Technical Paper Series

PROVIDE PROJECT

The Provincial Decision-making Enabling Project

Technical Paper 2004: 1

SeeResults: A Spreadsheet Application for the Analysis of CGE Model Results

Elsenburg

November 2004

PROVIDE PROJECT

The Provincial Decision-making Enabling Project

Overview

The Provincial Decision-Making Enabling (PROVIDE) Project aims to facilitate policy design by supplying policymakers with provincial and national level quantitative policy information. The project entails the development of a series of databases (in the format of Social Accounting Matrices) for use in Computable General Equilibrium models.

The National and Provincial Departments of Agriculture are the stakeholders and funders of the PROVIDE Project. The research team is located at Elsenburg in the Western Cape.

PROVIDE Research Team

Project Leader:	Cecilia Punt
Senior Researchers:	Kalie Pauw
	Melt van Schoor
Junior Researchers:	Benedict Gilimani
	Lillian Rantho
Technical Expert:	Scott McDonald
Associate Researchers:	Lindsay Chant
	Christine Valente

PROVIDE Contact Details

Private Bag X1
Elsenburg, 7607
South Africa

provide@elsenburg.com
+27-21-8085191
+27-21-8085210

For the original project proposal and a more detailed description of the project, please visit www.elsenburg.com/provide

SeeResults: A Spreadsheet Application for the Analysis of CGE Model Results[1]

Abstract

CGE models tend to produce large amounts of result data, and it can be difficult and time-consuming to view, import, process and keep track of these. SeeResults is an Excel-based spreadsheet application that has been developed within the PROVIDE project to facilitate viewing results and producing presentation output, such as charts. It effectively functions as a GDX (a binary format used for output from GAMS) file viewer, and has been designed to be flexible, user-friendly and immediately usable with minimal configuration.

This paper describes the rationale for the application, its main features, a number of screen captures and implementation notes.

[1] The main authors of this paper are Melt van Schoor, Dr Scott McDonald and Cecilia Punt. Thanks are also due to Lindsay Chant and Anju Keetharuth for initial testing and suggestions. Please direct correspondence to Melt van Schoor (meltvs@elsenburg.com).

Table of contents

List of figures

1. Introduction

One of the practicalities of CGE modelling is to deal with vast amounts of output data from the models. Indeed, one of the prime advantages of the CGE modelling approach is the level of detail and completeness of the output data, and that a new dataset is generated for every run of every model. However, it can be a tedious task for the modeller to search through these results in order to locate the relevant parts and subsequently to transform and present the results in, for example, a research report.

For users of GAMS (such as the PROVIDE Project), the situation has been much improved following the development of the GDX file format with associated viewer and GDX utilities, which allow GDX data to be exported to Excel[2]. However, certain limitations of these utilities imply that, first, there remains a need to export data to Excel *en masse* and, second that doing so remains onerous (and potentially error-prone) for anything over a handful of variable blocks at a time.

Mainly as a result of these problems, we have developed an Excel-based spreadsheet application, named "SeeResults", which is designed to expedite this process. Instead of merely automating the process of importing large datasets into Excel, the application has been implemented as a generalised GAMS (CGE) model result viewer. This is achieved by transparently importing data as it is needed rather than loading it *en masse*. Numerous additional features have been incorporated, such as a system of data definitions, which enables the application to categorise variable blocks, produce appropriate descriptions along with tables and charts, and to ensure consistent ordering of data elements in a series.

This paper serves to describe the application, the technical functioning thereof and the impact it is expected to have on the way in which CGE model results are analysed in the future inside and outside the PROVIDE project.

[2] http://www.gams.com/dd/docs/gams/gdxutils.pdf

2. Background: Handling CGE Model Results

GAMS models can produce a variety of output ("result") data, for example:

- Model variable values after solving (prices, quantities, incomes, etc.)
- SAMs constructed from these result model variables
- Percentage or absolute changes between base and result model variables
- Summary statistics calculated after solving the model (macro aggregates, Gini coefficients etc.)

Of course the scope and extent of the result variables[3] are limited only by the modeller's imagination. For a typical study (one that we are currently working on) the result set can comprise about 120 variable blocks, and this could easily be as high as 150 or 200. These are generated for *every* run of the model – typically a model adjustment means that all results need to be re-generated. Each of these variable blocks can have zero (for scalar) or more dimensions or indices (multidimensional data). Extra dimensions can be added to account for different experiments, closure rules, adjustment (shock) intensities or other differentiating characteristics of a data set. Taken together, this implies that a data set could have hundreds of thousands or even millions of data points.

Only a small proportion of this data is likely to make its way into the final research outputs, if only because of practical limitations in the amount of data that can be presented. However, the modeller still needs to be able to inspect all of the data in order to identify the interesting parts. Being able to view data graphically can be a major advantage during this process. GAMS output listings are not suitable for this task in most cases; hence there is a need to find another way of dealing with result data.

