Universidade Federal de São Carlos
Departamento de Engenharia de Produção

Otimização Linear Contínua e Discreta

PPGEP, UFSCar - Semestre 01/2025
Prof. Dr. Pedro Munari (munari@dep.ufscar.br)

Tópico 1.6 - Extra: Uso de Softwares de Otimização

Objetivos deste tópico

- Conhecer alguns softwares de otimização e linguagens de modelagem para problemas de programa linear e programação inteira;
- Praticar o uso desses softwares e linguagens na resolução de problemas.

Softwares de otimização

- Existem diversos *softwares* para a resolução de modelos de otimização na prática;

Softwares de otimização

- Existem diversos *softwares* para a resolução de modelos de otimização na prática;
- Dada a complexidade dos algoritmos envolvidos, a maioria dos softwares hoje se divide entre interfaces e *solvers*;

Softwares de otimização

- Existem diversos *softwares* para a resolução de modelos de otimização na prática;
- Dada a complexidade dos algoritmos envolvidos, a maioria dos softwares hoje se divide entre interfaces e *solvers*;
- Interface: Facilita a entrada de dados, escrita do modelo matemático, escolha dos parâmetros e configurações dos *solvers*; Permite troca de *solvers* facilmente. [Obs.: não necessariamente é interface gráfica; pode ser um pacote/biblioteca]

Softwares de otimização

- Existem diversos *softwares* para a resolução de modelos de otimização na prática;
- Dada a complexidade dos algoritmos envolvidos, a maioria dos softwares hoje se divide entre interfaces e *solvers*;
- Interface: Facilita a entrada de dados, escrita do modelo matemático, escolha dos parâmetros e configurações dos *solvers*; Permite troca de *solvers* facilmente. [Obs.: não necessariamente é interface gráfica; pode ser um pacote/biblioteca]
- *Solver*: É o que resolve o modelo matemático de fato, por meio de implementações dos algoritmos como simplex, *branch-and-bound*, *branch-and-cut*, e muitos outros

Softwares de otimização

▷ Exemplos de interfaces e *solvers*

Interface
Definição dos dados, modelo matemático e parâmetros.

Solver
Implementação dos algoritmos de otimização (simplex, branch-and-bound, etc.)

Softwares de otimização

- Exemplos de interface:
  - Excel, AMPL, GAMS, PIFOP, Pyomo (Python);
- Exemplos de *solver*:
  - CPLEX (IBM), Gurobi: comerciais ($$$) e os mais eficientes;
  - GLPK, COIN-OR e SCIP: livres, gratuitos e podem servir de alternativa aos softwares comerciais (são mais lentos, em geral).

Softwares de otimização

- Exemplos de interface:
  - Excel, AMPL, GAMS, PIFOP, Pyomo (Python);
- Exemplos de *solver*:
  - CPLEX (IBM), Gurobi: comerciais ($$$) e os mais eficientes;
  - GLPK, COIN-OR e SCIP: livres, gratuitos e podem servir de alternativa aos softwares comerciais (são mais lentos, em geral).
- Há muitos outros disponíveis;

Softwares de otimização

- Exemplos de interface:
  - Excel, AMPL, GAMS, PIFOP, Pyomo (Python);
- Exemplos de *solver*:
  - CPLEX (IBM), Gurobi: comerciais ($$$) e os mais eficientes;
  - GLPK, COIN-OR e SCIP: livres, gratuitos e podem servir de alternativa aos softwares comerciais (são mais lentos, em geral).
- Há muitos outros disponíveis;
- É possível interagir diretamente com os *solvers*, mas é mais complicado (em geral); assim como algumas interfaces tem seus próprios *solvers*, mas menos eficientes (em geral);

Softwares de otimização

- Exemplos de interface:
  - Excel, AMPL, GAMS, PIFOP, Pyomo (Python);
- Exemplos de *solver*:
  - CPLEX (IBM), Gurobi: comerciais ($$$) e os mais eficientes;
  - GLPK, COIN-OR e SCIP: livres, gratuitos e podem servir de alternativa aos softwares comerciais (são mais lentos, em geral).
- Há muitos outros disponíveis;
- É possível interagir diretamente com os *solvers*, mas é mais complicado (em geral); assim como algumas interfaces tem seus próprios *solvers*, mas menos eficientes (em geral);
- No Microsoft Excel, temos os suplemento *solver* nativo (limitado), mas há suplementos mais eficientes (*OpenSolver* e *SolverStudio*) que invocam esses outros *solvers* mencionados acima.

Softwares de otimização

- Em geral, os *softwares* usam o método simplex para a resolução dos problemas de programação linear e variações do método branch-and-bound para problemas de programação inteira.

Softwares de otimização

- Em geral, os *softwares* usam o método simplex para a resolução dos problemas de programação linear e variações do método branch-and-bound para problemas de programação inteira.
- Para iniciantes, temos dois exemplos bem simples:

Softwares de otimização

- Em geral, os *softwares* usam o método simplex para a resolução dos problemas de programação linear e variações do método branch-and-bound para problemas de programação inteira.
- Para iniciantes, temos dois exemplos bem simples:
  - Excel (Microsoft) – conta com o Solver (nativo) e outros suplementos mais eficientes (OpenSolver e SolverStudio);

Softwares de otimização

- Em geral, os *softwares* usam o método simplex para a resolução dos problemas de programação linear e variações do método branch-and-bound para problemas de programação inteira.
- Para iniciantes, temos dois exemplos bem simples:
  - Excel (Microsoft) – conta com o Solver (nativo) e outros suplementos mais eficientes (OpenSolver e SolverStudio);
  - `lp_solve` – simples, didático, livre e gratuito. Entrada de dados no formato `.lp`, amplamente conhecido e utilizado (mas não conveniente para problemas de grande porte).

Softwares de otimização

- Em geral, os *softwares* usam o método simplex para a resolução dos problemas de programação linear e variações do método branch-and-bound para problemas de programação inteira.
- Para iniciantes, temos dois exemplos bem simples:
  - Excel (Microsoft) – conta com o Solver (nativo) e outros suplementos mais eficientes (OpenSolver e SolverStudio);
  - `lp_solve` – simples, didático, livre e gratuito. Entrada de dados no formato `.lp`, amplamente conhecido e utilizado (mas não conveniente para problemas de grande porte).
- Para projetos mais avançados:

Softwares de otimização

- Em geral, os *softwares* usam o método simplex para a resolução dos problemas de programação linear e variações do método branch-and-bound para problemas de programação inteira.
- Para iniciantes, temos dois exemplos bem simples:
  - Excel (Microsoft) – conta com o Solver (nativo) e outros suplementos mais eficientes (OpenSolver e SolverStudio);
  - `lp_solve` – simples, didático, livre e gratuito. Entrada de dados no formato `.lp`, amplamente conhecido e utilizado (mas não conveniente para problemas de grande porte).
- Para projetos mais avançados:
  - Linguagem de modelagem algébrica (p.ex. AMPL, GAMS) ou bibliotecas em linguagem de programação (p.ex. Pyomo, Gurobipy, CPLEX Concert);

Softwares de otimização

- Em geral, os *softwares* usam o método simplex para a resolução dos problemas de programação linear e variações do método branch-and-bound para problemas de programação inteira.
- Para iniciantes, temos dois exemplos bem simples:
  - Excel (Microsoft) – conta com o Solver (nativo) e outros suplementos mais eficientes (OpenSolver e SolverStudio);
  - `lp_solve` – simples, didático, livre e gratuito. Entrada de dados no formato `.lp`, amplamente conhecido e utilizado (mas não conveniente para problemas de grande porte).
- Para projetos mais avançados:
  - Linguagem de modelagem algébrica (p.ex. AMPL, GAMS) ou bibliotecas em linguagem de programação (p.ex. Pyomo, Gurobipy, CPLEX Concert);
  - Uso de *solvers* mais robustos: CPLEX, Gurobi, HiGHS, Coin-OR, etc.

Softwares de otimização: iniciantes

- O Solver/Excel tem como principais vantagens:
  - Estar disponível em quase todo computador;

Softwares de otimização: iniciantes

- O Solver/Excel tem como principais vantagens:
  - Estar disponível em quase todo computador;
  - Entrada de dados por uma planilha do Excel.

Softwares de otimização: iniciantes

- O Solver/Excel tem como principais vantagens:
  - Estar disponível em quase todo computador;
  - Entrada de dados por uma planilha do Excel.
- Por outro lado, além limitar o número máximo de variáveis e restrições, pode ser ineficiente até mesmo em problemas relativamente pequenos.

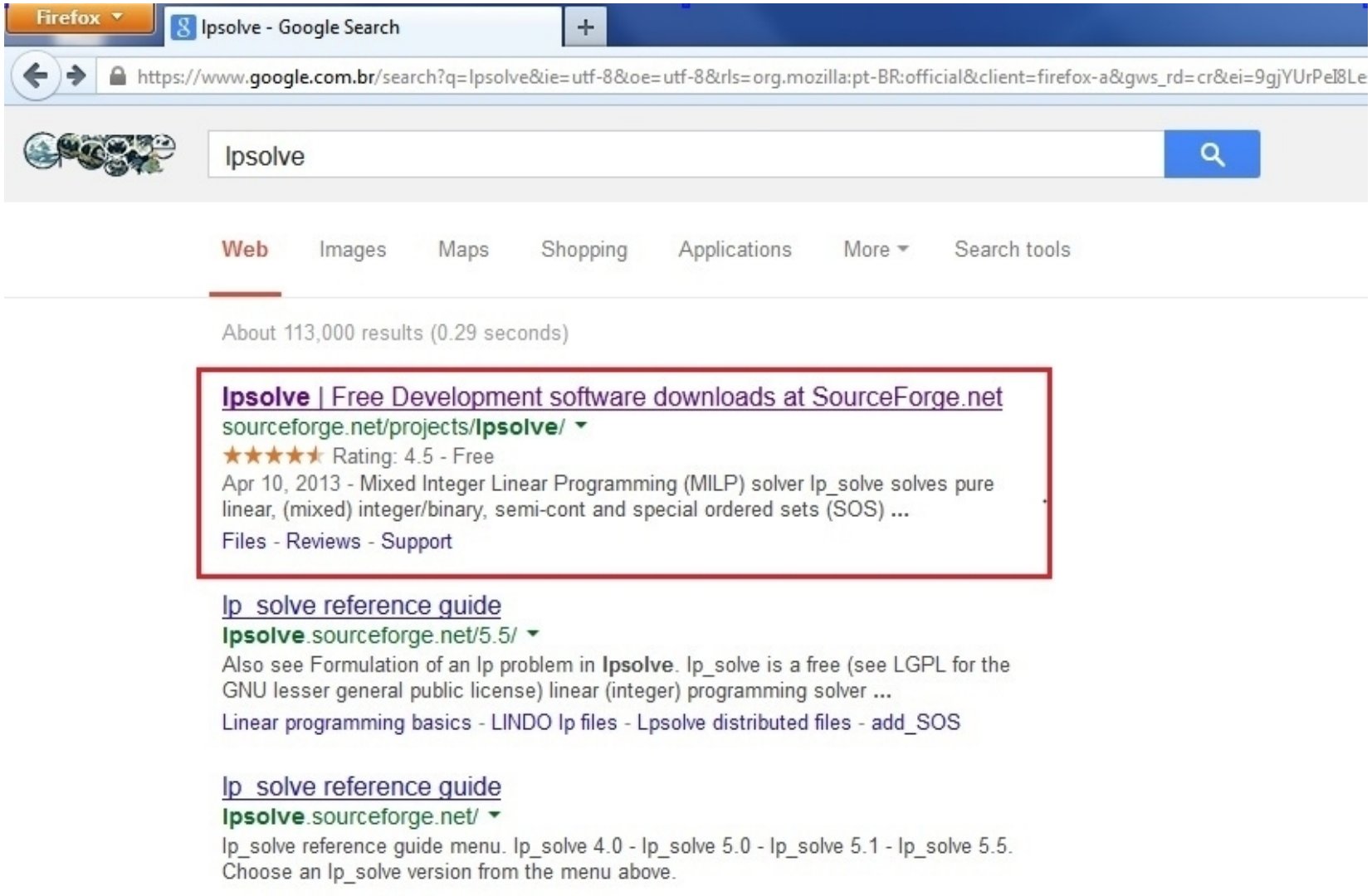
Softwares de otimização: iniciantes

- O Solver/Excel tem como principais vantagens:
  - Estar disponível em quase todo computador;
  - Entrada de dados por uma planilha do Excel.
- Por outro lado, além limitar o número máximo de variáveis e restrições, pode ser ineficiente até mesmo em problemas relativamente pequenos.
- O `lp_solve` é uma boa alternativa que permite a entrada de dados de forma bastante intuitiva;

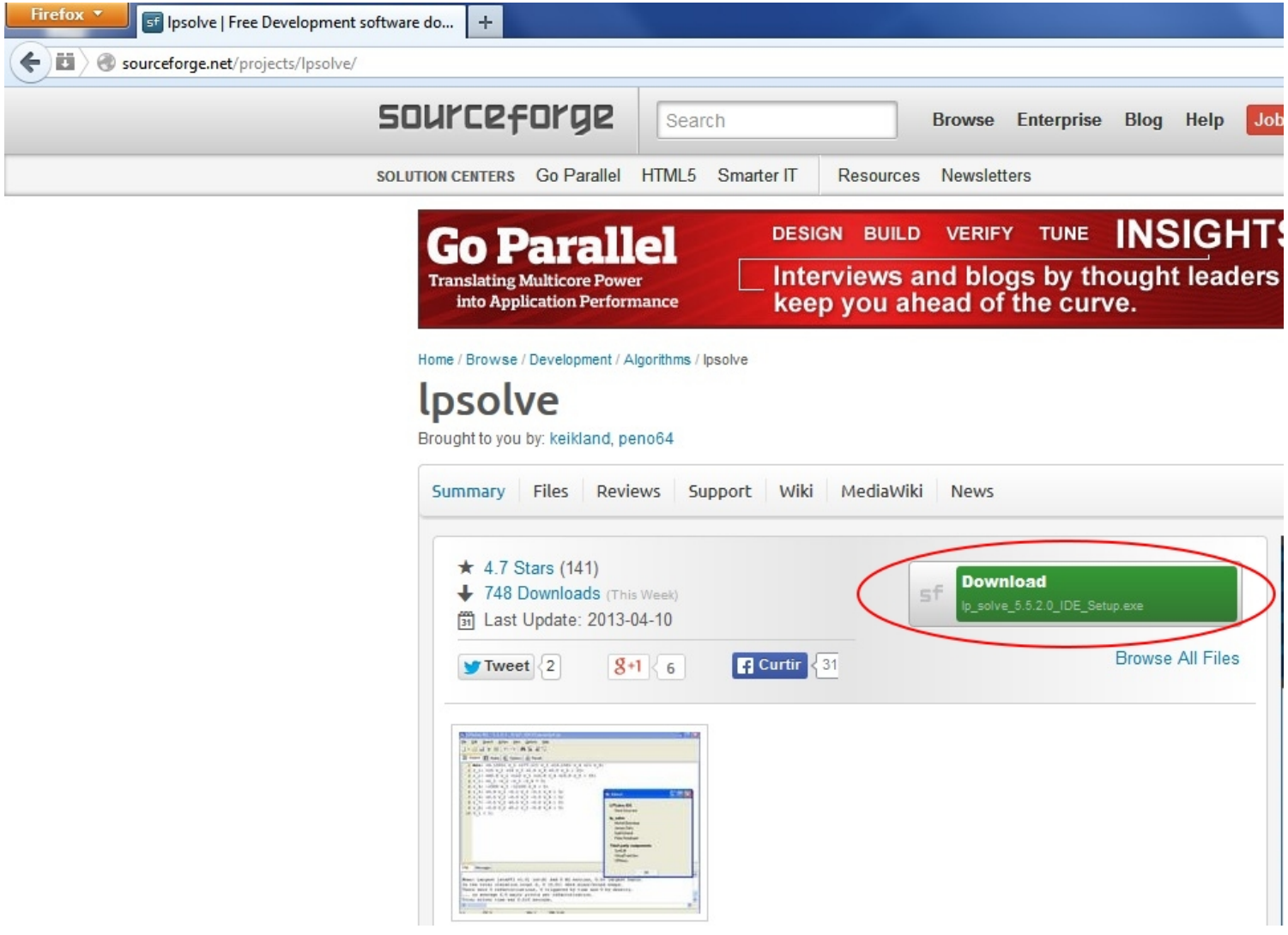
Softwares de otimização: iniciantes

- O Solver/Excel tem como principais vantagens:
  - Estar disponível em quase todo computador;
  - Entrada de dados por uma planilha do Excel.
- Por outro lado, além limitar o número máximo de variáveis e restrições, pode ser ineficiente até mesmo em problemas relativamente pequenos.
- O `lp_solve` é uma boa alternativa que permite a entrada de dados de forma bastante intuitiva;
  - Tem caráter mais didático, usado na resolução de problemas de pequeno porte.

