DATA SCIENCE
Introduction to GAMS

Antonio VIOLI

antonio.violi@unirc.it

What is GAMS?

- The General Algebraic Modeling System (GAMS) is a high-level modeling system for formulating and solving optimization models
- It is based on the algebraic representation of a mathematical model and provides a set of methods for their solution
- The representation of the model is independent of the machine used and the solver used
- http://www.gams.com/download/

Algebraic Representation

- A model may be defined by
  - the algebraic structure
  - data associated with a specific instance

$$\max \quad \sum_{j \in J} c_j x_j$$

$$\sum_{j \in J} a_{ij} x_j \leq b_i \quad i \in I$$

$$x_j \geq 0 \quad j \in J$$

Structure of a GAMS Model

- The main building blocks are
  - **SETS** - Indices necessary for the definition of the data and the model
  - **DATA** - specified within the model or acquired from external files
  - **VARIABLES and EQUATIONS** - used to define the decision variables and constraints (including the objective function) of the model

Example

$$
\begin{aligned}
\max \quad & Z = 30x_1 + 20x_2 + 10x_3 && (1)\\
& 2x_1 + x_2 + 3x_3 \leq 10 && (2)\\
& x_1 + x_2 + x_3 \leq 2 && (3)\\
& 2x_1 + x_2 + 5x_3 \leq 8 && (4)\\
& x_1, x_2, x_3 \geq 0 && (5)
\end{aligned}
$$

- SETS
  - i = { 1, 2, 3}
  - j = { 1, 2, 3}

Example(2)

- **DATA**
  - $A = \begin{matrix} 2 & 1 & 3 \\ 1 & 1 & 1 \\ 2 & 1 & 5 \end{matrix}$
  - $b^T = [10\ 2\ 8]$
  - $c^T = [30\ 20\ 10]$
- **VARIABLES** x_1, x_2, x_3
- **EQUATIONS** Are defined by (1)-(5)
  The objective function is treated as a constraint

SETS

- **SET** is used to represent the indices of the algebraic representation
- **SET** name comment (if any) /list of elements/;
- In our example
  - **SET** j column index /1,2,3/;
    SET i row index /1,2,3/;
- **SETS**
  j column index /1*3/
  i row index /1*3/;

- To define a set that has the same elements of a set already defined, we may use **ALIAS**

  ALIAS(j, k);

DATA

Numerical data in GAMS are entered as

- **Scalar**: A single real number
- **Parameter**: An indexed data collection of numbers
- **Table**: A syntactically convenient way to declare and initialize a parameter.

Scalar

- **SCALAR** is used to represent a single number
  SCALAR R /0.03/;

  SCALAR rate;

  rate = 1 -R;
- The example shows the declaration and immediate initialization of a scalar, R, and the declaration of another scalar, rate, which is not immediately initialized.
- Initialization can occur in a later assignment statement, as shown.

Parameter

- Parameters are data sets indexed by one or more indices. Indices are sets which are previously declared

  PARAMETER c(j) objective function coefficients / 1 30, 2 20, 3 10/;

  PARAMETER b(i) rhs constraints / 1 10, 2 2, 3 8/;

- It is possible to define a multidimensional paramter
  PARAMETER A(i,j) technology matrix
  / 1.1 2, 1.2 1, 1.3 3, 2.1 1, 2.2 1, 2.3 1, 3.1 2, 3.2 1, 3.3 5 /;

DATA

- A **TABLE** is just a syntactically convenient way to declare a multidimensional parameter.
  TABLE A(i,j) Technology matrix

	1	2	3
1	2	1	3
2	1	1	1
3	2	1	5

- The elements specified in a table must be positioned on the same row and column as the corresponding indices.
- Omitted entries correspond to zeroes.
- You may assign a specific value to an entry also by $A(\text{"1"}, \text{"2"}) = 1;$

Function and operators

GAMS provides a rich set of operators and built-in functions for forming expressions.

Operator	*Description*
`$`	Conditional operator, see page 20
`**`	exponentiation, a^b for $a > 0$ only.
`*, /`	multiplication and division.
`+, -`	addition and subtraction.

