1 Contents

Contents

XLS2GMS, Version 2.4, May 2004
Erwin Kalvelagen, GAMS Development Corp

Note: the bitmaps are best viewed with Large Fonts display settings.

2 Overview

Overview

XLS2GMS is a tool to convert spreadsheet data from a *Microsoft Excel spreadsheet* into GAMS readable format. The source is a MS Excel spreadsheet file (*.XLS) and the target is a *GAMS Include File*.

When running the executable **XLS2GMS.EXE** without command line parameters the tool will run interactively with a built-in GUI interface. Alternatively XLS2GMS can be run in batch mode which is useful when calling it directly from a GAMS model using the *$call* command.

The philosophy of the tool is to consider the content of a spreadsheet as **Text**. This text can contain GAMS statements, or parts of GAMS statements (e.g. the data part of a table statement). The text is exported to a GAMS include file where some spacing is introduced to maintain cell boundaries. This allows tables to be exported directly to GAMS include files.

XLS2GMS is written in Delphi (Pascal). The source is available upon request.

3 Requirements

Requirements

XLS2GMS runs only on PC's running Windows (95/98/NT/XP) and with MS Excel installed. Microsoft Excel is included in the MS Office suite.

4 Converting spreadsheet data to GAMS data

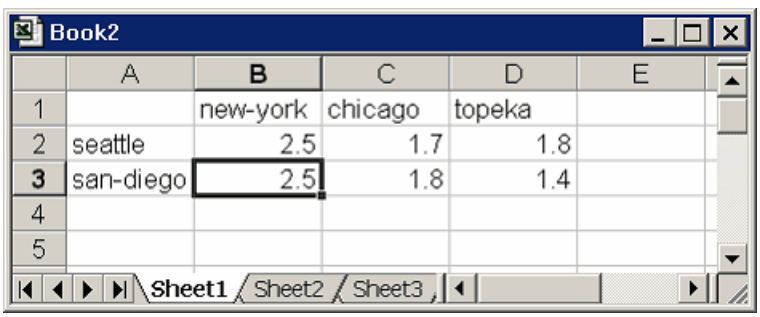
Converting spreadsheet data to GAMS data

Spreadsheet data are often differently organized than is suitable for import into a GAMS model. In some cases the data is scattered around different sheets, and in a format that is not compatible with a more structured multi-dimensional parameter as are used in a GAMS model. To export spreadsheet data to GAMS parameters, tools will either require a strict format to be used inside the spreadsheet or they will need to offer a complex specification step where the data representation in the spreadsheet is described so that it can be understood by the tool. This tool will use the first approach: the modeler is required to lay-out the data in the spreadsheet in a well defined format. Instead of defining a new format, we use the GAMS language syntax as the required representation. In effect the spreadsheet is considered as an alternative editor for GAMS source code.

As an example consider the GAMS table in the model **trnsport** which is part of the GAMS model library:

```
Table d(i,j)  distance in thousands of miles
                 new-york       chicago      topeka
    seattle         2.5           1.7          1.8
    san-diego       2.5           1.8          1.4  ;
```

This table can be expressed comfortably in a spreadsheet as follows:



	A	B	C	D	E
1		new-york	chicago	topeka	
2	seattle	2.5	1.7	1.8	
3	san-diego	2.5	1.8	1.4	
4					
5					

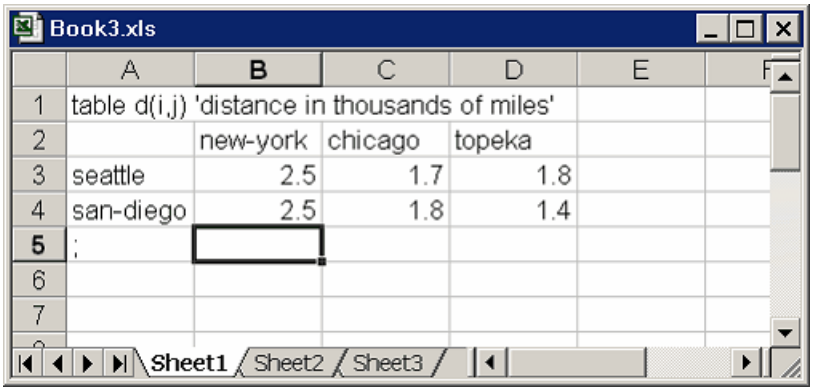
XLS2GMS can convert this table into a GAMS include file, which results in:

```
* -----------------------------------------------------
* XLS2GMS Version 2.4,  May 2004
* Erwin Kalvelagen, GAMS Development Corp.
* -----------------------------------------------------
* Application: Microsoft Excel
* Version:     9.0
* Workbook:    D:\gams projects\xls2gms\ver2.0\Book2.xls
* Sheet:       Sheet1
* Range:       $A$1:$D$3
* -----------------------------------------------------
          new-york chicago topeka
seattle   2.5      1.7     1.8
san-diego 2.5      1.8     1.4
* -----------------------------------------------------
```