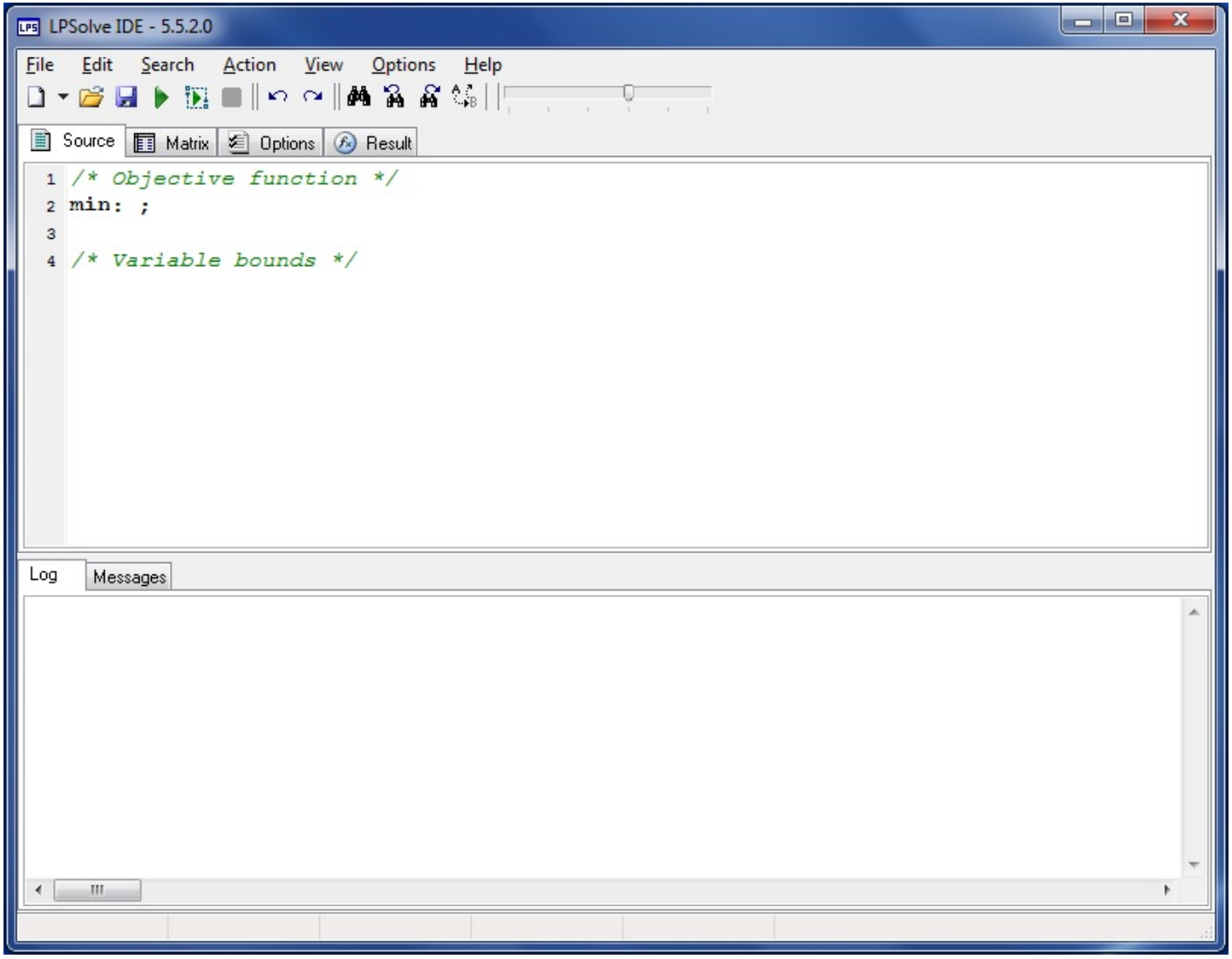
Para baixar e instalar o `lp_solve`:



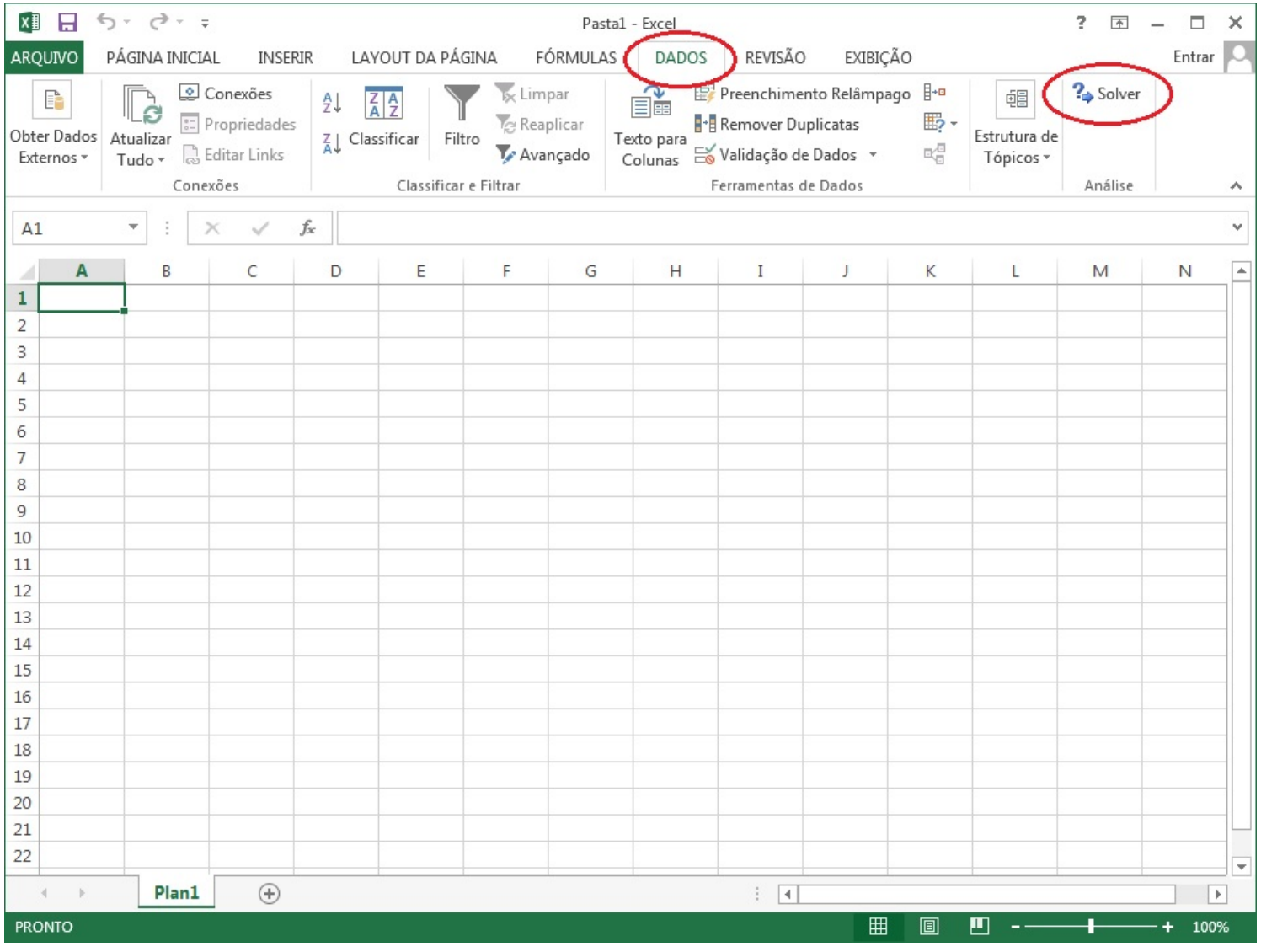
Para baixar e instalar o `lp_solve`:



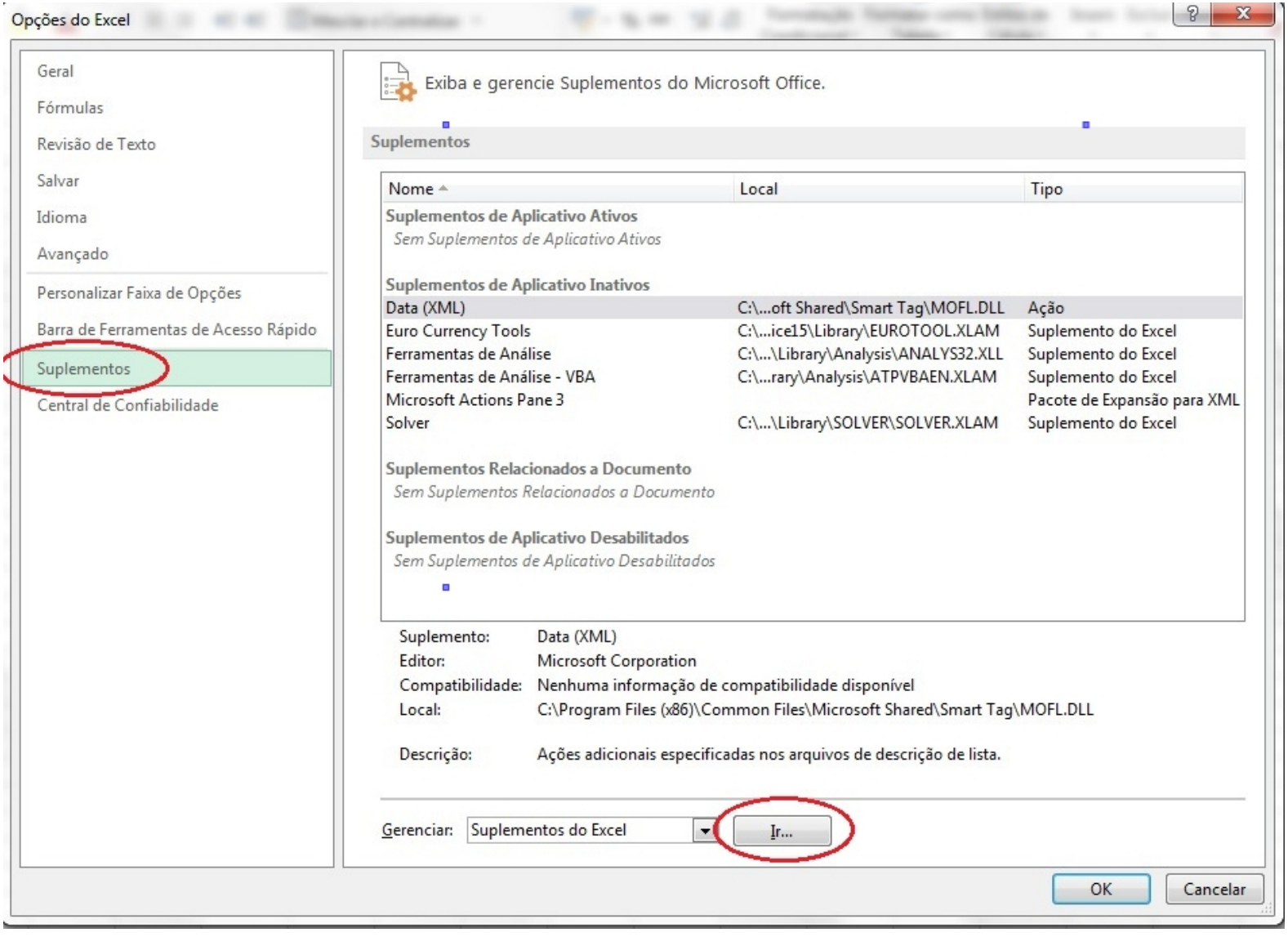
Tela inicial `lp_solve`:



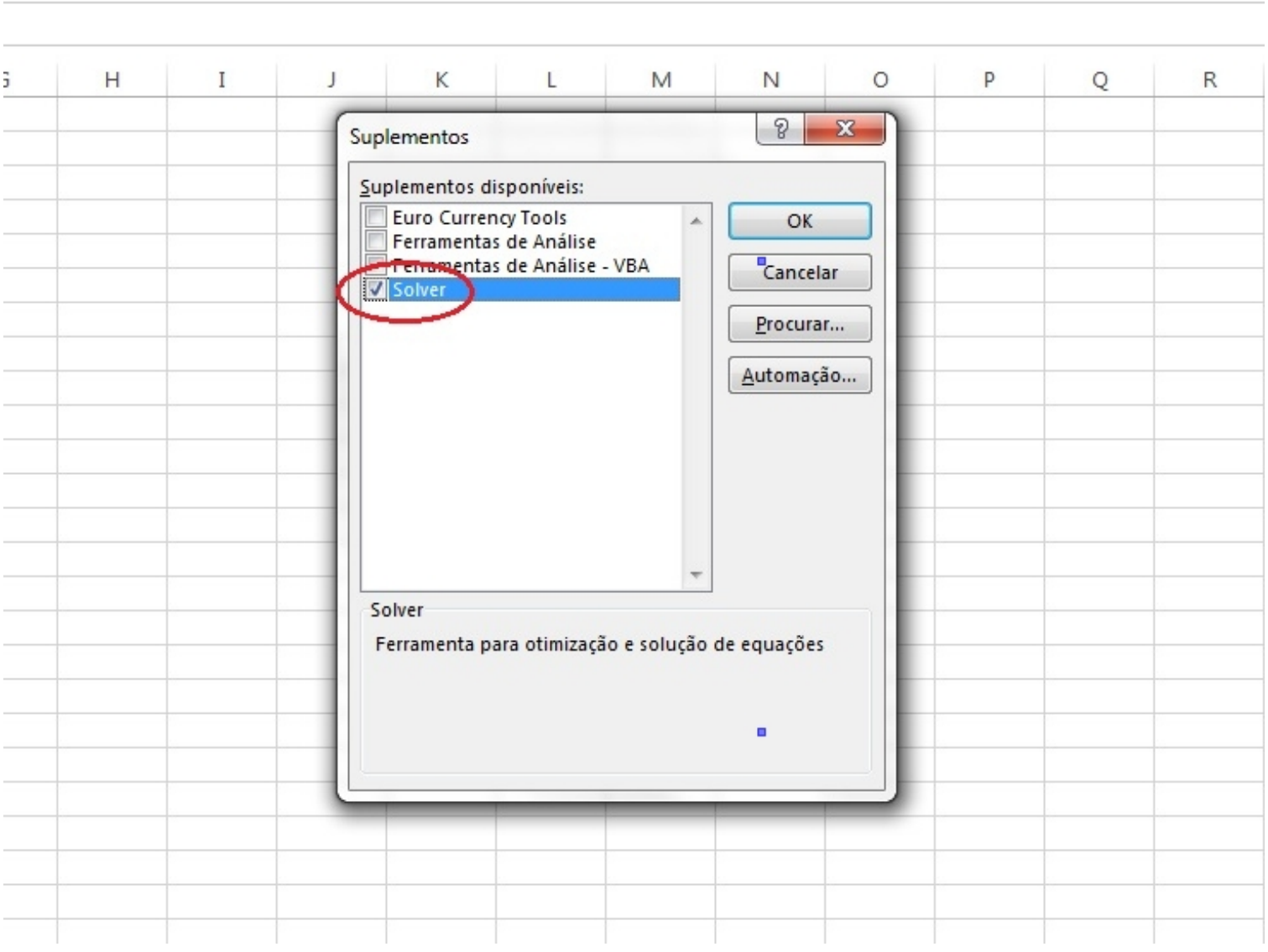
Para usar o Solver/Excel:



Caso não encontre, vá em Arquivo ≫ Opções ≫ Suplementos ≫ Ir:



Selecione a opção Solver e clique em OK:



Linguagem de modelagem algébrica

Linguagem de modelagem algébrica

- Para lidar com modelos mais complexos e que envolvam grande quantidade de dados, é essencial usar alguma linguagem de modelagem algébrica;

Linguagem de modelagem algébrica

- Para lidar com modelos mais complexos e que envolvam grande quantidade de dados, é essencial usar alguma linguagem de modelagem algébrica;
- Permitem entrar com o modelo na forma algébrica e, muitas vezes, trocar de *solver* facilmente;

Linguagem de modelagem algébrica

- Para lidar com modelos mais complexos e que envolvam grande quantidade de dados, é essencial usar alguma linguagem de modelagem algébrica;
- Permitem entrar com o modelo na forma algébrica e, muitas vezes, trocar de *solver* facilmente;
- Exemplos de linguagens: AMPL, GMPL, GAMS, Pyomo, etc.

Linguagem de modelagem algébrica

- Para lidar com modelos mais complexos e que envolvam grande quantidade de dados, é essencial usar alguma linguagem de modelagem algébrica;
- Permitem entrar com o modelo na forma algébrica e, muitas vezes, trocar de *solver* facilmente;
- Exemplos de linguagens: AMPL, GMPL, GAMS, Pyomo, etc.
- Algumas precisam ser utilizadas dentro de um software de modelagem que tipicamente leva o mesmo nome da linguagem;

Linguagem de modelagem algébrica

- Para lidar com modelos mais complexos e que envolvam grande quantidade de dados, é essencial usar alguma linguagem de modelagem algébrica;
- Permitem entrar com o modelo na forma algébrica e, muitas vezes, trocar de *solver* facilmente;
- Exemplos de linguagens: AMPL, GMPL, GAMS, Pyomo, etc.
- Algumas precisam ser utilizadas dentro de um software de modelagem que tipicamente leva o mesmo nome da linguagem;
  - Por exemplo, AMPL e GAMS são usadas no software de mesmo nome para modelagem e resolução do problema. Esses softwares **não** resolvem o problema, servindo apenas como uma interface que chama um *solver* (p.ex., CPLEX) para a resolução de fato.