In GAMS models, it is very easy to write result data to GDX files. The following code, which writes 4 parameters to a GDX file, provides an example:

```
Execute_Unload  'res_macro.gdx',
 rMACROTOT,
 nMACROTOT,
 rMACROSH,
 nMACROSH  ;
```

[3] In SeeResults, we use the term "variable block" to refer to blocks of result data with a common name. Tyically the results generated by a GAMS model that are of interest are the level (*.L) and marginal (*.M) values for the model variables, which are commonly stored in a GDX files as "parameters. SeeResults loads these GAMS/GDX "parameters" and calls them "variables" in respect of their origin.

The GDX format is technically superlative, GDX files being both extremely fast and small. Once the result data are in GDX format, they can be viewed using GAMS-IDE or GDXViewer, a specialised utility for viewing GDX files. Both of these include basic facilities for changing data orientation and the GDXViewer also includes basic charting (plot) facilties. While these are useful, there are a number of limitations. Filtering ability is limited, which can make it difficult to compare results within a smaller subset of elements from a large set. Charting facilities in GDXViewer are limited and not designed for presentation output. It is not possible to manipulate data, change formatting, perform calculations or cut and paste data to other applications (although there are data export facilities). In short, these facilities are suitable for viewing raw data initially but are not a complete solution for the processing of model result data.

Modellers are therefore required to export data to other applications for further work. At the present time, the logical choices are MS Excel[4] or a database application such as MS Access. Databases are powerful data storage engines and have powerful reorientation and filtering facilities. They can also deal with very large amounts of data efficiently. However, they typically have no or very limited charting abilities (on their own) and data manipulation is unnecessarily complex for the task at hand for non-specialists.

Excel has reasonably good charting facilities, is excellent for general data manipulation and provides the means to format data for presentation purposes. It can also provide limited database functionality such as reorientation and filtering via the pivot table feature. It is also useful as a means to distribute data, because it has a (very) wide user base. Excel does have certain reliability issues (it sometimes crashes for no apparent reason) and it is less efficient and somewhat restricted when dealing with large amounts of data[5]. On the balance, however, Excel remains our logical choice for this class of analyses.

From GDX to Excel

The problem therefore becomes to export model result data to Excel efficiently. GDXViewer has built-in facilities for exporting data to a number of formats, one of which is Excel. This is perfectly viable for one or a handful of variable blocks at a time, but is less suitable for large numbers of variable blocks.

[4] While there are other spreadsheet applications, Excel justifiably dominates the market at the moment, providing power functionality despite some reliability issues.

[5] It has built-in limitations of 255 columns and 65535 rows, which are occasionally problematic. For example, this limits the size of a SAM that can be displayed contiguously on a spreadsheet to 255 accounts.

A specialised tool called GDXXRW (part of the GDX utilities) can be used to better effect, but it requires exact configuration for each data item to be exported. These can either be supplied as command-line options or on an index sheet in a spreadsheet (for multiple variable blocks), listing the data items in a table. The latter approach is suitable, but requires careful typing of variable names and their target locations, especially when the layout has any degree of complexity. A simplified solution would be to put every variable block on its own worksheet. This works, but it produces cumbersome spreadsheets that are difficult to navigate. A navigation system that provides a menu of variables to choose from is therefore useful, but again this can be tedious to construct manually. An initial collection of macros to automate these tasks provided the impetus for the development of SeeResults.

3. Overview of Features

SeeResults originated from a collection of macros, but has developed into an integrated and generalised application. The application takes the form of a multidimensional data viewer. Users can reference or "load" data files and then browse through the variables contained in them. This format allows a great deal of generality while at the same time improving user-friendliness. SeeResults itself does not keep track of different datasets[6]; datasets are delineated by the files containing them, so that users can manage their data by using sensible file and directory (folder) names, etc.

The main goal has been to reduce the time and effort required of a modeller between the time when a model run has completed and the results can be viewed in a meaningful way. Several ancillary goals have been incorporated:

- To reduce human error
- To provide an efficient way to distribute result data to colleagues
- To provide an intuitive interface requiring minimal learning from new users
- To provide tables and charts that are easy to adapt for presentation quality output
- A high degree of responsiveness and stability

Sample Screens

The user interface is contained within only two worksheets. The first is the "Navigate" worksheet, and the second is the "Pivot Table Viewer". The Navigate worksheet is used to load data sources and data definitions (see below), and to select variables for viewing from amongst those available in the data sources. The Pivot Table Viewer is used to view data in a single variable block.

Figure 1 shows how a dataset is selected; note that two files are selected. The first one, a GDX file, is the data source. The second one is an optional data definition file. A data definition file is an Excel workbook containing two worksheets that describe the sets and variables in the GDX file. When a data definition file is used, certain additional meta-data is available to SeeResults, and it is used to provide greater ease of use and improve output

[6] An earlier attempt to incorporate such functionality was abandoned due to the large increase in complexity it entailed while offering little additional benefits.

presentation; for example, variable blocks are categorised, they have meaningful indices[7] and variable blocks and set elements can be shown with descriptions.

