Modeling with GAMS: Part II

University of Chicago
Booth School of Business
Kipp Martin

February 23, 2012
List of Files

- portOptSet.gms
- portOptSetInclude.gms
- portOptSetTable.gms
- portOptData.dat
- callGamsDemo.xlsm
- portOptData.xlsm
- callGAMS.m
- callGamsGetData.m
Outline

Motivation

Sets and Parameters

Separation of Model and Data

Call An Executable from VBA

Interfacing GAMS with Excel

Miscellaneous

Appendix – Windows

Calling and Interfacing with MATLAB
Motivation

Problem: How did we generate generate?

10.06*FS+17.64*IB+32.41*LG+32.36*LV+33.44*SG+24.56*SV = R1
S13.47*FS+7.51*IB+33.28*LG+12.93*LV+3.85*SG-6.70*SV = R3
45.42*FS-1.33*IB+41.46*LG+7.06*LV+58.68*SG+5.43*SV = R4
-21.93*FS+7.36*IB-23.26*LG-5.37*LV-9.02*SG+17.31*SV = R5

By hand? What if I had 500 stocks and several thousand scenarios (say daily returns rather than yearly returns)? Argh!!!
Motivation

In this module we are now going to combine:

- Excel
- VBA
- GAMS
- MATLAB
Motivation

Our Goal:

Excel
Control Macro
Excel
Data Input
Output

GAMS

Excel
Data Input
Output

Excel
Control Macro
Motivation

The first step is to rewrite our portfolio optimization model using sets and parameters.

Rather than write out the constraints “by hand” with all the numbers there, let’s write them in algebraic format:

OK, here we go!
Sets and Parameters

\[
\max \sum_{i=1}^{t} p_i R_i \\
\sum_{j=1}^{n} r_{ji} x_j = R_i, \quad i = 1, \ldots, t \\
R_i \geq \text{ReqRet}_i, \quad i = 1, \ldots, t \\
\sum_{j=1}^{n} x_j = 1
\]

- \( j = 1, \ldots, n \) indexes the set of funds
- \( i = 1, \ldots, t \) indexes the set of scenarios
- \( R_i \), the return variable defined over the set of scenarios
- \( x_j \), the investment quantity variable over the set of funds
Sets and Parameters

\[
\begin{align*}
\max & \sum_{i=1}^{t} p_i R_i \\
\sum_{j=1}^{n} r_{ji} x_j & = R_i, \quad i = 1, \ldots, t \\
R_i & \geq ReqRet_i, \quad i = 1, \ldots, t \\
\sum_{j=1}^{n} x_j & = 1
\end{align*}
\]

- \( r_{ji} \), the return parameter defined over both the fund and scenario sets
- \( p_i \), the probability parameter defined over the set of scenarios
- \( ReqRet_i \), required return parameter defined over the set of scenarios
Sets and Parameters

Concepts:

- A model
- Objective function
- Constraints
- An index over a set
- A variable defined over a set (or sets)
- A parameter defined over a set (or sets)
Sets and Parameters

In GAMS, the equation

$$\sum_{j=1}^{n} r_{ji} x_j = R_i$$

looks like

\[
\text{sum(fund, return(fund, scenario)*x(fund) ) =e= R(scenario)}
\]

Recall

- $j = 1, \ldots, n$ indexes the set of funds
- $i = 1, \ldots, t$ indexes the set of scenarios
Sets and Parameters

In GAMS, the equation

\[ R_i \geq ReqRet_i, \quad i = 1, \ldots, t \]

looks like

\[ R(\text{scenario}) \geq ReqRet(\text{scenario}) \]

Recall

- \( i = 1, \ldots, t \) indexes the set of scenarios
Sets and Parameters

In GAMS, the equation

\[ \sum_{j=1}^{n} x_j = 1 \]

looks like

\[ \text{sum(fund, } x(\text{fund}) ) =e= 1; \]

Recall

- \( j = 1, \ldots, n \) indexes the set of funds
Sets and Parameters

In GAMS, the objective function

$$\max \sum_{i=1}^{t} p_i R_i$$

looks like

$$z = e = \text{sum(scenario, Prob(scenario)*R(scenario))}$$

Recall

- $i = 1, \ldots, t$ indexes the set of scenarios
Sets and Parameters

SETS: In GAMS, we express the concept

- $j = 1, \ldots, n$ indexes the set of funds
- $i = 1, \ldots, t$ indexes the set of scenarios

```
SETS
  fund /ForeignStock, Intermediate-TermBond,
       Large-CapGrowth, Large-CapValue,
       Small-CapGrowth, Small-CapValue/

scenario /Year1, Year2, Year3, Year4, Year5/;
```