Linguagem de modelagem algébrica

- *A Mathematical Programming Language* (AMPL)

Linguagem de modelagem algébrica

▷ *A Mathematical Programming Language* (AMPL)

- AMPL é uma linguagem de modelagem algébrica dedicada a problemas de otimização, que pode ser usada em um software **comercial** que recebe o mesmo nome: `http://ampl.com/`

Linguagem de modelagem algébrica

▷ *A Mathematical Programming Language* (AMPL)

- AMPL é uma linguagem de modelagem algébrica dedicada a problemas de otimização, que pode ser usada em um software **comercial** que recebe o mesmo nome: `http://ampl.com/`
- Requer a aquisição de uma licença paga para uso comercial, mas permite o uso de licenças gratuitas, para alguns usos;

Linguagem de modelagem algébrica

▷ *A Mathematical Programming Language* (AMPL)

- AMPL é uma linguagem de modelagem algébrica dedicada a problemas de otimização, que pode ser usada em um software **comercial** que recebe o mesmo nome: `http://ampl.com/`
- Requer a aquisição de uma licença paga para uso comercial, mas permite o uso de licenças gratuitas, para alguns usos;
- É possível usar uma interface própria para edição e resolução, ou usá-la junto com o Python, Colab e outras linguagens;

Linguagem de modelagem algébrica

▷ *A Mathematical Programming Language* (AMPL)

- AMPL é uma linguagem de modelagem algébrica dedicada a problemas de otimização, que pode ser usada em um software **comercial** que recebe o mesmo nome: `http://ampl.com/`
- Requer a aquisição de uma licença paga para uso comercial, mas permite o uso de licenças gratuitas, para alguns usos;
- É possível usar uma interface própria para edição e resolução, ou usá-la junto com o Python, Colab e outras linguagens;
- Possui sintaxe intuitiva e permite separar o modelo e os dados em arquivos diferentes;

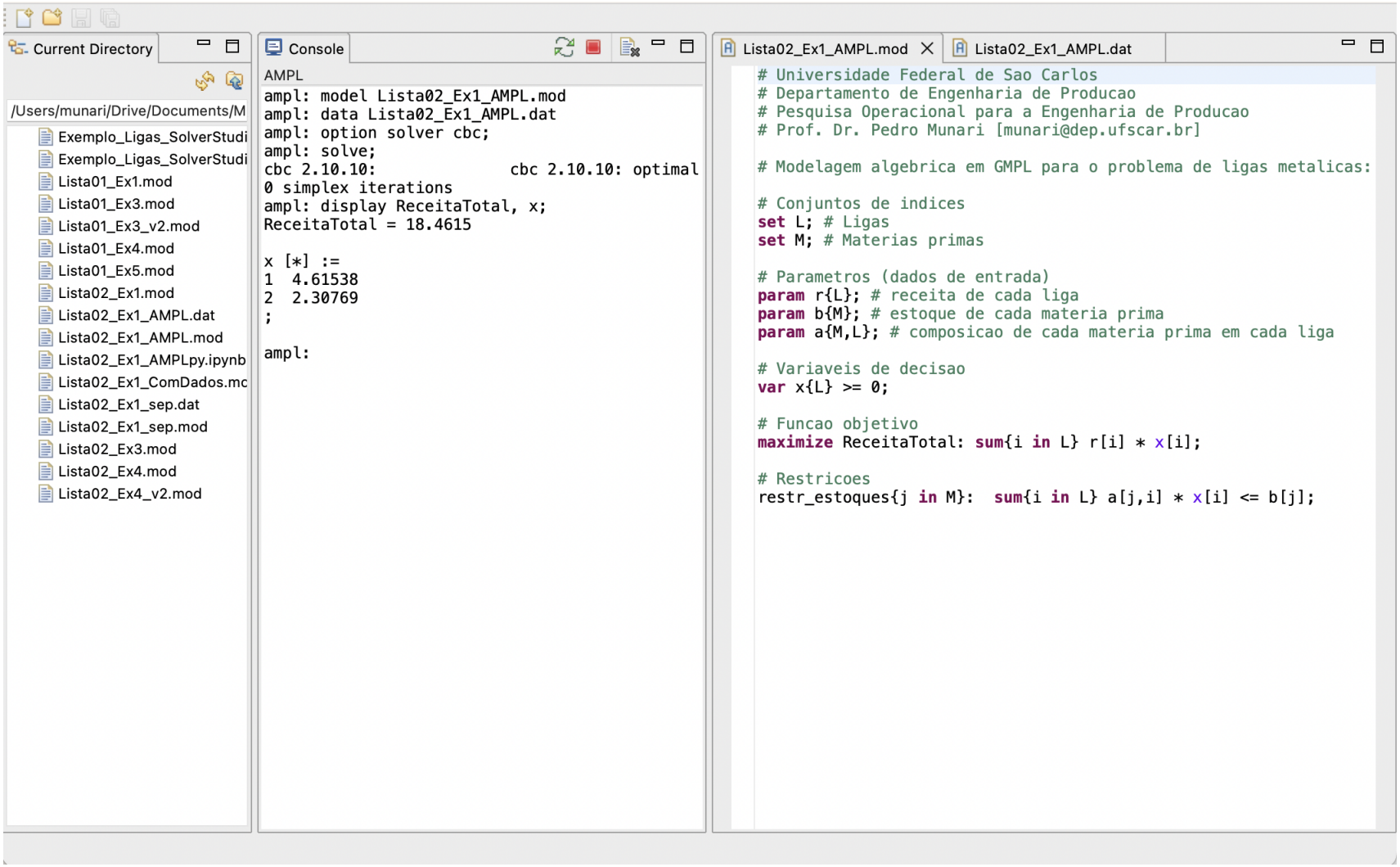
Linguagem de modelagem algébrica

▷ *A Mathematical Programming Language* (AMPL)

- AMPL é uma linguagem de modelagem algébrica dedicada a problemas de otimização, que pode ser usada em um software **comercial** que recebe o mesmo nome: `http://ampl.com/`
- Requer a aquisição de uma licença paga para uso comercial, mas permite o uso de licenças gratuitas, para alguns usos;
- É possível usar uma interface própria para edição e resolução, ou usá-la junto com o Python, Colab e outras linguagens;
- Possui sintaxe intuitiva e permite separar o modelo e os dados em arquivos diferentes;
- Viabiliza o uso de diversos solvers comerciais e livres (p.ex., CPLEX, Gurobi, Coin-OR, HIGHs).

Linguagem de modelagem algébrica

- *A Mathematical Programming Language* (AMPL)



Linguagem de modelagem algébrica

▷ http://ampl.com/python/

AMPL Products Solutions Documentation Resources Pricing Start free now Book a discovery call →

Experience the All-New Python Ecosystem & Solvers for Large-Scale Optimization

Natural mathematical modeling syntax + natural python integration.

Start now > Read the docs >

Integrate with amplpy

Our amplpy interface allows developers to access the features of AMPL from within Python.

Read the docs >

Start free now >

Collaborate with teams

AMPL Model Colaboratory is a collection of AMPL models in Jupyter Notebooks that run on platforms such as **Google Colab**, **Kaggle**, **Gradient**, and **AWS SageMaker**.

AMPL Colaboratory >

Easy to learn and teach optimization

Bring optimization to your courses easily with models on Google Colab, or a free, easily distributed AMPL for Courses license.

AMPL for Courses >

Deploy to your larger applications

Deployment is easy with solvers now available as python packages. Deploy in the cloud using Docker containers, or with options such as cloud functions (e.g. AWS Lambda and Azure Functions)

Linguagem de modelagem algébrica

▷ AMPLpy

[6]:

```
%%ampl_eval # Essa flag é para sinalizar que vamos colocar um código que deve ser processado pelo AMPL

reset; # Redefine o ambiente do AMPL
option solver cplex;# Escolhe o solver que vamos usar

# A partir deste ponto, vamos colocar o modelo algébrico, os dados, e os comandos para resolver e mostrar a solução

# Conjuntos de indices
set L; # Ligas
set M; # Materias primas

# Parametros (dados de entrada)
param r{L}; # receita de cada liga
param b{M}; # estoque de cada materia prima
param a{M,L}; # composicao de cada materia prima em cada liga

# Variaveis de decisao
var x{L} >= 0;

# Funcao objetivo
maximize ReceitaTotal: sum{i in L} r[i] * x[i];

# Restricoes
restr_estoques{j in M}:  sum{i in L} a[j,i] * x[i] <= b[j];

data;

set L := 1 2;
set M := Cobre Zinco Chumbo;

param: r :=
    1   3
    2   2;
```

Linguagem de modelagem algébrica

- *GNU Mathematical Programming Language* (GMPL)

Linguagem de modelagem algébrica

▷ *GNU Mathematical Programming Language* (GMPL)

- GMPL é um linguagem de modelagem algébrica da GNU para interação com o solver GLPK (*GNU Linear Programming Kit*);
- GMPL é um subconjunto da AMPL, mas é livre e gratuita; GMPL e AMPL são compatíveis na grande maioria dos comandos e, nesses casos, podem ser usadas nos mesmos softwares;

Linguagem de modelagem algébrica

▷ *GNU Mathematical Programming Language* (GMPL)

- GMPL é um linguagem de modelagem algébrica da GNU para interação com o solver GLPK (*GNU Linear Programming Kit*);
- GMPL é um subconjunto da AMPL, mas é livre e gratuita; GMPL e AMPL são compatíveis na grande maioria dos comandos e, nesses casos, podem ser usadas nos mesmos softwares;
- Pode ser usada por meio do software livre `Gusek` (gratuito): `http://gusek.sourceforge.net/`

AMPL/GMPL

- Gusek: http://gusek.sourceforge.net/

File Edit Search View Tools Options Language Buffers Help

1 GMPL_Ligas2.mod | 2 GMPL_Ligas2.dat

```
1  # Universidade Federal de Sao Carlos
2  # Departamento de Engenharia de Produção
3  # Pesquisa Operacional para a Engenharia de Produção
4  # Prof. Dr. Pedro Munari [munari@dep.ufscar.br]
5
6  # Modelagem algébrica em GMPL para o problema de ligas metalicas : MODELO
7
8  # Conjuntos de indices
9  set L; # Ligas
10 set M; # Materias primas
11
12 # Parametros (dados de entrada)
13 param r{L}; # receita de cada liga
14 param b{M}; # estoque de cada materia prima
15 param a{M, L}; # composicao de cada materia prima em cada liga
16
17 # Variaveis de decisao
18 var x{L} >= 0;
19
20 # Funcao objetivo
21 maximize ReceitaTotal: sum{i in L} r[i] * x[i];
22
23 # Restricoes
24 restr_estoques{j in M}: sum{i in L} a[j,i] * x[i] <= b[j];
25
26 # Resolver o problema
27 solve;
28
29 # Escrever valor otimo e solucao otima
30 display ReceitaTotal, x;
```

```
>C:\Drive\Documents\MyCourses\Pesquisa Operacional para Eng
GLPSOL: GLPK LP/MIP Solver, v4.60
Parameter(s) specified in the command line:
 --cover --clique --gomory --mir -m GMPL_Ligas2.mod -d GMPL
Reading model section from GMPL_Ligas2.mod...
33 lines were read
Reading data section from GMPL_Ligas2.dat...
30 lines were read
Generating ReceitaTotal...
Generating restr_estoques...
Model has been successfully generated
GLPK Simplex Optimizer, v4.60
4 rows, 2 columns, 8 non-zeros
Preprocessing...
3 rows, 2 columns, 6 non-zeros
Scaling...
 A: min|aij| =  1.000e-01  max|aij| =  5.000e-01  ratio =
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part is 3
*     0: obj =  -0.000000000e+00 inf =   0.000e+00 (2)
*     2: obj =   1.846153846e+01 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.1 Mb (120672 bytes)
Display statement at line 30
ReceitaTotal.val = 18.4615384615385
x[1].val = 4.61538461538462
x[2].val = 2.30769230769231
Model has been successfully processed
>Exit code: 0    Time: 0.223
```

Linguagem de modelagem algébrica

▷ *GNU Mathematical Programming Language* (GMPL)

- GMPL é um linguagem de modelagem algébrica da GNU para interação com o solver GLPK (*GNU Linear Programming Kit*);
- GMPL é um subconjunto da AMPL, mas é livre e gratuita; GMPL e AMPL são compatíveis na grande maioria dos comandos e, nesses casos, podem ser usadas nos mesmos softwares;
- Pode ser usada por meio do software livre `Gusek` (gratuito): `http://gusek.sourceforge.net/`

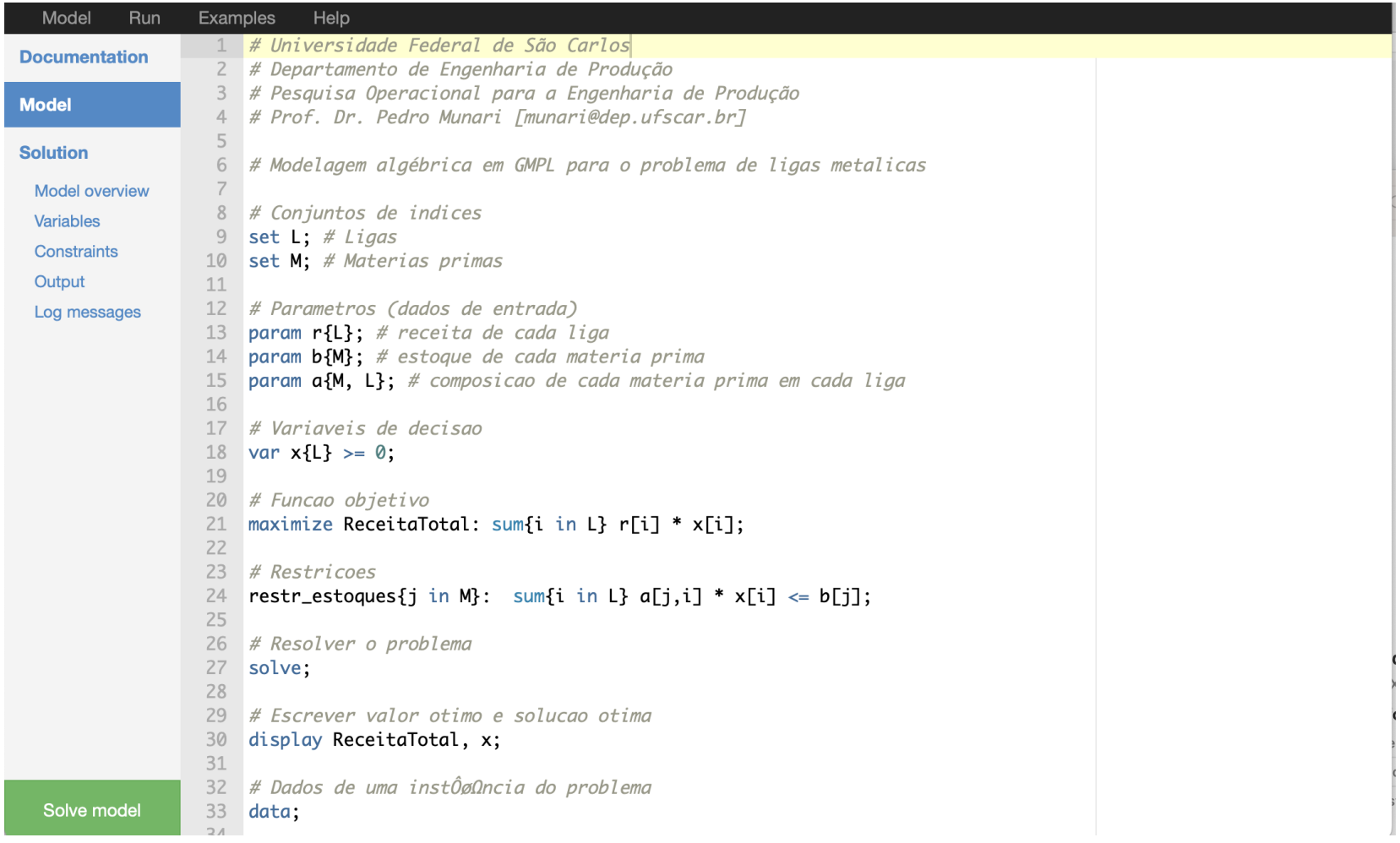
Linguagem de modelagem algébrica

▷ *GNU Mathematical Programming Language* (GMPL)

- GMPL é um linguagem de modelagem algébrica da GNU para interação com o solver GLPK (*GNU Linear Programming Kit*);
- GMPL é um subconjunto da AMPL, mas é livre e gratuita; GMPL e AMPL são compatíveis na grande maioria dos comandos e, nesses casos, podem ser usadas nos mesmos softwares;
- Pode ser usada por meio do software livre `Gusek` (gratuito): `http://gusek.sourceforge.net/`
- Pelo Online-Optimizer (gratuito): `https://online-optimizer.appspot.com/`

AMPL/GMPL

- Online-Optimizer: `https://online-optimizer.appspot.com/`



Linguagem de modelagem algébrica

▷ *GNU Mathematical Programming Language* (GMPL)

- GMPL é um linguagem de modelagem algébrica da GNU para interação com o solver GLPK (*GNU Linear Programming Kit*);
- GMPL é um subconjunto da AMPL, mas é livre e gratuita; GMPL e AMPL são compatíveis na grande maioria dos comandos e, nesses casos, podem ser usadas nos mesmos softwares;
- Pode ser usada por meio do software livre `Gusek` (gratuito): `http://gusek.sourceforge.net/`
- Pelo Online-Optimizer (gratuito): `https://online-optimizer.appspot.com/`

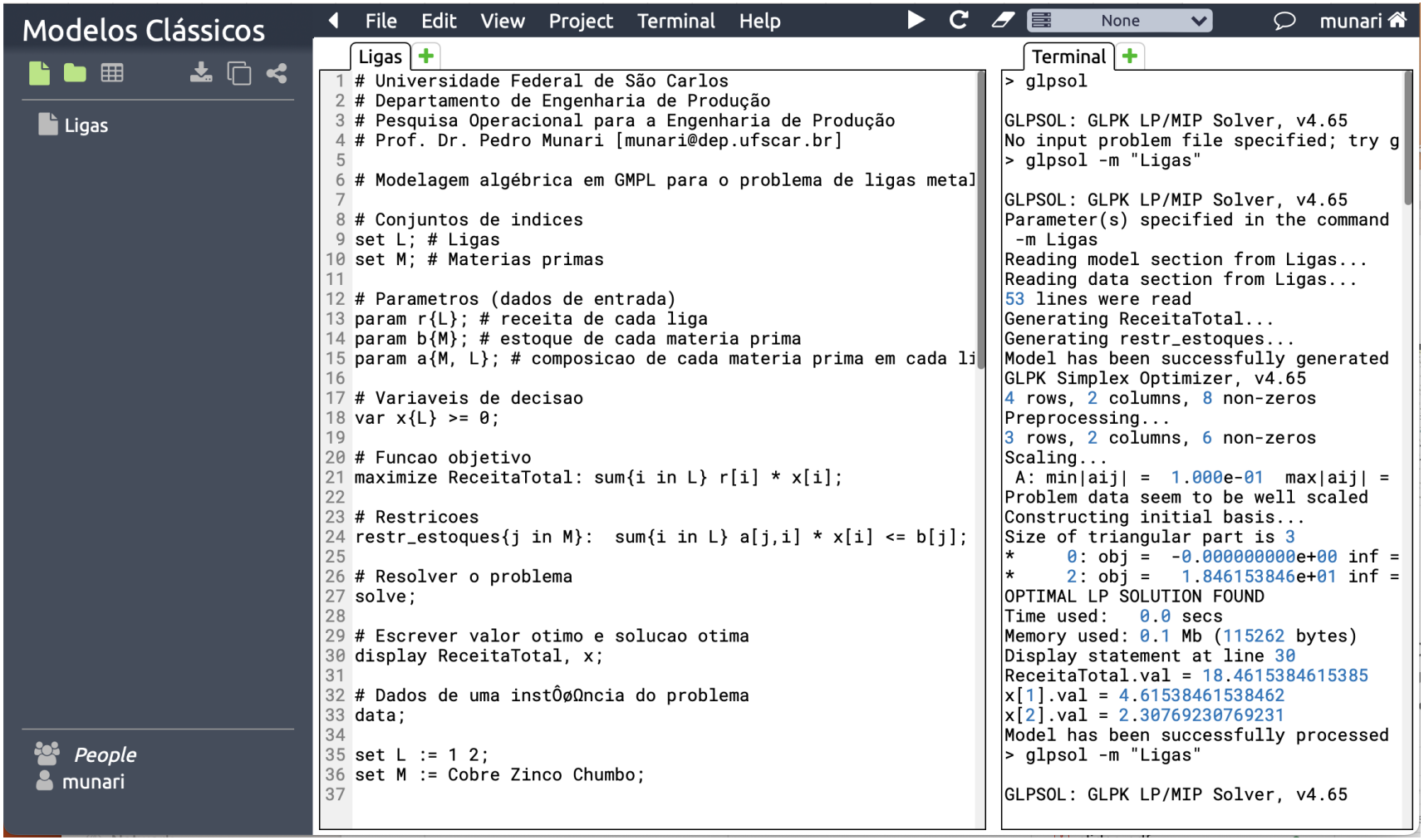
Linguagem de modelagem algébrica

▷ *GNU Mathematical Programming Language* (GMPL)

