□□□□□□ `S`, `I`, `R`
# □ □□□□□□□□ □□□□□□□ □□□□□□□□□□□□□□ □□□□□:
#
# $$ R_0 = \frac{\beta}{\gamma}, \quad \gamma = \frac{1}{infection\

params_basic = Dict(
    :Ns => [1000, 1000, 1000],
    :□_und => [0.5, 0.5, 0.5],
    :□_det => [0.05, 0.05, 0.05],
    :infection_period => 14,
    :detection_time => 7,
    :death_rate => 0.02,
    :reinfection_probability => 0.1,
    :Is => [0, 0, 1],
    :seed => 42,
    :n_steps => 100,
)
```

```
model = initialize_sir(; params_basic...)

times = Int[]
S_vals = Int[]
I_vals = Int[]
R_vals = Int[]

for step in 1:params_basic[:n_steps]
    Agents.step!(model, 1)
    push!(times, step)
    push!(S_vals, susceptible_count(model))
    push!(I_vals, infected_count(model))
    push!(R_vals, recovered_count(model))
end

basic_df = DataFrame(
    time = times,
    susceptible = S_vals,
    infected = I_vals,
    recovered = R_vals,
)

□ = 1 / params_basic[:infection_period]
R0 = params_basic[:□_und][1] / □

p_basic = plot(
    basic_df.time,
    basic_df.susceptible,
```

```
    label = "S",
    xlabel = "□□□",
    ylabel = "□□□□□□□□□□□",
    title = "□□□□□□□ □□□□□□□□ SIR",
)
plot!(p_basic, basic_df.time, basic_df.infected, label = "I")
plot!(p_basic, basic_df.time, basic_df.recovered, label = "R")
savefig(p_basic, plotsdir("extra_1_basic_sir.png"))

println("□□□□□□□ □□□□□□□□□□□□□□ □□□□□ R0 = $(round(R0, digits=3))")

# ## 2. □□□□□□□□□□□ □□□□□□
#
# □□□□□□ □□□□□□□□□, □□□ □□□□□ □□□□□□□□□□□ □□□□□□□□ `□`
# □□□□□□□□□ □□□□□□□□, □□ □□□□ □□□ □□□□□ □□□□□□□□□□□□□ □□□□□□□□□ 5%
# □□ □□□□□ □□□□□□□□□□□ □□□□□□□□□.
#
# □□□□□□□□□□ □□□□□□□□□ □□□□□□□□□□□□ □ □□□□□□□□□□□□□ □□□□□□□:
#
# $$ R_0 = 1 \Rightarrow \beta = \gamma. $$

beta_range = 0.05:0.05:1.0
threshold_beta = nothing
threshold_peak = nothing
population_total = sum(params_basic[:Ns])

threshold_results = DataFrame(
    beta = Float64[],
```

```
    peak_fraction = Float64[],
)

for beta in beta_range
    model_thr = initialize_sir(;
        Ns = params_basic[:Ns],
        β_und = fill(beta, 3),
        β_det = fill(beta / 10, 3),
        infection_period = params_basic[:infection_period],
        detection_time = params_basic[:detection_time],
        death_rate = params_basic[:death_rate],
        reinfection_probability = params_basic[:reinfection_probabi
        Is = params_basic[:Is],
        seed = params_basic[:seed],
        n_steps = params_basic[:n_steps],
    )

    peak_fraction = 0.0

    for step in 1:params_basic[:n_steps]
        Agents.step!(model_thr, 1)
        frac = infected_count(model_thr) / population_total
        if frac > peak_fraction
            peak_fraction = frac
        end
    end

    push!(threshold_results, (beta, peak_fraction))
```

```
    if threshold_beta === nothing && peak_fraction > 0.05
        threshold_beta = beta
        threshold_peak = peak_fraction
    end
end

theoretical_beta = □

p_threshold = plot(
    threshold_results.beta,
    threshold_results.peak_fraction,
    label = "□□□□□□□ □□□□ I",
    xlabel = "□",
    ylabel = "□□□□ □□□□□□□□□□□□□□",
    title = "□□□□□□□□□□□□ □□□□□□ □□□□□□□□",
)
hline!(p_threshold, [0.05], label = "□□□□□ 5%")
vline!(p_threshold, [theoretical_beta], label = "□□□□□□□□□□□□□ □□□□□ R₀
savefig(p_threshold, plotsdir("extra_2_threshold.png"))

println("□□□□□□□□□□□□□ □□□□□ beta = $(round(theoretical_beta, digits=3)
if threshold_beta !== nothing
    println("□□□□□□□□□ □□□□□ beta = $(threshold_beta), □□□ = $(round(th
else
    println("□ □□□□□□□□□ beta □□□□□ □□ □□□□□□")
end

CSV.write(datadir("extra_2_threshold_results.csv"), threshold_resul
```

```
# ## 3. □□□□□□ □□□□□□□□□□□□□□□
#
# □□ □□□□□ □□□□□□ □□□□□□□□ □□□□□□□ □□□□□□□□□□□□□ □□□□□□□□□□ □□□ □□□□□□ □□□□□□□□
# □ □□□□□□□□□□, □□□ □□□ □□□□□□□ □□ □□□□□□ □□□□□□□□□ □ □□ □□□□□□□□□ □ □□□□□□□ □□□□□□□.
#
# □□□□□□ □□□□□□□□□□□□□ □□□□□□□□□:
#
# - □□□□□□ 1: `□ = 0.2`
# - □□□□□□ 2: `□ = 0.5`
# - □□□□□□ 3: `□ = 0.8`

params_hetero = Dict(
    :Ns => [1000, 1000, 1000],
    :□_und => [0.2, 0.5, 0.8],
    :□_det => [0.02, 0.05, 0.08],
    :infection_period => 14,
    :detection_time => 7,
    :death_rate => 0.02,
    :reinfection_probability => 0.1,
    :Is => [0, 0, 1],
    :seed => 42,
    :n_steps => 100,
)

model_hetero = initialize_sir(; params_hetero...)

