Methods for `format` provide very flexible ways to convert them to character strings. We have an `axis` method to use them to label graphs. Lots of methods are needed, for `all.equal`, `as.character`, `c`, `cut`, `mean`, `round`, `seq`, `str`, ... And these need to check for appropriateness, so for example sums of dates are not well-defined, whereas means are.

We have also provided convenience functions like `weekdays`, `months` and `quarters`, which either extract information from the `POSIXlt` list or convert using an appropriate `format` argument in a call to the `format` method. The `POSIXt` method for the (new) generic function `julian` converts to Julian dates (the number of days since some origin, often 1970-01-01).

The future

We believe that the date-time classes in base R now provide sufficient flexibility and facilities to cover almost all applications and hence that they should now be used in preference to earlier and more limited systems. Perhaps most important is that these classes be used in inter-system applications such as database connectivity.

References

International Organization for Standardization (1988, 1997, ...) ISO 8601. Data elements and interchange formats – Information interchange – Representation of dates and times. The 1997 version is available on-line at `ftp://ftp.qsl.net/pub/g1smd/8601v03.pdf`.

Kline, K. and Kline, D. (2001) *SQL in a Nutshell.* O'Reilly.

Lewine, D. (1991) *POSIX Programmer's Guide. Writing Portable UNIX Programs.* O'Reilly & Associates.

Brian D. Ripley
University of Oxford, UK
`ripley@stats.ox.ac.uk`

Kurt Hornik
Wirtschaftsuniversität Wien, Austria
Technische Universität Wien, Austria
`Kurt.Hornik@R-project.org`

Installing R under Windows

by Brian D. Ripley

Very few Windows users will have ever experienced compiling a large system, as binary installations of Windows software are universal. Further, users are used to installing software by a point-and-click interface with a minimum of reading of instructions, most often none. The expectation is

> Insert the CD.
>
> If it doesn't auto-run, double-click on a file called `Setup.exe` in the top directory.
>
> Go through a few 'Wizard' pages, then watch a progress bar as files are installed, then click on Finish.

Contrast this with 'untar the sources, run `./configure`, `make` then `make install`'. Each in its own way is simple, but it is really horses for courses.

Every since Guido Masarotto put out a version of R for Windows as a set of `zip` files we have been looking for a way to install R in a style that experienced Windows users will find natural. At last we believe we have found one.

When I first looked at this a couple of years ago most packages (even Open Source ones) used a commercial installer such as InstallShield or Wise. Although I had a copy of InstallShield, I was put off by its size, complexity and the experiences I gleaned, notably from Fabrice Popineau with his `fptex` installation.

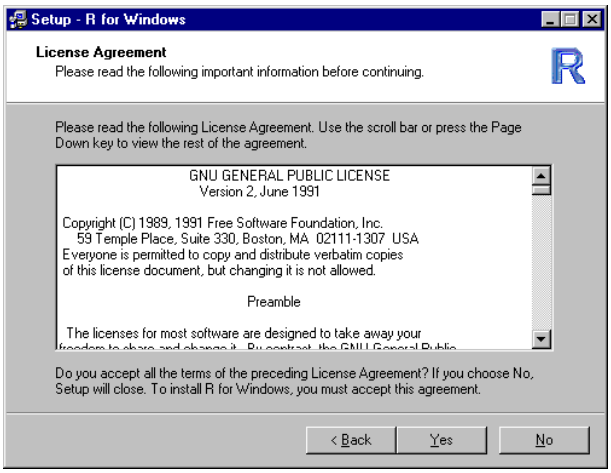
Shortly afterwards, MicroSoft introduced their own installer for their Office 2000 suite. This works in almost the same way, except that one double-clicks on a file with extension `.msi`. There is a development kit for this installer and I had expected it to become the installer of choice, but it seems rarely used. (The `Perl` and now `Tcl` ports to Windows do use it.) That makes one of its disadvantages serious: unless you have a current version of Windows (ME or 2000) you need to download the installer `InstMsi.exe`, which is around 1.5Mb and whose installation needs privileges an ordinary user may not have.

