

Preparação do trabalho (para usa em casa)

- Para o trabalho é necessário instalar Julia: <https://julialang.org>,
- e instalar os pacotes necessários em Julia

```
> <caminho>/bin/julia
julia> import Pkg
julia> Pkg.add("JuMP")
julia> Pkg.add("GLPKMathProgInterface")
julia> Pkg.add("IJulia")
```
- Caso não já está instalado, também é necessário instalar o GNU Linear Programming Kit (GLPK)
 - Windows: Baixar e instalar o GNU Linear Programming Kit (GLPK) em <http://gnuwin32.sourceforge.net/packages/glpk.htm>.
 - Linux: `sudo apt-get install glpk`
- A documentação de Julia/JuMP está disponível em <http://www.juliaopt.org/JuMP.jl/stable>.

Exemplo

Para demonstrar a especificação de um problema nos formatos CPLEX lp e GNU mathprog, considere o exemplo de um importador de Whisky:

Um importador de Whisky tem as seguintes restrições de importação

- no máximo 2000 garrafas de *Johnny Ballantine* por 70 R\$ cada uma,
- no máximo 2500 garrafas de *Old Gargantua* por 50 R\$ cada uma,
- no máximo 1200 garrafas de *Misty Deluxe* por 40 R\$ cada uma.

Dos Whiskies importados ele produz três misturas A , B , C , que ele vende por 68 R\$, 57 R\$ e 45 R\$, respectivamente. As misturas são

- A : no mínimo 60% Johnny Ballantine, no máximo 20% Misty Deluxe,
- B : no mínimo 15% Johnny Ballantine, no máximo 60% Misty Deluxe,
- C : no máximo 50% Misty Deluxe.

Quais seriam as misturas ótimas, e quantas garrafas de cada mistura devem ser produzidas para maximizar o lucro? Formule como programa linear.

Observações:

- Use como variáveis o número de garrafas $x_{m,i}$ da marca m usadas na mistura i .
- Desconsidere a integralidade das garrafas.

Especificação em Julia

<http://nbviewer.jupyter.org/url/www.inf.ufrgs.br/~mrpritt/oc/whisky.ipynb>
<http://www.inf.ufrgs.br/~mrpritt/oc/whisky.jl>

A solução é

```
using JuMP
using GLPK
using Formatting

I=collect(1:3); V=collect(1:3);

m = Model()
set_optimizer(m, GLPK.Optimizer);
@variable(m, x[i in I, v in V] >= 0);

pv=[68, 57, 45];

ci=[70, 50, 40];

@variable(m, y[v in V]) ## total vendas
@variable(m, z[i in I]) ## total importação
@objective(m, Max, sum(pv[v]*y[v] for v in V)-sum(ci[i]*z[i]
    for i in I))

@constraint(m, [v in V], y[v]==sum(x[i,v] for i in I))
@constraint(m, [i in I], z[i]==sum(x[i,v] for v in V));

li=[2000,2500,1200];

@constraint(m, [i in I], z[i] <= li[i]);

@constraint(m,x[1,1] >= 0.6 *y[1]) ## no mínimo 60% Johnny
    Ballantine em A
@constraint(m, x[3,1] <= 0.2 *y[1]) ## no máximo 20% Misty
    Deluxe em A

@constraint(m, x[1,2] >= 0.15*y[2]) ## no mínimo 15% Johnny
    Ballantine em B
@constraint(m, x[3,2] <= 0.6 *y[2]) ## no máximo 60% Misty
    Deluxe em B
```

```
@constraint(m, x[3,3] <= 0.6 *y[3]); ## no máximo 50% Misty
    Deluxe em C

println(m)
optimize!(m)

printfmt("O lucor máximo é {:.2f}.\n",objective_value(m))
printfmt("Importar {:.2f} garrafas de Johnny Ballantine, {:.2f}
    } garrafas de Old Gargantua, e {:.2f} garrafas de Misty
    Deluxe.",value(z[1]),value(z[2]),value(z[3]) )
for v in V
    printfmt("Mistura {}: {:.2f} garrafas.\n","ABC"[v],value(y
        [v]))
    if value(y[v])>0
        printfmt(" {:.2f}% JB, {:.2f}% OG, {:.2f}% MD.\n","ABC"
            [v],value(x[1,v])/value(y[v]),value(x[2,v])/value(y[v]
                ),value(x[3,v])/value(y[v]))
    end
end
end
```