hetero_total = DataFrame(
    time = Int[],
```

```
    infected = Int[],
)

hetero_city = DataFrame(
    time = Int[],
    city = Int[],
    infected = Int[],
)

for step in 1:params_hetero[:n_steps]
    Agents.step!(model_hetero, 1)

    push!(hetero_total, (step, infected_count(model_hetero)))

    for city in 1:3
        city_inf = count(a.status == :I && a.pos == city for a in a
        push!(hetero_city, (step, city, city_inf))
    end
end

p_total = plot(
    hetero_total.time,
    hetero_total.infected,
    label = "□□□□□ □□□□□□□□ I",
    xlabel = "□□□",
    ylabel = "□□□□□ □□□□□□□□□□□□□□",
    title = "□□□□□□□□□□□□□□: □□□□□ □□□□□□□□",
)
```

```
savefig(p_total, plotsdir("extra_3_heterogeneity_total.png"))

p_city = plot(
    xlabel = "□□□",
    ylabel = "□□□□□ □□□□□□□□□□□□□□",
    title = "□□□□□□□□□□□□□□: □□□□□□□□ □□ □□□□□□□",
)
for city in 1:3
    subdf = hetero_city[hetero_city.city .== city, :]
    plot!(p_city, subdf.time, subdf.infected, label = "□□□□□ $city")
end
savefig(p_city, plotsdir("extra_3_heterogeneity_by_city.png"))

CSV.write(datadir("extra_3_heterogeneity_total.csv"), hetero_total)
CSV.write(datadir("extra_3_heterogeneity_by_city.csv"), hetero_city

save(datadir("extra_1_basic_result.jld2"), Dict(
    "R0" => R0,
    "basic_df" => basic_df,
    "threshold_beta" => threshold_beta,
    "threshold_peak" => threshold_peak,
))

# ## □□□□□
#
# □ □□□□□□□□□□ □□□□□□□□□□ □□□□□□□□□□□□□□ □□□□□□□ 1-3 □□□□:
#
# - □□□□□□□□ □□□□□□□ □□□□□□□□□□ `S`, `I`, `R`;
```

```
# - [illegible] [illegible] [illegible] [illegible] `R₀`;
# - [illegible] [illegible] [illegible] [illegible] [illegible] `[illegible]`;
# - [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible].