In May 1999 I decided to write a simple installer, `rwinst.exe`, using the GraphApp toolkit that Guido had used to write the R for Windows GUI, and this has been in use since. But it was not ideal, for

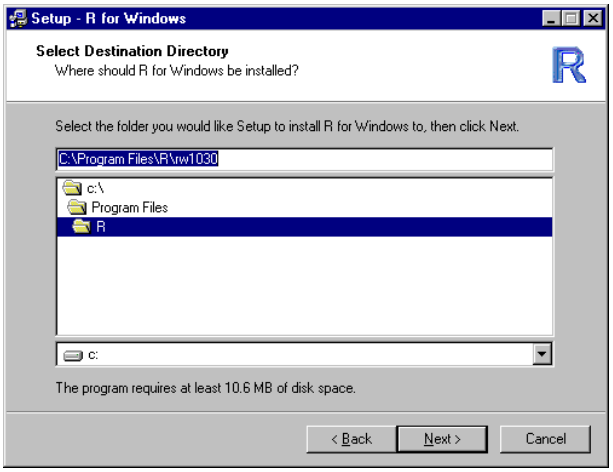
- It did not use a completely standard 'look-and-feel'.
- It was hard to maintain, and mistakes in an installer are 'mission-critical'.
- Users found it hard to cope with needing several files to do the installation.

The prospect of *recommended* packages at version 1.3.0 was going to require twice as many files, and forced a re-think in early 2001. I had earlier looked at Inno Setup by Jordan Russell (`www.jrsoftware.org`)