`lt, <`
`gt, >`
`eq, =`
`le, <=`
`ge, >=`
`ne, <>`
	less than
	greater than
	equals
	less than or equal to
	greater than or equal to
	not equal to

`not`	Logical Not

`and`
`or`
`xor`
	Logical And
	Logical Or
	Logical Exclusive Or

Functions and operators

Function	Description	Note
`ord`	Ordinate value of index	5
`card`	Cardinality of set	5
`sum`	Summation over set	4
`prod`	Product over set	4
`smin`	Minimum over set	4
`smax`	Maximum over set	4
`errorf(x)`	integral of std. normal from $-\infty$ to x	1
`exp(x)`	exponential, e^x	
`log(x)`	natural log (for $x > 0$)	
`log10(x)`	base-10 log (for $x > 0$)	
`normal(x,y)`	normal distribution; mean x, std.dev y	3
`uniform(x,y)`	uniform distribution in [x, y[	3
`abs(x)`	absolute value	1
`ceil(x)`	smallest integer $\geq$ x	2
`floor(x)`	largest integer $\leq$ x	2
`mapval(x)`	mapping function (see User's Guide)	2
`max(x,y,...)`	maximum of arguments	1
`min(x,y,...)`	minimum of arguments	1
`mod(x,y)`	remainder (modulo)	2
`power(x,y)`	power; y must be an integer	
`round(x)`	rounding to nearest integer	2
`round(x,y)`	rouding to y decimal places	2
`sign(x)`	-1, 0 or 1 depending on the sign of x	2
`sqr(x)`	square of x	
`sqrt(x)`	square root	
`trunc(x)`	rounding towards 0	2
`arctan(x)`	arcus tangent, result in radians	
`cos(x)`	cos, x in radians	
`sin(x)`	sin, x in radians	

The function SUM

The functions **SUM** has two arguments where the first must be a set (or index) expression.

SUM (j,c(j))

Variables

- Variable declarations are used to declare the variables used in a model.
- Variables can be continuous or discrete or some mixture of the two.
- Continuous variables are allowed to take on a range of variables between some (possibly infinite) lower and upper bounds
- Discrete variables must take on an integer value between some finite bounds.

Variable Declarations and Types		Default Bounds	
`Variable`	Continuous	-INF	INF
`Positive Variable`	Continuous	0	INF
`Binary Variable`	Discrete	0	1
`Integer Variable`	Discrete	0	100
`Semicont Variable`	Either 0 or in [LO; UP]	0	INF
`Semiint Variable`	Either 0 or in {LO, LO+1, ..., UP}	0	100

Variables attributes

- After declaration of a variable, it is always possible to change its bounds:
  - POSITIVE VARIABLES x(j);
    x.LO(j) = 1; x.UP(j) = 10;

    x.FX("2") = 8;
- Here, an array of variables is declared as non-negative (default bounds 0 and ∞), but then the bounds are reset to 1 and 10, by setting .LO and .UP attributes.
- By the .FX attribute it is possible to assign a given value to a variable
- In the example, the value of x_2 is set to 8
- The objective function is dealt as continuous variable **VARIABLE**

Equations

- Equations are used to declare and define model constraints

  EQUATIONS constr(i), objective;

  constr(i).. sum(j, A(i,j)*x(j) =L= b(i);
  objective.. z =E= sum(j, c(j)*x(j));

- We declare a set of constraints **constr(i)** and an individual constraint, **objective**.
- They are then defined (indicated by the .. symbol).
- Each of the constraints constr(i) is a less-or-equal inequality constraint, as indicated by =L=
- The objective constraint is an equality indicated by =E=
- Greater-or-equal constraints are specified using =G=

In our example

- **positive variables** x(j);
- **variables** z;
- **equations** constr(i), objective;

- **constr(i)**.. sum(j, A(i,j)*x(j) =L= b(i);
- **objective**.. z =E= sum(j, c(j)*x(j));

Model declaration

- Model declarations serve to collect the constraints and variables that are part of the model, and to name the model.
- **MODEL** mymodel /objective, constr/;
- Between the slashes is listed the names (without indices) of any constraints that should be part of the model mymodel.
- If all the constraints defined in the source file up to this point are part of the model, one can write:
- **MODEL** mymodel /ALL/;

Solution

- The **SOLVE** statement has the general form

  SOLVE modelname MINIMIZING objvar USING modeltype;