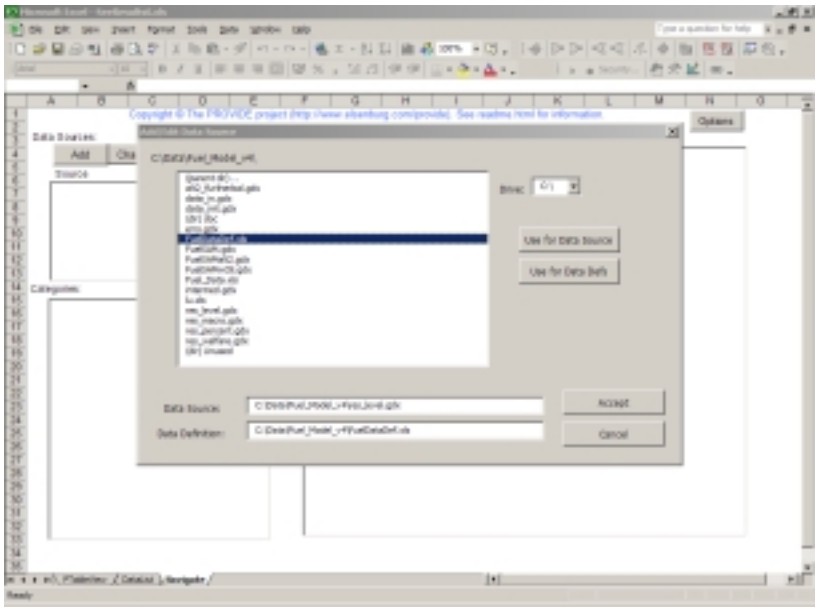

Figure 1. Selecting a data source

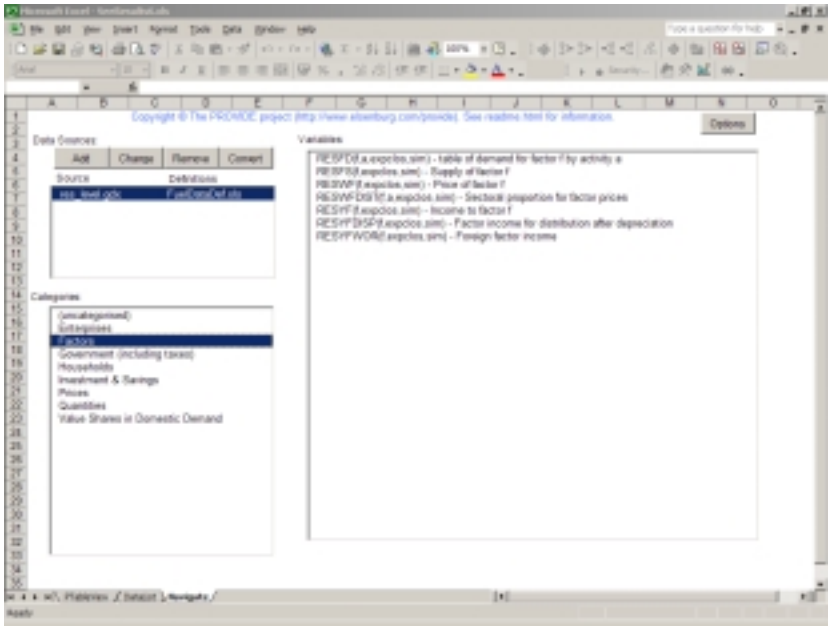

Figure 2. Selecting a variable block for viewing

[7] GDX files, unfortunately, do not store the names of indices.

When a data source is selected, its contents are listed according to the defined categories (if any), as can be seen in Figure 2. At this stage, no actual data have been imported, so the procedure is fast. Notice the categorisations, descriptions and index names, all of which are supplied from the data definition file.

Once a variable block is selected for viewing, the data are imported, loaded into a pivot table and displayed in the pivot table viewer, as shown in Figure 3.