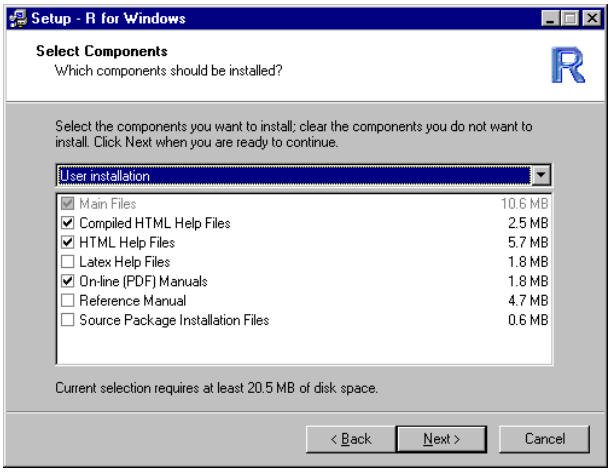
(a) The 'welcome' screen



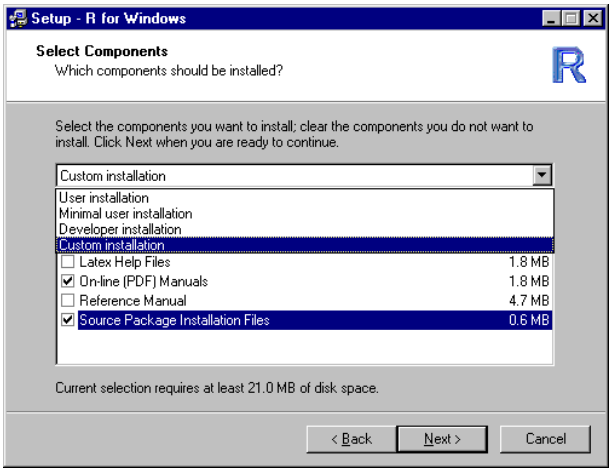
(b) The licence – GPL2 of course



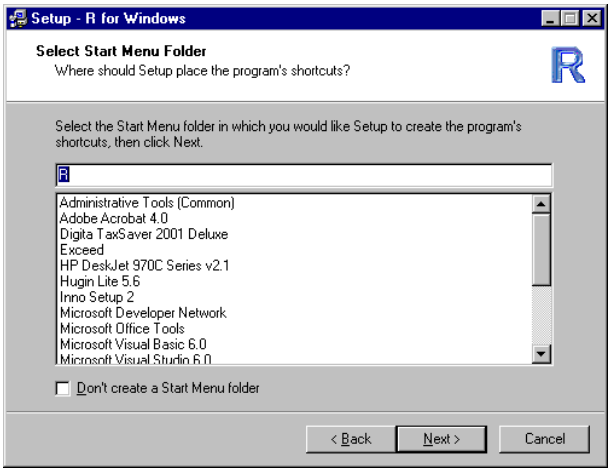
(c) Choose the installation folder



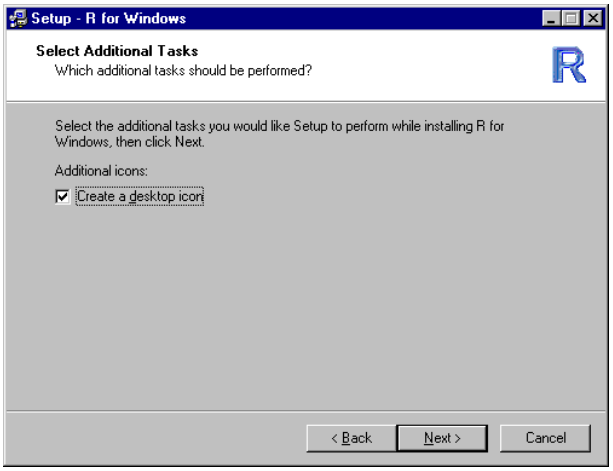
(d) Select components as required



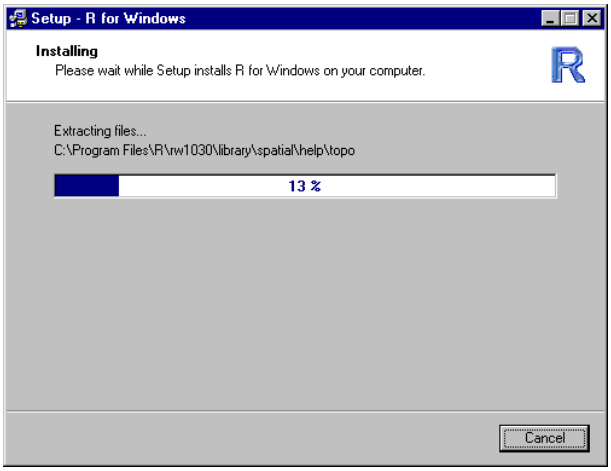
(e) There is a list of standard selections, plus 'custom'



(f) Choose a group name for the Start menu



(g) Do you want a desktop icon?



(h) Installing the files

and had found it too limited. However, an enhanced version 2 was by then in beta release and was used to make an alternative installer for R 1.2.3. Inno Setup 2 is now released and has been adopted for the recommended installers for R 1.3.0. The figures show the installation sequence using `SetupR.exe` for 1.3.0. Note that there is a fair degree of control but with sensible defaults. For example, the commercial installers do not usually offer the option **not** to add to the start menu, nor to add a desktop icon (and that is labelled by the R version number).

The `SetupR.exe` installation is a single file of about 15Mb, which can conveniently be distributed on CD-R or downloaded (over a fast enough Internet connection).

One issue that had been a concern was that it is thought that some users need to install R from floppies, and the `.zip` files had been designed where possible[1] to fit on a single floppy. How much demand there was/is is hard to gauge, and such users almost by definition are unlikely to be fairly represented by email contact. As R grew, maintaining a division into floppy-sized pieces became harder, and needed constant vigilance. Fortunately Inno Setup provided a simple option to produce (exactly) floppy-sized pieces, and we have an alternative distribution consisting of `miniR.exe` and `miniR-1.bin` which fit on one floppy, and `mini-2.bin` to `mini-6.bin` which fit on five further floppies. The six floppies are very full, and doubtless the next release will need seven.

Inno Setup is a Delphi program which is freely available (including source). It is programmed by a script of instructions. For a project the size of R that script needs to be a few thousand lines, but is generated automatically by a `Perl` script from the distribution files. I had a workable installer running in a couple of hours, have spent less than a day on it in total, and future maintenance of this installer should be a much easier task

Uninstalling

Surely you wouldn't want to remove something as useful as R?

One good reason to remove R is to clear space before upgrading to a new version. In the past the only way to do this was to delete the whole directory containing R, say `c:\R\rw1021`. That did remove everything, as R touched nothing else (it did not use the Registry nor the system directories), but it would also wipe out any customizations the user had made, as well as all the add-on packages installed in `c:\R\rw1021\library`.

