RExcel - Using R from within Excel

Overview

The Excel addin `RExcel.xla` allows to use **R** from within Excel. The package additionally contains some Excel workbooks demonstrating different techniques for using **R** in Excel.

There are two types of servers, forground and background. The background server ist totally hidden from the user, all interaction with **R** has to be done in Excel. The foreground server allows direct access to the **R** GUI command line while working in Excel. The backgound server is installed with the R(D)COM server, the background server depends on the {**R**}-package `rcom`, which has to be installed >from CRAN.

Usage

There are at least three different ways of using **R** from within Excel

Scratchpad mode
: Writing **R** Code directly in an Excel worksheet and transferring scalar, vector, and matrix variables between **R** and Excel

Macro mode
: Writing macros using VBA and the macros supplied by `RExcel.xla`, attaching the macros to menu items or toolbar items

Worksheet functions
: **R** can be called directly in functions in worksheet cells

Scratchpad Mode

The RExcel menu contains the following items

R Start
: Initiates a connection to **R**
: If necessary starts an **R** process to be accessed >from Excel
: For the details of executing code at startup see section *Startup*

Close R
: When this process is running, **R Start** changes to **Close R**

Run Code
: Selecting a range (only one column wide) in Excel containing valid **R** code and then choosing this menu item executes the selected code

Get
: Gets the value of an **R** variable into the active Excel cell or range
: Possible values are scalars and matrices of numbers or strings, or dataframes.

Put
: Puts the values of the selected Excel range into an **R** variable.
: Possible values are scalars and matrices of numbers or strings, or dataframes.
: If a dataframe is to be put, the first row of the Excel range must contain the variable names for the dataframe.
: `Put` for dataframes only transfers data in visible rows and columns, hidden rows and columns will not be transferred.

Copy Code
: Puts the content of the selected range on the Windows clipboard, assuming that it is **R** code, and wraps the code in VBA procedure calls to make it ready for inclusion in VBA macros

Debug R
: Switches to debug mode: all the calls to **R** will be displayed in a popup window before they are executed

Debug R
: Switches to debug mode: all the calls to **R** will be displayed in a popup window before they are executed

Error Log
: Opens additional window with error messages and debugging information

Options
: Allows to change some aspects of the layout of the RExcel menu structure and startup delay for foreground server

Set R server
: Allows to select type and location of server. If library `rcom` is installed, allows to change `RExcel`'s so it can use the foreground server. Also allows to (de)activate an RCommander menu item on the RExcel menu

RExcel Help
: Displays this help file

R Help
: Displays the **R** help file

About RExcel
: Displays version information about the various software components involved

Get and **Put** interactively prompt for the name of the **R** variable or **R** expression.
The value of the **R** variable or expression only may be a numeric or string scalar, vector, or matrix, or a dataframe, otherwise RExcel will produce an error. The user dialog also allows to have row names for arrays and dataframes to be transferred. In that case, the first column if the selected Excel range will be used as vector of row names. For dataframes, column names always are tranferred. For arrays, column names are transferred optionally. In this case, the first row of the selected Excel range will be used for the column names.

While Excel as a connection to an **R** process, the context menu for cells (accessed by right clicking on a cell or selecting a range and then right clicking) contains the menu items **Run R**, **Get R Value**, **Put R Var**, **Get R Dataframe**, and **Get R Dataframe** which perform the same functions as the corresponding menu items in the **RExcel** menu.
The structure of this additional menu items may be changed in RExcel's **Options** menu.

Additionally, while Excel as a connection to an **R** process, the picture context menu of Excel has an additional menu item, `Prettify R graphics`, which helps formatting pictures pasted >from an **R** graphics window. Metafiles copied to the clipboard from **R** and then pasted into Excel have transparent background and no borders. Applying this menu item to such graphics will change the formatting to nontransparent background and borders.

Some ways of using these techniques are illustrated in the example file `RDemoDev.xls`, available as `Data transfer` on the Demo Worksheets menu.

Macro mode

VBA macros for accessing R can be written using the following VBA procedures and functions

`RInterface.StartRServer()`
: Starts the **R** server.

For the details of executing code at startup see section *Startup*

`RInterface.StopRServer()`

Stops the **R** server

`RInterface.RRun(commandstring)`

Executes `commandstring`

`RInterface.PutArray(varname,range,WithRowNames:=False,WithColNames:=False)`

Assigns the contents of range to **R** variable var.
If the named parameters `WithRownames` or `WithColNames` are True,
contents of the first row and/or the first columns of `range`
will be transferred as row and/or column names for the **R** object
and not be included in the values of the **R** object.