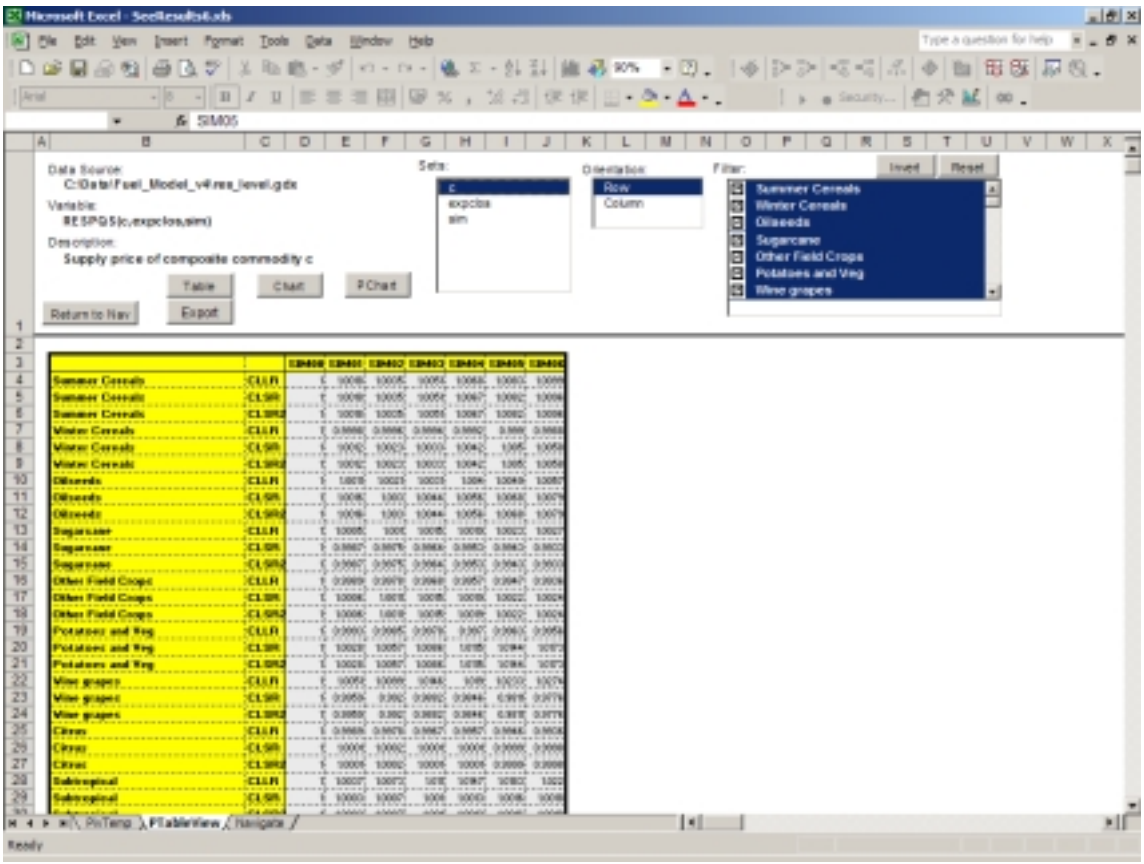

Figure 3. The Pivot Table Viewer

The data shown on the screen is a static copy of an underlying Excel pivot table. The controls allow configuration of the pivot table for changing data orientation and filters. When charts are made using the [Chart] button on this page, it is drawn from the static data, and uses a configured template, and is therefore usually suitable for presentation output with minimal or no changes. The actual underlying pivot table is accessible on the "PivTemp" worksheet.

Additional Features

The above screenshots illustrate the basic operation of SeeResults. The interface is as simple as possible, and the technical details with regard to the actual importing and loading of data are suppressed. Apart from this basic operation, there are a few additional features that deserve mention:

- Parameters in GDX files (shown as variables in SeeResults) can be converted to a special-format Excel file that can be viewed using SeeResults in the same way as a GDX file. Conversion can take some time, but the resulting file has three advantages – it contains its own data definitions (which simplify file management for the user), it reduces the delay while the program loads a variable block and it produces a file that can be viewed on systems that do not have the GDX utilities installed[8].

- "Persistent filters" refers to an option that preserves filtering between different variable blocks (and even between data sources) based on set names. It allows a user to select certain set elements of interest and then look at different data that are dimensioned on those sets. For example, it is useful when a user wishes to investigate what happens to different prices, quantities, etc for a particular commodity.

- Numerous small enhancements, such as support for aliases, shortcuts for moving between worksheets, selectable descriptions for set elements in table and chart outputs, and a simple file browser that makes it easier to specify data sources.

There is also complete documentation explaining the features and operation in detail, which is included in this text as an appendix.

[8] SeeResults require the GDX utilities to be installed in order to view GDX files.

4. Notes on Implementation

There are a number of design and other implementation principles that have proved valuable during the implementation of SeeResults. This section briefly mentions some of these.

Integrated application

Originally we had envisaged a number of tools to accomplish a number of separate tasks related to viewing GAMS model result data. While these were useful, they inevitably resulted in an intricate web of tools that are difficult to use except by those who designed them, and were highly specific to a particular mode of working. Collecting these tools into an application had numerous advantages. Most computer-literate people are familiar with the mode of operation; to "run the application and then load the data" comes almost as second nature to many people.

One earlier problem was that GDX data could only be exported to a workbook that is not open, otherwise resulting in an error, which caused considerable exasperation. An integrated application could solve this issue by adopting a "pull" rather than a "push" strategy to import data. Internally, SeeResults creates a temporary workbook for data importing and then reads the data from that workbook before deleting it again.

Use of existing functionality

SeeResults makes heavy use of the GDX utilities[9] for listing and importing data. The GDX utilities were designed for this task, and it would have been wasteful to attempt to reprogram their functionality. The fact that they are straightforward command-line operated utilities greatly simplified this endeavour.

Excel is conceptually a very powerful programming environment, allowing use of the spreadsheet itself as a data construct and combining the power of Visual Basic with the ability to use spreadsheet functions in Visual Basic code. All configuration data are stored on hidden worksheets, so that they can be saved along with the workbook application. This is easier to implement than (for example) maintaining configuration files.

