Introdução ao JuMP.jl

Modelagem para Otimização Matemática em Julia

O que é JuMP.jl?

JuMP (Julia for Mathematical Programming) é uma linguagem de modelagem de código aberto para otimização matemática em Julia.

- **Não é um solver:** JuMP não resolve os problemas diretamente. Ele fornece uma sintaxe clara e de alto nível para *descrever* o problema.
- **Interface com Solvers:** JuMP se conecta a dezenas de solvers de última geração (comerciais e de código aberto) como HiGHS, Gurobi, CPLEX, Ipopt, etc.
- **Sintaxe Algébrica:** Você escreve as equações matemáticas de forma muito parecida com a forma como as escreve no papel.
- **Alto Desempenho:** Por ser construído em Julia, ele se beneficia da velocidade e da capacidade de composição da linguagem.

A Anatomia de um Modelo JuMP

O fluxo de trabalho para resolver qualquer problema em JuMP segue estes passos:

1. **Modelo (`Model`):** Criar um "contêiner" para o seu problema.
2. **Solver:** Anexar a ferramenta que resolverá o problema.
3. **Variáveis (`@variable`):** Declarar as incógnitas do problema.
4. **Objetivo (`@objective`):** Definir a função que você quer maximizar ou minimizar.
5. **Restrições (`@constraint`):** Adicionar as regras, limites e equações que as variáveis devem obedecer.
6. **Otimizar (`optimize!`):** Mandar o solver resolver o problema.
7. **Analisar os Resultados:** Verificar o status da solução, o valor ótimo e os valores das variáveis.

Nosso Exemplo: O Problema do Fazendeiro

Um fazendeiro tem **100 hectares** de terra e quer decidir quanto plantar de **trigo (x)** e **cevada (y)** para maximizar seu lucro.

Dados do Problema:

Cultura	Lucro (€/ha)	Fertilizante (kg/ha)	Pesticida (L/ha)
Trigo (x)	150	20	5
Cevada (y)	220	35	3

Recursos Disponíveis:

- Fertilizante: **2500 kg**
- Pesticida: **400 L**

Nosso Exemplo: Formulação Matemática

O objetivo é encontrar os valores de `x` e `y` que resolvem:

Maximizar (Lucro):

$$150x + 220y$$

Sujeito a (Restrições):

1. **Terra:** $x + y \leq 100$
2. **Fertilizante:** $20x + 35y \leq 2500$
3. **Pesticida:** $5x + 3y \leq 400$
4. **Não-negatividade:** $x, y \geq 0$

Este é um problema clássico de **Programação Linear (PL)**.

Mão na Massa: Implementação em JuMP (1/3)

Vamos traduzir o problema para o código JuMP.

Passo 1: Preparar o ambiente

Primeiro, importamos os pacotes e criamos o modelo, já conectando-o a um solver.

Usaremos o **HiGHS**, um excelente solver de código aberto para PL.

```
# Importa os pacotes necessários
using JuMP
using HiGHS

# 1. Cria um modelo vazio
# 2. Anexa o solver HiGHS
model = Model(HiGHS.Optimizer)
```

Adicionar a formulação matemática (2/3)

```
# 3: Declarar variáveis
@variable(model, x >= 0, base_name = "trigo")
@variable(model, y >= 0, base_name = "cevada")

# 4: Definir a função objetivo
@objective(model, Max, 150x + 220y)

# 5: Adicionar as restrições
@constraint(model, c_terra, x + y <= 100)
@constraint(model, c_fertilizante, 20x + 35y <= 2500)
@constraint(model, c_pesticida, 5x + 3y <= 400)
```

Adicionar a formulação matemática (3/3)

```
# Imprime o modelo para visualização (opcional)
print(model)

# 6: Otimizar
optimize!(model)

# 7: Analisar os resultados
println("--- Resultados da Otimização ---")
println("Status da solução: ", termination_status(model))
println("Lucro Máximo: ", objective_value(model))
println("\n--- Plano de Plantio ---")
println("Hectares de Trigo (x): ", value(x))
println("Hectares de Cevada (y): ", value(y))
```