end

main()
```

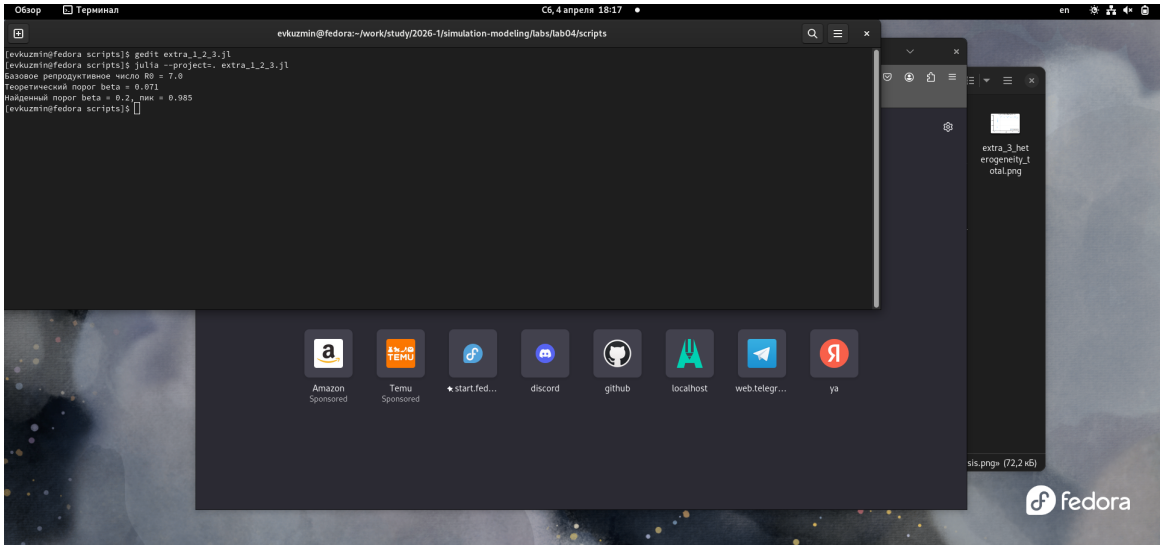


Рисунок 5.1: Создание и выполнение скрипта дополнительных заданий 1–3

На рисунке показан запуск скрипта, объединяющего дополнительные задания 1–3.

5.1.1 Базовая динамика

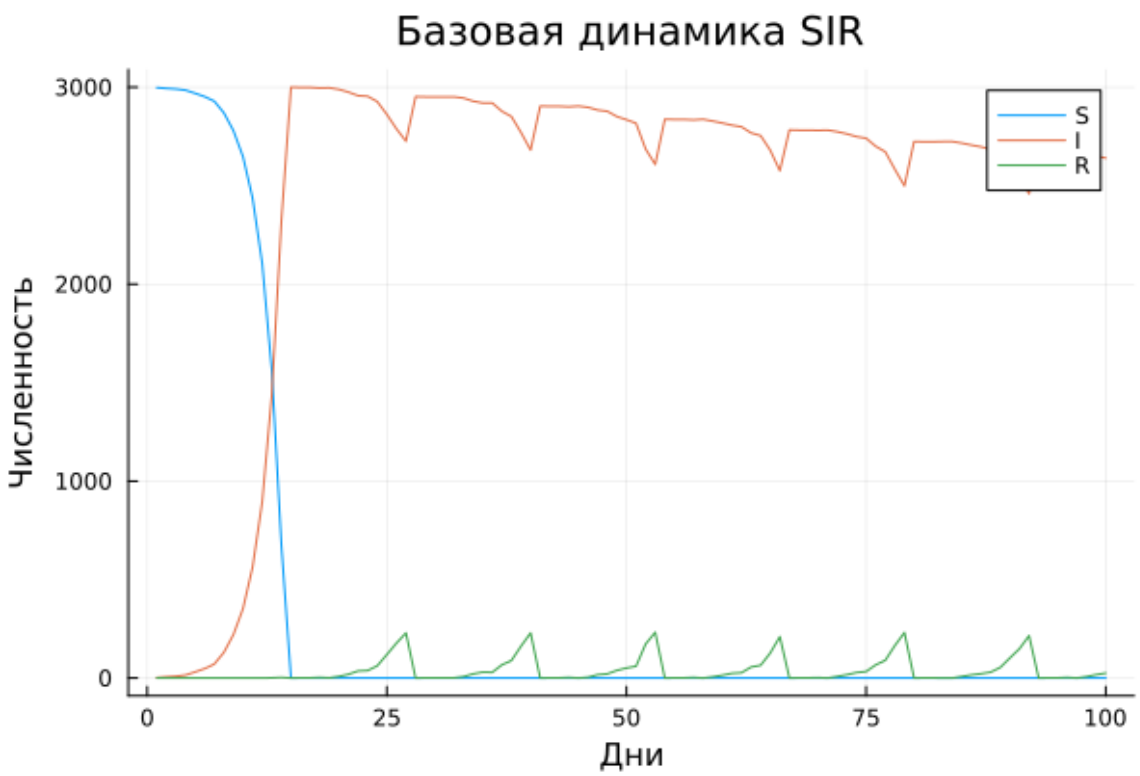


Рисунок 5.2: Базовая динамика SIR для дополнительных заданий

На рисунке показана базовая динамика системы SIR для дополнительных заданий. Видно, что эпидемический процесс проходит стадию роста, достигает пика и затем затухает за счёт уменьшения числа восприимчивых и накопления выздоровевших.

5.1.2 Исследование порога

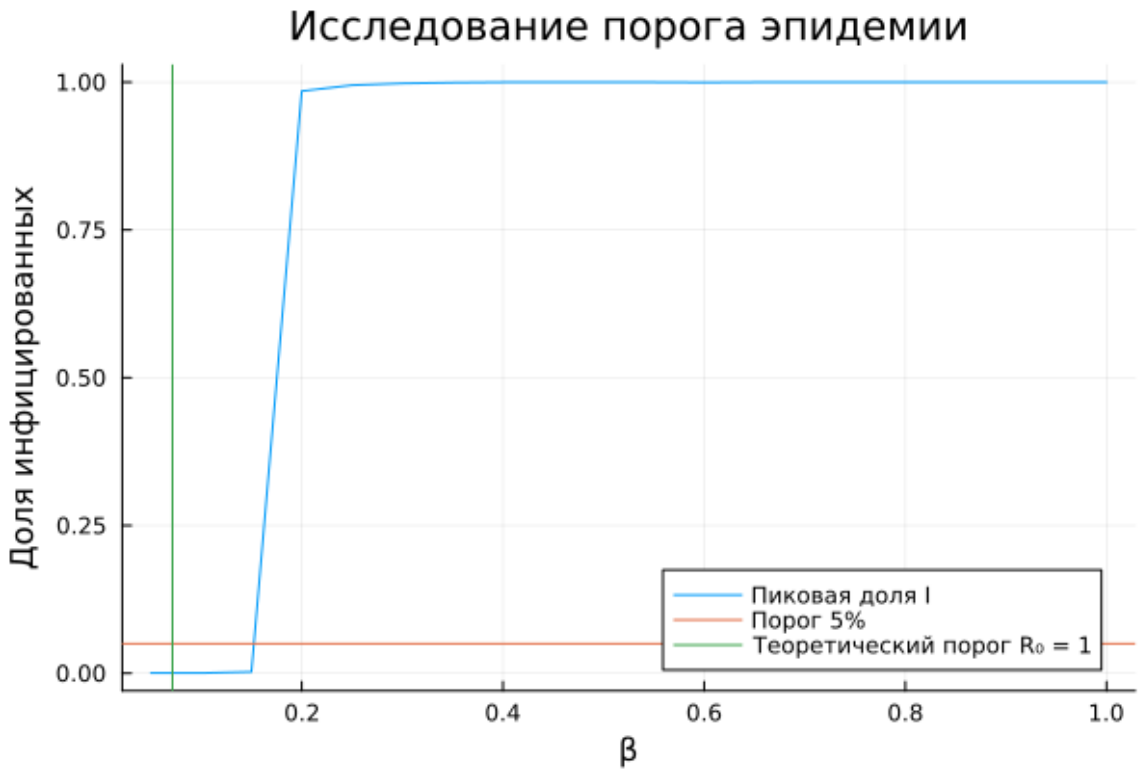


Рисунок 5.3: Исследование порога эпидемии

На рисунке показано исследование эпидемического порога. Горизонтальная линия соответствует уровню 5% популяции, а вертикальная — теоретическому порогу R_0 = 1. Сравнение позволяет сопоставить теоретические оценки и наблюдаемые результаты моделирования.

5.1.3 Гетерогенность

Рисунок 5.4: Динамика инфицированных по городам при гетерогенных параметрах

На рисунке показаны траектории числа инфицированных в отдельных городах при различных значениях β. Более высокая заразность в отдельном городе приводит к более интенсивному локальному развитию эпидемии.

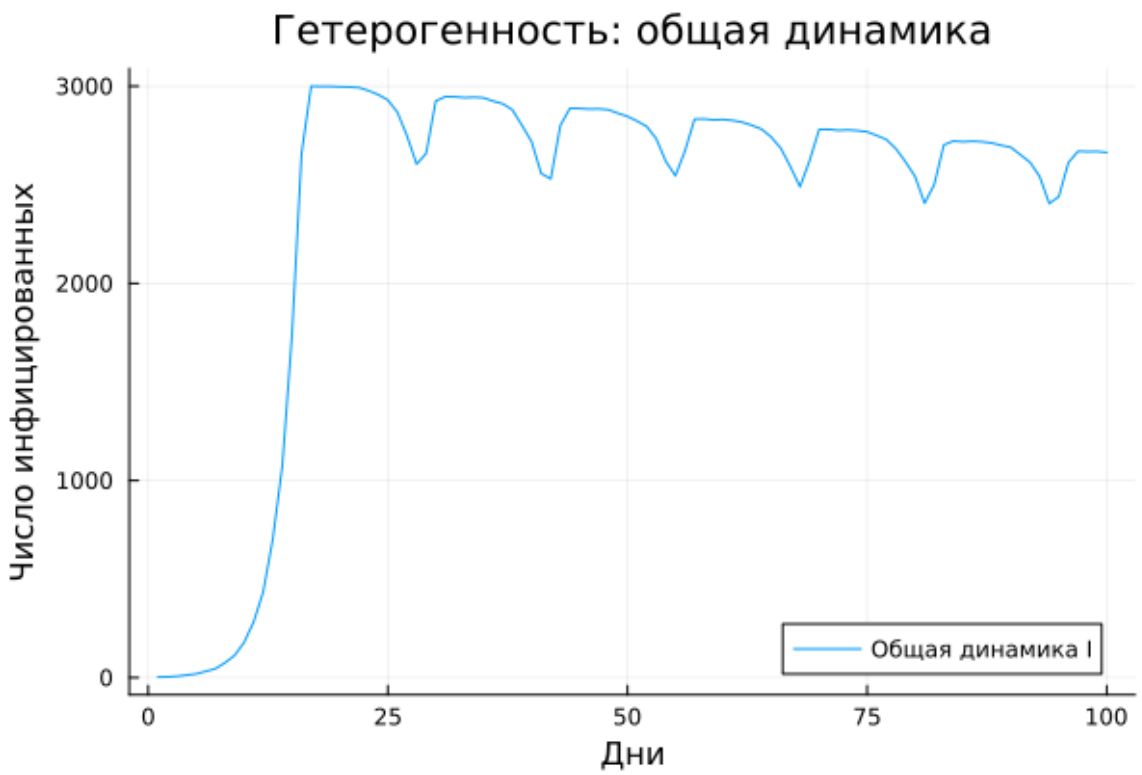


Рисунок 5.5: Общая динамика при гетерогенных параметрах

На рисунке представлена общая динамика инфицированных при гетерогенных параметрах. Введение неоднородности делает поведение системы более сложным по сравнению с однородным случаем.

5.2 Дополнительные задания 4–5

Задания 4 и 5 объединены в один скрипт, так как обе постановки связаны с миграцией и ограничением перемещений между городами.

Листинг 5.2. Скрипт дополнительных заданий 4–5

extra_4_5.jl