  - modelname is the model to be solved
  - objvar is the variable whose value should be minimized (MINIMIZING) or maximized MAXIMIZING)
  - model type indicates the type of model to be solved

- GAMS will select a default solver that is capable of solving the indicated model type, or a desired solver can be specified

  OPTION LP = CPLEX;

Model type

Model Type		Kinds of variables and constraints
LP	Linear Program	linear
MIP	Mixed-integer	linear, discrete
RMIP	Relaxed MIP	as MIP; solved as an LP
NLP	Non-linear Prog	linear, non-linear
DNLP	Discontinuous NLP	linear, non-linear, non-diff. constraints
MINLP	Mixed-Integer NLP	linear, discrete, non-linear
RMINLP	Relaxed MINLP	as MINLP; solved as a NLP
MCP	Mixed Complementarity Problem	complementarity constraints
CNS	Constrained Nonlinear System	LP or NLP without objective

Model type

- The output from GAMS contains many aids for checking and comprehending a model
- Once compiled, if no errors occur, GAMS produces an output file with extension .lst, that contains
  - A copy of the file
  - A description of the model
  - Some statistics on the model
  - The summary of the solution
- In case of errors, a new line marked with **** is added below the line containing the error

Model Statistics

BLOCKS OF EQUATIONS	2	SINGLE EQUATIONS	4
BLOCKS OF VARIABLES	2	SINGLE VARIABLES	4
NON ZERO ELEMENTS	13		
GENERATION TIME =	0.156 SECONDS	3.9 Mb	
EXECUTION TIME =	0.156 SECONDS	3.9 Mb	

Model Statistics

SOLVE SUMMARY

MODEL mymodel OBJECTIVE z

TYPE LP DIRECTION MAXIMIZE

SOLVER CPLEX FROM LINE 26

**** SOLVER STATUS 1 Normal Completion

**** MODEL STATUS 1 Optimal

**** OBJECTIVE VALUE 60.0000

RESOURCE USAGE, LIMIT 0.016 1000.000

ITERATION COUNT, LIMIT 3 2000000000

**** MODEL STATUS 1 **OPTIMAL**
****MODEL STATUS 3 **UNBOUNDED**
****MODEL STATUS 4 **INFEASIBLE**
****MODEL STATUS 8 **INTEGER SOLUTION**

Solution report

Optimal solution found.

Objective : 60.000000

	LOWER	LEVEL	UPPER	MARGINAL
---- EQU objective	.	.	.	1.000

---- EQU constr

	LOWER	LEVEL	UPPER	MARGINAL
1	-INF	4.000	10.000	.
2	-INF	2.000	2.000	30.000
3	-INF	4.000	8.000	.

	LOWER	LEVEL	UPPER	MARGINAL
---- VAR z	-INF	60.000	+INF	.