SETS is a GAMS keyword
Sets and Parameters

**VARIABLES**: In GAMS, we express the concept

- $R_i$, the return variable defined over the set of scenarios
- $x_j$, the investment quantity variable over the set of funds

as

```
VARIABLES
    R(scenario),
    z "maximize the portfolio return" ;

POSITIVE VARIABLES
    x(fund) "investment levels"
```

**VARIABLES** is a GAMS keyword
Sets and Parameters

PARAMETERS: In GAMS, we express the concept

- $\text{ReqRet}_i$, required return parameter defined over the set of scenarios
- $p_i$, the probability parameter defined over the set of scenarios

as

PARAMETERS

\[
\text{ReqRet}(\text{scenario}) \\
/ \text{Year1 2, Year2 2, Year3 2,} \\
\text{Year4 2, Year5 2} / \\
\]

\[
\text{Prob}(\text{scenario}) \\
/ \text{Year1 .2, Year2 .2, Year3 .2,} \\
\text{Year4 .2, Year5 .2} / ;
\]

PARAMETERS is a GAMS keyword
Sets and Parameters

Let’s examine again the objective function:

\[ z = \sum_{\text{scenario}} \text{Prob}(\text{scenario}) \times R(\text{scenario}) \]

What is:

- \text{scenario}
- \text{Prob}
- \text{R}

How does GAMS evaluate

\[ z = \sum_{\text{scenario}} \text{Prob}(\text{scenario}) \times R(\text{scenario}) \]
PARAMETERS (continued): In GAMS, we express the concept $r_{ji}$, the return parameter defined over both the fund and scenario sets as

```gams
PARAMETERS
return(fund,scenario)
/
ForeignStock.Year1 = 10.06, Small-CapValue.Year1 = 24.56,
ForeignStock.Year2 = 13.12, Small-CapValue.Year2 = 25.32,
ForeignStock.Year3 = 13.47, Small-CapValue.Year3 = -6.7,
ForeignStock.Year4 = 45.42, Small-CapValue.Year4 = 5.43,
ForeignStock.Year5 = -21.93, Small-CapValue.Year5 = 17.31,
.
.
./;

PARAMETERS is a GAMS keyword
```
PARAMETERS (continued): In GAMS, we alternatively express the concept

- $r_{ji}$, the return parameter defined over both the fund and scenario sets

as

```
TABLE return(fund,scenario) "the returns table"
     Year1  Year2  Year3  Year4  Year5
ForeignStock    10.06  13.12  13.47  45.42  -21.93
Intermediate-TermBond 17.64  3.25   7.51  -1.33   7.36
Large-CapGrowth    32.41  18.71  33.28  41.46  -23.26
Large-CapValue     32.36  20.61  12.93   7.06  -5.37
Small-CapGrowth     33.44  19.40  3.85  58.68  -9.02
Small-CapValue     24.56  25.32  -6.7   5.43  17.31
```

**TABLE** is a GAMS keyword
Sets and Parameters

See:

- portOptSet.gms
- portOptSetTable.gms
It is a good idea to break the GAMS model into the **model** and the **data**.

See the file `portOptSet.gms` that illustrates the model and data we have just defined.

I really like to separate things out. See the file `portOptSetInclude.gms` where there is **no data**.

See the file `portOptData.dat` for the data.