- GMPL é um linguagem de modelagem algébrica da GNU para interação com o solver GLPK (*GNU Linear Programming Kit*);
- GMPL é um subconjunto da AMPL, mas é livre e gratuita; GMPL e AMPL são compatíveis na grande maioria dos comandos e, nesses casos, podem ser usadas nos mesmos softwares;
- Pode ser usada por meio do software livre `Gusek` (gratuito): `http://gusek.sourceforge.net/`
- Pelo Online-Optimizer (gratuito): `https://online-optimizer.appspot.com/`
- E pelo PIFOP (gratuito): `https://pifop.com/`;

AMPL/GMPL

- PIFOP: https://pifop.com/



AMPL/GMPL

▷ Exemplo

Uma metalúrgica produz dois tipos de ligas metálicas. Cada liga é composta de proporções diferentes de cobre, zinco e chumbo, os quais estão disponíveis em quantidades limitadas em estoque. Deseja-se determinar quanto produzir de cada liga metálica, de modo a maximizar a receita bruta, satisfazendo-se as seguintes composições das ligas e a disponibilidade de matéria-prima em estoque:

Matéria-prima	Liga 1	Liga 2	Estoque
Cobre	50%	30%	3 ton
Zinco	10%	20%	1 ton
Chumbo	40%	50%	3 ton
Preço venda	3 mil	2 mil	(R$ por ton)

AMPL/GMPL

- Modelo com dados

$$
\begin{aligned}
\text{max} \quad & 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 0, x_2 \geq 0
\end{aligned}
$$

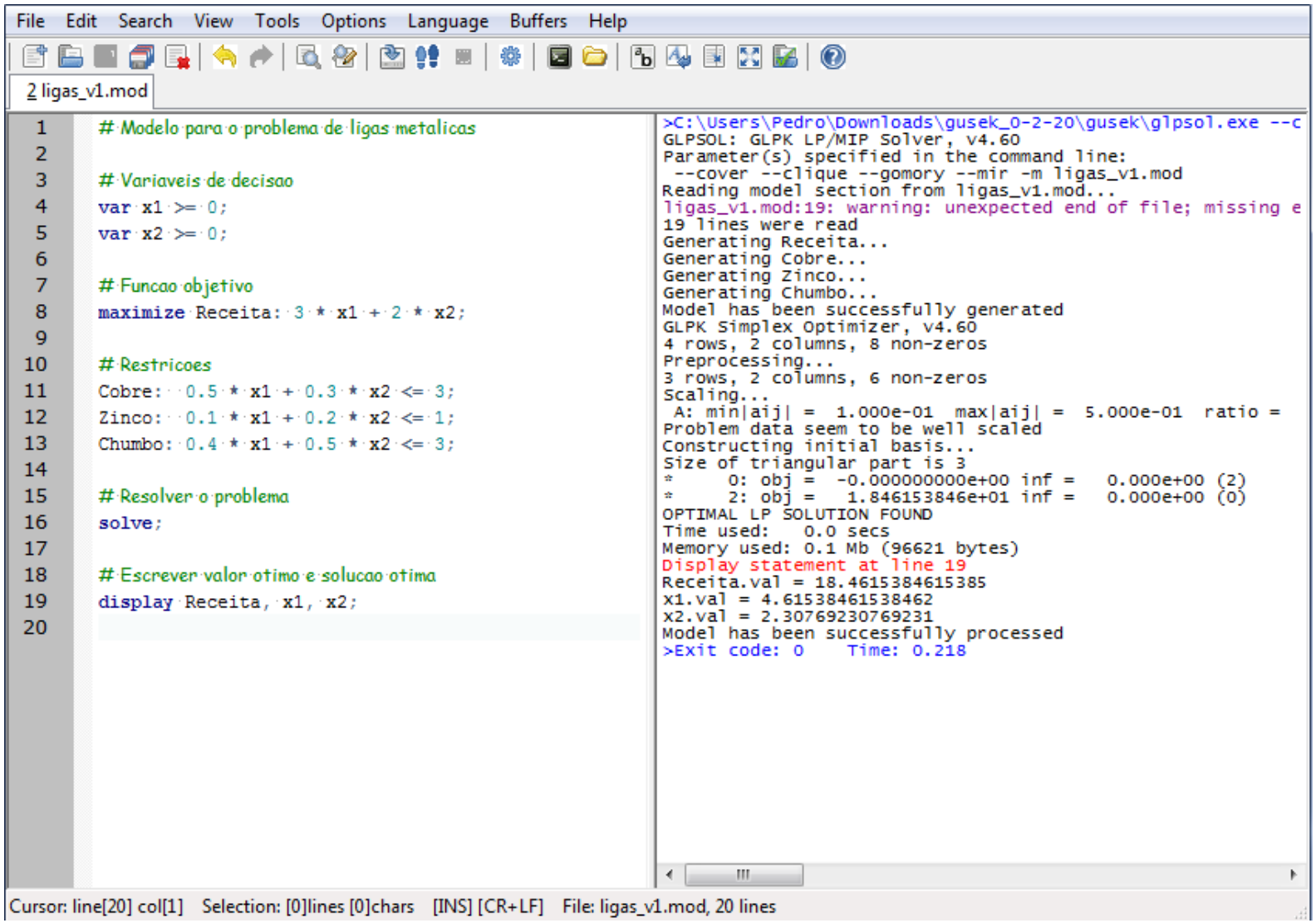
AMPL/GMPL

- Modelo com dados (evitar!)

```
# Variaveis de decisao
var x1 >= 0;
var x2 >= 0;

# Funcao objetivo
maximize Receita: 3 * x1 + 2 * x2;

# Restricoes
Cobre:  0.5 * x1 + 0.3 * x2 <= 3;
Zinco:  0.1 * x1 + 0.2 * x2 <= 1;
Chumbo: 0.4 * x1 + 0.5 * x2 <= 3;
```

AMPL/GMPL

- Gusek



AMPL/GMPL

▷ Exemplo

- A modelagem anterior explicita os dados diretamente no modelo, semelhante ao que é feito, por exemplo, no `lp_solve`;

AMPL/GMPL

▷ Exemplo

- A modelagem anterior explicita os dados diretamente no modelo, semelhante ao que é feito, por exemplo, no `lp_solve`;
- Embora as linguagens algébricas permitam isso, esta **não** é a forma ideal de usá-las;

AMPL/GMPL

▷ Exemplo

- A modelagem anterior explicita os dados diretamente no modelo, semelhante ao que é feito, por exemplo, no `lp_solve`;
- Embora as linguagens algébricas permitam isso, esta **não** é a forma ideal de usá-las;
- O potencial delas está em permitir a definição do **modelo algébrico**, isto é, sem os dados de um problema específico;

AMPL/GMPL

▷ Exemplo

- A modelagem anterior explicita os dados diretamente no modelo, semelhante ao que é feito, por exemplo, no `lp_solve`;
- Embora as linguagens algébricas permitam isso, esta **não** é a forma ideal de usá-las;
- O potencial delas está em permitir a definição do **modelo algébrico**, isto é, sem os dados de um problema específico;
- Vamos então definir um modelo algébrico para o exemplo.

AMPL/GMPL

▷ Exemplo

Uma metalúrgica produz dois tipos de ligas metálicas. Cada liga é composta de proporções diferentes de cobre, zinco e chumbo, os quais estão disponíveis em quantidades limitadas em estoque. Deseja-se determinar quanto produzir de cada liga metálica, de modo a maximizar a receita bruta, satisfazendo-se as seguintes composições das ligas e a disponibilidade de matéria-prima em estoque:

Matéria-prima	Liga 1	Liga 2	Estoque
Cobre	50%	30%	3 ton
Zinco	10%	20%	1 ton
Chumbo	40%	50%	3 ton
Preço venda	3 mil	2 mil	(R$ por ton)

AMPL/GMPL

▷ Exemplo: modelo algébrico

- Conjuntos

AMPL/GMPL

▷ Exemplo: modelo algébrico

- Conjuntos
  - Ligas: $L = \{1, \ldots, n\}$;

AMPL/GMPL

▷ Exemplo: modelo algébrico

- Conjuntos
  - Ligas: $L = \{1, \ldots, n\}$;
  - Matérias-primas: $M = \{1, \ldots, m\}$;

AMPL/GMPL

▷ Exemplo: modelo algébrico

- Conjuntos
  - Ligas: $L = \{1, \ldots, n\}$;
  - Matérias-primas: $M = \{1, \ldots, m\}$;
- Parâmetros

AMPL/GMPL

▷ Exemplo: modelo algébrico

- Conjuntos
  - Ligas: $L = \{1, \ldots, n\}$;
  - Matérias-primas: $M = \{1, \ldots, m\}$;
- Parâmetros
  - Receita: $r_i, \forall i \in L$;

AMPL/GMPL

▷ Exemplo: modelo algébrico

- Conjuntos
  - Ligas: $L = \{1, \ldots, n\}$;
  - Matérias-primas: $M = \{1, \ldots, m\}$;
- Parâmetros
  - Receita: $r_i, \forall i \in L$;
  - Estoque: $b_j, \forall j \in M$;

AMPL/GMPL

▷ Exemplo: modelo algébrico

- Conjuntos
  - Ligas: $L = \{1, \ldots, n\}$;
  - Matérias-primas: $M = \{1, \ldots, m\}$;
- Parâmetros
  - Receita: $r_i, \forall i \in L$;
  - Estoque: $b_j, \forall j \in M$;
  - Composição das ligas: $a_{ji}, \forall j \in M, \forall i \in L$;

AMPL/GMPL

▷ Exemplo: modelo algébrico

- Conjuntos
  - Ligas: $L = \{1, \ldots, n\}$;
  - Matérias-primas: $M = \{1, \ldots, m\}$;
- Parâmetros
  - Receita: $r_i, \forall i \in L$;
  - Estoque: $b_j, \forall j \in M$;
  - Composição das ligas: $a_{ji}, \forall j \in M, \forall i \in L$;
- Variáveis de decisão

AMPL/GMPL

▷ Exemplo: modelo algébrico

- Conjuntos
  - Ligas: $L = \{1, \ldots, n\}$;
  - Matérias-primas: $M = \{1, \ldots, m\}$;
- Parâmetros
  - Receita: $r_i, \forall i \in L$;
  - Estoque: $b_j, \forall j \in M$;
  - Composição das ligas: $a_{ji}, \forall j \in M, \forall i \in L$;
- Variáveis de decisão
  - $x_i \geq 0, \forall i \in L$: quantidade a ser produzida da liga i.

AMPL/GMPL

- Exemplo: modelo algébrico

$$\begin{aligned} \max \quad & \sum_{i \in L} r_i x_i \\ \text{s.a} \quad & \sum_{i \in L} a_{ji} x_i \leq b_j, \ \forall j \in M \\ & x_i \geq 0, \ \forall i \in L. \end{aligned}$$

AMPL/GMPL

▷ Exemplo: modelo algébrico

- Conjuntos
  - Ligas: $L = \{1, \ldots, n\}$;
  - Matérias-primas: $M = \{1, \ldots, m\}$;
- Parâmetros
  - Receita: $r_i, \forall i \in L$;
  - Estoque: $b_j, \forall j \in M$;
  - Composição das ligas: $a_{ji}, \forall j \in M, \forall i \in L$;
- Variáveis de decisão
  - $x_i \geq 0, \forall i \in L$: quantidade a ser produzida da liga i.

AMPL/GMPL

▷ Exemplo: definição do modelo

```
# Conjuntos de indices
set L; # Ligas
set M; # Materias primas
```

AMPL/GMPL

▷ Exemplo: definição do modelo

```
# Conjuntos de indices
set L; # Ligas
set M; # Materias primas

# Parametros (dados de entrada)
param r{L}; # receita de cada liga
param b{M}; # estoque de cada materia prima
param a{M, L}; # composicao de cada materia prima em cada liga
```

AMPL/GMPL

▷ Exemplo: definição do modelo

```
# Conjuntos de indices
set L; # Ligas
set M; # Materias primas

# Parametros (dados de entrada)
param r{L}; # receita de cada liga
param b{M}; # estoque de cada materia prima
param a{M, L}; # composicao de cada materia prima em cada liga

# Variaveis de decisao
var x{L} >= 0;
```

AMPL/GMPL

▷ Exemplo: definição do modelo

```
# Conjuntos de indices
set L; # Ligas
set M; # Materias primas

# Parametros (dados de entrada)
param r{L}; # receita de cada liga
param b{M}; # estoque de cada materia prima
param a{M, L}; # composicao de cada materia prima em cada liga

# Variaveis de decisao
var x{L} >= 0;

# Funcao objetivo
maximize ReceitaTotal: sum{i in L} r[i] * x[i];
```

AMPL/GMPL

▷ Exemplo: definição do modelo

```
# Conjuntos de indices
set L; # Ligas
set M; # Materias primas

# Parametros (dados de entrada)
param r{L}; # receita de cada liga
param b{M}; # estoque de cada materia prima
param a{M, L}; # composicao de cada materia prima em cada liga

# Variaveis de decisao
var x{L} >= 0;

# Funcao objetivo
maximize ReceitaTotal: sum{i in L} r[i] * x[i];

# Restricoes
restr_estoques{j in M}:  sum{i in L} a[j,i] * x[i] <= b[j];
```

AMPL/GMPL

- Exemplo: modelo algébrico

$$\begin{aligned} \max \quad & \sum_{i \in L} r_i x_i \\ \text{s.a} \quad & \sum_{i \in L} a_{ji} x_i \leq b_j, \ \forall j \in M \\ & x_i \geq 0, \ \forall i \in L. \end{aligned}$$

AMPL/GMPL

▷ Exemplo

Uma metalúrgica produz dois tipos de ligas metálicas. Cada liga é composta de proporções diferentes de cobre, zinco e chumbo, os quais estão disponíveis em quantidades limitadas em estoque. Deseja-se determinar quanto produzir de cada liga metálica, de modo a maximizar a receita bruta, satisfazendo-se as seguintes composições das ligas e a disponibilidade de matéria-prima em estoque:

Matéria-prima	Liga 1	Liga 2	Estoque
Cobre	50%	30%	3 ton
Zinco	10%	20%	1 ton
Chumbo	40%	50%	3 ton
Preço venda	3 mil	2 mil	(R$ por ton)

AMPL/GMPL

▷ Exemplo: definição dos dados

```
# Dados de uma instância do problema
data;
```

AMPL/GMPL

▷ Exemplo: definição dos dados

```
# Dados de uma instância do problema
data;

set L := 1 2;
set M := Cobre Zinco Chumbo;
```

AMPL/GMPL

▷ Exemplo: definição dos dados

```
# Dados de uma instância do problema
data;

set L := 1 2;
set M := Cobre Zinco Chumbo;

param: r :=
1    3
2    2;
```

AMPL/GMPL

▷ Exemplo: definição dos dados

```
# Dados de uma instância do problema
data;

set L := 1 2;
set M := Cobre Zinco Chumbo;

param: r :=
1    3
2    2;

param: b :=
Cobre  3
Zinco  1
Chumbo 3;
```

AMPL/GMPL

▷ Exemplo: definição dos dados

```
# Dados de uma instância do problema
data;

set L := 1 2;
set M := Cobre Zinco Chumbo;

param: r :=
1    3
2    2;

param: b :=
Cobre  3
Zinco  1
Chumbo 3;

param a : 1 2 :=
Cobre   0.5  0.3
Zinco   0.1  0.2
Chumbo  0.4  0.5;
end;
```

AMPL/GMPL

- Exemplo: definição dos comandos para resolução e *output*

```
# Escolher o solver (APENAS NO AMPL)
option solver cbc;
```

AMPL/GMPL

- Exemplo: definição dos comandos para resolução e *output*

```
# Escolher o solver (APENAS NO AMPL)
option solver cbc;

# Resolver o problema
solve;
```

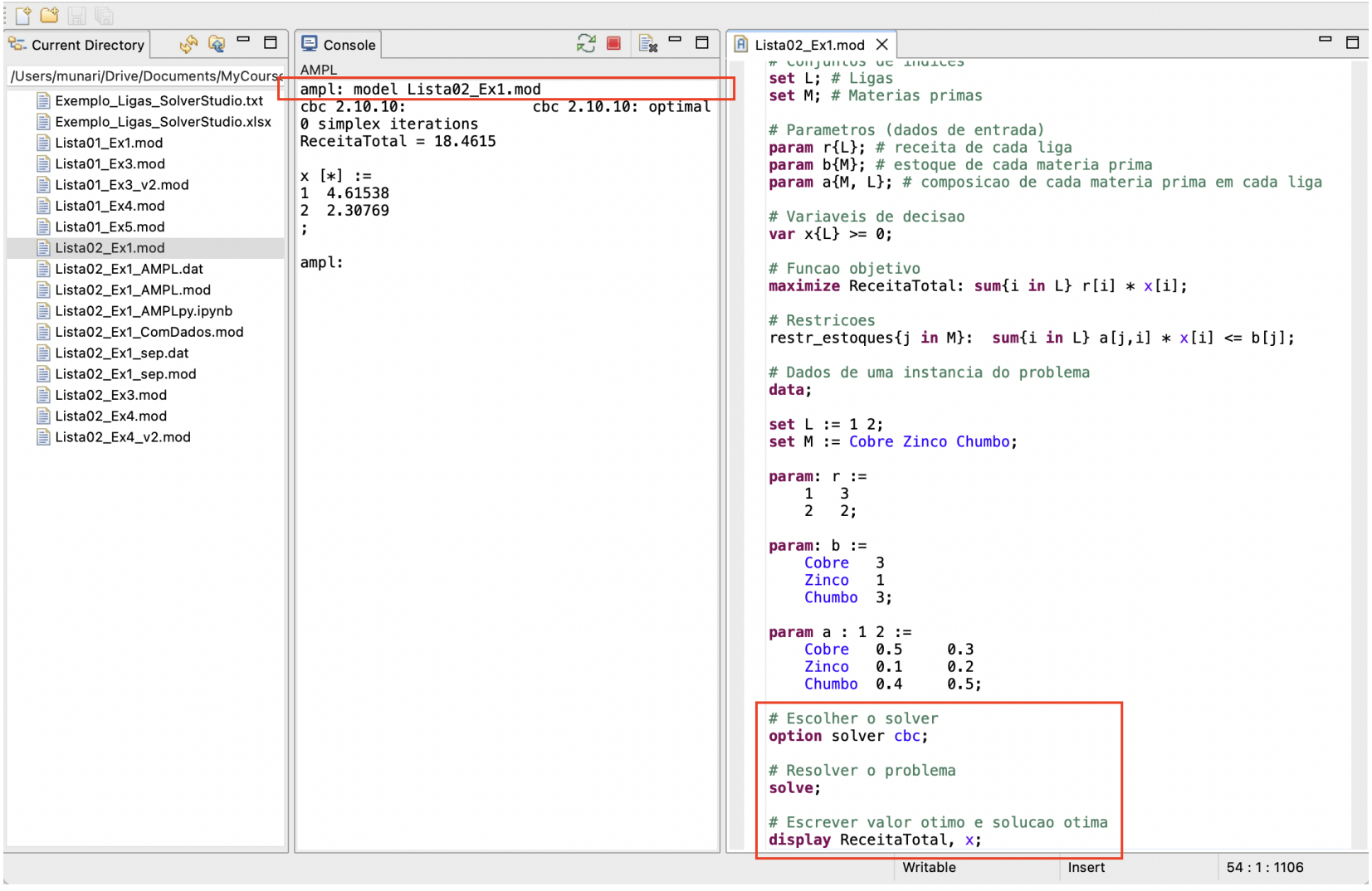
AMPL/GMPL

- Exemplo: definição dos comandos para resolução e *output*

```
# Escolher o solver (APENAS NO AMPL)
option solver cbc;

# Resolver o problema
solve;

# Escrever valor otimo e solucao otima
display ReceitaTotal, x;
```