The tool will try to keep cells in a column aligned so that table statements can be used in the GAMS model. The above file **book2.inc** can be imported directly into a GAMS model by:

```
Table d(i,j)  distance in thousands of miles
$include book2.inc
;
```

As the tool does not expect any special formatting, we could have include the table statement in the spreadsheet, as in:



This would result in:

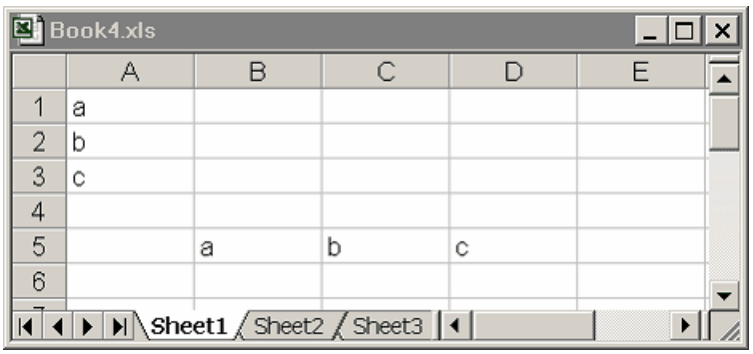
```
* ----------------------------------------------------
* XLS2GMS Version 2.4,  May 2004
* Erwin Kalvelagen, GAMS Development Corp.
* ----------------------------------------------------
* Application: Microsoft Excel
* Version:     9.0
* Workbook:    D:\gams projects\xls2gms\ver2.0\Book3.xls
* Sheet:       Sheet1
* Range:       $A$1:$D$5
* ----------------------------------------------------
table d(i,j) 'distance in thousands of miles'
          new-york chicago topeka
seattle   2.5      1.7     1.8
san-diego 2.5      1.8     1.4
;
* ----------------------------------------------------
```

In some cases the data will need to be copied and massaged to fit into the above format. It is convenient to add a sheet dedicated for this purpose to your workbook. This interface sheet can be filled either manually, with formulas that automatically update values, or by macro's (either recorded or programmed in VBA).

5 Importing sets

Importing sets

Sets can be directly imported if they are organized vertically. The following picture shows a spreadsheet with two sets with elements {a,b,c} organized vertically (A1:A3) and horizontally (B5:D5).



The first set can be imported directly using:

```
set i /
$call =xls2gms r=a1:a3 i=book4.xls o=set1.inc
$include set1.inc
/;
display i;
```

The second set is somewhat more difficult, as we need to add a separating comma between the elements. This can be accomplished by:

```
set j /
$call =xls2gms r=b5:d5 s="," i=book4.xls o=set2.inc
$include set2.inc
/;
display j;
```