```
# # [illegible] 4-5
# ## [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible]:
#
# - [illegible] [illegible] [illegible] [illegible] [illegible] [illegible];
# - [illegible] [illegible] [illegible] [illegible].
#
# [illegible] [illegible] [illegible], [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]: [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible], [illegible] [illegible] [illegible].

using DrWatson
@quickactivate "project"

using Agents, DataFrames, Plots, CSV, Statistics
using JLD2

include(srcdir("sir_model.jl"))

# ## [illegible] [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible].
# [illegible] [illegible] [illegible], [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible].

function create_migration_matrix(C, intensity)
```

```
    M = ones(C, C) .* intensity / (C - 1)
    for i in 1:C
        M[i, i] = 1 - intensity
    end
    return M
end

# ## [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible]:
#
# - [illegible] [illegible], [illegible] [illegible] [illegible] [illegible];
# - [illegible] [illegible] [illegible].
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

function peak_time_without_quarantine(p)
    migration_rates = create_migration_matrix(p[:C], p[:migration_i

    model = initialize_sir(;
        Ns = p[:Ns],
        [illegible]_und = p[:[illegible]_und],
        [illegible]_det = p[:[illegible]_det],
        infection_period = p[:infection_period],
        detection_time = p[:detection_time],
        death_rate = p[:death_rate],
        reinfection_probability = p[:reinfection_probability],
        Is = p[:Is],
```

```
        seed = p[:seed],
        migration_rates = migration_rates,
    )

    infected_frac(model) =
        count(a.status == :I for a in allagents(model)) / nagents(m

    peak = 0.0
    peak_step = 0

    for step in 1:p[:n_steps]
        agent_ids = collect(allids(model))
        for id in agent_ids
            agent = try
                model[id]
            catch
                nothing
            end
            if agent !== nothing
                sir_agent_step!(agent, model)
            end
        end

        frac = infected_frac(model)
        if frac > peak
            peak = frac
            peak_step = step
        end
```

```
    end

    return (peak_time = peak_step, peak_value = peak)
end

# ## □□□□□□□□ □ □□□□□□□□□□□
#
# □ □□□□ □□□□□□□□ □□□□□□□□□□□□ □□□□□□□ □□□□□□□□□□□ □□□□:
# □□□□ □□□□ □□□□□□□□□□□□□□□ □ □□□□□□ □□□□□□□□□ □□□□□□□□ □□□□□,
# □□□□□□□□□ □□ □□□□□ □□□□□□ □□□□□□□□□□□.

function peak_time_with_quarantine(p)
    migration_rates = create_migration_matrix(p[:C], p[:migration_i

    model = initialize_sir(;
        Ns = p[:Ns],
        □_und = p[:□_und],
        □_det = p[:□_det],
        infection_period = p[:infection_period],
        detection_time = p[:detection_time],
        death_rate = p[:death_rate],
        reinfection_probability = p[:reinfection_probability],
        Is = p[:Is],
        seed = p[:seed],
        migration_rates = copy(migration_rates),
    )

    infected_frac(model) =
```

```
        count(a.status == :I for a in allagents(model)) / nagents(m

    peak = 0.0
    peak_step = 0

    quarantine_threshold = p[:quarantine_threshold]
    quarantined_cities = falses(p[:C])

    for step in 1:p[:n_steps]
        for city in 1:p[:C]
            city_population = count(a.pos == city for a in allagent
            city_infected = count(a.status == :I && a.pos == city f

            if city_population > 0
                city_frac = city_infected / city_population
                if city_frac > quarantine_threshold && !quarantined
                    model.migration_rates[city, :] .= 0.0
                    model.migration_rates[city, city] = 1.0
                    quarantined_cities[city] = true
                end
            end
        end

        agent_ids = collect(allids(model))
        for id in agent_ids
            agent = try
                model[id]
            catch
```

