

Ronald Horst¹

Abstract

This article summarizes recent work in G7, the database and econometric software package developed by Inforum and its partners. A report is presented of recent development efforts, several applications of the new tools, and possibilities for future work.

1. Introduction

G7 is the flagship tool offered by Inforum² for data construction and analysis, econometrics, and to support the formation of large-scale structural econometric models. Together with a suite of associated programs, it has been under continual development for many years, with contributions from Inforum staff and partners. This document offers a brief summary of work on *G7* that recently has been completed. While many improvements have been made, we focus here on two significant extensions. These two extensions have been used to simplify tedious tasks that previously were error-prone and time consuming. We believe they offer tremendous additional potential that has yet to be exploited.

The two new features are 1) an interface between *G7* and Microsoft Excel that allows *G7* to read and write data in spreadsheet formats and 2) the introduction of flow-control methods to the *G7* scripting language. Documentation and sample programs are available on the Internet, and they soon will be included in the *G7* help files.³

Following sections of this document introduce the tools in greater detail, describes their current capabilities and the use of the tools, and presents simple sample programs. We conclude with plans to extend these features in future work.

2. New Features of *G7*

The key new features of *G7* provide a means of reading Excel files without first converting them to Lotus format or to text files, a means of recording data in the form of an Excel spreadsheet, and also the ability to control the execution of scripts based on the conditions of arguments. Details are provided in the following sections.

2.1 An Excel Interface

¹ Ronald Horst, Inforum, University of Maryland, College Park, USA. Ronald.Horst@gmail.com.

² For more information about Inforum, please see our website at www.Inforum.umd.edu.

³ For the *G7* software and for additional documentation on the material introduced here, please visit the Inforum web site at www.Inforum.umd.edu. User's Guides for the features presented here may be found on the "Inforum World Conference XIV" page.

It seems that Excel spreadsheets have become a standard means of distributing economic data. In the past, it was difficult to convert spreadsheet data into a format that *G7* could understand. Several steps were required. These steps were tedious, error-prone, and would need to be repeated every time that the data were updated. If a simple, one-step means of converting Excel data into *G7* data banks were available, then much time could be saved. If *G7* data also could be written to Excel files for distribution to clients, then better still.

At present, *G7* has no means of reading a file in Excel format directly from the disk. Such direct access is desirable, because data files often become large and require significant time to read. Direct access allows the greatest speed for reading and writing spreadsheets. At this point, *G7* instead relies on Microsoft Excel to operate as a server. Unfortunately, use of these capabilities thus requires that Excel be installed on the same machine.

To read data from a spreadsheet, *G7* first signals Windows to start Excel, if it is not running already. Excel then is instructed to open the desired spreadsheet and go to a particular worksheet. When the worksheet containing the data is opened, *G7* sends instructions for Excel to report the data contained in certain cells. Currently, *G7* can send requests for data that may be stored in one of several possible arrangements. The cells can be arranged in free format, which are interpreted as a time series or a vector element, or an entire vector may be read for multiple years, or a single year of a matrix may be read. Time series data is stored in the *G7* workspace. Vector and matrix data is stored in the default Vam file.

Data may be written to existing Excel worksheets. At this time, *G7* cannot create new workbooks or worksheets. Writing of time series and single vector and matrix elements is supported, as is the writing of entire matrices.

2.2 Flow Control Capabilities

The most common flow control methods are if-else statements. These are common in most compiled programming languages and in some scripting languages. Current capabilities in *G7* are limited; we will discuss extensions in the conclusion. However, even modest capability has yielded significant benefit: complicated and repetitive code has been simplified, the need for tedious debugging has been reduced, and the saved time and energy has been put to better uses. Resulting work thus tends to be of higher quality since it is produced by shorter, simpler code that is easier to write, to understand, and to debug.

The if-else routines compare two arguments provided by execution of the script, or they compare one argument to a string, a number, or a variable defined by *G7*. Arguments may be passed to the file containing the if-else routine or they may be iterators in a loop. Available comparisons allow checks for equality and inequality, less-than and greater-than, and similar conditions between two numbers. Note that these numbers are arguments; at this

Excel server.

We turn now to a more complicated example. The complete demo is available on the Inforum site. We offer an abridged version here, with only the details of primary importance. We begin, assuming that a Vam file has been opened and has been declared the default. The data read from the spreadsheet are stored as vector elements in a Vam bank. Writing to a vector element, instead of to the workspace, is forced by providing a bank letter (“c”) corresponding to a Vam bank in front of the name of the vector. Note that columns may be specified either by the Excel column letters or by the column number.