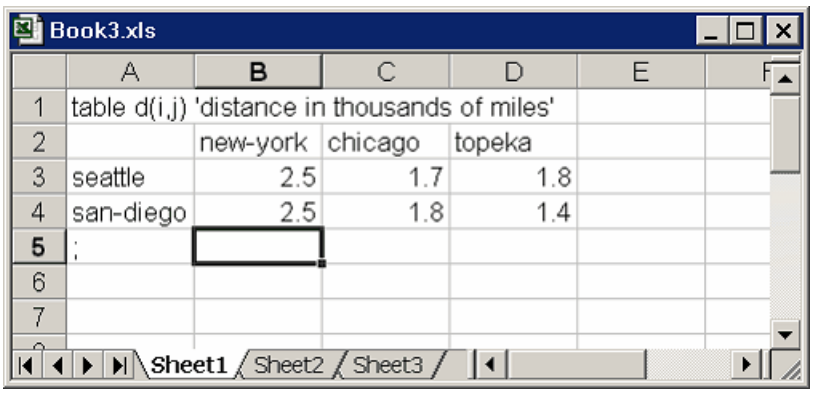
This will generate the include file:

```
* ----------------------------------------------------
* XLS2GMS Version 2.4,  May 2004
* Erwin Kalvelagen, GAMS Development Corp.
* ----------------------------------------------------
* Application: Microsoft Excel
* Version:     9.0
* Workbook:    D:\gams projects\xls2gms\ver2.0\Book4.xls
* Sheet:       Sheet1
* Range:       $B$5:$D$5
* ----------------------------------------------------
a,b,c
* ----------------------------------------------------
```

6 Importing sets and tables

Importing sets and tables

The table



	A	B	C	D	E
1	table d(i,j) 'distance in thousands of miles'				
2		new-york	chicago	topeka	
3	seattle	2.5	1.7	1.8	
4	san-diego	2.5	1.8	1.4	
5	;				
6					
7					

Sheet1 Sheet2 Sheet3

can be considered to contain three pieces of GAMS data:

- The set i (seattle, san-diego)
- The set j (new-york, chicago, topeka)
- The distances

All this information can be read as follows:

```
set i /
$call =xls2gms r=a3:a4 i=book3.xls o=seti.inc
$include seti.inc
/;

set j /
$call =xls2gms r=b2:d2 s="," i=book3.xls o=setj.inc
$include setj.inc
/;

table d(i,j)
$call =xls2gms r=a2:d4 i=book3.xls o=pard.inc
$include pard.inc
;

display i,j,d;
```

The above $call statements can be combined onto one as follows:

```
$onecho > book3a.txt
i=%system.fp%book3.xls
r1=a3:a4
o1=seti.inc
r2=b2:d2
o2=setj.inc
s2=","
r3=a2:d4
o3=pard.inc
$offecho

$call =xls2gms @book3a.txt

set i /
$include seti.inc
/;

set j /
$include setj.inc
/;

table d(i,j)
$include pard.inc
;

display i,j,d;
```

The command file generated by the $onecho statement looks like:

```
i=E:\wtools\ver000\examples\book3.xls
r1=a3:a4
o1=seti.inc
r2=b2:d2
o2=setj.inc
s2=","
r3=a2:d4
o3=pard.inc
```

7 Multidimensional parameters

Importing multidimensional parameters

Consider the data table:

data.txt

	A	B	C	D	E	F	G	H	I	J
1	livestock yield time series (kg per head)									
2					1974	1975	1976	1977	1978	1979
3										
4		sheep	.	meat	10.6	11.42	10.6	9.38	8.97	6.93
5		sheep	.	milk	23.7	24.1	24.2	24.2	24	23.9
6		sheep	.	wool	1.3	1.3	1.3	1.3	1.3	1.3
7		sheep	.	hide	0.5	0.6	0.6	0.5	0.6	0.4
8		goat	.	meat	6.39	7.31	8.68	7.31	6.39	6.85
9		goat	.	milk	37.7	38.1	38.2	38.2	38.3	37.8
10		goat	.	wool	0.6	0.6	0.6	0.6	0.6	0.6
11		goat	.	hide	0.2	0.3	0.3	0.3	0.2	0.3
12		angora	.	meat	1.77	1.77	2.66	2.21	1.77	1.77
13		angora	.	milk	14.9	15.2	14.8	15.2	14.8	15
14		angora	.	wool	1.6	1.6	1.6	1.6	1.6	1.4
15		angora	.	hide	0.1	0.1	0.1	0.1	0.1	0.1
16		cattle	.	meat	24.59	25.12	21.42	23	18.25	25.12
17		cattle	.	milk	210	208.1	219.8	213.8	214.8	217.5
18		cattle	.	hide	3.3	3.4	2.9	3	2.6	3.3
19		buffalo	.	meat	43.73	45.42	40.61	37.21	32.2	32.68
20		buffalo	.	milk	267.1	269.2	263.8	219.6	275.5	285.1
21		buffalo	.	hide	4.1	3.4	3	2.4	2.5	2.6
22		poultry	.	meat	2.24	2.24	2.24	2.24	2.24	2.24
23		poultry	.	egg	62.4	62.2	64.2	78.3	76.4	73.3
24										