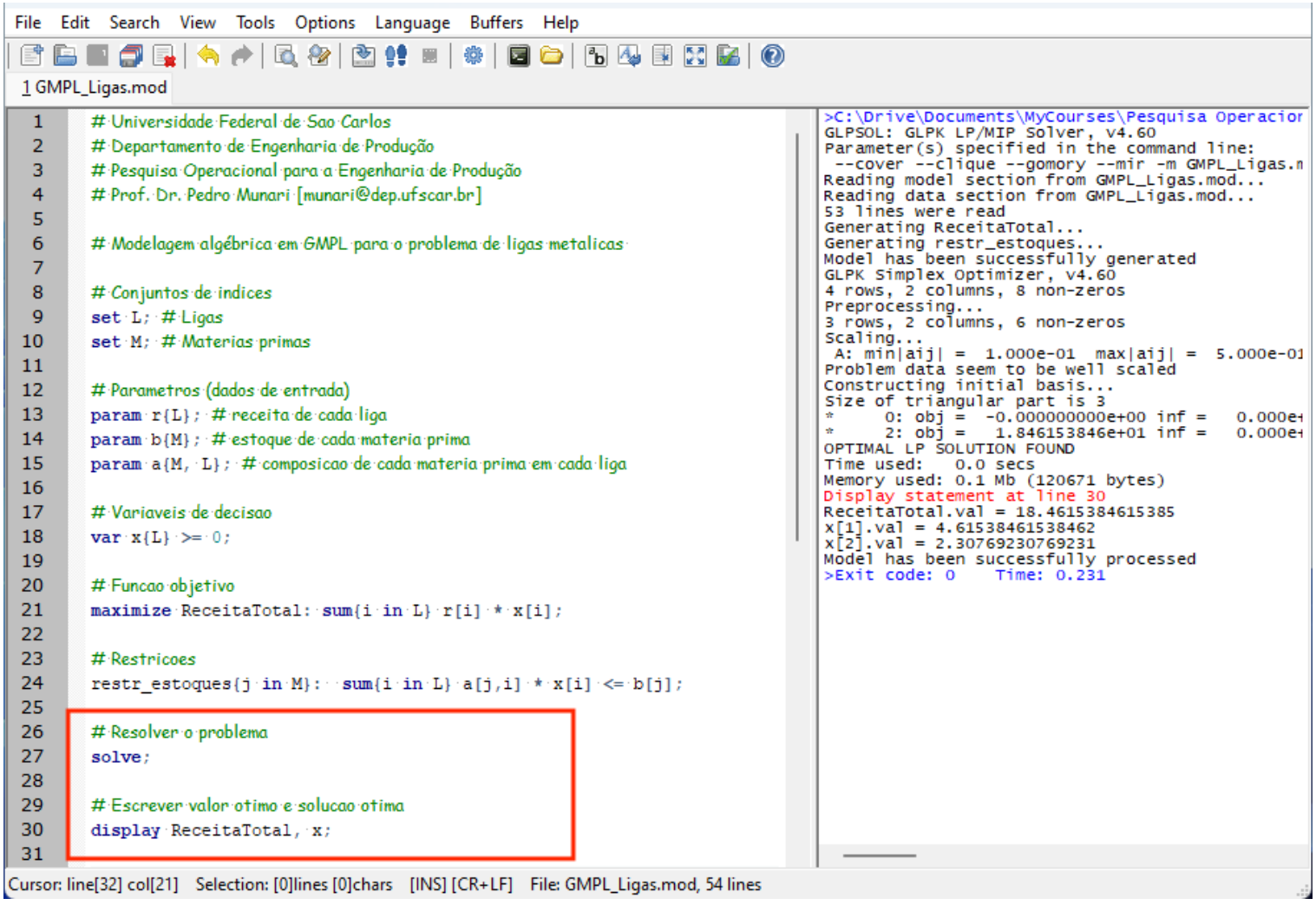
AMPL/GMPL

▷ AMPL



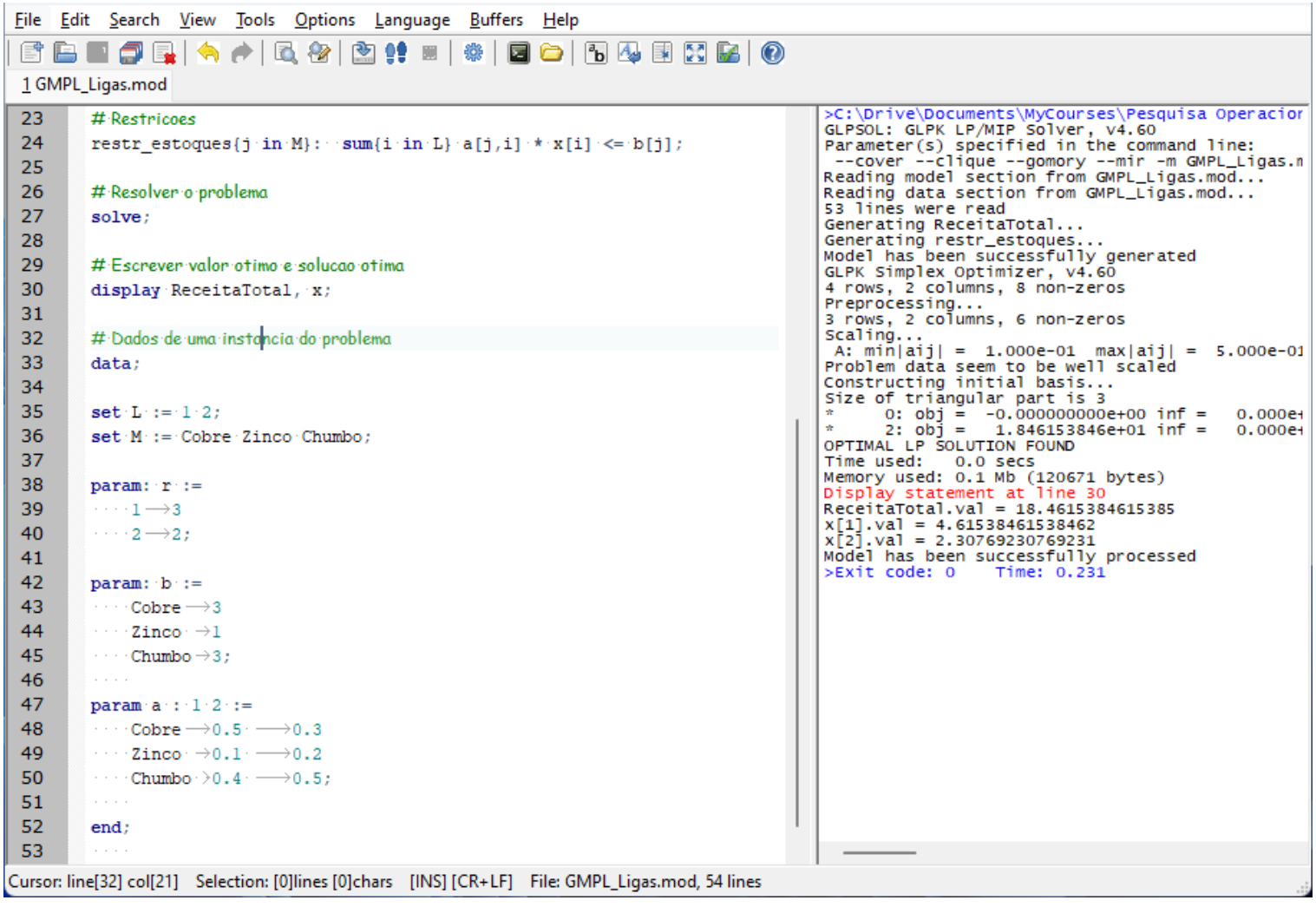
AMPL/GMPL

- Gusek



AMPL/GMPL

- Gusek



AMPL/GMPL

▷ Importante!

- Embora não seja o ideal, nesse primeiro exemplo definimos um único arquivo `.mod`, contendo todo o código para a definição do modelo, dos dados e dos comandos para resolução e *output*;

AMPL/GMPL

▷ Importante!

- Embora não seja o ideal, nesse primeiro exemplo definimos um único arquivo `.mod`, contendo todo o código para a definição do modelo, dos dados e dos comandos para resolução e *output*;
- Observe que no Gusek definimos os comandos `solve` e `display` **antes** da definição dos dados (o mesmo deve ser feito no PIFOP e otimizador *online*);

AMPL/GMPL

▷ Importante!

- Embora não seja o ideal, nesse primeiro exemplo definimos um único arquivo `.mod`, contendo todo o código para a definição do modelo, dos dados e dos comandos para resolução e *output*;
- Observe que no Gusek definimos os comandos `solve` e `display` **antes** da definição dos dados (o mesmo deve ser feito no PIFOP e otimizador *online*);
- Já no AMPL, os comandos `solve` e `display` devem vir **depois** da definição dos dados, ou serem digitados diretamente no *console* da interface.

AMPL/GMPL

▷ Importante!

- O ideal é dividirmos o código em **dois arquivos**: a definição do modelo fica no `.mod`, e a definição dos dados fica no `.dat`;

AMPL/GMPL

▷ Importante!

- O ideal é dividirmos o código em **dois arquivos**: a definição do modelo fica no `.mod`, e a definição dos dados fica no `.dat`;
- Essa separação é importante, principalmente em projetos grandes, e permite resolver diferentes instâncias (dados de cenários específicos) sem alterar o modelo;

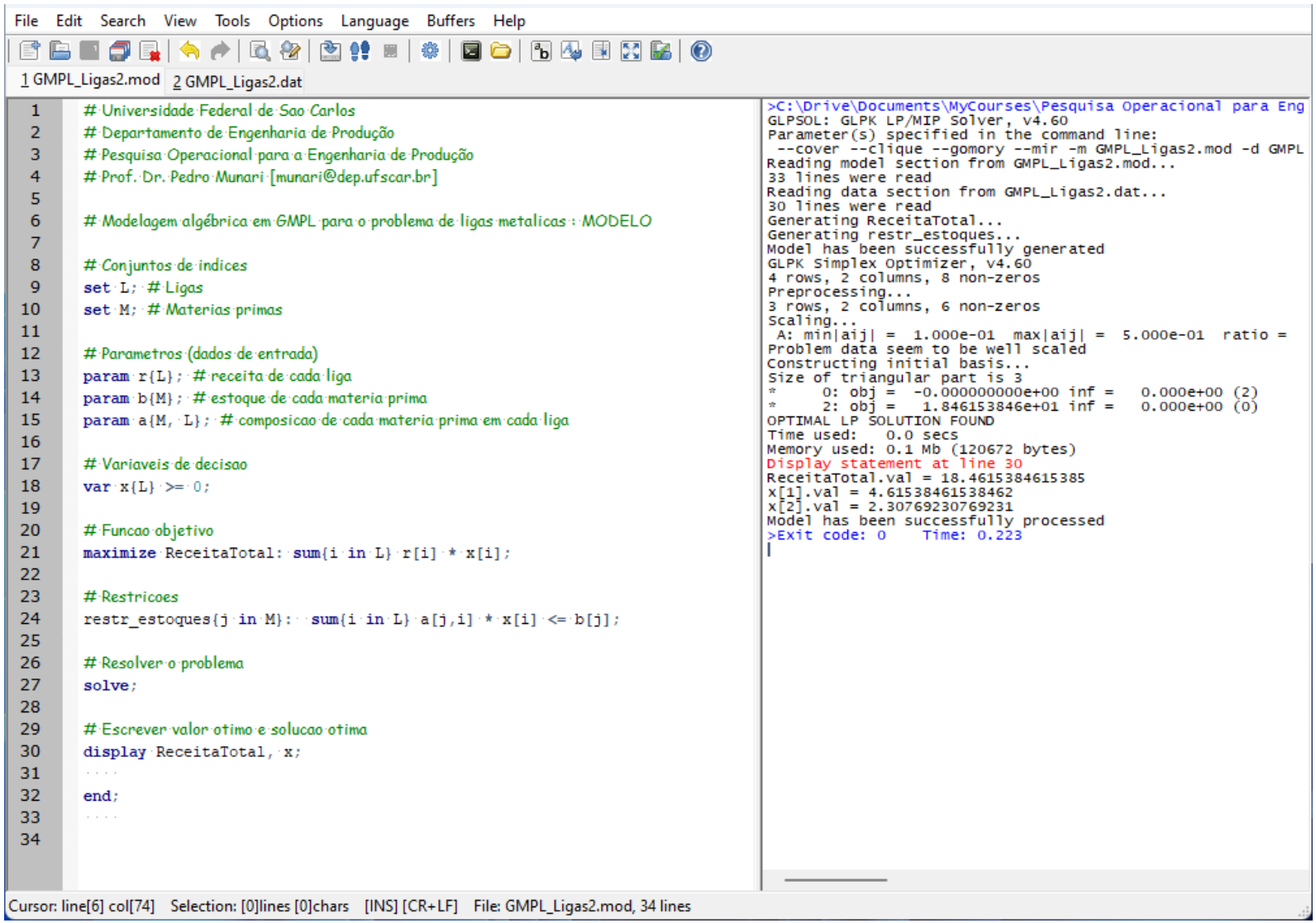
AMPL/GMPL

▷ Importante!

- O ideal é dividirmos o código em **dois arquivos**: a definição do modelo fica no `.mod`, e a definição dos dados fica no `.dat`;
- Essa separação é importante, principalmente em projetos grandes, e permite resolver diferentes instâncias (dados de cenários específicos) sem alterar o modelo;
- Os comandos de resolução podem ser inseridos no `.mod`, ou digitados diretamente na interface no caso do AMPL.