---- VAR x

	LOWER	LEVEL	UPPER	MARGINAL
1	.	2.000	+INF	.
2	.	.	+INF	-10.000
3	.	.	+INF	-20.000

VARIABLE x.L

1 2.000

VARIABLE z.L = 60.000

EXECUTION TIME = 0.000 SECONDS 3 MB 24.3.3 r48116 WEX-WEI

Display

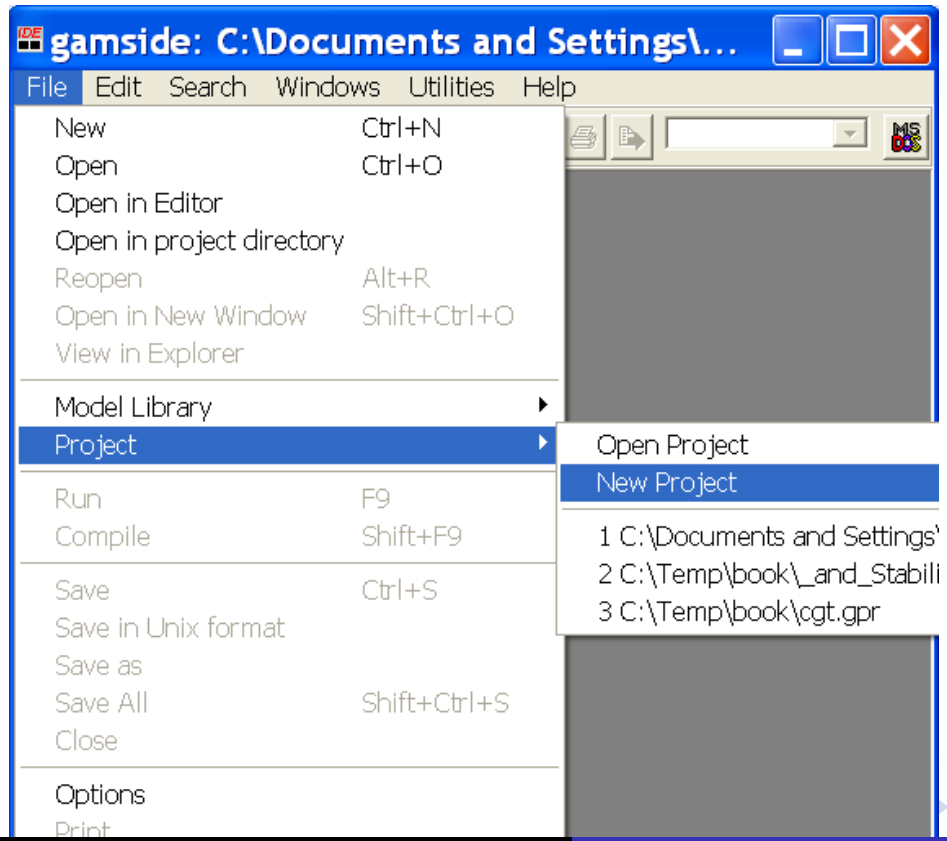
- The easiest way to output data and results is the DISPLAY statement:
- DISPLAY A, c, b;
- Data (scalars and parameters) are specified without indices
- DISPLAY x.l, x.lo, x.up, x.m:
- To display variables, such as x, it is necessary to specify which attribute we wish to display:
  - x.l is the level values
  - x.m the marginal values
  - x.up upper bound
  - X.l lower bound

Getting started

- GAMS provides a graphical user interface (Integrated Development Environment) that facilitates managing the files involved in a GAMS project.