data

In this spreadsheet the first two columns are index columns. To make this valid GAMS syntax we need to insert a dot between the index elements. A simple way is to insert a (narrow) column with a dot in each cell. This way we can import this table as:

```
Sets
  l    'livestock types'  /sheep,goat,angora,cattle,buffalo,mule,poultry/
  cl   'livestk comm'     /meat,milk,wool,hide,egg/
  ty   'time periods - years' / 1974*1979 /
;


$onecho > yield.txt
I="%system.fp%yield.xls"
R=data!B2:J23
O=yield.inc
$offecho

$call =xls2gms @yield.txt


Table yieldtl(l,cl,ty)  livestock "yield" time series (kg per head)
$include yield.inc
;

display yieldtl;
```

8 interactiveuse

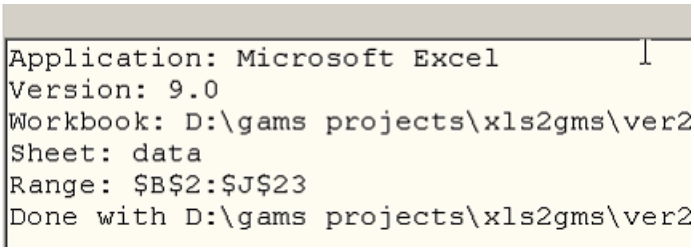
Interactive use

When the tool is called without command line parameters, it will startup interactively. Using it this way, one can specify the spreadsheet file (.XLS file), the range and the final destination file (a GAMS include file) using the built-in interactive environment. The main screen contains a number of buttons and edit boxes, which are explained below.

Input file (*.XLS). This is the combo box to specify the input file. The file must be a valid MS Excel spreadsheet file (*.XLS). The browse button can be used to launch a file open dialog which makes it easier to specify a file. The file may be located on a remote machine using the notation *\\machine\directory\file.xls*.

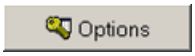
Range. The range can be left empty in which case the whole first sheet is taken. Otherwise the range can be a single cell (e.g. A1), a block (e.g. B2:J23), or a region within a sheet (e.g. Sheet1!A1:C10). The range can also be a name if the spreadsheet contains named ranges. The browse button will start Excel allowing you to interactively select a range.

Output GAMS Include file (*.INC). If you want to create a GAMS include file, then specify here the destination file. The include file will be an ASCII file that can be read by GAMS using the *$include* command. If the include file already exists, it will be overwritten.



Progress Memo. This memo field is used to show progress of the application. Also error messages from the database are printed here. This is a read-only field.

The edit boxes above all have a drop down list which can be used to access quickly file names and queries that have been used earlier (even from a previous session).



options button will pop up a window where you can specify a number of options.

help button will show this help.

the **OK button** is pressed the query will be executed and an include file will be generated.

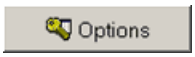
the **batch button** will give information on how the current extract command can be executed directly from GAMS in a batch environment. The batch call will be displayed and can be copied onto the clipboard. In the IDE press Ctrl-C or choose Edit|Paste to copy the contents of the clipboard to a GAMS text file.

close button will exit the application. The current settings will be saved in an INI file so when you run XLS2GMS again all current settings will be restored.

9 Options

Options

The **Options** window can be created by pressing the options button:



The following options are available in the options window:

Quote blanks: Quote strings if they contain blanks or embedded quotes. If a string does not contain a blank or embedded quotes, it will remain unquoted. If the string contains a single quote the quotes applied will be double quotes and if the string contains already a double quote, the surrounding quotes will be single quotes. (In the special case the string contains both, double quotes are replaced by single quotes). For more information see this example.

Mute: Don’t include the extra informational text (such as used query etc.) in the include file.

No listing: Surround the include file by *$offlisting* and *$onlisting* so that the data will not be echoed to the listing file. This can be useful for very large data files, where the listing file would become too large to be handled comfortably.