```
                nothing
            end
            if agent !== nothing
                sir_agent_step!(agent, model)
            end
        end

        frac = infected_frac(model)
        if frac > peak
            peak = frac
            peak_step = step
        end
    end

    return (peak_time = peak_step, peak_value = peak)
end

function main()

# ## [illegible] [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] 0.0 [illegible] 0.5
# [illegible] 0.1. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible] [illegible].

    migration_intensities = 0.0:0.1:0.5
    seeds = [42, 43, 44]
```

```
    params_list = []

    for mig in migration_intensities
        for s in seeds
            push!(
                params_list,
                Dict(
                    :migration_intensity => mig,
                    :C => 3,
                    :Ns => [1000, 1000, 1000],
                    :□_und => [0.5, 0.5, 0.5],
                    :□_det => [0.05, 0.05, 0.05],
                    :infection_period => 14,
                    :detection_time => 7,
                    :death_rate => 0.02,
                    :reinfection_probability => 0.1,
                    :Is => [1, 0, 0],
                    :seed => s,
                    :n_steps => 150,
                    :quarantine_threshold => 0.1,
                ),
            )
        end
    end

# ## □□□□□□ □□□□□□□□□□□□□
#
# □□□ □□□□□□□ □□□□□□ □□□□□□□□□□ □□□□□□□□□□□ □□□ □□□□□□□:
```

```
#
# - [illegible] [illegible];
# - [illegible] [illegible].
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

    results = DataFrame(
        migration_intensity = Float64[],
        seed = Int[],
        mode = String[],
        peak_time = Float64[],
        peak_value = Float64[],
    )

    for params in params_list
        res_no_q = peak_time_without_quarantine(params)
        push!(results, (
            params[:migration_intensity],
            params[:seed],
            "[illegible] [illegible]",
            res_no_q.peak_time,
            res_no_q.peak_value,
        ))

        res_q = peak_time_with_quarantine(params)
        push!(results, (
            params[:migration_intensity],
            params[:seed],
```

```
            "□ □□□□□□□□□□",
            res_q.peak_time,
            res_q.peak_value,
        ))

        println("□□□□□□□□ □□□□□□□□□□□□: migration = $(params[:migration_i
    end

# ## □□□□□□□□□□ □□□□□□□□□□□

#
# □□□□□□ □□□□□□□ □□□□□□□□□□□ □□□□□□□□□□□ □ CSV-□□□□.

    CSV.write(datadir("extra_4_5_results.csv"), results)

# ## □□□□□□□□□□ □□□□□□□□□□□

#
# □□□ □□□□□□□□ □□□□□□□□ □□□□□□□□□□□□□ □□□□□□□□ □ □□□ □□□□□□□ □□□□□□
# (□ □□□□□□□□□□ □ □□□ □□□□□□□□□) □□□□□□□□□□□ □□□□□□□ □□□□□□□□:
#
# - □□□□□□□ □□ □□□□;
# - □□□□□□□□ □□□□.

    grouped = combine(
        groupby(results, [:migration_intensity, :mode]),
        :peak_time => mean => :mean_peak_time,
        :peak_value => mean => :mean_peak_value,
    )
```

```
# ## [illegible]
#
# [illegible]
# [illegible] [illegible].

    p_time = plot(
        xlabel = "[illegible]",
        ylabel = "[illegible]",
        title = "[illegible]: [illegible]",
    )

    for mode in unique(grouped.mode)
        subdf = grouped[grouped.mode .== mode, :]
        plot!(p_time, subdf.migration_intensity, subdf.mean_peak_ti
    end
    savefig(p_time, plotsdir("extra_4_5_peak_time.png"))

# ## [illegible]
#
# [illegible]
# [illegible] [illegible].

    p_peak = plot(
        xlabel = "[illegible]",
        ylabel = "[illegible]",
        title = "[illegible]: [illegible]",
    )
```

```
    for mode in unique(grouped.mode)
        subdf = grouped[grouped.mode .== mode, :]
        plot!(p_peak, subdf.migration_intensity, subdf.mean_peak_va
    end
    savefig(p_peak, plotsdir("extra_4_5_peak_value.png"))

# ## [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible] JLD2.

    save(datadir("extra_4_5_results.jld2"), Dict(
        "results" => results,
        "grouped" => grouped,
    ))
end

main()