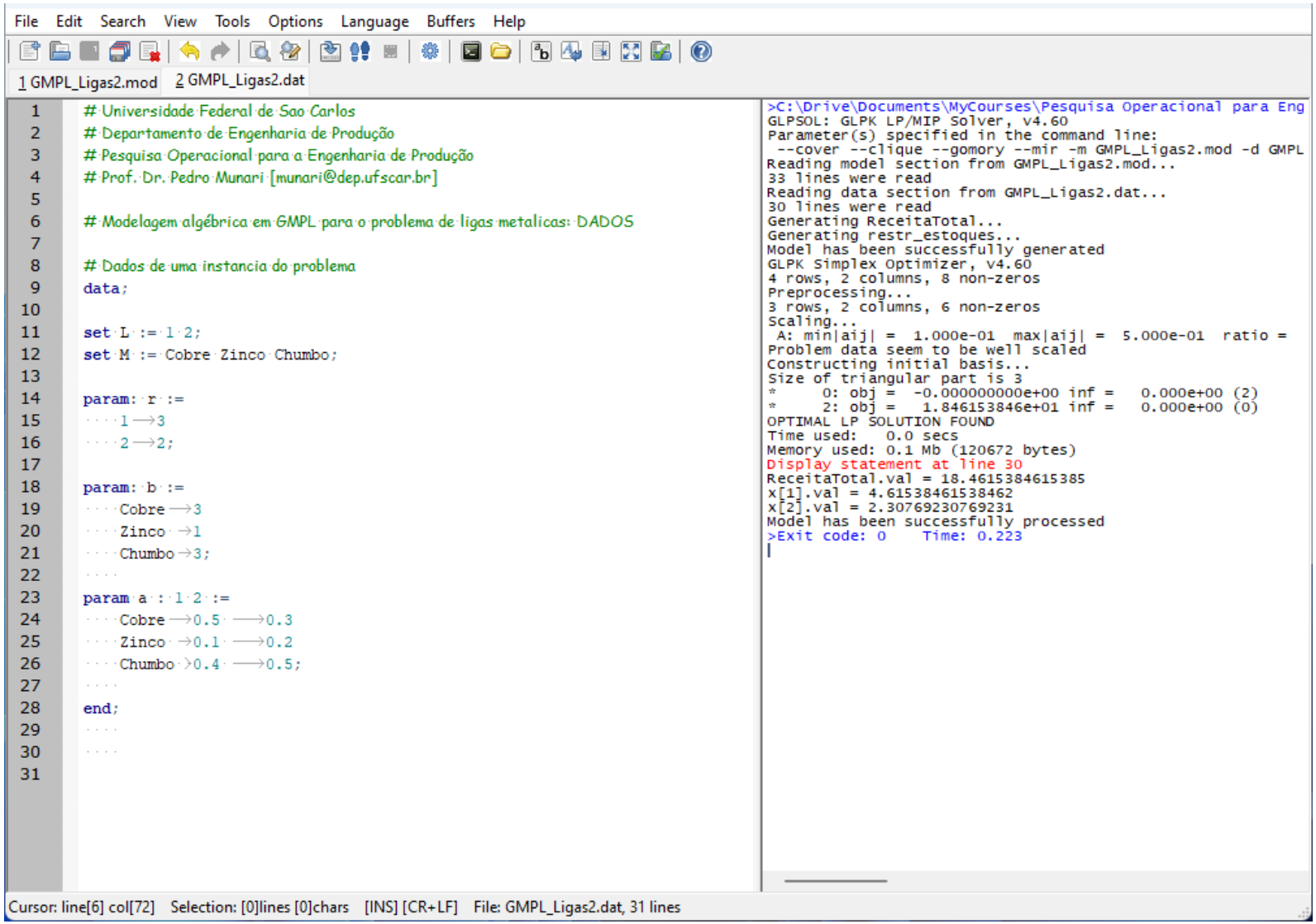
AMPL/GMPL

- Gusek: Arquivo `.mod`



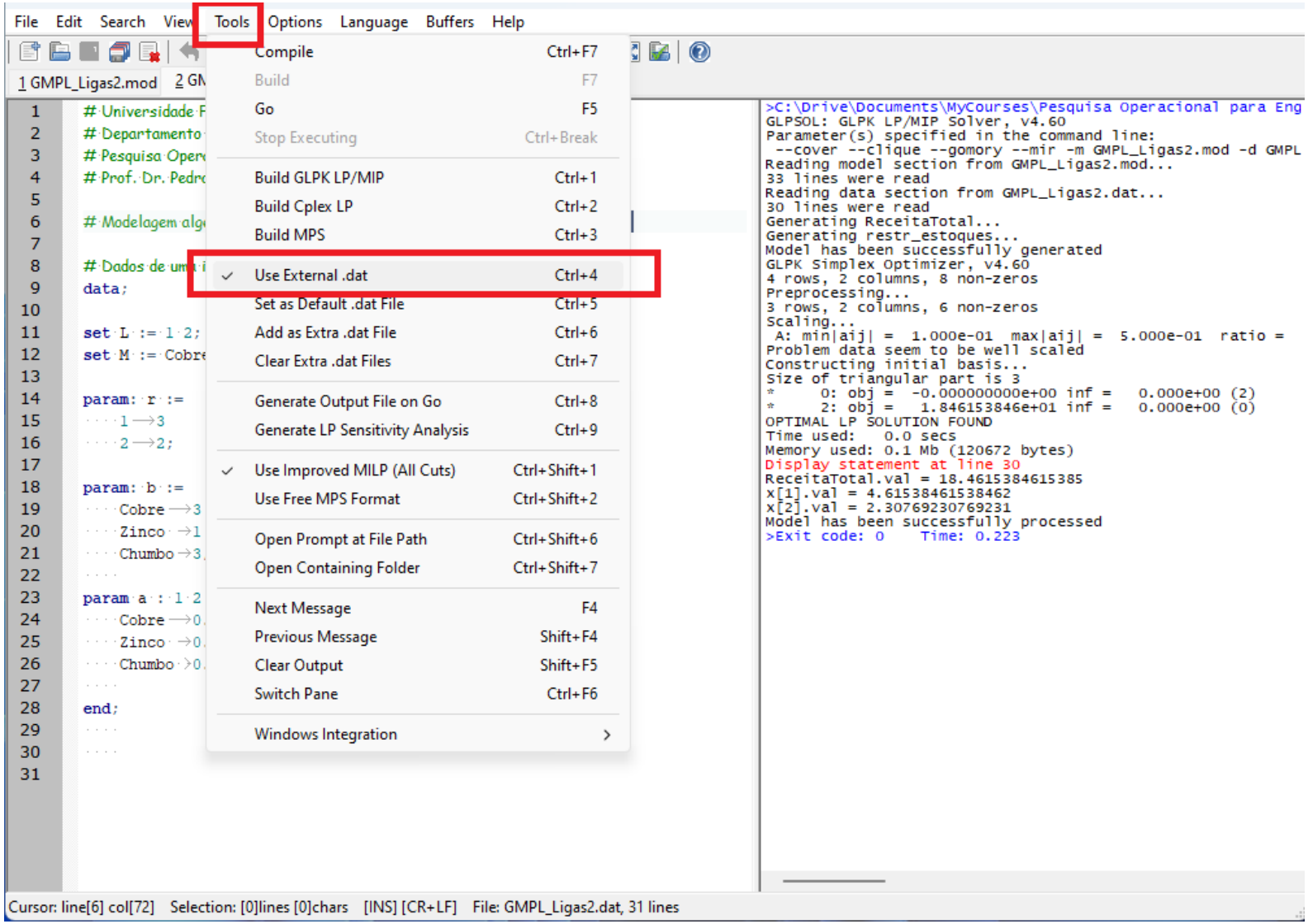
AMPL/GMPL

- Gusek: Arquivo .dat



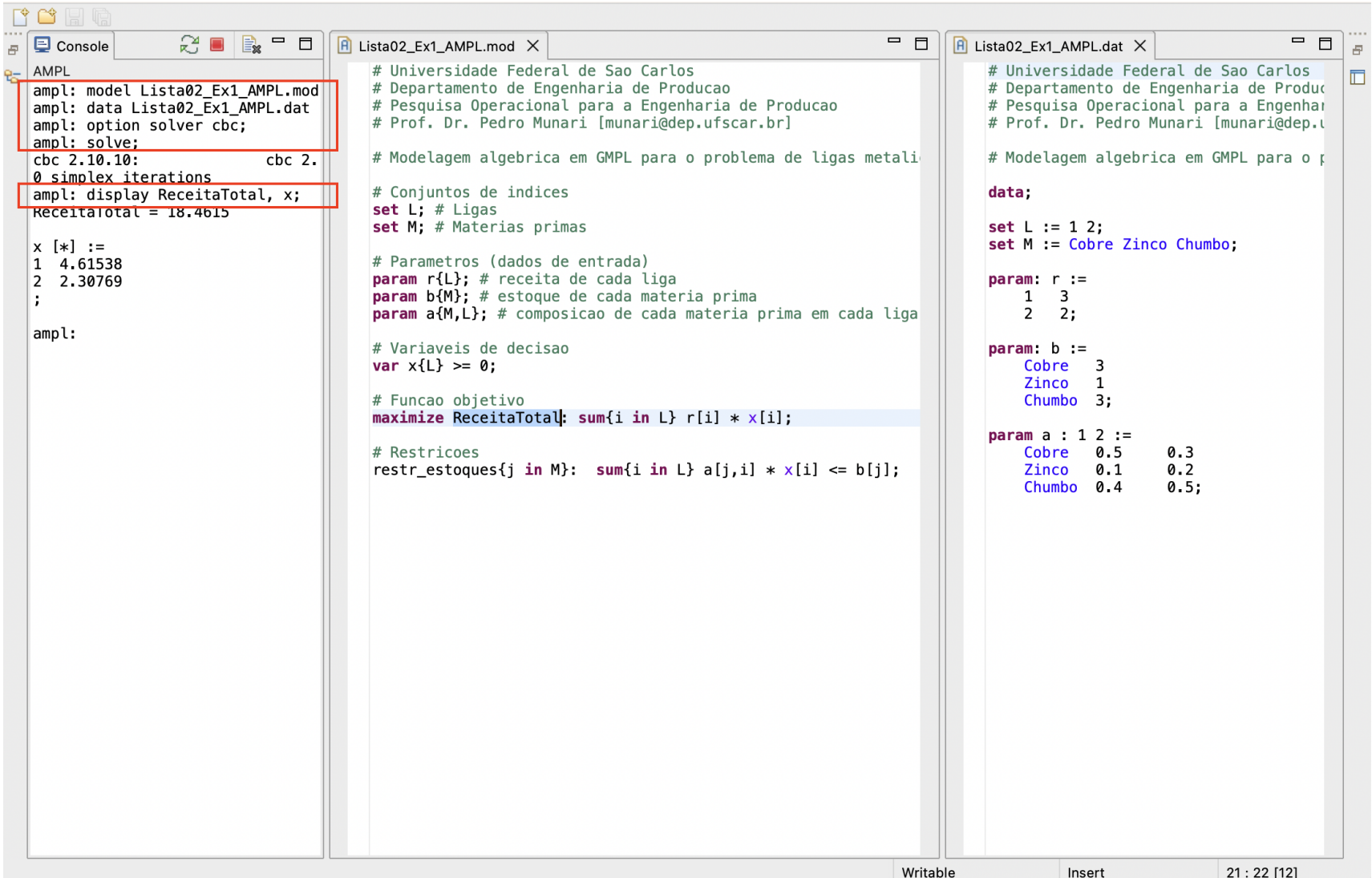
AMPL/GMPL

- Gusek: Antes de resolver...



AMPL/GMPL

- AMPL: Arquivos `.mod`, `.dat` e comandos na interface



AMPL/GMPL

▷ Sintaxe

- Definição dos conjuntos de índices, usando o comando `set`:

  ```
  set L;
  ```

AMPL/GMPL

▷ Sintaxe

- Definição dos conjuntos de índices, usando o comando `set`:
  `set L;`
- Definição de parâmetros, usando o comando `param`:
  `param a{M, L};`

AMPL/GMPL

▷ Sintaxe

- Definição dos conjuntos de índices, usando o comando `set`:

```
set L;
```

- Definição de parâmetros, usando o comando `param`:

```
param a{M, L};
```

- Definição de variáveis, usando o comando `var`:

```
var x{L} >= 0;
var y{Produtos} >= 0, integer;
var w{V} binary;
```

AMPL/GMPL

▷ Sintaxe

- Definição dos conjuntos de índices, usando o comando `set`:

```
set L;
```

- Definição de parâmetros, usando o comando `param`:

```
param a{M, L};
```

- Definição de variáveis, usando o comando `var`:

```
var x{L} >= 0;
var y{Produtos} >= 0, integer;
var w{V} binary;
```

- Definição da função objetivo, usando o comando `maximize` ou `minimize`:

```
maximize ReceitaTotal:  sum{i in L} r[i] * x[i];
```

AMPL/GMPL

▷ Sintaxe

- Definição de restrições, com o domínio e nome (opcional):

```
estoques{j in M}:  sum{i in L} a[j,i] * x[i] <= b[j];
```

AMPL/GMPL

▷ Sintaxe

- Definição de restrições, com o domínio e nome (opcional):
  `estoques{j in M}:  sum{i in L} a[j,i] * x[i] <= b[j];`
- Resolução do problema, por meio do comando `solve`;

AMPL/GMPL

▷ Sintaxe

- Definição de restrições, com o domínio e nome (opcional):

```
estoques{j in M}:  sum{i in L} a[j,i] * x[i] <= b[j];
```

- Resolução do problema, por meio do comando `solve`;
- Impressão de dados e resultados, por meio do comando `display`;

```
display ReceitaTotal, x;
display {i in T, k in E: x[i,k] > 0} x[i,k];
```

AMPL/GMPL

▷ Sintaxe

- Definição de restrições, com o domínio e nome (opcional):

```
estoques{j in M}:  sum{i in L} a[j,i] * x[i] <= b[j];
```

- Resolução do problema, por meio do comando `solve`;
- Impressão de dados e resultados, por meio do comando `display`;

```
display ReceitaTotal, x;
display {i in T, k in E: x[i,k] > 0} x[i,k];
```

- Entrada de dados da instância, por meio do comando `data`;

AMPL/GMPL

▷ Sintaxe

```
# Dados de uma instância do problema
data;

set L := 1 2;
set M := Cobre Zinco Chumbo;

param: r :=
1 3
2 2;

param: b :=
Cobre  3
Zinco  1
Chumbo 3;

param a : 1 2 :=
Cobre   0.5  0.3
Zinco   0.1  0.2
Chumbo  0.4  0.5;

end;
```

AMPL/GMPL

▷ AMPLpy: AMPL + Python

- Podemos usar AMPL/GMPL diretamente no Python, por meio da biblioteca AMPLpy;

AMPL/GMPL

▷ AMPLpy: AMPL + Python

- Podemos usar AMPL/GMPL diretamente no Python, por meio da biblioteca AMPLpy;
- A sintaxe da linguagem é exatamente a mesma;

AMPL/GMPL

▷ AMPLpy: AMPL + Python

- Podemos usar AMPL/GMPL diretamente no Python, por meio da biblioteca AMPLpy;
- A sintaxe da linguagem é exatamente a mesma;
- Não precisamos instalar a interface do AMPL, ou nenhum outro software similar, necessitando apenas do Python e da biblioteca AMPLpy;

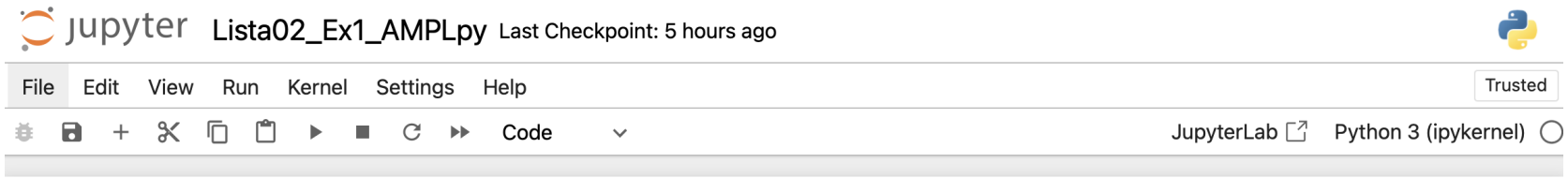
AMPL/GMPL

▷ AMPLpy: AMPL + Python

- Podemos usar AMPL/GMPL diretamente no Python, por meio da biblioteca AMPLpy;
- A sintaxe da linguagem é exatamente a mesma;
- Não precisamos instalar a interface do AMPL, ou nenhum outro software similar, necessitando apenas do Python e da biblioteca AMPLpy;
- Site com vários exemplos: `https://colab.ampl.com`;

AMPL/GMPL

▷ AMPLpy: AMPL + Python



[1]:
```
# Universidade Federal de São Carlos
# Departamento de Engenharia de Produção
# Pesquisa Operacional para a Engenharia de Produção
# Prof. Dr. Pedro Munari [munari@dep.ufscar.br]

# Modelagem algébrica em AMPLpy para o problema de ligas metálicas
# Mais informações: https://colab.ampl.com
```

Comandos iniciais

[3]:
```
# Instalar AMPL no Python, caso ainda não tenha instalado
%pip install -q amplpy
```

[4]:
```
# Importar as bibliotecas do AMPL
from amplpy import AMPL, ampl_notebook
ampl = ampl_notebook(
    modules=["highs","coin","cplex"],  # carregar solvers para uso
    license_uuid="default"  # numero da licença do software
)
```

AMPL/GMPL

▷ AMPLpy: AMPL + Python

jupyter Lista02_Ex1_AMPLpy Last Checkpoint: 5 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

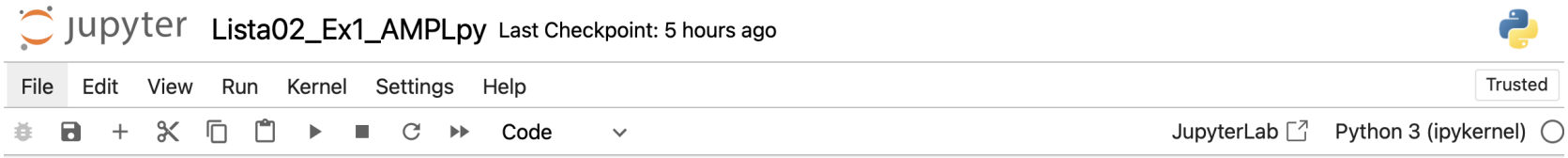
Exemplo de forma de uso 1: colocamos o modelo, dados e comandos em uma única célula

[6]:

```
%%ampl_eval # Essa flag é para sinalizar que vamos colocar um código que deve ser processado pelo AMPL

reset; # Redefine o ambiente do AMPL
option solver cplex;# Escolhe o solver que vamos usar

# A partir deste ponto, vamos colocar o modelo algébrico, os dados, e os comandos para resolver e mostrar a sol

# Conjuntos de indices
set L; # Ligas
set M; # Materias primas

# Parametros (dados de entrada)
param r{L}; # receita de cada liga
param b{M}; # estoque de cada materia prima
param a{M,L}; # composicao de cada materia prima em cada liga

# Variaveis de decisao
var x{L} >= 0;

# Funcao objetivo
maximize ReceitaTotal: sum{i in L} r[i] * x[i];

# Restricoes
restr_estoques{j in M}:  sum{i in L} a[j,i] * x[i] <= b[j];

data;

set L := 1 2;
set M := Cobre Zinco Chumbo;
```

AMPL/GMPL

▷ AMPLpy: AMPL + Python



Exemplo de forma de uso 2: lemos o modelo, dados e comandos de um único arquivo

[8]:
```
%%ampl_eval # Essa flag é para sinalizar que vamos colocar um código que deve ser processado pelo AMPL

reset; # Redefine o ambiente do AMPL
option solver cplex; # Escolhe o solver que vamos usar
model Lista02_Ex1.mod # Le um arquivo .mod contendo modelo, dados e comandos para resolver e imprimir saída
```

```
CPLEX 22.1.1:  - Version identifier: 22.1.1.0 | 2022-11-28 | 9160aff4d
 - CPXPARAM_Simplex_Display                            0
 - CPXPARAM_MIP_Display                                0
 - CPXPARAM_Barrier_Display                            0
CPLEX 22.1.1: optimal solution; objective 18.46153846
2 simplex iterations
ReceitaTotal = 18.4615

x [*] :=
1  4.61538
2  2.30769
;
```

Exemplo de forma de uso 3: lemos o modelo e os dados de arquivos separados

[10]:
```
%%ampl_eval # Essa flag é para sinalizar que vamos colocar um código que deve ser processado pelo AMPL

reset; # Redefine o ambiente do AMPL
option solver cplex; # Escolhe o solver que vamos usar

model Lista02_Ex1_AMPL.mod # Le um arquivo .mod contendo apenas o modelo
data  Lista02_Ex1_AMPL.dat # Le um arquivo .dat contendo apenas os dados
```

Pyomo

- Linguagem de modelagem em Python

Pyomo

▷ Linguagem de modelagem em Python

- Solvers comerciais oferecem bibliotecas amigáveis em Python para definição dos modelos de forma similar a uma linguagem algébrica;
- Por exemplo: DOCplex (IBM CPLEX); e gurobipy (Gurobi);

Pyomo

▷ Linguagem de modelagem em Python

- Solvers comerciais oferecem bibliotecas amigáveis em Python para definição dos modelos de forma similar a uma linguagem algébrica;
- Por exemplo: DOCplex (IBM CPLEX); e gurobipy (Gurobi);
- Possuem comunicação direta com o solver, mas são específicas para uso do solver correspondente, impossibilitando a troca de solvers, caso necessário;