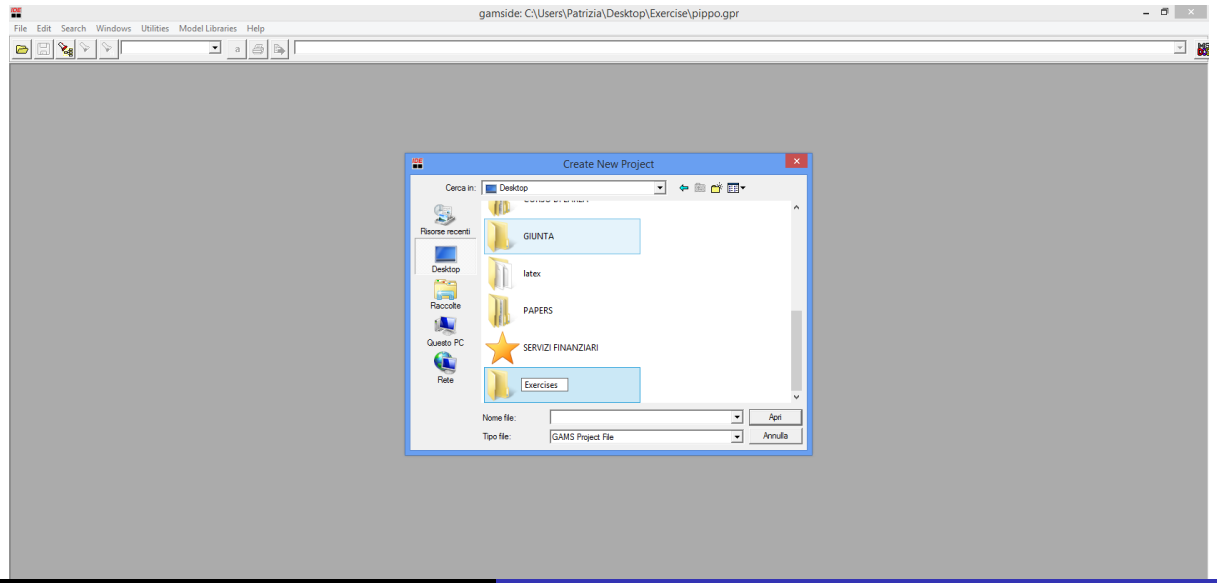
Create New GAMS Project

- Choose from the GAMSIDE: File → Project → New project



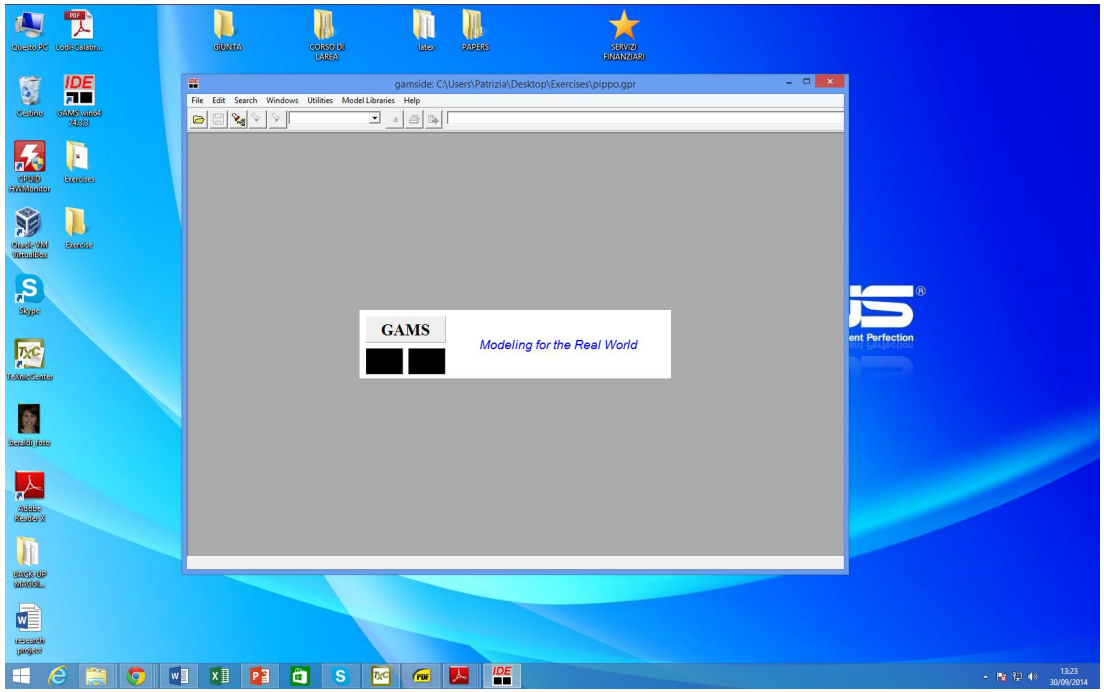
Name New GAMS Project

- On desktop or where you prefer create a new directory by pressing the folder icon.
- Name the new folder Exercises
- Double click on Exercises folder
- Type pippo in the File Name box
- Press Open