# ## [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] 4-5 [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible],
# [illegible] [illegible] [illegible] [illegible] [illegible].
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible],
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].
```

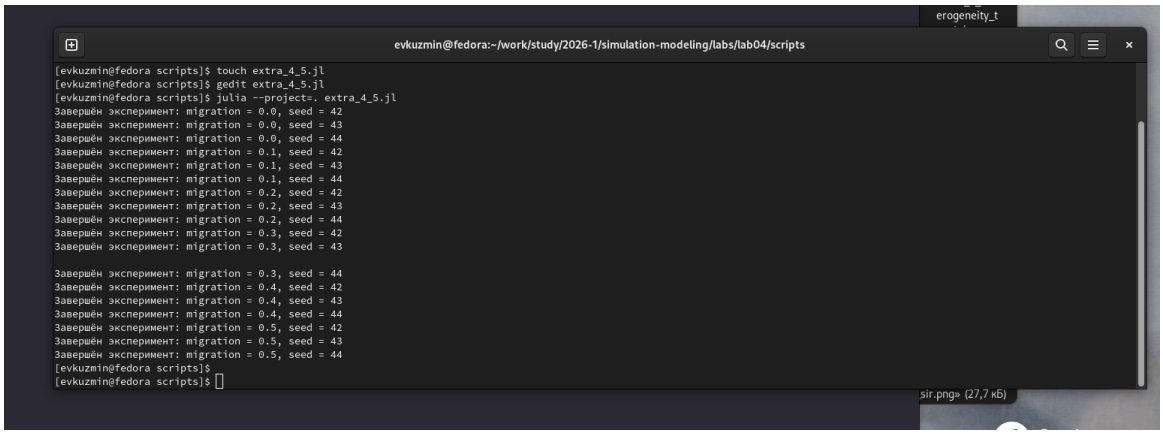


Рисунок 5.6: Создание и выполнение скрипта дополнительных заданий 4–5

На рисунке показан запуск скрипта дополнительных заданий 4–5.

5.2.1 Влияние миграции и карантина на время до пика

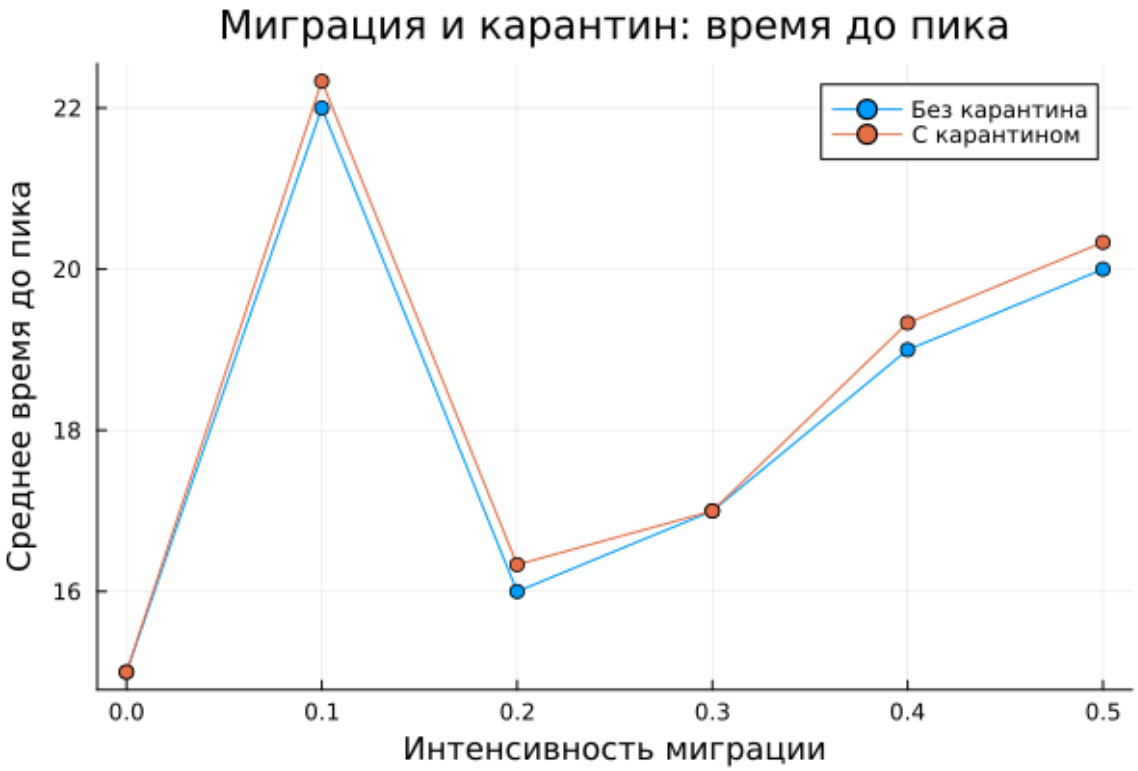


Рисунок 5.7: Сравнение времени достижения пика с карантином и без карантина

На рисунке показано, как меняется среднее время достижения пика при различных значениях миграции. Сравнение режимов с карантином и без карантина позволяет оценить, насколько ограничение перемещений замедляет развитие эпидемии.

5.2.2 Влияние миграции и карантина на величину пика

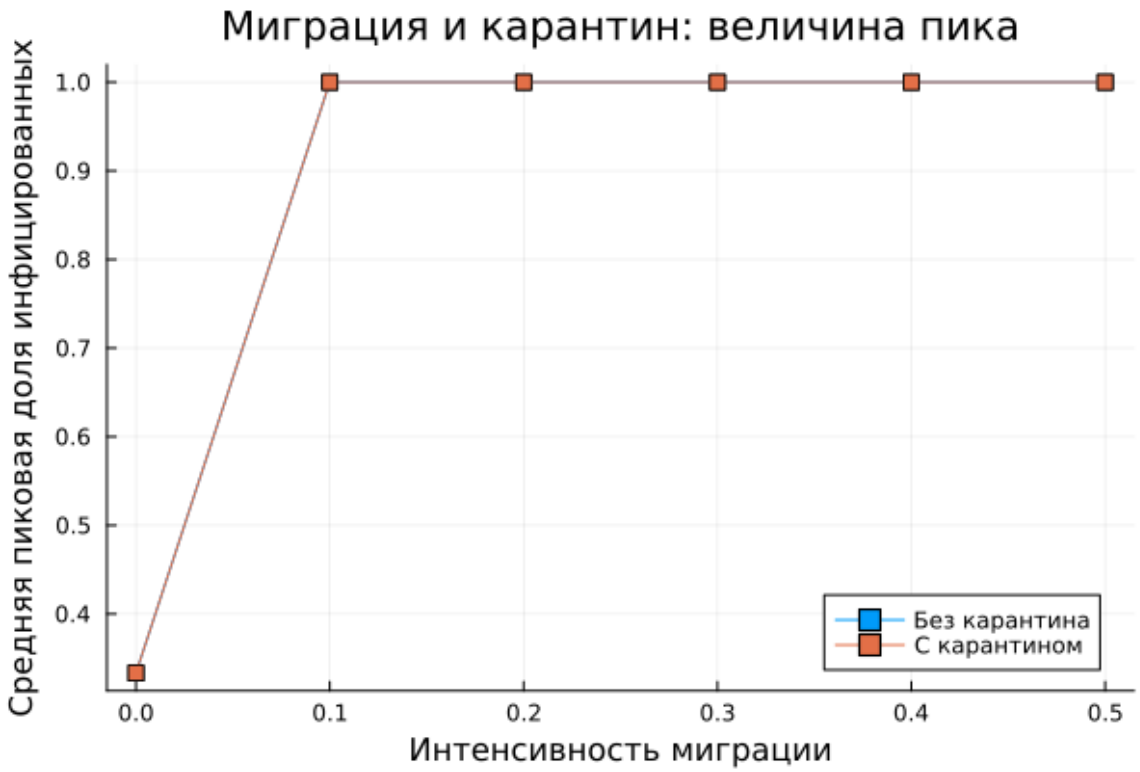


Рисунок 5.8: Сравнение величины пика с карантином и без карантина

На рисунке показана зависимость величины пика эпидемии от интенсивности миграции для режимов с карантином и без карантина. Это позволяет оценить эффективность карантинных мер по снижению максимальной нагрузки.

5.3 Дополнительное задание 6

В дополнительном задании 6 решалась задача оптимизации параметров модели при ограничении на пиковую долю инфицированных.

Листинг 5.3. Скрипт дополнительного задания 6

extra_6.jl