Inno Setup automatically provides an uninstaller, and an R installation made with `setupR.exe` or `miniR.exe` can be removed via the R group on the `Start` menu, from the Control Panel[2] or by running `Uninst.exe` in the top directory of the installation. This does only uninstall the files it installed, so that added packages are left. It also removes the Registry entries it set.

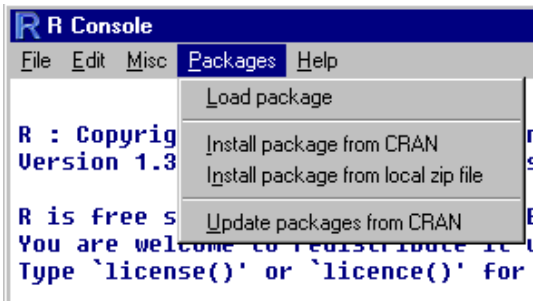
This makes upgrading easy. Just uninstall the old version, install the new one and move the added packages across to `c:\R\rw1030\library` (say). **Note:** some packages may need to be updated to work with new versions of R, so as a final step run `update.packages()`. Which brings us to ...

Installing packages

Windows' users have almost all expected to have packages pre-compiled for them, and to have a

[1] this was never possible for the `.zip` file containing the 4.6Mb PDF version of the reference manual.
[2] for sufficiently privileged users only

simple way to install them. Another function of `rwinst.exe` was to provide a point-and-click way to install pre-compiled packages. Now `rwinst.exe` is no longer supported[3], we needed another way to ease the installation of packages. This is provided by the Packages menu introduced in R 1.3.0. This takes advantage of the new facilities to download files from URLs to allow installation of packages either from a local zip file or directly from a CRAN node.



Other items on that menu provide shortcuts to `library()` and `update.packages()`.

Brian D. Ripley
University of Oxford, UK
ripley@stats.ox.ac.uk

Spatial Statistics in R

by Brian D. Ripley

The following two articles discuss two recent spatial statistics packages for R, and the Editors suggest that I write an overview of the area.

There are several packages available, almost all of which originated as S code and from quite different communities. As a result there is considerable overlap in their functionality. Many are quite old (pre-dating classes in S, for example, and from an era of much lower levels of computing resources). In roughly historical order there are

akima An R interface by Albrecht Gebhardt to spatial spline interpolation Fortran code by H. Akima. Closely modelled on the `interp` function in S-PLUS.

tripack Delaunay triangulation of spatial data. An R interface by Albrecht Gebhardt to Fortran code by R. J. Renka.

spatial Now part of the **VR** bundle. Contains trend-surface analysis, kriging and point-process code originally written by B. D. Ripley in 1990–1 for teaching support.

VR version 6.2-6 includes enhanced trend-surface code contributed by Roger Bivand.

sgeostat Geostatistical modelling code written for S by James J. Majure and ported to R by Albrecht Gebhardt.

splancs Originally commercial code for spatial and space-time point patterns by Barry Rowlingson. Roger Bivand has made a GPL-ed version available for R, with contributions from Giovanni Petris.

spatstat Code for point pattern analysis originally written for S-PLUS 5.1 by Adrian Baddeley and Rolf Turner. The version on the website (`http://www.maths.uwa.edu.au/~adrian/spatstat.html`) is said to work with R 1.0.1.

geoR Geostatistical code by Paulo J. Ribeiro. See the next article. Paulo does not mention that he also has a **geoS**.

RandomFields Specialized code to simulate from continuous random fields by Martin Schlather. See the next but one article.

Which should you use? Spatial statistics is not a single subject, and it depends which part you want. It is natural for me to use the classification of Ripley (1981).

Smoothing and interpolation is still a range of methods. Package **akima** provides one reasonable interpolation method, and package **spatial** provides the commonest of smoothing methods, trend surfaces. Kriging can be either interpolation or smoothing, and is covered at various depths in **spatial**, **sgeostat** and **geoR**.

For **spatial autocorrelation** there is nothing available yet. For spatial lattice process (e.g. CAR processes) there is no specialized code, but `gls` from **nlme** could be used.

For **spatial point patterns**, **spatial** provides the basic tools, and methods to fit Strauss processes. **splancs** handles polygonal boundaries, and it and **spatstat** have a wider range of methods (by no means all of which I would recommend).

[3] it can still be compiled from the sources.