[9] GDXXRW.EXE and GDXDUMP.EXE.

Another major usage of Excel functionality is the use of Pivot Tables for data reorientation and filtering in multidimensional data. Excel's pivot tables are ideally suited to the task of storing and presenting reasonable amounts of data and allowing the display format to be changed after data has been imported to Excel. It performs SeeResult's technically demanding data handling tasks more efficiently than would have been possible using custom Visual Basic code.

Flexible use

The application has been designed to be as simple and flexible as possible. There is little that a user could do that will result in something illogical or produce an error. Some actualisations of this concept were to let data definition files be optional, to tolerate misspecified data definitions[10] and to leave user's data management to the operating system.

Modular Visual Basic Code

SeeResults makes use of Visual Basic for Applications, which comes bundled with MS Office applications. Writing relatively tidy and modularised code ensures fewer errors, better generality and reusability. For example, there are different modules dealing with each GDX function (listing and reading data), management of internal settings, executing external programs (used by GDX module), creating pivot tables, reading data definitions and reading result data from Excel sources. The code dealing with worksheet controls are located in the module associated with each worksheet. These contain application-specific functions that call the more general functions from the other modules.

[10] For example, if a variable name is misspelled, it will be ignored and the user will see that one of the variables does not have descriptions, but it will not generate an error.

5. Concluding Comments

SeeResults was created with a specific purpose in mind, namely to automate the process of loading data from GDX files into Excel. In actual fact, we have created an application which removes the need for loading entire datasets into Excel by allowing a spreadsheet application to function as a GDX data viewer, reading individual variable blocks from GDX on the fly.

By focusing on core requirements and keeping the interface as simple as possible, we have developed a general application that does not unduly restrict the user in terms of how and for what it can be used. In the meantime, it has proved very effective at its original task, namely for result data analysis in the PROVIDE project. By allowing its use by others, we hope to gain insights into possible future improvements while making it easier for modellers to concentrate on analysing their model results.

SeeResults is a work in progress, and it is expected that it will evolve over time to account for additional features, bug fixes and other changes. There is considerable scope for added functionality, particularly in the final data analysis and presentation, but the generality and simplicity should not be sacrificed in the process, and SeeResults should not interfere with modeller's own preferred methods of working.

6. Appendix A: Documentation

The following documentation is distributed along with the program as an HTML file (web page). Note that the documentation below is only illustrative, users are referred to the latest version of the document for working purposes, available at http://www.elsenburg.com/provide.

SeeResults Documentation

Contents

A. General Information

This text provides documentation for SeeResults, an Excel-based spreadsheet application that has been designed to aid the viewing of results from GAMS CGE models. The program is designed to work with the GDX data file format, which is easy to use for output in GAMS. The workbook acts as a GDX file viewer, but with the option of using a special data definition file to describe and categorise results. Additionally, it is possible to store results in a special format which would allow users without GAMS/GDX installed on their computers to view them.

The latest version of the program is available at http://www.elsenburg.com/provide/seeresults.

Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. A copy of the GNU GPL, gpl.txt, is included with the programme files. Do NOT remove this notice if you redistribute this programme If you use this programme you should acknowledge the source and not claim authorship. Altered versions should be clearly distinguished from the original.

If you wish to make the software available for wider distribution, for example on a website or packaged with other software, we request that you contact us first. If you wish to acknowledge the software in your research outputs, please reference "PROVIDE (2004). 'SeeResults: A Spreadsheet Application for CGE Model Result Analysis'. *PROVIDE Technical Paper 2004:1*. Department of Agricultural Economics: Elsenburg. Available online at http://www.elsenburg.com/provide/seeresults."

Programming on this version and compilation of this text and the SeeResults package was done by Melt van Schoor, with substantial inputs by Dr Scott McDonald, Lindsay Chant and Cecilia Punt. We also wish to thank Anju Keetharuth for essential testing and useful comments. Please contact Melt van Schoor (meltvs@elsenburg.com) with bug reports, suggestions and questions, but keep in mind that the program is unsupported. This text was last edited on 26 November 2004 (the program itself may have been updated since then).

It is recommended that you read at least sections B and C of this text before using the program.

B. Prerequisites

At a minimum, you need need a recent version of Windows and Excel and a data file that you wish to view. If you wish to view GDX files (which is likely if you want to view results from your own models) you will also need to have the GDX utilities installed. To utilise the data description features of SeeResults, you will also need a data definition file.

- **Windows.**The program has been tested on different versions of Windows XP and (updated) Windows 2000. It may or may not work on Windows NT and is unlikely to work on Windows 98 or 95. Please ensure that you have the latest service packs installed, especially for Windows 2000 - older versions of Windows 2000 will not work with SeeResults.