Sample 3.1.2

```
x1 open C0301e.xls
x1 open worksheet 1
do{
  x1 read %1 27 down c.gdpN%2 1990 1990
  }(3-4 6-7 9-10) (1-6)m
x1 exit
```

Finally, we present code to read a block of data and store it as a 17X17 matrix. Note that the `x1 matread` instruction should be on one line. First, blocks of data in the spreadsheet are indicated by listing their rows and columns. Next, the name of the matrix is given in which the data should be stored, along with the matrix rows and columns. Finally, the year is provided for which the matrix should be stored.

Sample 3.1.3

```
x1 open C0319e.xls
x1 open worksheet 1
x1 matread c(2-18) r(14-17, 19-20, 22-24,26-29,31-32, 34-35) ...
  c.AM c(1-17) r(1-17) 2000
```

3.2 Flow Control Capabilities

We now demonstrate some of the flow control features offered by *G7*. The samples presented are rather useless, but they illustrate the means by which useful instructions may be written. For example, strings presented for comparison may be regions of a country. Blocks of code could be tailored to specific regions, while remaining code would be identical for all regions. In this way, repetition can be eliminated and code can be simplified.

The first sample shows how if-else statements can be used to perform operations based on the iterator value in a loop. Useful examples include 1) special operations may be performed on particular sectors, and 2) reports may be displayed for particular iterations.

Here, we display iterate over values 1 to 10, and we display the iteration numbers 1-3, 5, and 10.

Sample 3.2.1

```
do{
  if( %1 < 3 || %1 == 5 || %1 == 10){
    ic Iteration %1
  }
}(1-10)
```

The second example demonstrates the combined use of if-else statements and the ability of *G7* to support user-defined functions. The function defined below may be passed 0 to 3 arguments. The if-else statements compare the number of arguments, which *G7* defines for if-else comparisons as the variable “NARGS,” to various integers. A statement is printed based on the results of the comparisons.

Sample 3.1.2

```
function TEST{
  if( NARGS == 0 ){
    ic Zero arguments were passed to the function TEST.
  }
  else if( NARGS == 1 ){
    ic The argument %1 was passed to the function TEST.
  }
  else{
    ic Two arguments, %1 and %2, were passed to TEST.
  }
}
TEST
TEST ONE_ARGUMENT
TEST FIRST_ARGUMENT SECOND_ARGUMENT
```

The third example demonstrates several tests, first for the comparison of strings and then for the comparison of numbers. The arguments may be replaced by the standard *G7* variables (i.e. %1, %2, etc.) defined for “add” files and loop iterators.

Sample 3.1.3

```
if( Clopper != Almon ){
  ic String inequality test failed.
  ic Clopper is not the same as Almon.
}
if( A <= C ){
  ic Character less than or equal to comparisons.
}
if( -1 < -0.5 ){
```

```

        ic Integer evaluation with negatives.
    }
    if( ( 2 < 3 && 3 <= 4 ) || A <= C ){
        ic Multiple evaluations with AND, OR, and parentheses.
    }

```

4. Conclusions

The features described here are important extensions. They simplify and speed the work of the applied economist. At this point, however, they remain early, exploratory attempts, and many possible extensions and revisions promise additional benefits.

As noted earlier, *G7* currently has no ability to access Excel spreadsheets directly. Instead, it relies on an interface with the Microsoft Excel program, which in turn reads and writes data files. This approach unfortunately limits speed and requires that the user purchase and install Excel in order to employ these features. At least two possibilities exist to address these problems. First, the greatest levels of speed could be reached by integrating code, which may be available on the Internet, to read spreadsheets directly. There may be some licensing restrictions and other difficulties with integrating such code, but the approach likely is feasible. The other alternative is to employ an interface with OpenOffice, which can read and write in Excel format. Benefits include cost, since OpenOffice is free, and portability, since OpenOffice is available for all major operating systems. Another option is to include both direct access capabilities and possible reliance on third-party software (i.e. Excel or OpenOffice). In this way, high-speed access would be available, and other features of the third-party software could be exploited if the software is installed on the user's machine.

The flow control capabilities currently implemented merely scratch the surface of what is possible and useful. At this point, only arguments passed to the routine essentially as text can be evaluated. In particular, information stored as data cannot be evaluated, so the capabilities currently do not include tests such as convergence checks. In addition to allowing data elements to be read, it also might be useful to allow expressions to be evaluated, such as `if(i < 2*j)`.

Other concepts of flow control also can be defined and implemented. The existing approach evaluates an expression, where the expression amounts to one or perhaps several comparisons of integers or strings. If the expression is true, then a block of code is executed, operating on multiple time periods. Another type of comparison would evaluate two time series, where the comparison would be performed on multiple periods. A period-by-period result would be generated depending on the results of the comparison. For example, a function could be defined to work with the existing “f” routine:

```
f z = @if( x>y, x, y)
```

where elements of the time series **z** are the maximum values of **x** and **y**.