Separator. This option allows you to set a separator string to be written between cell entries. By default this is a blank. In some cases it can be useful to make this a comma. See example for an example where this is used to import sets. Note: when this option is set to an empty string, the results may not be syntactically correct for GAMS. As it is difficult to see the difference between a single blank and an empty string, the user interface will give some feedback for these cases. When an empty string is used, a warning is written to the include file.

Range Seperator. Multiple ranges can be separated by a range separator symbol. By default this is a semi-colon. When certain non-US locales are used, the semi-colon is a list separator symbol which can be used in multi-area ranges. In case of such a conflict, it is possible to change the range separator symbol.

Append. Append to the output file .

Browse Read-Only. When the Browse Range button is pressed, we launch Excel and try to load the currently specified input file. If this option is checked then the input file is loaded as read-only. If this option is not checked the file is loaded normally, in which case you can change and save it.

The buttons have an obvious functionality:

OK button will accept the changes made.

Cancel button wil ignore the changes made, and all option settings will revert to their previous values.

Help button will show this help text.

10 Batch use

Batch use

When calling **XLS2GMS** directly from GAMS we want to specify all command and options directly from the command line or from a command file. An example is:

```
C:\tmp>xls2gms "I=c:\My Documents\test.xls" O=test.inc
```

This call will perform its task without user intervention. The batch facility can be used from inside a GAMS model, e.g.:

```
table c(i,j) 'data from spreadsheet' /
$call =xls2gms I=C:\tmp\test.xls O=C:\tmp\data.inc R=B1:E10
$include C:\tmp\data.inc
/;
```

The $call statement is rather error prone and you will need to spend a little it of time to get the call correct and reliable.

All the possible command line options are listed in command line arguments section. A proper **batch**

call will at least contain the following command line parameters:

1. I=inputfilename. O=outputincludefile

If you only specify **I=inputfilename** then the interactive user interface is started with an initial setting of the input file name edit box equal to the name given in the command line argument. Only if an input file and an output file is provided, the call will be considered a batch invocation.

11 Command-line Arguments

Command line arguments

I=inputfile

This option is required and specifies the name of the .XLS file containing the Access database. If the file contains blanks the name should be surrounded by double quotes. It is advised to use absolute paths, so Access has no confusion what file to open. If only the file name is used without a path, the file is searched in the current directory (this is the project directory when running under the IDE). On a network UNC names can be used, and files from another computer can be accessed, e.g. "\\hostname\c\my documents\a.xls." This option is required for batch processing.

O=outputincludefile

option specifies the name of the output file. The format of the output file will be a GAMS include file for a parameter statement. Make sure the directory is writable. UNC names can be used. An output file must be specified for batch operation.

R=range

range is an optional argument. If not specified the whole first sheet is taken. Otherwise the range can be a single cell (e.g. **A1**), a block (e.g. **B2:J23**), or a region within a sheet (e.g. **Sheet1!A1:C10**). To specify a whole sheet use: **Sheet2!**. The range can also be a name if the spreadsheet contains named ranges. Both global names (e.g. **R=rangename**) and sheet specific range names (e.g. **R=Sheet2!rangename**) are recognized.

To select multiple ranges in one go, you can specify: **R=range1;range2;range3**. This is just a short-hand for three separate invocations of **xls2gms**. A multiple-area range can be specified by **R=area1,area2,area3**. Before exporting, a new range is created consisting of the union of the areas. This can be used to drop certain rows or columns from a table.

D

Debug. This option can be used for debugging purposes. If specified the import filter will no run minimized but "restored", i.e. as a normal window. In addition the program will not terminate until the user clicks the Close button. This allows you to monitor possible errors during execution of **xls2gms**.

B If this parameter is specified, strings that have blanks in them will be quoted. If the string is already quoted this step is not performed. If the name contains an embedded single quote, the surrounding quotes will be double quotes. If the name already contains a double quote, the surrounding quotes will be single quotes. If both single and double quotes are present in the string, then all double quotes are replaced by single quotes and the surrounding quotes will be double quotes. By default this option is turned off.

M Run in modest or mute mode: no additional information, such as version number etc. is written to the listing file.

L Embed the data in $*offlisting*, $*onlisting*. A quick way to reduce the size of the listing file.

@filename
@"file name" Causes the program to read options from a file. If the file name contains blanks, it can be surrounded by double quotes. The option file contains one option per line, in the same syntax as if it were specified on the command line.