```
# # [illegible] 6
# ## [illegible] SIR
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].
# [illegible] [illegible] [illegible] [illegible] [illegible], [illegible]:
#
# - [illegible] [illegible] [illegible];
# - [illegible] [illegible] [illegible] 30% [illegible] [illegible] [illegible] [illegible].
#
# [illegible] [illegible] [illegible] [illegible] [illegible]:
#
# - [illegible] [illegible] `[illegible]_und`;
# - [illegible] [illegible] [illegible] `detection_time`;
# - [illegible] [illegible] `death_rate`.
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible], [illegible] [illegible] [illegible] [illegible] [illegible].

using DrWatson
@quickactivate "project"

using BlackBoxOptim, Random, Statistics
```

```
using Agents
using JLD2
using DataFrames
using CSV
using Plots

include(srcdir("sir_model.jl"))

# ## [illegible] [illegible]
#
# [illegible] [illegible] [illegible] [illegible] `x`, [illegible]:
#
# - `x[1]` — [illegible] [illegible] `[illegible]_und`;
# - `x[2]` — [illegible] [illegible] [illegible];
# - `x[3]` — [illegible] [illegible].
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] 30% [illegible],
# [illegible] [illegible] [illegible] [illegible] [illegible].

function cost_with_constraint(x)
    infected_frac(model) =
        count(a.status == :I for a in allagents(model)) / nagents(m

    total_population = 1500
    dead_frac(model) = (total_population - nagents(model)) / total_
```

```
peak_vals = Float64[]
dead_vals = Float64[]

for rep in 1:2
    model = initialize_sir(;
        Ns = [500, 500, 500],
        β_und = fill(x[1], 3),
        β_det = fill(x[1] / 10, 3),
        infection_period = 10,
        detection_time = round(Int, x[2]),
        death_rate = x[3],
        reinfection_probability = 0.1,
        Is = [0, 0, 1],
        seed = 42 + rep,
        n_steps = 50,
    )

    peak_infected = 0.0

    for step in 1:50
        Agents.step!(model, 1)
        frac = infected_frac(model)
        if frac > peak_infected
            peak_infected = frac
        end
    end

    push!(peak_vals, peak_infected)
```

```
        push!(dead_vals, dead_frac(model))
    end

    mean_peak = mean(peak_vals)
    mean_dead = mean(dead_vals)

    penalty = 0.0
    if mean_peak > 0.30
        penalty = 10.0 * (mean_peak - 0.30)
    end

    return mean_dead + penalty
end

function main()

# ## □□□□□□ □□□□□□□□□□□
#
# □□□□□□□□□□□□ □□□□□□□□ □□□□□□□□□□ □□□□□□, □□□□□□□ □□□□□□ □□□□□□□□
# □□□ □□□□□□□ □□□□□□□□□□□□□□□□ □□□□□□□□□□□ □ □□□□□□□□ □□□□□□□□□.

    result = bboptimize(
        cost_with_constraint;
        Method = :random_search,
        SearchRange = [
            (0.1, 1.0),
            (3.0, 14.0),
            (0.01, 0.1),
```

```
        ],
        NumDimensions = 3,
        MaxTime = 20,
        TraceMode = :silent,
    )

# ## □□□□□□□□□□ □□□□□□□□□ □□□□□□□□□□
#
# □□□□□ □□□□□□□□□□ □□□□□□□□□□□ □□□□□□□□□□□ □□□□□□□□□ □□□□□□□□□
# □ □□□□□□□□ □□□□□□□ □□□□□□□.

    best = best_candidate(result)
    fitness = best_fitness(result)

    println("□□□□□□□□□□□ □□□□□□□□□:")
    println("□_und = $(best[1])")
    println("□□□□□ □□□□□□□□□ = $(round(Int, best[2])) □□□□")
    println("□□□□□□□□□□ = $(best[3])")
    println("□□□□□□□□ □□□□□□□ □□□□□□□ = $(fitness)")

# ## □□□□□□□□□□□ □□□□□□
#
# □□□ □□□□□□□□□ □□□□□□□□□□ □□□□□□□□ □□□□□□□□□ □□□□□□ □□□□□□,
# □□□□□ □□□□□□□ □□□□□□□□□□□□ □□□ □□□□□□□□□□□□□□ □ □□□□ □□□□□□□.

    model_check = initialize_sir(;
        Ns = [500, 500, 500],
        □_und = fill(best[1], 3),
```

```
        ⊠_det = fill(best[1] / 10, 3),
        infection_period = 10,
        detection_time = round(Int, best[2]),
        death_rate = best[3],
        reinfection_probability = 0.1,
        Is = [0, 0, 1],
        seed = 123,
        n_steps = 50,
    )

    times = Int[]
    infected_vals = Float64[]
    dead_vals = Float64[]

    total_population = 1500
    peak_check = 0.0

    for step in 1:50
        Agents.step!(model_check, 1)

        current_infected = infected_count(model_check) / total_popu
        current_dead = (total_population - total_count(model_check)

        push!(times, step)
        push!(infected_vals, current_infected)
        push!(dead_vals, current_dead)

        if current_infected > peak_check
```

```
            peak_check = current_infected
        end
    end

    check_df = DataFrame(
        time = times,
        infected_fraction = infected_vals,
        dead_fraction = dead_vals,
    )

    println("□□□□□□□□□□□ □□□ □□□□□□□□□□□□□□ = $(round(peak_check, digits=3
    println("□□□□□ 30% □□□□□□□□: $(peak_check <= 0.30)")