- **Excel.** To use SeeResults you should have Excel 2000 or later (97 will not work) installed. It is important that your macro security settings will allow unsigned VB macros to execute. To check this, from Excel click Tools->Options->Security->Macro Security and ensure you selected either Low or Medium security level (not High). This procedure may differ slightly according to your version of Excel.

- **Data file.** You should either have a GDX file containing your results or an XLS data file that has been created by SeeResults. Note:

    - In GDX files Only **parameters** are of interest - any other items (e.g. sets or variables) in the GDX file will be ignored. The term "variables" in the this text refers to model results stored as parameters and not to GDX/GAMS variables.

    - Due to Excel limitations, parameters with more than 65530 nonzero data points will not be viewable using SeeResults.

    - GDX files do not store zeroes, but if you set zero values to EPS in your GAMS code before writing the GDX file, SeeResults will treat and display them as zeroes (for example, RESPXC(c,sim)$(NOT RESPXC(c,sim)) = EPS).

  - While you will be able to select any XLS file as a data source, only the special-format files created by SeeResults will work correctly. For more information on the special Excel data files see Section H.

- **GDX utilities.** To use GDX files you need to have the GDX utilities installed. These form part of GAMS and therefore you should consult GAMS documentation for details on how to install them. Additionally, unless these utilities are installed in C:\GAMS\, you need to tell SeeResults where to find them by setting the relevant options.

- **Data definition file.** This is an Excel file describing the data that is being viewed, which enables SeeResults to use variable categories, variable descriptions, correct set names, correct element ordering and set element descriptions. The use of a data defintion file is optional. See E. Data Definitions for details.

C. Brief Tutorial

This section briefly explains the process of viewing the results in a GDX file. It only covers the most important concepts, while the remainder of the documentation provides a detailed treatment. We assume that you meet the prerequisites listed in the previous section (note especially the directive about the Excel macro security option) and that you are testing the program using the example files "Sample_Results.gdx" and "Sample_DataDef.xls", which are included in the package.

Before you begin, you should check that SeeResults is able to use the GDX utilities. To do this, open SeeResults.xls and then click on [Options]. Turn the "Pause after execution" option on and then click on the [Test] buttons. If you can see the utilities running, it means that SeeResults is able to use the utilities. If you see a message such as "The system cannot find the path specified", then try changing the location options to reflect the path where GAMS is installed on your system and test again. Remember to turn off the "Pause after execution" option when you are satisfied.

You can then add a data source. To do this, you need to click on [Add] and then select the data file. In the dialog, the first time you double-click on a file, it will be selected as a data source and the second time you double-click it will be selected as a data definition file. Selecting a data definition file is optional. After you click on [Accept] you should see your data source and data definition file (if any) in the data source box. If not, it could mean either that you chose illegal options (e.g. files with the wrong extensions) or that the files you selected could not be found or read for some reason.

Data definition files are Excel files that contain information about sets and variables that make part of a result set. The easiest way to create a data definition file is to look at the example and adjust it to your own data set. It is acceptable to define only some of the data in a result set; the principle is that whatever is available will be used as far as it fits the data.

Data sources are stored as pairs of filenames (source and definition) in the Data Sources box. You can define multiple data sources at a time for easy navigation between them. When you save the workbook (i.e. SeeResults.xls) you effectively save the contents of the data sources box, the options and persistent filters if you use them. No model result data are ever saved in the viewer.

After defining a data source, you should be able to see one or more categories in the Categories box just below the Data Source box. If you did not select a data definition file, there will be no categorisation, and all parameters will fall in the "(uncategorised)" category. Clicking on a category will show all the variables in the GDX file under that category.

Clicking (or double-clicking) on a variable name will lead you to the pivot table viewer (assuming the parameter has at least one dimension and at least one data point). The data for that variable should then be displayed according to a default layout. From here you can change orientation options for each set (index), and select to view charts or export a table. Experiment with these options to see how they work.

You can always return to the Navigate worksheet by pressing Ctrl-Shift-R and to the pivot table viewer by pressing Ctrl-Shift-P.

D. Options

This section describes the various options that you can change by clicking on [Options] from the main Navigate worksheet. All options are global, meaning they apply to all data that you view with the workbook (SeeResults.xls) in which you set the options. If you change any options, be sure to save them by saving the SeeResults.xls workbook. By default, the workbook is **not automatically saved** when you close it (and the usual "Do you want to save.." window does not appear!).
The Options are:

1. **GDXXRW.EXE location**. This specifies the full path to the GDXXRW.EXE utility on your system. This must be correct in order to view GDX data. Use the [Test] button to see if it runs (turn on the "pause after execution" option on first).

2. **GDXDUMP.EXE location**. This specifies the full path to the GDXDUMP.EXE utility on your system. This must be correct in order to view GDX data. Use the [Test] button to see if it runs (turn on the "pause after execution" option on first).

3. **Pause after execution**. This will pause each time after the utilities are run. Turn it on only for debugging purposes.

4. **Save settings before closing**. This option will force an automatic save when the SeeResults.xls workbook is closed, effectively saving your options, data source references and persistent filters (see below). This option was added because the usual "Do you wish to save..." dialog box does not appear when you close SeeResults.xls.