In the file `portOptSetInclude.gms` we replace the data with the statement

```
$include portOptData.dat
```
Separation of Model and Data

Here is my **SETS** declaration:

```plaintext
SETS
    fund, scenario;
```

then in the input file **portOptData.dat** I have

```plaintext
SETS
    fund /ForeignStock, Intermediate-TermBond,
         Large-CapGrowth, Large-CapValue,
         Small-CapGrowth, Small-CapValue/

    scenario /Year1, Year2, Year3, Year4, Year5/ ;
```
Separation of Model and Data

Here is my **PARAMETERS** declaration:

\[
\begin{align*}
\text{PARAMETERS} \\
\quad \text{return "scenario returns", ReqRet "min return",} \\
\quad \text{Prob "scenario probabilities";}
\end{align*}
\]

then in the input file **portOptData.dat** I have

\[
\begin{align*}
\text{PARAMETERS} \\
\quad \text{ReqRet(scenario)} \\
\quad \quad \text{/ Year1 2, Year2 2, Year3 2,} \\
\quad \quad \text{Year4 2, Year5 2 /} \\
\quad \text{Prob(scenario)} \\
\quad \quad \text{/ Year1 .2, Year2 .2, Year3 .2,} \\
\quad \quad \text{Year4 .2, Year5 .2 /;}
\end{align*}
\]
Separation of Model and Data

In the `portOptData.dat` file I also have:

```
PARAMETERS
return(fund,scenario)
/
ForeignStock.Year1 = 10.06,       Small-CapValue.Year1 = 24.56,
ForeignStock.Year2 = 13.12,       Small-CapValue.Year2 = 25.32,
ForeignStock.Year3 = 13.47,       Small-CapValue.Year3 = -6.7,
ForeignStock.Year4 = 45.42,       Small-CapValue.Year4 = 5.43,
ForeignStock.Year5 = -21.93,      Small-CapValue.Year5 = 17.31,
.                                .
.                                .
.                                .
/
```
Develop a Style: Here is my style.

- Part I: The Model

- Part II: Data – with an include.

- Part III: The Solution
Separation of Model and Data

Part I: The Model Components – see page 29 of User’s Guide

- Set Declarations
- Parameter Declarations
- Variable Declarations
- Equation Declarations
- Equation Definitions
- Model Definitions
Separation of Model and Data

Part II: The Data Components – see page 29 of User’s Guide

- Set Definitions
- Parameter Definitions
- Assignments
- Displays

Note the difference between definition and declaration.
Separation of Model and Data

Part III: The Solution Components – see page 29 of User’s Guide

- Solver options
- Solve
- Displays

Develop your own style!
See the example \texttt{callGamsDemo.xlsm}.

**Key Idea:** Call GAMS from inside an Excel spreadsheet.

We use (Windows and Mac) the following code:

\begin{verbatim}
result = Shell(executablePath, vbNormalFocus)
\end{verbatim}

**\texttt{Shell}:** A VBA function that can call an executable program on your computer.
Calling an Executable

**Shell**: A VBA function that can call an executable program on your computer. The function has two arguments

- **Argument 1**: the path to the executable

- **Argument 2**: what to do with the command window – use `vbNormalFocus`

**Critical to have the path variable correct.**
Calling an Executable

For Windows

gamsExe = "C:\Program Files\GAMS23.6\gams.exe "
gamsMod = " C:\gamsdir\portOptExcel\portOpt.gms "
gamsOut = "Output=C:\gamsdir\portOptExcel\result.txt"

gamsExecPath = gamsExe & gamsMod & gamsOut

For Mac OS X

gamsExecPath = "Macintosh HD:Users:kmartin:runGams.sh"

Don’t forget in the terminal window do

chmod u+x runGams.sh
Important: The Shell command issues an asynchronous call to the executable.

What is the difference between a synchronous and asynchronous call?

We add the code:

```plaintext
hourTime = Hour(Now())
minuteTime = Minute(Now())
secondTime = Second(Now()) + 3
waitTime = TimeSerial(hourTime, minuteTime, secondTime)
```
Now for some serious fun!

Clearly writing all the numbers into `portOptData.dat` is not something you ever want to do by hand.

We now get those data from an Excel spreadsheet.

We also write the result to a spreadsheet. Is this neat or what.
Interfacing GAMS with Excel

Here is the concept:

![Diagram showing the interaction between GAMS, Excel Control Macro, and Excel Data Input Output.]
Interfacing GAMS with Excel

Here is the workflow (implemented in portOptData.xlsm):

**Step 1:** Use VBA to process the necessary data and write the file (e.g. portOptData.dat) that the GAMS model needs.

**Step 2:** Use VBA to call GAMS.

**Step 3:** Use VBA to read the GAMS output file.
In `portOptData.xlsm` we have:

```vba
Sub Main()
    Call GenData
    Call RunGAMS
    Call GetModelResult
End Sub
```

The **GenData** procedure:

- Read the spreadsheet **Data** and put the information into a string.

