An Eli-Excel Interface

Written by HF, 2012-2-15

1. Overview

This writing is about a group of Eli functions which control Eli-Excel interface. These functions have many helpful names for their arguments to make it easier to access Excel. It is convenient to have few simple lines to transfer data from Eli environment to Excel, and to read formatted data from Excel to Eli. We use Excel 2003 for this first edition to link Excel with Eli because it is a very basic version. This interface provides basic operations such like clear, insert, delete, read and write. At present, we haven't implemented functions which set colors and formulas, draw pictures or accept general data types.

The functions are implemented in Eli 0.02c and tested with Excel 2003. When you start to use these functions, you'd better make sure the workbook you want to use is not in use. Note that if any unpredictable error occurs, one solution is to terminate all possible EXCEL.exe via task manager. I don't know whether it works well in other versions. This is a simple version for Eli to communicate with Excel. If you have other requirement or encounter any problem, don't hesitate to inform me so that I can adapt the code where necessary.

1.1 Eli

Eli is an array programming language system based on APL but uses ascii-character set. It has a large number of primitive functions, monadic and dyadic, and several operators, for programming arrays. These primitives manipulate data efficiently, and encourage an elegant and succinct style of programming to realize one's ideas quickly. It is still a powerful tool for processing data and getting result promptly.

1.2 Microsoft Excel

Microsoft Excel is a widely used commercial spreadsheet application which features calculation, graphic tools, pivot tables, and a macro programming language called VBA(Visual Basic for Application). It allows ones to store, manipulate, analyze, and visualize data. It has been a extensively applied on many platforms.

Microsoft Excel has the basic features of all spreadsheets: using a grid of cells arranged in numbered rows and letter-named columns to organize and manipulations data. It supports charts, graphs or histograms generated from specific groups of cells. The generated graphic component can either be embedded within the current sheet, or added as a separate object. A more elaborate Excel application can automatically poll data from external databases or measuring instruments using an update schedule, analyze the results, make a Word report or Power Point slide, and e-mail these presentations on a regular basis to a list of recipients.

The latest version of Excel is Excel 2010 (version 14) included in Office 2010.

2. Implementation

We use OLE to link Eli with Excel. This interface simplifies multi-workbook and multi-worksheet problem in Excel and allows one to process at one Excel sheet at a time. It means that a user should close an Excel file after processing before he handles the next Excel file. There are one variable and eight functions in this interface to operate on the Excel side.

The main structure of Excel data is organized in threelevels.

- A workbook (excel name)
- several sheets
- rectangle cells

We are familiar with cells embedded in a sheet. A cell can contain data of various types such as integer, double, time, etc. It is not difficult to locate a cell. A cell with coordinate 'B5'. means the cell is in the fifth row and the second column. One can represent a group of cells, called '*area*', by 'B5:H16'; 'B5:H16' indicates a matrix with top left cell 'B5' and bottom right cell 'H16'. Since two opposite pairs of vertices can determine the same matrix, 'B16:H5' presents the same matrix as the one above. So there are two ways to represent a a matrix of cells. An empty string('') means *all* cells. In the following table, LARG is the left argument to the function listed, and RARG is the right argument to the function.

List of variable and functions

Name	Arguments		Description
	LARG	RARG	
[]OF (variable)		Name	the location of a file (include excel)
[]OP	0/1	'xls'	RARG: name or path; to open/close a workbook
[]XR	'Sheet'	'area'	RARG:'', 'B5' or 'B5:H16'; read data from a sheet
[]XW	'Sheet'	'data'	RARG: data; write data to a sheet
[]XN	'old'	'new'	To replace an old sheet name by a new sheet name
[]XD		'name'	To delete an existing sheet from an excel file
[]XC		'name'	To clear a sheet
[]XXR	'Sheet'	'area'	A high level version of []XR
[]XXW	'Sheet'	'data'	A high level version of []XW

2.1 Path

In order to transmit arguments flexibly, a variable and a function are used to identify paths. They are []OF and []OP. The default path of file directory is related to the workspace path. Using "Options->Workspace Path" to change a workspace path.

```
[]OF<-'demo'
```

[]OF is designed to store an Excel name or user defined path(contains file name). Before using this interface, it is important to give an existing file name to []OF. Currently, it is restricted to just store an Excel name. Note that there should be no extra suffix behind file name since suffix will be introduced by []OP as shown below.

```
0 []OP 'xls'
```

[]OP is a dyadic function. The left argument must be 0 or 1 correcponding to open and close of a file respectively. The right argument the type of a file for handling, 'xls' means 'Excel 2003'. If an workbook is not closed, opening this new workbook would cause an error. Furthermore, we intend to import more extensions here in the near future. Note that `1 []OP 'xls'` will close the active workbook and sheet.

All the functions below:

```
[]XR, []XW, []XN, []XD, []XC
```

need []OF and []OP for configuration. These five functions work when a workbook is opened or closed. This mode is designed for user who intends to do frequent operations on one sheet or among multiple sheets within a workbook.

2.2 Operations

Sheets usually are frequently used to uniformize stored data. Hence, several operations are needed to manipulate sheets. Before one calls these operations, the status of a workbook must be *open*.

```
w<-[]XD 'orange'
```

[]XD is a monadic function which only requires a sheet name as input. It will delete this sheet in the designated workbook if it exists or does nothing. In case the sheet really exists, it will be successfully deleted and return a 0. Otherwise, it returns 1.

```
w<-[]XC 'apple'
```

[]XC is a monadic function which also needs a sheet name as input. It will return a cleaned sheet. If a sheet is cleaned successfully, it would return 0.Otherwise, it returns 1. Note that the main reason for introducing this function is that writing data to a workbook would not necessarily clean the whole sheet before writing. So it is a prudent design to do so.

```
w<-'banana' []XN 'notyou'
```

[]XN is an dyadic function which requires two sheet names as input for users who want to change a sheet's name during data processing. The left argument stands for the old name and the right argument for the new name. It replaces the old sheet name by the new sheet name. If it fails to

replace the old sheet name, it returns 1; otherwise it returns 0.

2.3 Read/Write

Excel allows various types of data to be stored in one sheet. But one cell can only contain one type of data. Eli allows numerical data of boolean, integral and floating point types, and character strings to communicate with Excel. For string type data, Eli can only read from one string cell but can write a symbol or a character matrix to a sheet. Apart from this case, each cell in a selected area must have same type.

```
w<-'apple' []XR 'B2:B7'
```

[]XR([]XW) is designed for reading or writing data repteatedly when it is likely that the whole data set can not be transferred to a sheet in one operation or need multiple sheets to receive data.

```
Z<-'apple' []XXR 'B2:B7'
```

[]XXR([]XXW) does not carry multiple operations to open/close a workbook, it is convenient for reading (writing) data to one sheet of a workbook. Note that []XXR([]XXW) requires a correct path string to be stored in []OF.