5. **Alias indicator character**. If you specify a character here, any string consisting of repetitions of the specified character is considered a postfix that can be added to set names to indicate an alias for the same set name without the postfix. For example, if you enter 'p' for this option, and define the set 'c' (in your data definition file), you can index variables on 'c', 'cp' or 'cpp', etc. without having to define 'cp' or 'cpp'.

6. **Item labels**. In the data definitions, set elements are defined with two alternative descriptions (See Section E). The element name itself functions as a third possible description. Use this option to select which of these you wish to appear on rendered data tables and charts. Note: if you convert data to .xls format, it will permanently have the descriptions that were in effect at the time of conversion.

7. **Persistent filters**. When you turn this option on, each time you change filters for a set on the pivot table viewer, the filter will be saved. The next time that the same set is read into the pivot table viewer, the previous filter will be re-applied automatically. You can always use the [Reset] function to turn the filter off.

E. Data Definitions

Data definitions describe the data that you view. For SeeResults, data definitions are entirely optional. Furthermore, even if you do use data definitions, you may describe the data only partially. The principle is that any data definitions available will be used by the viewer (as far as it fits the data). Depending on which components of the data definition

are available, the following changes to SeeResults's behaviour occur when data definitions are used:

- Set (index) names are used instead of dim1, dim2, dim3,...
- Variables are categorised on the Navigate worksheet.
- Set elements are described (depending on the selected "Item labels" option).
- Set elements are ordered consistently, in the same order as they appear in the data definition.

Data definitions are expressed in a data definition file. A data definition file is an Excel workbook (.xls file) that contains two special worksheets, one of which is named "Sets" and another "Variables". As can be expected, these sheets define sets and variables respectively. To use a data definition file, reference it when you define data sources, or add a data definition file reference to an existing data source by selecting it and clicking [Change]. If your data source is an Excel file, you may use the same file as the data source, as long as it contains the two mentioned worksheets.

An example data definition file is available.

Set Definitions

Sets are defined on a worksheet named "Sets" in the data definition file. The first set name must appear in cell A2 on the worksheet. The format is as follows:

	A	B	C	D	E	...
1	Set 1 Description			Set 2 Description		...
2	[Set 1 Name]	ShortDescr	LongDescr	[Set 2 Name]	ShortDescr	...
3	[Set 1 Elem 1 Name]	[Set 1 Elem 1 Short Descr]	[Set 1 Elem 1 Long Descr]	[Set 2 Elem 1 Name]	[Set 2 Elem 1 Short Descr]	...
4	[Set 1 Elem 2 Name]	[Set 1 Elem 2 Short Descr]	[Set 1 Elem 2 Long Descr]	[Set 2 Elem 2 Name]	[Set 2 Elem 2 Short Descr]	...
...	...	...	...	...	...	...

In the above schema, each cell with bracketed text is read by SeeResults. Each set has a name and a number of elements and each element has a name and two descriptions. It is vital to preserve the cell layout exactly, for example each set must be exactly three columns away from the previous one. You may leave out long (or short) element descriptions, but then you still need to preserve open columns. Set names should correspond to the set names used in the GAMS model and element names must correspond to the element names in the GAMS model. For element descriptions, you can use anything, but avoid the use of special (non-alphanumeric) characters or overly long descriptions.

Variable Definitions

Variable blocks are defined on a worksheet named "Variables" in the data definition file. The first variable block name must appear in cell A3 on the worksheet. Note that each of the rows in this table represent a variable block with potentially numerous individual variables. The format is as follows:

	A	B	C	D
1				
2	Variable	Indices	Category	Description
3	[Variable 1 Name]	[ind1,ind2,ind3,...]	[Variable 1 Category]	[Variable 1 description]
4	[Variable 2 Name]	[ind1,ind2,ind3,...]	[Variable 2 Category]	[Variable 2 Description]
...	...	...	...	...

Again, only bracketed cells are read by SeeResults. Each variable block has one or more indexes, which are comma-seperated without any other characters (e.g. a space after each comma). Variable names should correspond to your GAMS model variable declarations exactly, while categories and descriptions can be anything (again avoid the use of non-alphanumberic characters). To ensure that variables in the same categories are grouped together on the Navigate worksheet, be sure to specify exactly the same categories for the relevant variables.

F. The Navigate Worksheet

The Navigate worksheet is the main interface in SeeResults. It should open automatically whenever you open the SeeResults program. From here, you can change options, define data sources and navigate variables.

To change options, click on the [Options] button. A special options window will pop up allowing you to change global options. See the relevant section for details.

Defining data sources is neccessary before you can view any data. A data source is essentially a reference to a data file (a GDX or a suitable XLS file) and optionally also a data definition file. You can use the [Add], [Change] and [Remove] buttons to manage your data sources. Both [Add] and [Change] will open a special file browser window which allows you to select the data and optional data definition file. In the file browser window, you can generally double-click on relevant items to activate them - directories to change them and files to select them. The first time you double-click on a file, it will be selected as the data source and the second time you double-click on a file it will be selected as the data definition file. However, this will only work if the boxes are empty. Alternatively, you can use the [Use for Data Source] and [Use for Data Defs] buttons. A number of conditions will trigger an error condition which will result in your selections being rejected. For example, you will not be able to select files with invalid extensions or files which are inaccessible.

