Пример решения задачи оптимизации портфеля через линейное программирование в Julia

Постановка задачи

Инвестор хочет распределить $10,000 между 3 активами:

Актив	Ожидаемая доходность	Риск (σ)	Минимальная доля	Максимальная доля
A	8%	10%	10%	50%
B	12%	15%	10%	50%
C	6%	5%	10%	50%

Ковариационная матрица: - Cov(A,B) = 0.5% - Cov(A,C) = -0.2% - Cov(B,C) = 0.3%

Для задачи ЛП, ковариационная матрица и риск игнорируются.

[ ]:
```
MIN_SHARE = 0.1          # Минимальная доля актива
MAX_SHARE = 0.5          # Максимальная доля актива
RETURNS = [0.08, 0.12, 0.06]  # Доходности активов A, B, C
```

```
[ Info: Starting sender/receiver loops
```

[ ]:
```
3-element Vector{Float64}:
 0.08
 0.12
 0.06
```

Инициализация окружения

Подключим необходимые пакеты:

[ ]:
```
# Установка пакетов (раскомментировать при первом запуске)
# import Pkg
# Pkg.add("JuMP")
# Pkg.add("GLPK")
```

[ ]:
```
using JuMP, GLPK;
```

Создание модели

Инициализируем модель с решателе GLPK:

[ ]:
```
model = Model(GLPK.Optimizer)
```

[ ]:
```
A JuMP Model
├ solver: GLPK
├ objective_sense: FEASIBILITY_SENSE
├ num_variables: 0
├ num_constraints: 0
└ Names registered in the model: none
```

Определение переменных

Доли инвестиций в каждый актив (с ограничениями):

[ ]:
```
@variable(model, MIN_SHARE <= x_A <= MAX_SHARE)  # Доля актива A
@variable(model, MIN_SHARE <= x_B <= MAX_SHARE)  # Доля актива B
@variable(model, MIN_SHARE <= x_C <= MAX_SHARE)  # Доля актива C
```

[ ]:
x_C

Добавление ограничений

Главное условие: полное распределение капитала

[ ]:
```
@constraint(model, sum_constraint, x_A + x_B + x_C == 1)
```

[ ]:
x_A + x_B + x_C = 1

Определение целевой функции

Максимизация ожидаемой доходности:

[ ]:
```
@objective(model, Max, RETURNS[1]*x_A + RETURNS[2]*x_B +␣
↪RETURNS[3]*x_C)
```

[ ]:
$0.08x\_A + 0.12x\_B + 0.06x\_C$

Решение задачи

Запуск оптимизации:

[ ]:
```
optimize!(model)
```

Результаты оптимизации

Выведем оптимальное распределение:

[ ]:
```
println("Оптимальные доли:")
println("A: ", round(value(x_A), digits=3))
println("B: ", round(value(x_B), digits=3))
println("C: ", round(value(x_C), digits=3))
println("\nОжидаемая доходность портфеля: ",␣
  ↪round(objective_value(model)*100, digits=2), "%")
```

```
Оптимальные доли:
A: 0.4
B: 0.5
C: 0.1

Ожидаемая доходность портфеля: 9.8%
```