Laboratório 1

Questão 1 (Empresa de vidro)

Formule e resolve o problema da empresa de vidro (questão 1 da primeira lista de exercícios).

- Qual a solução ótima e o seu valor?
- Supõe que temos 2 horas semanais extras. Aumentando as horas semanais de qual planta leva ao maior aumento do lucro?

Questão 2 (Trisabor)

Formule e resolve o problema da empresa “Trisabor” (questão 2 da primeira lista de exercícios).

- Qual a solução e o seu valor?
- Como podemos modificar os lucros, tal que vale a pena vender pães simples?

Questão 3 (Rações)

Resolve o problema do fabricante de rações da primeira lista usando Julia/JuMP.

- a) Formula o problema primeiramente explicitamente, com todos coeficientes.
- b) Depois estuda uma forma mais compacta, usando vetores e matrizes para os dados, e somatórios. Começa por definir os dados. Por exemplo, usando índices $i = 1, 2, 3$ para nutrientes (Cálcio, Proteína, e Carboidratos) e $j = 1, 2, 3$ para ingredientes (Soja, Milho, Cana) podemos definir uma matriz c_{ij} de composição nutritiva por:

```
julia> c=[[0.2 1 3]; [50 9 0]; [0.8 2 2]]  
3×3 Array{Float64,2}:  
 0.2  1.0  3.0  
50.0  9.0  0.0  
 0.8  2.0  2.0
```

Depois re-escreve a função objetivo e as restrições de forma compacta.

Questão 4 (Sorte)

Resolve o problema de investimentos em negócios da primeira lista.

Questão 5 (Atlântico gauchês)

Resolve o problema do hotel “Atlântico gauchês” da primeira lista.

- a) Verifica que o problema não tem solução.
- b) Qual a menor modificação do sistema tal que existe uma solução?

Questão 6 (Refinar óleo)

Resolve da mistura de outros óleos. Procura manter a formulação genérica, tal que ela poderia ser generalizada facilmente para mais óleos.

Questão 7 (Otimizar estudos)

Implemente o modelo de otimizar os estudos e resolve.

Questão 8 (Formulação Matemática)

Resolve o problema dos pipelines.

- a) Qual o fluxo máximo que pode ser transportado de “A” para “B”?
- b) O que acontece se a capacidade de “A” para “Norte” é reduzida para 200?
- c) Qual a menor modificação da rede tal que o valor ótimo é somente 100?

Criando iPython notebooks com Julia A forma mais simples de trabalhar com a Julia é usar um Notebook. O primeiro passo é rodar um servidor para notebooks na máquina local:

```
> <caminho>/bin/julia  
julia> using IJulia  
julia> notebook()
```

Isso abre um navegador no browser.

Um Notebook consiste de uma série de células. Uma célula pode conter texto (markup) ou código. Células com código podem ser executadas diretamente no Notebook e o resultado aparece no Notebook.

Em <http://nbviewer.jupyter.org/url/www.inf.ufrgs.br/~mrpritt/oc/oils.ipynb> tem um notebook visualizado. Vocês podem baixar o Notebook e gravar no diretório do navegador. O Notebook vai aparecer no navegador, e vocês podem abri-lo e rodar. Tem várias possibilidades, por exemplo vocês podem rodar tudo com **Cell >> Run All**, ou célula por célula por clicar em **Run** ou usar **⬆ + Enter**.