N=inifilename Use a different Inifile than the standard **xls2gms.ini** located in the same directory as the executable **xls2gms.exe**.

A Append to output files instead of overwriting them.

G="x" Sets the range separator symbol

S="x" Sets the output separator symbol

As invocations of **xls2gms** are reasonably expensive (a copy of Excel is started), there is a way to optimize related calls. From version 1.4, xls2gms allows multiple ranges to be read and multiple include files to be written in one swoop. The syntax is best explained by showing an illustrative example:

```
$call =xls2gms I=c:\tmp\x.xls R1=range1 R1=range2 R2=range3 O1=c:\tmp\f1.inc
O2=c:\tmp\f2.inc
```

In this example the ranges 'range1' and 'range2' are written to the file 'f1.inc' while the range 'range3' will go to file 'f2.inc'. In general the ranges specified with **R*n*** will be written to the file specified with **O *n***. If multiple ranges are specified, they are written sequentially to the output file.

```
$call =xls2gms I=c:\tmp\x.xls R=range1 R=range2 O=c:\tmp\f1.inc
```

In this example the ranges 'range1' and 'range2' are written to 'f1.inc'.

12 $CALL command

The $CALL command

The **$CALL** command in GAMS will execute an external program at compile time. There are two forms:

```
$call externalprogram
$call =externalprogram
```

The version without the leading '=' calls the external through the command processor (command.com

or cmd.exe). The second version with the '=', bypasses the command processor and directly executes the external program. We mention some of the differences:

1. Some commands are not external programs but built-in commands of the command processor. Examples are COPY, DIR, DEL, ERASE, CD, MKDIR, MD, REN, TYPE. If you want to execute these commands you will need to use the form `$call externalprogram` which uses the command processor.. If you want to execute a batch file (.bat or .cmd file) then you will need to use the form `$call externalprogram`. If it is important to stop with an appropriate error message if the external program does not exist, only use the form `$call =externalprogram`. The other form is not reliable in this respect. This can lead to surprising results and the situation is often difficult to debug, so in general we would recommend to use the form: `$call =externalprogram..`
When calling pure Windows programs it is important to call the second form. The first form will not wait until the external Windows program has finished. If it is important to use a command processor in the invocation of a Windows program, use the START command, as in: `$call start /w externalwindowsprogram`. Otherwise, it is preferred to use: `$call =externalwindowsprogram`.

In general it is recommended to use the `$call =externalprogram` version for its better error-handling.

When command line arguments need to be passed to the external program, they can be added to the line, separated by blanks:

```
$call externalprogram parameter1 parameter2
$call =externalprogram parameter1 parameter2
```

The total length of the command line can not exceed 255 characters. If the program name or the parameters contain blanks or quotes you will need to quote them. You can use single or double quotes. In general the following syntax will work:

```
$call '"external program" "parameter 1" "parameter 2"'
$call ="external program" "parameter 1" "parameter 2"
```

It is noted that the first form needs additional quotes around the whole command line due to bugs in the parsing of the $call in GAMS. The second form work without additional quotes *only if the = appears outside the double quotes*.

13 Command files

Command files

Parameters can be specified in a command file. This is important if the length of the command line exceeds 255 characters, which is a hard limit on the length that GAMS allows for command lines. Instead of specifying a long command line as in:

```
$call =xls2gms I="c:\My Documents\test.xls" O="c:\My Documents\data.inc"
R=”Sheet2!A1:F15”
```

we can use a command line like:

```
$call =xls2gms @"c:\My Documents\options.txt"
```

The command file **c:\My Documents\options.txt** can look like:

```
I=c:\My Documents\test.xls
```

```
O=c:\My Documents\data.inc
R=Sheet2!A1:F15
```

It is possible to write the command file from inside a GAMS model using the $echo command. The following example will illustrate this:

```
$set cmdfile "trnsport.txt"
$echo "I=trnsport.xls" > "%cmdfile%"
$echo "O=trnsport.inc" >> "%cmdfile%"
$call =xls2gms @"%cmdfile%"

$include trnsport.inc
```

Newer versions of GAMS allow:

```
$set cmdfile trnsport.txt
$onecho > "%cmdfile%"
I=trnsport.xls
O=trnsport.inc
$offecho
$call =xls2gms @"%cmdfile%"

$include trnsport.inc
```