`RInterface.PutArrayFromVBA(Rvarname,VBAvarname)`

Assigns the contents of the VBA variable `VBAvarname` to **R** variable `Rvarname`.

`RInterface.GetArray(Rexpr,range)`

Puts the value of **R** expression `Rexpr` into Excel range `range`
`Rexpression` hast to have a value of scalar, vector, or matrix.
Values of type lists or dataframe will produce errors.

`RInterface.GetArrayToVBA(Rexpr)`

Returns the value of **R** expression `Rexpr` for further use in a VBA program
The same restrictions as for `RInterface.GetArray` apply.

`RInterface.PutDataframe(varname,range,WithRowNames:=False,RespectHidden:=False)`

Assigns the contents of range interpreted as dataframe (with variable names in the top row) to **R** variable var.
The named parameter `WithRowNames` controls if the first column of `range` is transferred as a dataframe variable or as row names of the dataframe.
The first row of the Excel range is always used for column names in the dataframe.
If the named parameter `RespectHidden` is True, data in hidden rows and columns will not be transferred. If the parameter is False, the whole range, including hidden rows and columns, will be transferred.

`RInterface.GetDataframe(varname,range)`

Puts the value of **R** variable `var` (which needs to be a dataframe) into Excel range `range`, putting variable names in the first row of the range

`RInterface.RunRFile(filename)`

Executes the commands in `filename` (file is on client computer)

`RInterface.PretttifyGraph(picture)`

Modifies a picture object by adding borders and setting nontransparent background

For example macros using these macros see example file `RDemoMacro.xls`, available as `Writing Macros` on the Demo Worksheets menu.

Many functions in **R** return compound objects, not arrays. These compound objects cannot be trasferred from **R** to Excel with `RInterface.GetArray`. The component has to be extracted. Therefore,
`RInterface.GetArray("lm(y~x)",Range("C1"))` will produce an error.
`RInterface.GetArray("lm(y~x)$coefficients",Range("C1"))` will transfer the regression coefficients to a range in Excel.

Worksheet functions

In Excel worksheets, the following functions can be used

`RPut(var,range,...)`

Assigns the value(s) from range to the **R** variable `var` and returns a string containing the name of `var`

`RStrPut(var,range,...)`

Assigns the value(s) from range to the **R** variable `var` as string(s) and returns a string containing the name of `var`

`RPutDataframe(var,range,attach,...)`

Assigns the value(s) from range to the **R** variable `var` as dataframe. The first row of the range has to contain the names of the variables. If the argument `attach` is given and has the Boolean value `TRUE`, then the dataframe is attached. For any other value of this argument, it is not attached. Returns a string containing the name of `var`.

`RFactor(var,...)`

Converts the **R** variable into a factor. `var` has to be a fully qualified name in **R**. For a nonattached dataframe `var` hast to be given in the form `dataframe$var`. Returns a string containing the name of `var`. Accepts only one name as argument.

`RFactorLevels(var,levels,...)`

Converts the **R** variable into an ordered factor. `levels` is a range containing the ordered factor levels. `var` behaves like the same argument of `RFactor` Returns a string containing the name of `var`.

`REval(expression,...)`

Returns the result of evaluating `expression`

`Component(args)`

Takes a series of strings and builds a string which is a R expression where the second and further arguments extract list components from the expression so far (used to get components from functions returning lists)

`RApply(function,args,...)`

Applies `function` to the arguments given by `args`. The values contained in the cells become the arguments of the function call.

`RApplyC(function,component,args,...)`

Applies `function` to the arguments given by `args`. The values contained in the cells become the arguments of the function call. Result of function is supposed to be an object. Return value of `RApplyC` ist the element component of this object.

`RApplyA(function,argstring,...)`

Applies function to the arguments given by `argstring`. `argstring` is a string with all the arguments for the function call.

`RApplyAC(function,component,args,...)`

Applies function to the arguments given by `argstring`. `argstring` is a string with all the arguments for the function call. Result of function is supposed to be an object. Return value of `RApplyAC` ist the element component of this object.

`RExec(range,...)`

Executes the contents of `range` but does not get a return value from R; returns the string "Done:" and text of executed R code.

`RCall(function,args,...)`

Applies function to the arguments given by `args`. `function` is a procedure, i.e. has no valid return value in R. The values contained in the cells become the arguments of the function call; returns the string "Done:" and text of executed R code.

