Embedding R in Windows applications, and executing R remotely

Thomas Baier and Erich Neuwirth

February 15, 2004

R is object oriented, and objects are the standard way of packing analysis results in R.

Many programming environments and applications programs in Windows can act as as (D)COM clients, which is the standard way of accessing objects exposed by (D)COM servers. Particularly, all Microsoft office programs are (D)COM clients and therefore can access any (D)COM server. Therefore, in encapsulating R as a (D)COM server is a natural choice to make R functionality accessible to other programs.

To embed R in other programs, therefore, one of the key questions is what kind of objects are exposed to these applications hosting R, and how these objects are exposed.

There are two different routes that can be taken here. We can either choose to expose R objects "as is", with all their richness, or we can choose a minimalist approach and only offer objects of types which can be handled easily by programs which normally do not employ rich object types for the data they usually handle.

The difference can be very well illustrated when R is embedded in Excel. A spreadsheet essentially has 6 data types, scalars, vectors, and matrices of either numbers or strings. If we want to make R functionality a part of the spreadsheet functionality, it is sufficient that the R (D)COM server exposed this type od data objects.

On the other hand, VBA (the programming language built into Excel) allows to work with any type of object. Therefore, the whole R object model, and even user-defined new object types, can be made accessible in VBA, and therefore be used in Excel.

The question is, how is R being used in connection with Excel. When the programmes "thinks R" and uses Excel just as a convenient data source and data editor, the full object model makes sense. Then, programming is done in R and VBA, and data and results are just transferred from time to time between worksheets and R. This way, Excel becomes a convenience item for R, but conceptually R is the center of the programming model.

If we want to use R as an extension of Excel worksheets, and only as subroutines accessible from VBA, the minimalist approach seems more adapted. In this case, calls to R will only return objects which can directly be embedded in worksheets. One of the key concepts or spreadsheet programs is automatic recalculation. Using data types which can immediately be embedded in the worksheet makes R calculations become part of Excel's automatic recalculation, thereby offering facilities not offered by R itself. Using only simple data types like arrays allows very fast implementation of the interface. Using the full R object model adds another level of complexity and therefore probably slows down calculation considerably. Calculation speed is very important for reasonable automatic recalculation, therefore this approach R leads to a less natural spreadsheet extension. Additionally, if the R server is executed on another machine than the client, transfer speed also plays an important role, and using only native Windows data types speeds up things considerably.

The relative merits of the 2 different approaches also heavily depend on the experience of the programmer using R as an embedded library. To be able to use the full R object hierarchy, one has to be rather knowledgeable about R's object model, and understand the relationships between different kinds of objects. Making R objects fully accessible in applications really puts just another kind of syntactic glue (in the case of Excel the glue is VBA) on top of R's objects.

Using the minimalist approach allows simpler access to R functions in other applications. The interface to R can be kept much simpler. Of course, the price to pay is that we do not have the full richness of R accessible in the application environment directly. It is, however, always possible to encapsulate everything in R code which only returns the simple data types.

If we separate R core functionality, especially the statistical methods needed in an application, from the data handling and interface code, it makes sense to write the core functionality as R functions returning only simple data types. Then, all the additional code (GUIs, spreadsheet functions) can be written without detailed knowledge of the R object hierarchy.

Another problem when we embed R into another application is who of the two partners is the authority for the state of data objects. When we transfer data to R, we assign the data to variables. What happens if the data in the hosting application is changed? Will these changes automatically propagate to R? As soon as we use variables in both applications, we have to be very careful about keeping variables synchronized.

If we apply a strict functional model, R only exposing (stateless) functions, and not (stateful) data objects, then we elegantly avoid this problem. To be able to apply that model, all the functions supplied by R have to return data types which can immediately be represented in the hosting application.

We will show some applications with both approaches, and we will demonstrate how the different approaches influence calculation speed.