14 Multiple-area ranges and post-processing

Multiple-area ranges and post-processing

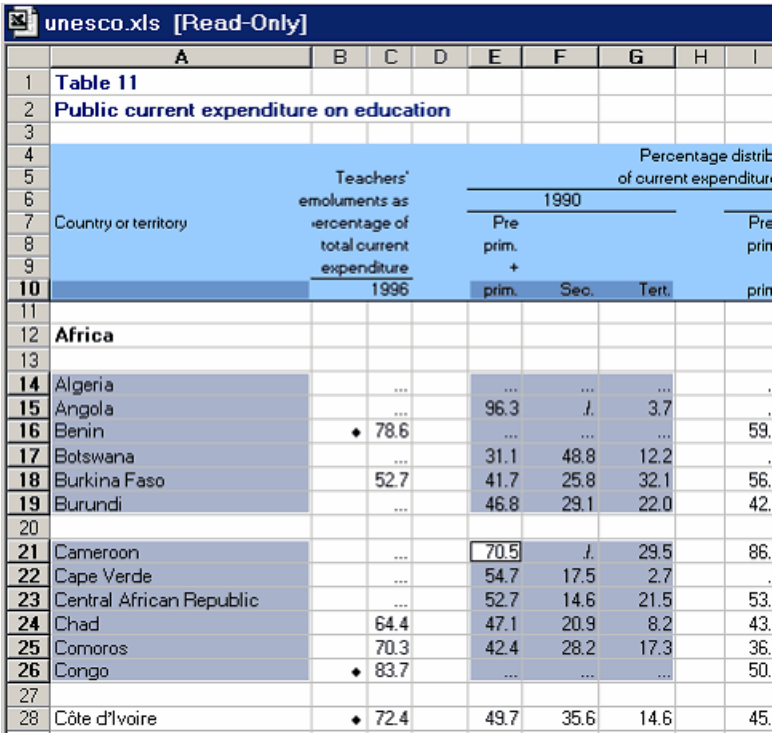
The following fragment is from a spreadsheet from Unesco:

unesco.xls [Read-Only]

	A	B	C	D	E	F	G	H	I	J
1	Table 11									
2	Public current expenditure on education									
3										
4							Percentage distribution			
5		Teachers'					of current expenditure by level			
6		emoluments as				1990				1996
7	Country or territory	ercentage of			Pre				Pre-	
8		total current			prim.				prim.	
9		expenditure			+				+	
10			1996		prim.	Sec.	Tert.		prim.	Sec
11										
12	Africa									
13										
14	Algeria		...		...	...	...		...	..
15	Angola		...		96.3	.!	3.7		...	..
16	Benin	◆	78.6		...	...	...		59.1	21.7
17	Botswana		...		31.1	48.8	12.2		...	..
18	Burkina Faso		52.7		41.7	25.8	32.1		56.6	25.1
19	Burundi		...		46.8	29.1	22.0		42.7	36.7
20										
21	Cameroon		...		70.5	.!	29.5		86.8	.!
22	Cape Verde		...		54.7	17.5	2.7		...	..
23	Central African Republic		...		52.7	14.6	21.5		53.2	16.5
24	Chad		64.4		47.1	20.9	8.2		43.5	24.2
25	Comoros		70.3		42.4	28.2	17.3		36.6	35.1
26	Congo	◆	83.7		...	...	...		50.4	11.8
27										

Assume we want to extract the 1990 percentage distribution of current expenditure for the countries Algeria through Congo. The range to select is not a contiguous area but consists of several pieces. In

Excel we can use the mouse and the Ctrl key to make a multiple selection:



unesco.xls [Read-Only]