`RCallA(function,argstring,...)`

Applies function to the arguments given by `argstring`. `function` is a procedure, i.e. has no valid return value in R. `args` is a string with all the arguments for the function call; returns the string "Done:" and text of executed R code.

`RSetEval(varname,expression,...)`

Assigns an R expression to an R variable. Return value of `RSetEval` is the name of the assingee variable

`RSetApply(varname,function,args,...)`
: Applies `function` to `args` (Excel values), assigns result to R variable. Return value of `RSetApply` is the name of the assingee variable.

`RSetApplyA(varname,function,argstring,...)`
: Applies `function` to arguments `argstring` (R values, arguments given as string), assigns result to R variable. Return value of `RSetApplyA` is the name of the assingee variable.

`MakeArgs(argrange,transpose=FALSE)`
: Creates a string with unnamed and named R arguments from an Excel range. If transpose=FALSE argument range is oriented columnwise, otherwise rowwise. Used to build argument strings for `RApplyA`, `RApplyAC`, `RCallA`, and `RSetApplyA`.

`RNumber(number)`
: Converts a number to a string with the decimal separators needed by **R**

`RPut` and `RStrPut` assign matrix values when necessary.

`REval` and `RApply` may return arrays and therefore may be used as Excel array functions.

All these functions accept dummy arguments. Dummy arguments are needed so that Excel performs automatic recalculation in the correct order. A range referenced in a dummy argument will be recalculated before the range containing the reference.

The functions `RPut`, `RStrPut`, `RPutDataframe`, `RFactor`, `RFactorLevels`, `REval`, `REvalC`, `RApply`, `RApplyC`, `RApplyA`, `RApplyAC`, `RExec`, `RCall`, `RCallA`, `RSetEval`, `RSetApply`, and `RSetApplyA` accept dummy arguments. Dummy arguments are needed so that Excel performs automatic recalculation in the correct order. A range referenced in a dummy argument will be recalculated before the range containing the reference.

For the functions `RPut`, `RStrPut`, `RPutDataframe`, `RFactor`, `RFactorLevels`, `REval`, `REvalC`, `RApplyA`, `RApplyAC`, `RExec`, `RCallA`, `RSetEval`, and `RSetApplyA` it is clear which arguments are dummy arguments.

For `RApply`, `RApplyC`, `RCall`, and `RSetApply`, all arguments including and after an argument with the value "depends" are considered dummy arguments.

`REval`, `RApply`, and `RApplyC` can be used to define VBA functions called from worksheet cells. In that case, even a description for Excel's function wizard can be added for these VBA functions.

`AddDescription(name,text)`
: Adds a description to be used by Excel's function wizard for a user defined VBA function

`AddDescription` should be called from an `Auto_Open` macro for the worksheet containing the definition of the VBA function.

`RProc` is deprecated. `RExec` should be used instead.

`RVarSet` is deprecated. `RSetEval` should be used instead.

When a workbook containing any of these functions is loaded into Excel, a connection to **R** is initiated immediately.
For the details of executing code at startup see section *Startup*

When using **R** worksheet functions, debug mode can produce quite a lot of interactive messages. Therefore, it should be used very cautiously under these circumstances.

Examples on how to use these functions can be found in the example files `RDemoRecalc.xls`, `RDemoDens.xls`, and `RDemoGraph.xls`, available as `Worksheet functions`, and `Graphics with sliders`, and `Interactive graphics` on the Demo Worksheets menu. `Interactive graphics` needs R version 1.9.1 or higher.

Data transfer and missing values

RExcel works with the following data types:

- numeric data (integer and real)
- strings
- date and time
- complex numbers

RExcel handles missing values in Excel and R. Excel has the special code `#N/A` for missing values. It can either be typed manually or can be produced by the function `NA()`. **R** used the symbol `NA` for missing values. Furthermore, Excel has codes for different numerical errors, and **R** has NaN for numerical errors. Since error handling is somewhat different in Excel and **R**, RExcel allows to select from 3 different methods `Excel mode`, `Loose mode`, and `Strict mode` when transferring data from Excel to **R** or back.

For numeric ranges, these 3 modes behave in the following way (when transferring from Excel to **R**):

`Excel mode`
: Empty cells become 0, #N/A becomes NA, numeric errors become NaN.

`Loose mode`
: Empty cells become NA, #N/A becomes NA, numeric errors become NA.

`Strict mode`
: Empty cells become NA, #N/A becomes NA, numeric errors become NaN.