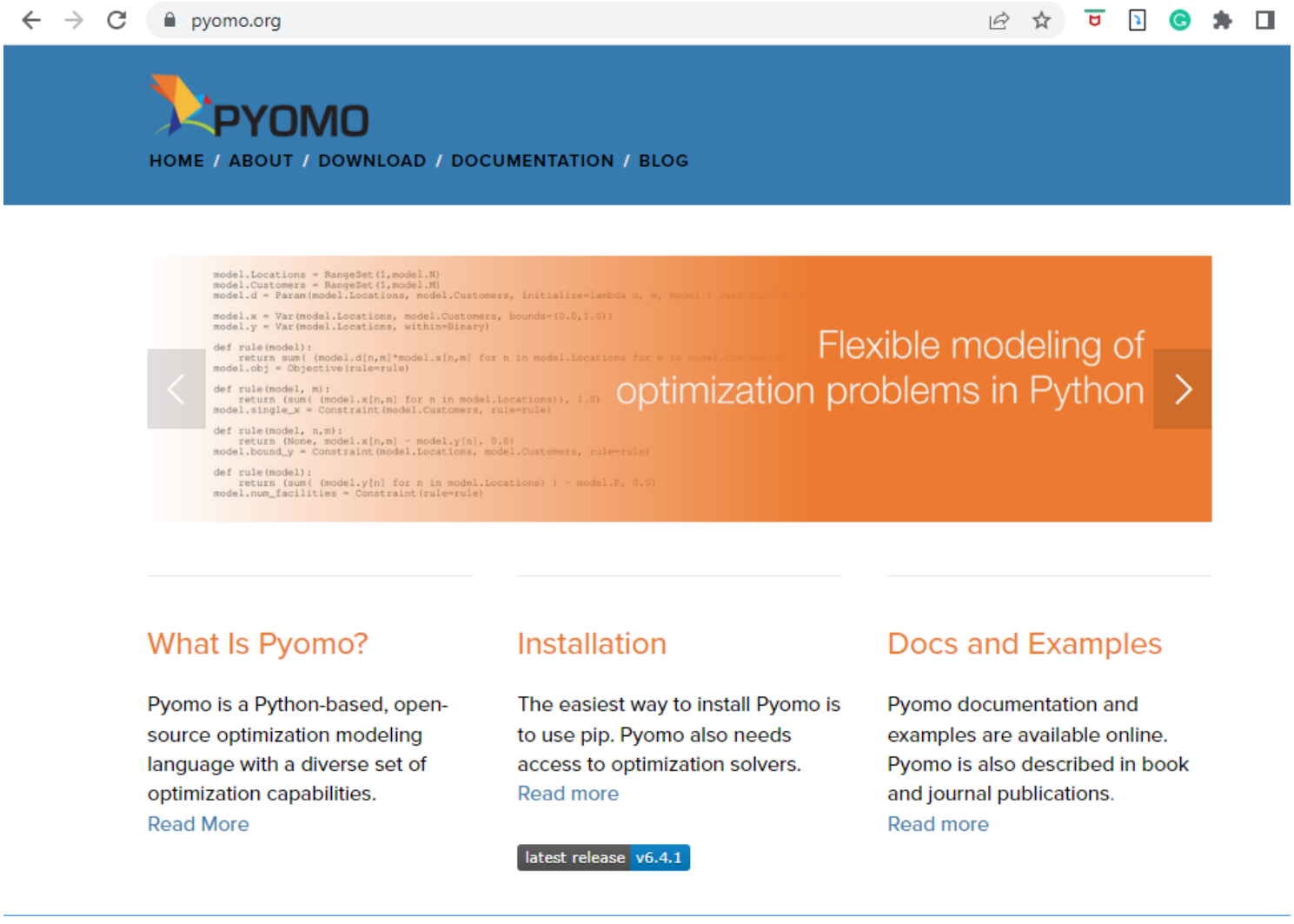
Pyomo

▷ Linguagem de modelagem em Python

- Solvers comerciais oferecem bibliotecas amigáveis em Python para definição dos modelos de forma similar a uma linguagem algébrica;
- Por exemplo: DOCplex (IBM CPLEX); e gurobipy (Gurobi);
- Possuem comunicação direta com o solver, mas são específicas para uso do solver correspondente, impossibilitando a troca de solvers, caso necessário;
- Assim, exitem outras bibliotecas que permitem implementar os modelos em Python e usando uma linguagem próxima da algébrica, mas que são independentes do solver e, portanto, permitem trocar de solvers facilmente.

Pyomo

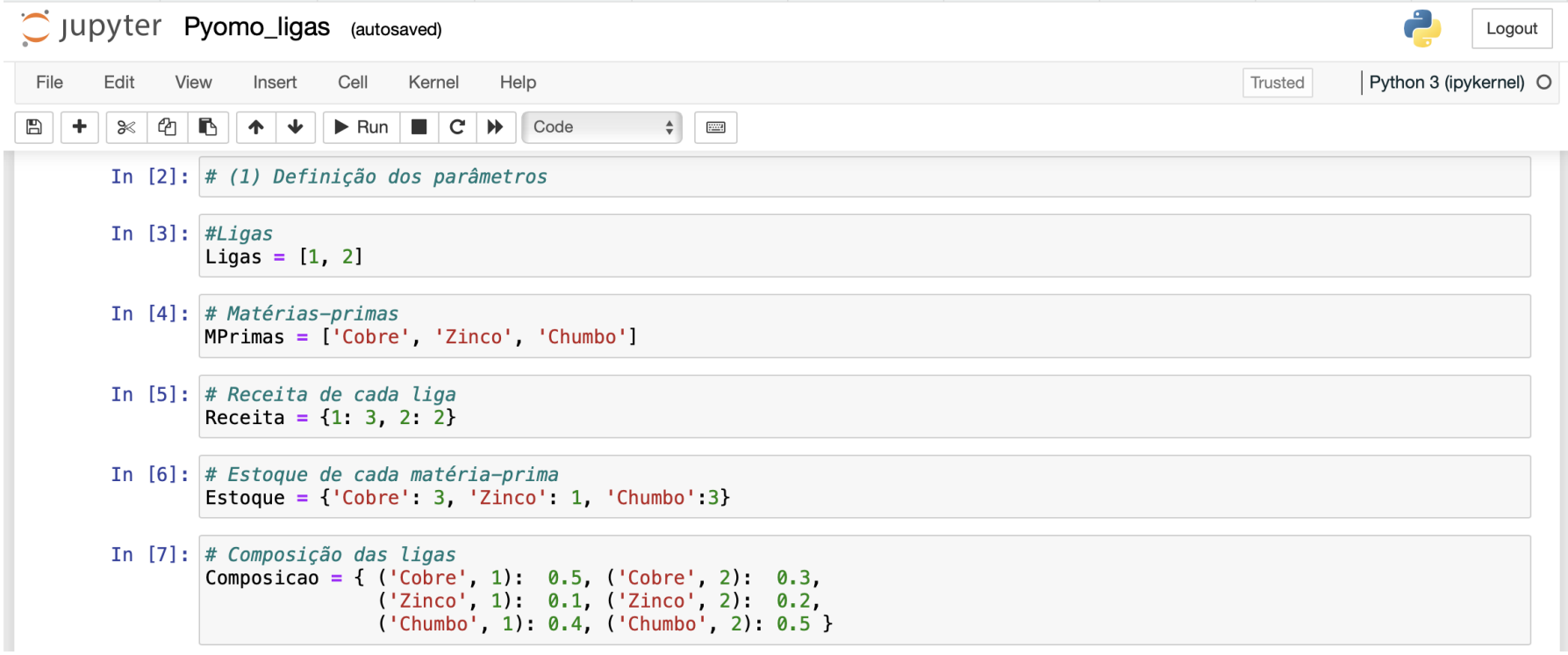
- Linguagem de modelagem em Python



Pyomo

▷ Linguagem de modelagem em Python

- O usuário tem toda liberdade para definir os dados da forma que desejar, podendo ler de planilhas ou outras fontes;



Pyomo

▷ Linguagem de modelagem em Python

- O modelo é definido de forma independente dos dados;

In [8]:
```
# (2) Definição do modelo matemático no Pyomo
```

In [9]:
```
# Cabeçalhos para incluir as bibliotecas do Pyomo
import pyomo.environ as pyo
from pyomo.environ import *
from pyomo.opt import SolverFactory
```

In [10]:
```
# Definição do modelo matemático no Pyomo
modelo = pyo.ConcreteModel()
```

In [11]:
```
# Definição dos conjuntos no Pyomo
L = pyo.Set(initialize = Ligas)
M = pyo.Set(initialize = MPrimas)
```

In [12]:
```
# Definição dos parâmetros do modelo
modelo.r = pyo.Param(L, initialize = Receita, within = pyo.NonNegativeReals)
modelo.b = pyo.Param(M, initialize = Estoque, within = pyo.NonNegativeReals)
modelo.a = pyo.Param(M * L, initialize = Composicao, within = pyo.NonNegativeReals)
```

In [13]:
```
#Definição das variáveis de decisão: quantidade produzida de cada liga
modelo.x = pyo.Var(L, within = NonNegativeReals)
```

In [14]:
```
# Definição da função objetivo
modelo.fObjetivo = pyo.Objective(expr= sum( modelo.r[i] * modelo.x[i] for i in L ), sense = pyo.maximize)
```

In [15]:
```
# Definição das restrições
modelo.restrEstoque = pyo.ConstraintList()
for j in M:
    modelo.restrEstoque.add(expr= sum( modelo.a[j,i] * modelo.x[i] for i in L) <= modelo.b[j])
```

Pyomo

▷ Exemplo: modelo algébrico

$$
\begin{aligned}
\max \quad & \sum_{i \in L} r_i x_i \\
\text{s.a} \quad & \sum_{i \in L} a_{ji} x_i \leq b_j, \ \forall j \in M \\
& x_i \geq 0, \ \forall i \in L.
\end{aligned}
$$

Pyomo

▷ Linguagem de modelagem em Python

In [16]:
```
# (3) Resolução por um solver de otimização (GLPK, COIN, CPLEX, GUROBI)
```

In [17]:
```
# Resolução pelo solver GLPK (outros podem ser usados mudando-se apenas a linha a seguir)
solver = pyo.SolverFactory('glpk')
resultado = solver.solve(modelo, tee=True)
```

```
GLPSOL--GLPK LP/MIP Solver 5.0
Parameter(s) specified in the command line:
 --write /var/folders/np/g95pcx850ll5r6223805znv40000gn/T/tmpi7gsu9ly.glpk.raw
 --wglp /var/folders/np/g95pcx850ll5r6223805znv40000gn/T/tmp4u29gwo1.glpk.glp
 --cpxlp /var/folders/np/g95pcx850ll5r6223805znv40000gn/T/tmppraf3cyg.pyomo.lp
Reading problem data from '/var/folders/np/g95pcx850ll5r6223805znv40000gn/T/tmppraf3cyg.pyomo.lp'...
4 rows, 3 columns, 7 non-zeros
31 lines were read
Writing problem data to '/var/folders/np/g95pcx850ll5r6223805znv40000gn/T/tmp4u29gwo1.glpk.glp'...
23 lines were written
GLPK Simplex Optimizer 5.0
4 rows, 3 columns, 7 non-zeros
Preprocessing...
3 rows, 2 columns, 6 non-zeros
Scaling...
 A: min|aij| =  1.000e-01  max|aij| =  5.000e-01  ratio =  5.000e+00
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part is 3
*     0: obj =  -0.000000000e+00 inf =   0.000e+00 (2)
*     2: obj =   1.846153846e+01 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.0 Mb (40424 bytes)
Writing basic solution to '/var/folders/np/g95pcx850ll5r6223805znv40000gn/T/tmpi7gsu9ly.glpk.raw'...
16 lines were written
```

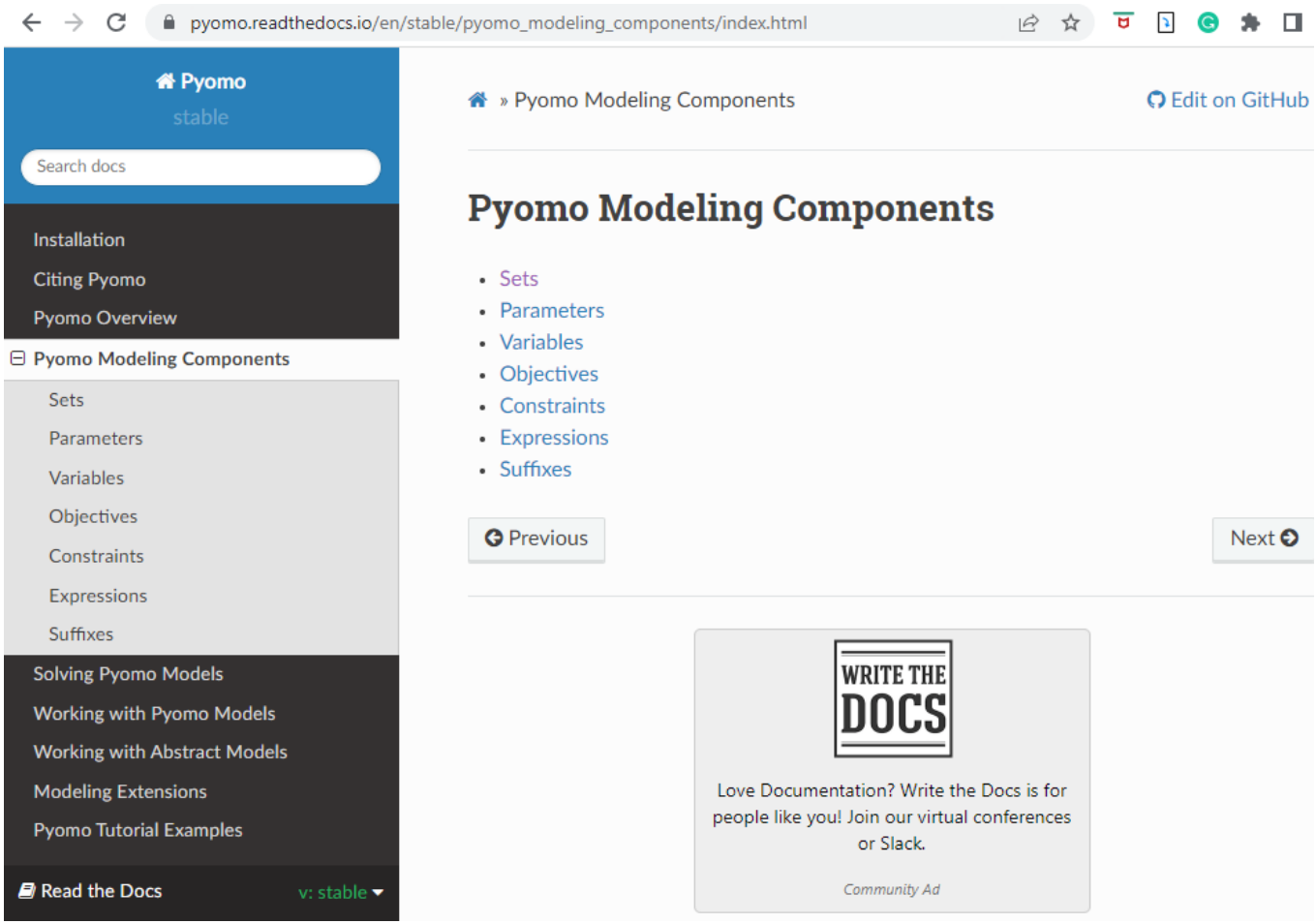
In [18]:
```
# Imprime a solução
if (resultado.solver.status == SolverStatus.ok) and (resultado.solver.termination_condition != TerminationCondition.
    print(modelo.x.get_values())
else:
    print('Não foi possível encontrar uma solução!')
```

```
{1: 4.61538461538462, 2: 2.30769230769231}
```

Pyomo

- Muito bem documentada!

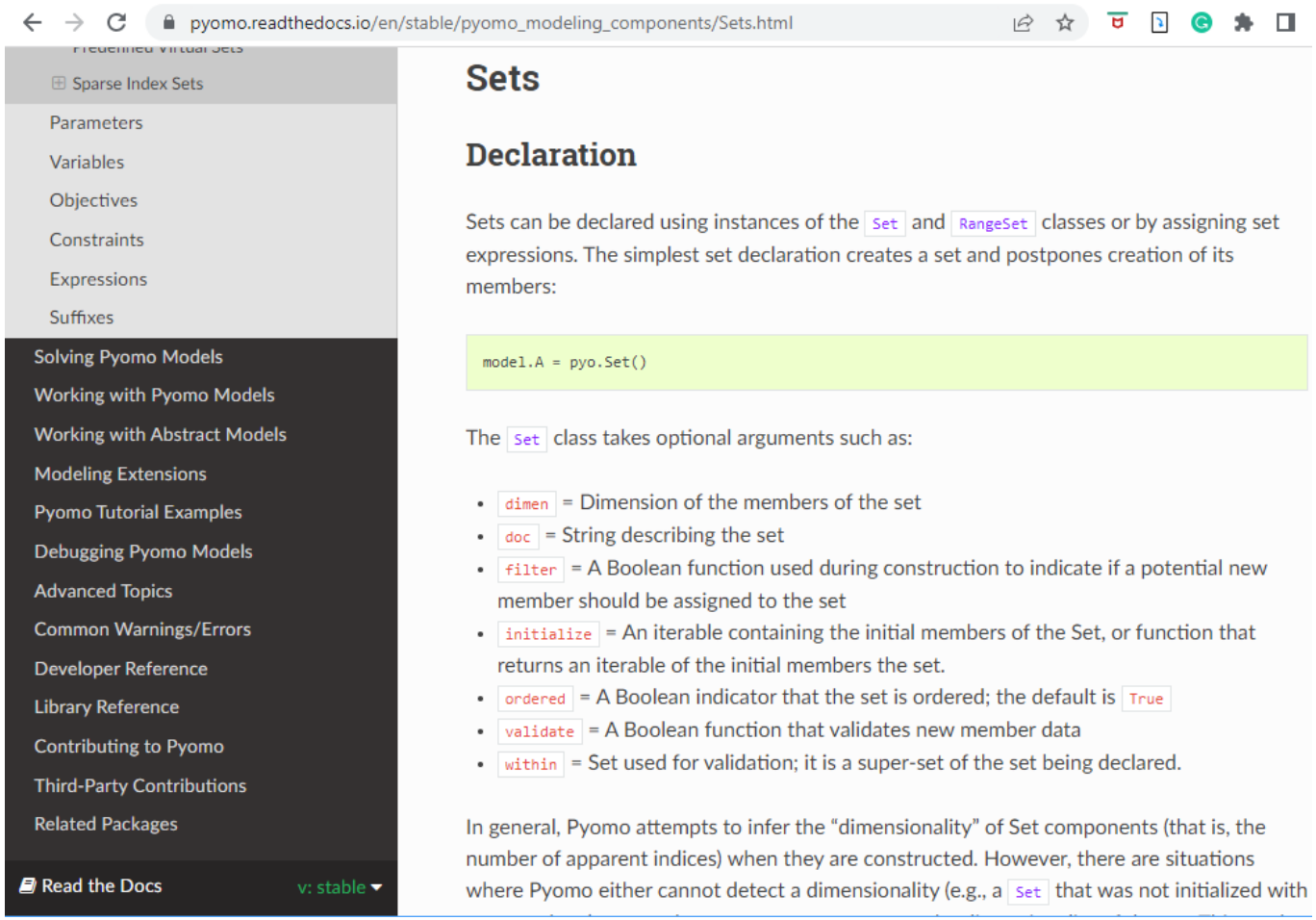
`https://pyomo.readthedocs.io/en/stable/index.html`



Pyomo

- Muito bem documentada!