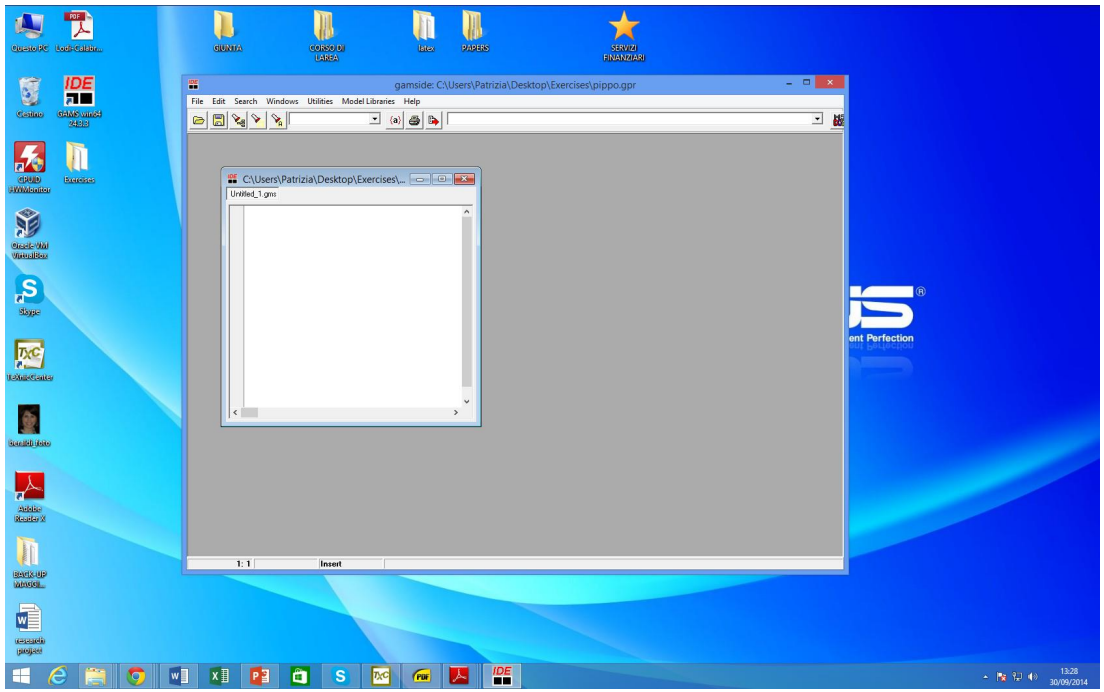
New GAMS Project

The GAMS window should now show the new pippo.gpr project window

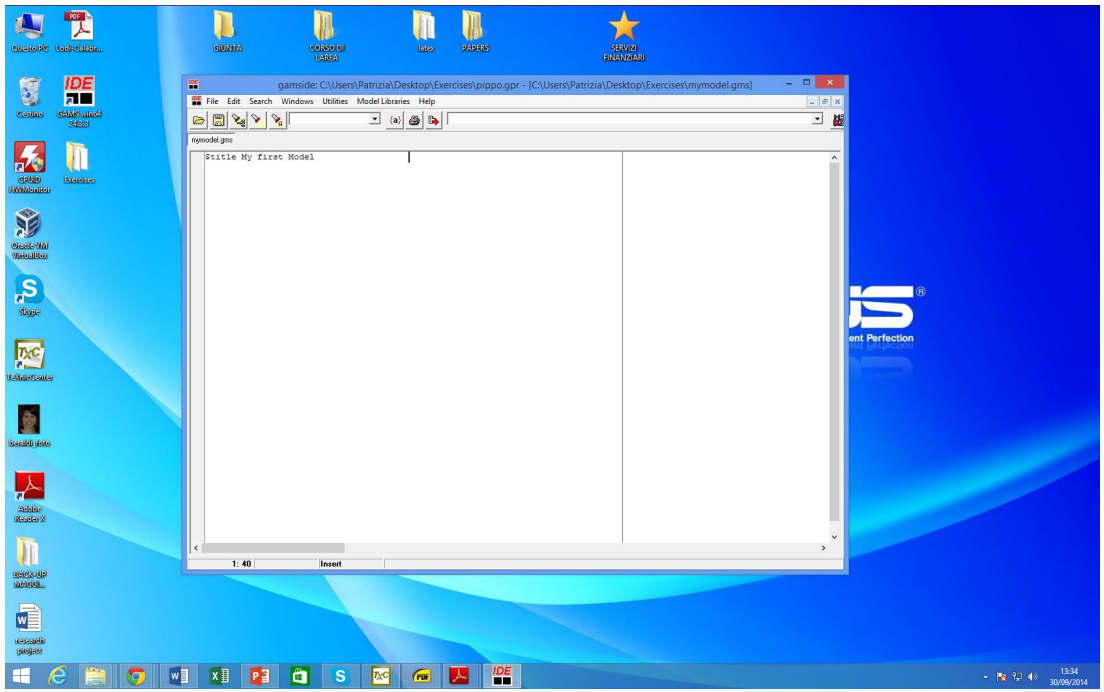


Create a new GAMS file

- Select: File $\rightarrow$ New
- You should see the new file Untitled1.gms



Let's try


gamside: C:\Users\Patrizia\Desktop\Exercises\pippo.gpr - [C:\Users\Patrizia\Desktop\Exercises\mymodel.gms]
File Edit Search Windows Utilities Model Libraries Help
mymodel.gms
$title My first Model
1: 40
Insert