For string (character) ranges, these 3 modes behave in the following way (when transferring from Excel to **R**):

`Excel mode` and `Loose mode`
: Empty cells become empty strings, #N/A becomes NA.

`Strict mode`
: Empty cells become NA, #N/A becomes NA.

For numeric ranges, these 3 modes behave in the following way (when transferring from **R** to Excel):

`Excel mode` and `Loose mode`
: NA becomes empty cells, NaN becomes #N/A.

`Strict mode`
: NA becomes #N/A, NaN becomes #NUM!.

For string (character) ranges, these 3 modes behave in the following way (when transferring from **R** to Excel):

`Excel mode` and `Loose mode`
: Empty strings becomes empty cells (empty strings), NA becomes empty cells (empty strings).

`Strict mode`
: Empty strings become #N/A, NA becomes #N/A.

Server types and locations

RExcel can handle 4 different types of R servers, and the servers can be located either on the same machine as the RExcel client, or on a remote machine. This makes fo a total of 8 different configurations.

Additional background information about the servers is given in the file `RExcelReadme.txt` contained in the package(s).

The server types are

Background
: Server runs invisibly in background, interaction with R only throgh Excel.

Foreground
: Server runs visibly in foreground, interaction with R either in Excel or from the RGui command line. Only available if R library `rcom` is installed on the same machine as `RExcel`.

Serverpool exclusive
: R server is managed in a serverpool. Only one instance of Excel can access one R process.

Serverpool shared
: R server is managed in a Serverpool. Multiple instances of Excel or other COM clients can access one R process.

When using **R** servers from a serverpool (either exclusive or shared) , programs like RServerManagerAdministrator (installed with R(D)COM server) manage the serverpool.

The menu item Set R server allows to select the server type, server name (for remote servers), and R process name (for servers from a serverpool).

Remote machine name can be either textual names or IP addresses.

Startup

When RExcel initiates the connection to **R**, a few things happen.

When the server mode is set to background server, a new **R** process is started with the (D)COM server.

When server mode is set to foreground, RExcel checks if there is an **R** process started from the command which has loaded the `rcom` library. If this is the case, RExcel connects to this process, otherwise, a new **R** process is started and `rcom` is loaded. This process will also perform all site specific startup actions (defined e.g. in `$RHOME\etc\RProfile.site`).

Then, RExcel looks for a file `RExcelStart.R` in the directory where RExcel is located. This is the place to define site specific customizations of RExcel.

Finally, RExcel looks for a file `RExcelStart.R` in the current directory. If Excel's current workbook is a previously saved `.xls` file, the directory containing this file is the current directory. Otherwise, it is the (configuration dependent) default Excel data directory.

Developing Applications

It is possible to develop Excel applications in a way the user does not see **R** directly. The **R** macros can be called from custom menus, and the RExcel menu can stay completely hidden.

To hide the Excel menu, the first line in module `AAConfigParams` in `RExcel.xla` should be changed to

```
Public Const DisplayRExcelMenu = False
```

Additionally, an Excel workbook using the macros (not the functions) in RExcel.xla must have a reference to `RExcelVBAlib` in the Tools->Reference list in the VBA development environment of Excel.

Type and location of the server can be set from Excel macros.

`SetServerType(servertype)`
: accepts the following predefined constants for `servertype`:

```
stLocalBackground
stRemoteBackground
stLocalForeground
stRemoteForeground
stLocalPoolExclusive
stRemotePoolExclusive
stLocalPoolShared
stRemotePoolShared
```

`SetServerName(servername)`
: takes a valid server name as string. Only to be used for servertypes `stRemoteBackground`, `stRemoteForeground`, `stRemotePoolExlusive`, and `stRemotePoolShared`

`SetProcessName(processname)`
: takes a valid R process name (managed by RServerManager). Only to be used for servertypes `stLocalPoolExclusive`, `stRemotePoolExclusive`, `stLocalPoolShared`, and `stRemotePoolShared`

Notes for users

When RExcel is used to create graphics (either from macros or from cell formulas), the window containing the **R** graphics may get focus. To get focus back to Excel, the user has to click on Excel. This first click will not change the current selection in Excel. To select a specicic cell, the user has to click on the cell once again. So, when just clicking on a cell in Excel only once, the cell might not be immediately selected. A second click is necessary.

Author(s)

Thomas Baier and Erich Neuwirth

References

The current version and development versions of the `R(D)COM server`, `rcom`, and `RExcel` always are available from http://rcom.univie.ac.at/

[Package Index]