- Open a file **portOptData.dat** and write the string to the file.
The **GenData** procedure uses several VBA concepts we have studied:

- Reads information from named ranges into a string
- Opens a text file
- writes the string to the text file
The procedure **RunGAMS**

- Call the GAMS executable (use code from `callGamsDemo.xlsm`)

- The GAMS executable reads `portOptData.dat`

- The GAMS executable solves the portfolio optimization problem

- The GAMS executable writes the solution to the model to the file `portOptSetAns.dat`
The procedure `GetModelResult`

- Open GAMS output file `portOptSetAns.dat`
- Read the GAMS output file `portOptSetAns.dat` into a string
- Parse the string and put contents into an array
- Write the array to the spreadsheet `Result`
Interfacing GAMS with Excel

Run the example **portOptData.xlsm**. Be careful!

- I assume the GAMS executable is in the directory
  C:\Program Files\GAMS23.7

- I have all of the other files in
  C:\gamsdir\portOptExcel\
A good overview of using GAMS: http://www.gams.com/mccarl/mccarlhtml/index.html?

See the tutorial:

docs/gams/Tutorial.pdf

See the full-blown user guide:

docs/bigdocs/GAMSUsersGuide:

If you are familiar with the **PATH** command, putting the GAMS executable in your PATH command is a good idea.

You may wish to create a project for your GAMS model (Windows IDE only)

Make use of .lst file for debugging in the IDE.

Make use of absolute rather than relative paths

See **finlib_ml** for numerous financial models and data sets

GAMS might be useful in your other coursework
In order to call GAMS from Excel you need to be running as an Administrator. Here are two fixes:

**Fix 1: Always Open Excel “As Administrator”**

1. Navigate to the Excel executable. Excel is often located in "C:\Program Files\Microsoft Office\Office12\excel.exe" or something similar.

2. Right click on excel.exe and select ”run as administrator”.
In order to call GAMS from Excel you need to be running as an Administrator. Here are two fixes:

**Fix 2: More Generic Solution**

1. Go to your control panel
2. Click on turn on/off user account control
3. Make sure that user account control is turned off
4. Do the Windows thing and reboot

My thanks to Miles Loh for this information.
Calling GAMS

It is easy to call GAMS from MATLAB. (See the m-file callGAMS.m).

You can call any executable from MATLAB. Use an `!` before the command.

```
!"C:\Program Files\Gams23.7\gams.exe" C:\gamsdir\portOptExcel\portOpt.gms
```

**In-class Exercise:** Download `callGAMS.m` and run it. It should create an output file named `portOptAns.dat`. 
We want GAMS to write output to a simple text file that we read with MATLAB. This is illustrated in `portOpt.gms`

```plaintext
file portOptAns /C:\gamsdir\portOptExcel\portOptAns.dat/;
put portOptAns
put 'z,' z.l /
put 'FS,' FS.l/
put 'IB,' IB.l/
put 'LG,' LG.l/
put 'LV,' LV.l/
put 'SG,' SG.l/
put 'SV,' SV.l/
putclose portOptAns
```
Calling GAMS

\[
\begin{align*}
z, & \quad 17.33 \\
FS, & \quad 0.00 \\
IB, & \quad 0.00 \\
LG, & \quad 0.11 \\
LV, & \quad 0.00 \\
SG, & \quad 0.41 \\
SV, & \quad 0.48 \\
\end{align*}
\]

**Key Idea:** We now read `portOptAns.dat` in MATLAB.
Calling GAMS

fid = fopen('portOptAns.dat');
%read the first line
%get the optimal value
    fscanf(fid, ’%[^,]’ , 1)
    fscanf(fid, ’%[,]’ , 1)  %burn off comma
    optimalValue = fscanf(fid, ’%f’ , 1)
fgetl(fid)
kount = 1
while ~feof(fid);

    stock = fscanf(fid, ’%[^,]’ , 1)
    fscanf(fid, ’%[,]’ , 1)  %burn off comma
    investment(kount, 1) = fscanf(fid, ’%f’ , 1)
    kount = kount + 1
    fgetl(fid)
end
Calling GAMS

**In-class Exercise:** Combine

- `callGAMS.m`
- `callGamsGetData.m`

and run the GAMS model and get the answer into the `investments` array.