# ## □□□□□□□□□□ □□□□□□□ □□□□□□□□□□□□ □□□□□□□
#
# □□ □□□□□□□ □□□□□□□□□□□□□:
#
# - □□□□ □□□□□□□□□□□□□□;
# - □□□□ □□□□□□□□;
# - □□□□□□□□□□□□□□□ □□□□□ □□□□□□□□□□□ 30%.

    p = plot(
        check_df.time,
        check_df.infected_fraction,
        label = "□□□□ □□□□□□□□□□□□□□",
        xlabel = "□□□",
        ylabel = "□□□□",
        title = "□□□□□□□□ □□□□□□□□□□□□ □□□□□□□",
```

```
    )

    plot!(p, check_df.time, check_df.dead_fraction, label = "□□□□ □□□
    hline!(p, [0.30], label = "□□□□□□□□□□□ 30%")

    savefig(p, plotsdir("extra_6_optimization_check.png"))

# ## □□□□□□□□□□ □□□□□□□□□□□
#
# □□□□□□□□ □□□□□□□□□ □□□□□□□□□, □□□□□□□□ □□□□□□□ □□□□□□□
# □ □□□□□□□□□□ □□□□□□□□□□□□ □□□□□□□.

    CSV.write(datadir("extra_6_check.csv"), check_df)

    save(
        datadir("extra_6_optimization_result.jld2"),
        Dict(
            "best" => best,
            "fitness" => fitness,
            "peak_check" => peak_check,
            "check_df" => check_df,
        ),
    )

end

main()
```

```
# ## [illegible]
#
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible],
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
# [illegible]. [illegible] [illegible] [illegible], [illegible] [illegible]
# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].
```

5.3.1 Проверка найденного оптимального решения

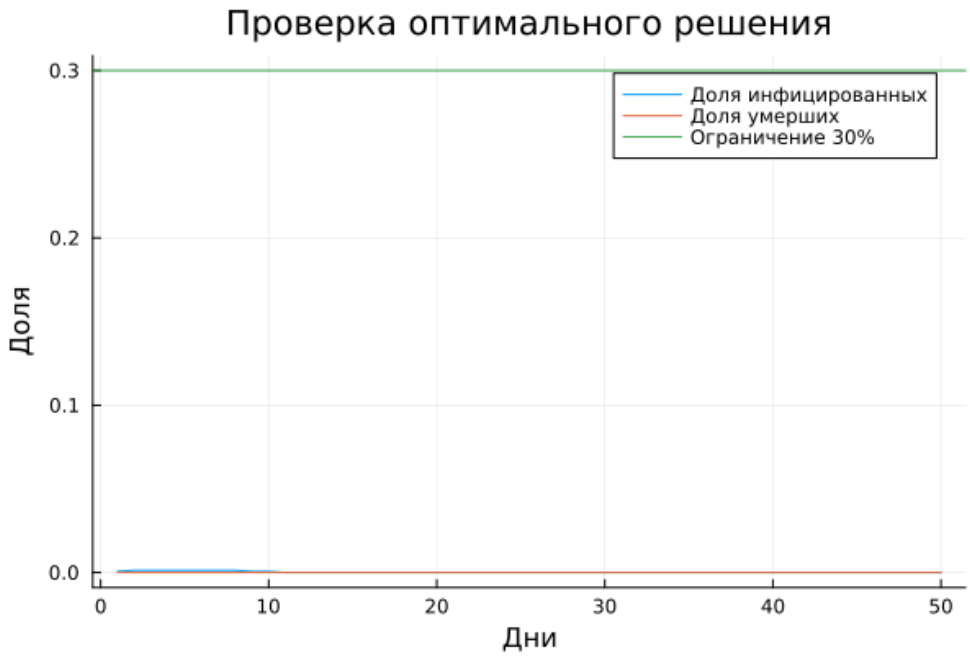


Рисунок 5.9: Проверка оптимального решения для дополнительного задания 6

На рисунке показана проверка найденного оптимального решения. Горизонтальная линия соответствует ограничению на пиковую долю инфицированных. По графику можно оценить, удовлетворяет ли найденное решение заданному ограничению.

6 Итоговые выводы

В ходе выполнения лабораторной работы была реализована агентная модель SIR и проведено её подробное исследование.

В работе были выполнены:

- построение базовой модели;
- подготовка обычных и literate-версий скриптов;
- генерация notebook и Quarto-документации;
- параметрические исследования коэффициента заражения;
- анализ миграции между городами;
- решение задачи оптимизации параметров;
- выполнение дополнительных заданий, связанных с порогом эпидемии, гетерогенностью, карантинными мерами и ограниченной оптимизацией.

Полученные результаты показывают, что агентная модель SIR позволяет исследовать широкий круг эпидемиологических сценариев. Изменение параметров модели существенно влияет на время достижения пика, число инфицированных, число умерших и характер распространения инфекции между городами.

7 Список литературы

1. Kermack, W. O., McKendrick, A. G. A Contribution to the Mathematical Theory of Epidemics. Proceedings of the Royal Society A, 1927.
2. Agents.jl — Julia package for agent-based modeling. Available at: https://juliadynamics.github.io/Age
3. Datseris, G. et al. Agents.jl: A performant and feature-full agent-based modeling software of minimal code complexity. Simulation, 2022.
4. DrWatson.jl — reproducible scientific projects in Julia. Available at: https://juliadynamics.github.io/Dr
5. Literate.jl — package for literate programming in Julia. Available at: https://fredrikekre.github.io/Liter
6. BlackBoxOptim.jl — optimization package for Julia. Available at: https://github.com/robertfeldt/Black
7. Методические указания по лабораторной работе №4 по дисциплине «Имитационное моделирование», РУДН, 2026.