Once data sources have been defined, you should see a number of categories in the Categories box on the Navigate worksheet. If this fails, check the GDXDUMP.EXE setting and ensure that your data file is valid. If you have not specified a data definition file, or if your data definition file does not contain categories for the variables in the data file, there will only be a single category, named "(uncategorised)".

Select a category by clicking on it. This should load a list of variables in that category into the Variables box. Click (or double-click) on a variable to view it. If you select a variable block containing valid data (at least one dimension and at least one data point), you will be taken to the pivot table viewer (see below). If you see an error message, you may have a problem with GDXDUMP.EXE.

You can return to the Navigate worksheet from any location in the same workbook by activating a set macro with the key combination Ctrl-Shift-R.

G. The Pivot Table Viewer

This worksheet enables you to extract, manipulate and chart data. When you select a variable from the Navigate worksheet, the variable block will be "loaded" into the pivot table viewer. A default layout is used to display variables and a table is drawn initally. To change the layout, you need to change the settings per field. Fields (or indices) for the current variable are listed in the first list box. Once you select a field, you can then select Orientation (Row or Column) and filters for it using the other two list boxes. Once you are satisfied, click on the [Table] button to redraw the table. Whenever a table is created, you are seeing a static formatted copy of a temporary pivot table that has been configured using the options you specify. You can see the actual pivot table on the "PivTemp" worksheet if you wish.

The other options available are [Chart], [PChart] and [Export]. [Chart] will produce a chart based on the table currently displayed, while [PChart] will produce a pivot chart based on the underlying pivot table. [Export] will create a static copy of the data on a new worksheet in a new workbook. The worksheets that are created when you use [Export] will never be automatically deleted, because they are not in the SeeResults.xls workbook. Note that these options ([Chart], [PChart] and [Export]) will always produce results based on the data table currently displayed on the PTableView worksheet. Therefore, if you wish to change (e.g.) a filter and then view the chart again, you first need to click on [Table] to update the table with your new settings.
All pivot tables, charts, etc. are deleted automatically when the workbook is saved (except for exported tables). Also, they can be overwritten if you view another variable or the same variable with different settings. Orientation and filter settings for a particular variable are lost whenever you select another variable, except when the persistent filters option is turned on, in which case filters (but not orientations) will be maintained for a given set (even across different variables or data sources). If you are using persistent filters option, you can save the current set of filters by saving the SeeResults.xls workbook.

You can return to the pivot table viewer from any location in the same workbook by activating a set macro with the key combination Ctrl-Shift-P.

H. XLS Data Files

SeeResults can use either normal GDX files or special-format XLS files as data sources. You can create the special XLS files by defining a GDX data source (with or without a data defintion file) and then selecting the [Convert] button from the Navigate worksheet. This will prompt you for a filename, and then create a XLS file containing all of the **valid** variables in the GDX file. Scalar or empty variables are not included in the new file. Additionally, if your data source references a data definition file, the data definitions will automatically be copied to the new XLS file, so that the new file can be used simultaneously as a data source and a data definition file.

During the conversion process, it is strongly recommended that you wait for it to complete without using any other application on the computer. This is because the conversion process uses the windows clipboard, and also creates multiple processes. After a successful conversion (which can take some time) the new file will automatically appear as a new data source.

There are a number of advantages of XLS format files:

- They can be distributed to people who do not have the GDX utilities installed on their computers (SeeResults itself is still required!)
- Navigating between variables in the same file will be much faster.
- Data and data definitions are condensed into one file.

The main disadvantage of using XLS files are:

- They take up more space than equivalent GDX files.
- The conversion process itself can take a while to complete.
- After conversion, it is no longer possible to flip between set element descriptions (name, long, short) - the option that was in effect when the conversion was performed is permanently embedded in the xls data file.

-

Technical Papers in this Series

Number	Title	Date
TP2003: 1	Creating a 1995 IES Database in STATA	September 2003
TP2003: 2	Creating a 1995 OHS and a combined OHS-IES Database in STATA	September 2003
TP2003: 3	A Standard Computable General Equilibrium Model Version 3: Technical Documentation	September 2003
TP2003: 4	Measures of Poverty and Inequality: A Reference paper	October 2003
TP2004: 1	SeeResults: A spreadsheet Application for the Analysis of CGE Model Results	November 2004
TP2004: 2	The Organisation of Trade Data for inclusion in Social Accounting Matrix	December 2004
TP2004: 3	Creating a 2000 IES Database in Stata	August 2004
TP2004: 4	Forming Representative Household Groups in a SAM	July 2004

Other PROVIDE Publications

Background Paper Series
Working Papers
Research Reports