`https://pyomo.readthedocs.io/en/stable/index.html`



Pyomo

▷ Comentários

- Caso ainda não tenha o Pyomo ou GLPK instalados no Python, basta digitar os comandos a seguir no Anaconda Prompt em uma célula do Jupyter:

```
conda install -c conda-forge pyomo
conda install -c conda-forge glpk
```

No Colab:

```
!pip install -q pyomo
!apt-get install -y -qq glpk-utils
```

Pyomo

▷ Comentários

- Caso ainda não tenha o Pyomo ou GLPK instalados no Python, basta digitar os comandos a seguir no Anaconda Prompt em uma célula do Jupyter:

```
conda install -c conda-forge pyomo
conda install -c conda-forge glpk
```

No Colab:

```
!pip install -q pyomo
!apt-get install -y -qq glpk-utils
```

- Lembre-se que em Python, os índices começam em 0;

Pyomo

▷ Comentários

- Caso ainda não tenha o Pyomo ou GLPK instalados no Python, basta digitar os comandos a seguir no Anaconda Prompt em uma célula do Jupyter:

```
conda install -c conda-forge pyomo
conda install -c conda-forge glpk
```

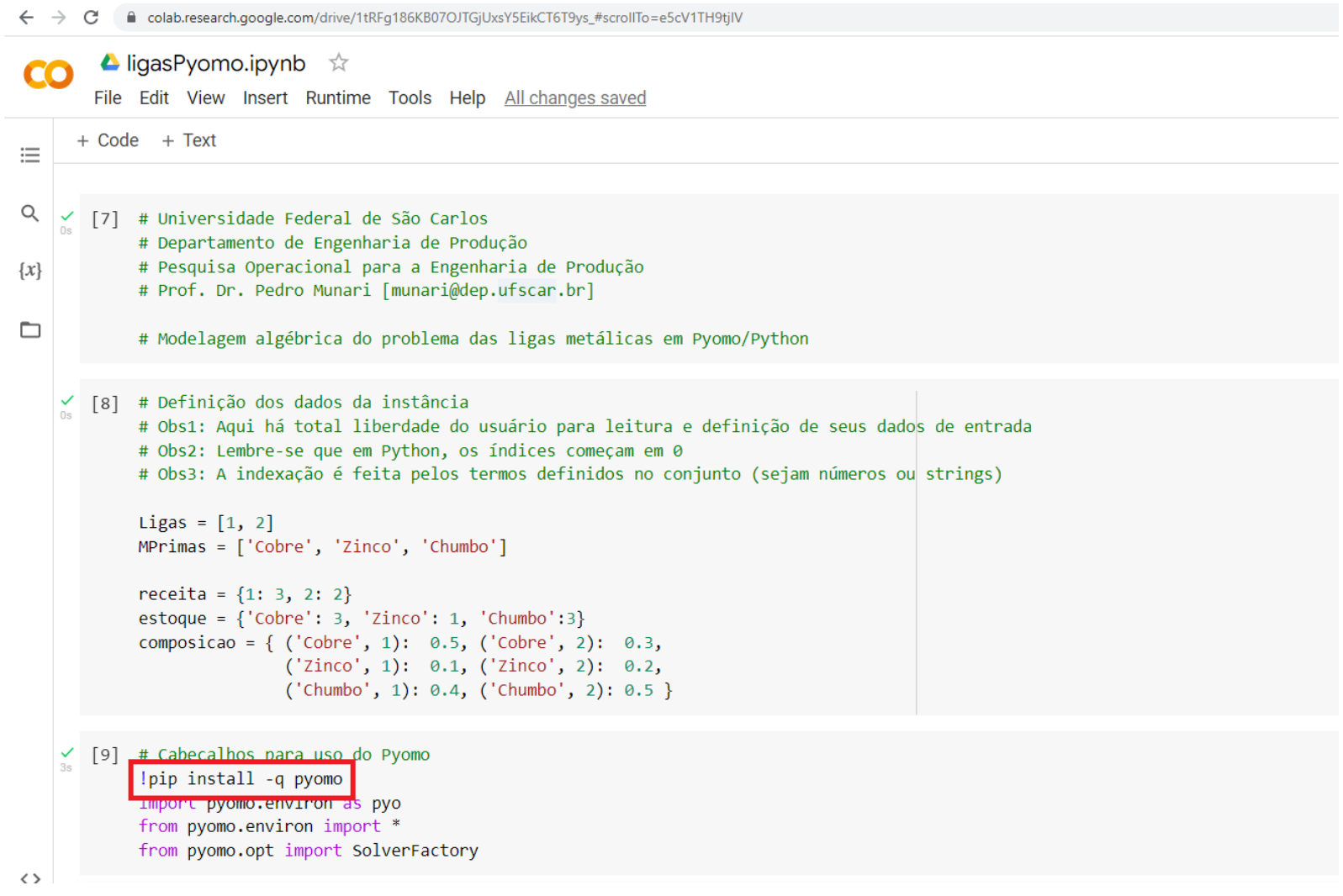
No Colab:

```
!pip install -q pyomo
!apt-get install -y -qq glpk-utils
```

- Lembre-se que em Python, os índices começam em 0;
- A indexação no modelo é feita pelos termos definidos no conjunto (sejam números ou strings).

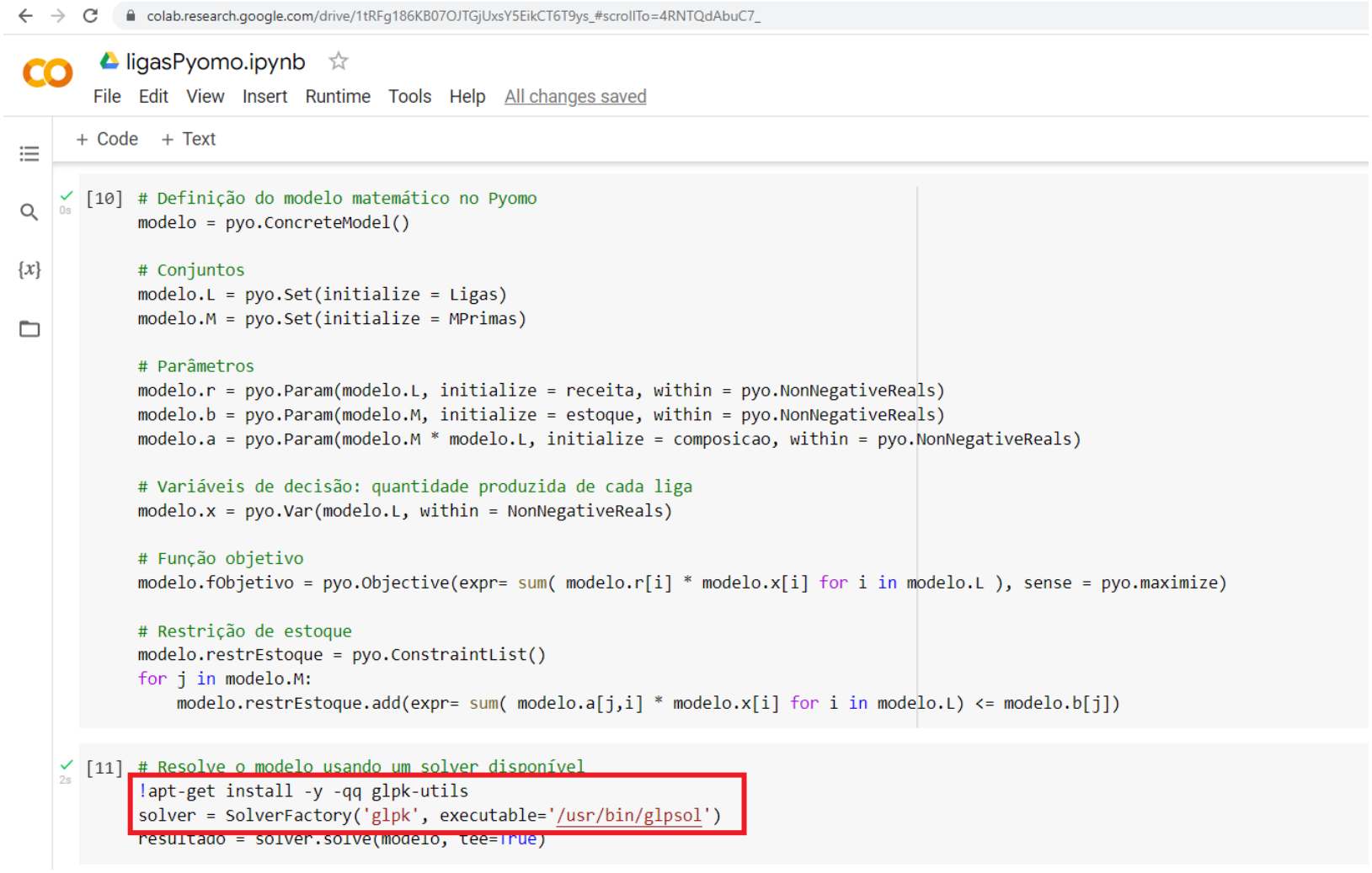
Pyomo

- Disponível também no *Google Colab*



Pyomo

- Disponível também no *Google Colab*



Vantagens e desvantagens

AMPL/GMPL:

- Linguagem intuitiva, simples e muito próxima do modelo matemático;

Vantagens e desvantagens

AMPL/GMPL:

- Linguagem intuitiva, simples e muito próxima do modelo matemático;
- Interfaces pagas e gratuitas, e mais intuitivas de usar;

Vantagens e desvantagens

AMPL/GMPL:

- Linguagem intuitiva, simples e muito próxima do modelo matemático;
- Interfaces pagas e gratuitas, e mais intuitivas de usar;
- Solvers comerciais e livres disponíveis;

Vantagens e desvantagens

AMPL/GMPL:

- Linguagem intuitiva, simples e muito próxima do modelo matemático;
- Interfaces pagas e gratuitas, e mais intuitivas de usar;
- Solvers comerciais e livres disponíveis;
- Permite trocar de solvers facilmente;

Vantagens e desvantagens

AMPL/GMPL:

- Linguagem intuitiva, simples e muito próxima do modelo matemático;
- Interfaces pagas e gratuitas, e mais intuitivas de usar;
- Solvers comerciais e livres disponíveis;
- Permite trocar de solvers facilmente;
- A entrada e saída de dados pode ser uma barreira;

Vantagens e desvantagens

AMPL/GMPL:

- Linguagem intuitiva, simples e muito próxima do modelo matemático;
- Interfaces pagas e gratuitas, e mais intuitivas de usar;
- Solvers comerciais e livres disponíveis;
- Permite trocar de solvers facilmente;
- A entrada e saída de dados pode ser uma barreira;
- Integração com outros softwares/códigos pode ser limitada ou complicada.

Vantagens e desvantagens

Pyomo:

- Linguagem mais próxima de uma linguagem de programação;

Vantagens e desvantagens

Pyomo:

- Linguagem mais próxima de uma linguagem de programação;
- Solvers comerciais e livres disponíveis;

Vantagens e desvantagens

Pyomo:

- Linguagem mais próxima de uma linguagem de programação;
- Solvers comerciais e livres disponíveis;
- Permite trocar de solvers facilmente;

Vantagens e desvantagens

Pyomo:

- Linguagem mais próxima de uma linguagem de programação;
- Solvers comerciais e livres disponíveis;
- Permite trocar de solvers facilmente;
- A entrada e saída de dados pelo Python;

Vantagens e desvantagens

Pyomo:

- Linguagem mais próxima de uma linguagem de programação;
- Solvers comerciais e livres disponíveis;
- Permite trocar de solvers facilmente;
- A entrada e saída de dados pelo Python;
- Integração direta com outros códigos em Python;

Vantagens e desvantagens

Pyomo:

- Linguagem mais próxima de uma linguagem de programação;
- Solvers comerciais e livres disponíveis;
- Permite trocar de solvers facilmente;
- A entrada e saída de dados pelo Python;
- Integração direta com outros códigos em Python;
- Também permite integração direta com Excel, por meio do `xlwings`;

Vantagens e desvantagens

Pyomo:

- Linguagem mais próxima de uma linguagem de programação;
- Solvers comerciais e livres disponíveis;
- Permite trocar de solvers facilmente;
- A entrada e saída de dados pelo Python;
- Integração direta com outros códigos em Python;
- Também permite integração direta com Excel, por meio do `xlwings`;
- Exige conhecimentos em Python! (Importantíssimo hoje em dia!)

Vantagens e desvantagens

Outras linguagens:

- As formas mais eficientes para integração com os solvers é por meio de bibliotecas nativas deles, principalmente quando reotimizações ou mudanças nos modelos são necessárias;

Vantagens e desvantagens

Outras linguagens:

- As formas mais eficientes para integração com os solvers é por meio de bibliotecas nativas deles, principalmente quando reotimizações ou mudanças nos modelos são necessárias;
- Em especial, as bibliotecas em C/C++ são as que oferecem mais recursos, liberdade para o usuário e eficiência computacional;

Vantagens e desvantagens

Outras linguagens:

- As formas mais eficientes para integração com os solvers é por meio de bibliotecas nativas deles, principalmente quando reotimizações ou mudanças nos modelos são necessárias;
- Em especial, as bibliotecas em C/C++ são as que oferecem mais recursos, liberdade para o usuário e eficiência computacional;
- Por outro lado, a dificuldade de usá-las é maior, pois exige conhecimento nessas linguagens.

Linguagem de modelagem algébrica

▷ Exercício 1

Uma empresa produz dois tipos de ligas metálicas. Cada liga é composta de proporções diferentes de Cobre, Zinco e Chumbo, disponíveis em quantidades limitadas em estoque.

Matéria-prima	Liga 1	Liga 2	Estoque
Cobre	50%	30%	3 ton
Zinco	10%	20%	1 ton
Chumbo	40%	50%	3 ton
Preço venda	3 mil	2 mil	(R$ por ton)

Deseja-se determinar quanto produzir de cada liga metálica, de modo a maximizar a receita bruta, satisfazendo-se as composições das ligas e a disponibilidade de matéria-prima em estoque.

(a) Modele o problema algebricamente;

(b) Resolva usando AMPL/GMPL;

(c) Resolva usando Pyomo por meio do Jupyter Notebook.

Linguagem de modelagem algébrica

▷ Exercício 2

Para os Exercícios 1, 3 e 4 da lista da Semana 1:

1. Resolva usando AMPL/GMPL;
2. Resolva usando Pyomo por meio do Jupyter Notebook.

Linguagem de modelagem algébrica

▷ Exercício 3

Duas fazendas (F1, F2) produtoras de carne costumam utilizar três centros de processamento (P1, P2, P3) como intermediários para atender seus principais clientes (C1, C2, C3). Nas tabelas a seguir, são apresentados os custos de transporte por tonelada de carne entre as fazendas, os centros de processamento e os clientes. F1 e F2 podem produzir no máximo 80 e 90 ton, respectivamente. A demanda de C1 é 50 ton, de C2 é 40 ton e de C3 é 65 ton. Deseja-se determinar quanto cada fazenda deve produzir e como deve ser feita a distribuição, de modo a minimizar os custos de transporte. Elabore um modelo de otimização (algébrico) e obtenha uma solução ótima usando AMPL/GMPL ou Pyomo. Descreva a melhor decisão a ser tomada.

De/Para	P1	P2	P3
F1	10	8	30
F2	27	6	11

De/Para	C1	C2	C3
P1	7	5	32
P2	5	4	3
P3	24	8	6

Linguagem de modelagem algébrica

▷ Exercício 4

Uma cooperativa agrícola composta de 3 fazendas, produz 3 tipos de culturas: milho, arroz e feijão. Cada tipo de cultura possui um limite máximo de área total que pode ser usada para sua produção. Além disso, cada cultura demanda uma certa quantidade de água. Para evitar concorrência entre os cooperados, a proporção de área cultivada deve ser a mesma em cada uma das fazendas. As tabelas a seguir apresentam os dados referentes ao consumo e disponibilidade de recursos, requisitos de plantio e lucro.

A cooperativa deseja determinar qual a área a ser usada para plantar cada cultura em cada fazenda, de modo a maximizar o lucro de produção da cooperativa. Elabore um modelo de otimização para apoiar a tomada de decisão da cooperativa e obtenha uma solução ótima usando GMPL ou Pyomo. Descreva a melhor decisão a ser tomada.

Linguagem de modelagem algébrica

▷ Exercício 4

Fazenda	Área total (Acres)	Água disponível (Litros)
1	400	1800
2	650	2200
3	350	950

Cultura	Área máxima (Acres)	Consumo de água (L/Acre)	Lucro (R$/Acre)
Milho	660	5,5	5000
Arroz	880	4	4000
Feijão	400	3,5	1800

Instalação dos softwares

- AMPL: https://ampl.com/start-free-now/
- `Gusek`: `http://gusek.sourceforge.net/`
- *Online-Optimizer*: `https://online-optimizer.appspot.com/`
- PIFOP: `https://pifop.com/`.

Instalação dos softwares

- Para o Python e o Jupyter Notebook, instale o Anaconda:
  `https://www.anaconda.com/download`
- Para instalar o Pyomo e GLPK, basta digitar os comandos a seguir em uma célula do Jupyter:

```
conda install -c conda-forge pyomo
conda install -c conda-forge glpk
```

  No Colab:

```
!pip install -q pyomo
!apt-get install -y -qq glpk-utils
```

- Para instalar o AMPLpy, digite em uma célula do Jupyter:

```
%pip install -q amplpy
```

Vídeo-aulas

- Vídeos com exemplos de como utilizar os softwares e linguagens apresentados nesta aula:

  `https://www.youtube.com/watch?v=sWhYRauuvEA`

  `https://www.youtube.com/watch?v=oGxG4pbQXoY`

  `https://www.youtube.com/watch?v=EJsh0gimhxE`

  `https://www.youtube.com/watch?v=XAwzm_2v4tI`

  `https://www.youtube.com/watch?v=PjVOSzLEa-s`

- Obrigado pela atenção!
- Dúvidas?