	A	B	C	D	E	F	G	H	I
1	Table 11								
2	Public current expenditure on education								
3									
4								Percentage distrib	
5		Teachers'						of current expenditur	
6		emoluments as				1990			
7	Country or territory	ercentage of			Pre				Pre
8		total current			prim.				prin
9		expenditure			+				
10			1996		prim.	Sec.	Tert.		prin
11									
12	Africa								
13									
14	Algeria		...		...	...	...		.
15	Angola		...		96.3	./.	3.7		.
16	Benin	◆	78.6		...	...	...		59.
17	Botswana		...		31.1	48.8	12.2		.
18	Burkina Faso		52.7		41.7	25.8	32.1		56.
19	Burundi		...		46.8	29.1	22.0		42.
20									
21	Cameroon		...		70.5	./.	29.5		86.
22	Cape Verde		...		54.7	17.5	2.7		.
23	Central African Republic		...		52.7	14.6	21.5		53.
24	Chad		64.4		47.1	20.9	8.2		43.
25	Comoros		70.3		42.4	28.2	17.3		36.
26	Congo	◆	83.7		...	...	...		50.
27									
28	Côte d'Ivoire	◆	72.4		49.7	35.6	14.6		45.

The range is `A10,E10:G10,A14:A19,E14:G19,A21:A26,E21:G26`, where the comma's indicate the range is a multiple-area range. In this case we have six pieces. It is important that Excel understands that the union of the pieces forms a rectangular area. If this is not the case an error will be raised. (You can check this yourself by selecting a multi-area range and copying it to a new sheet: this operation will fail if the areas together don't form a rectangle).

The extracted text file will look like:

```
* -----------------------------------------------------
* XLS2GMS Version 2.3,  March 2004
* Erwin Kalvelagen, GAMS Development Corp.
* -----------------------------------------------------
* Application: Microsoft Excel
* Version:     9.0
* Workbook:    D:\gams projects\xls2gms\ver2.0\unesco.xls
* Sheet:       Sheet1
* Range:       $A$10,$E$10:$G$10,$A$14:$A$19,$E$14:$G$19,$A$21:$A$26,$E$21:$G$26
* -----------------------------------------------------
                           prim. Sec. Tert.
Algeria                    ...   ...  ...
Angola                     96.3  ./.  3.7
Benin                      ...   ...  ...
Botswana                   31.1  48.8 12.2
'Burkina Faso'             41.7  25.8 32.1
Burundi                    46.8  29.1 22
Cameroon                   70.5  ./.  29.5
'Cape Verde'               54.7  17.5 2.7
'Central African Republic' 52.7  14.6 21.5
Chad                       47.1  20.9 8.2
Comoros                    42.4  28.2 17.3
Congo                      ...   ...  ...
* -----------------------------------------------------
```

This file is not completely suitable for using in a GAMS model. The following edits would need to be made:

1. The header labels should get rid of the trailing dot. Cells with … should be replaced by blanks. Cells with ./. should be replaced by blanks

In the GAMS distribution a subdirectory **gbin** contains lots of interesting Unix utilities. Some of these are very suited to do string processing on text files, such as **sed** and **awk**. In this case we can use sed with some substitution commands:

`s/prim\./prim /` Replace "`prim.`" by "`prim` ". We need to be careful to keep the table alignment correctly, so we replace the dot by a blank instead of just removing the dot. A dot is a special character in **sed** (it means "any character") so we escape it by specifying "`\.`".

`s/Sec\./sec /` Replace "`Sec.`" by "`sec` ".

`s/Tert\./tert /` Replace "`Tert.`" by "`tert` ".

`s/\.\.\./ /g` Replace "`...`" by " ". The dots are escaped. We add **g** to indicate there there may be multiple instances on a line that must be replaced.

`s/\.\/\./ /g` Replace "`./.`" by " ". Both dots and '/' needs to be escaped.

The complete GAMS formulation can look like:

```
$ontext

   New version XLS2GMS ver 2.1 can handle
   multiple-area ranges.

$offtext


set c 'countries' /Algeria,Angola,Benin,Botswana,'Burkina Faso',Burundi,
                   Cameroon,'Cape Verde','Central African Republic',
                   Chad,Comoros,Congo/;
set exp 'percentage distribution of current expenditure' /prim,sec,tert/;


$onecho > commands.txt
I=%system.fp%unesco.xls
R=A10,E10:G10,A14:A19,E14:G19
O=unesco.inc
B
$offecho

$call =xls2gms @commands.txt

$onecho > sedcommands.txt
s/prim\./prim /
s/Sec\./sec /
s/Tert\./tert /
s/\.\.\./   /g
s/\.\/\./   /g
$offecho

$call sed.exe -f sedcommands.txt unesco.inc > unescox.inc


table distr(c,exp)
$include unescox.inc
;
display distr;
```