R for Windows FAQ

Frequently Asked Questions on R for Windows
Version 3.3.0

B. D. Ripley and D. J. Murdoch

Table of Contents

1 Introduction

This FAQ is for the Windows port of R: it describes features specific to that version. The main R FAQ can be found at

https://CRAN.R-project.org/doc/FAQ/R-FAQ.html.

The information here applies only to recent versions of R for Windows, ('3.1.0' or later). It is biased towards users of 64-bit Windows.

2 Installation and Usage

2.1 Where can I find the latest version?

Go to any CRAN site (see https://cran.r-project.org/mirrors.html for a list), navigate to the `bin/windows/base` directory and collect the file(s) you need. The current release is distributed as an installer '`3.3.0 -win.exe`' of about 65MB.

There are also links on that page to the '`r-patched`' and '`r-devel`' snapshots. These are frequently updated builds of development versions of R. The '`r-patched`' build includes bug fixes to the current release, and '`r-devel`' contains these as well as changes that are planned to eventually make it into the next '`x.y.0`' release.

2.2 How do I install R for Windows?

Current binary versions of R run on Windows XP or later, including on 64-bit versions: See Section 2.27 [Can I use R on 64-bit Windows?], page 11. The last version known to run on Windows 2000 was 2.12.2.

We only test on versions of Windows currently supported by Microsoft, mainly 64-bit Windows 7 and Server 2008.

Your file system must allow case-honouring long file names (as is likely except perhaps for some network-mounted systems). An installation takes up to 150MB of disk space.

If you want to be able to build packages from sources, we recommend that you choose an installation path not containing spaces. (Using a path with spaces in will probably work, but is little-tested.) Users of Vista/Windows 7/8/Server 2008/2012 installing for a single user using an account with administrator rights[1] should consider installing into a non-system area (such as `C:\R`). Installing to a network share (a filepath starting with `\\machine\...`) is not supported: such paths will need to mapped to a network drive.

To install use '`3.3.0 -win.exe`'. Just double-click on the icon and follow the instructions. If you have an account with Administrator privileges you will be able to install R in the `Program Files` area and to set all the optional registry entries; otherwise you will only be able to install R in your own file area. You may need to confirm that you want to proceed with installing a program from an 'unknown' or 'unidentified' publisher.

After installation you should choose a working directory for R. You will have a shortcut to `Rgui.exe` on your desktop and/or somewhere on the Start menu file tree, and perhaps also in the Quick Launch part of the taskbar (Vista and earlier). Right-click each shortcut, select Properties... and change the 'Start in' field to your working directory. (If your account was not the one used for installation, you may need to copy the shortcut before editing it.)

On some systems you will have two shortcuts, one for 32-bit with a label starting `R i386` and one for 64-bit starting `R x64` (see Section 2.28 [Should I run 32-bit or 64-bit R?], page 11)

You may also want to add command-line arguments at the end of the Target field (*after* any final double quote, and separated by a space), for example `--sdi --max-mem-size=1G`. You can also set environment variables at the end of the Target field, for example `R_LIBS=p:/myRlib`, and if you want to ensure that menus and messages are in (American) English, `LANGUAGE=en`.

[1] Non-administrator accounts will automatically be offered a default installation directory in the user area.

It is also possible to install from an MSI file, which will be of interest only for system administrators. For how to build the MSI file, see the 'R Installation and Administration Manual'.

2.3 How do I check an installation is not corrupt?

Relates to earlier installers, removed in R 2.11.0.

2.4 Can I customize the installation?

The normal way to customize the installation is by selecting components from the wizards shown by the installer. However, sysadmins might like to install R from scripts, and the following command-line flags are available for use with the installer.

'`/SILENT`' only show the installation progress window and error messages.

'`/VERYSILENT`'
: only show error messages.

'`/DIR="x:\dirname"`'
: set the default installation directory

'`/GROUP="folder name"`'
: set the default Start-menu group name

'`/COMPONENTS="comma separated list of component names"`'
: set the initial list of components: Components are named '`main`', '`i386`', '`x64`' and '`translations`'.

It is also possible to save the settings used to a file and later reload those settings using

'`/SAVEINF="filename"`'
: save the settings to the specified file. Don't forget to use the quotes if the filename contains spaces.

'`/LOADINF="filename"`'
: instructs the installer to load the settings from the specified file after having checked the command line.

A successful installation has exit code 0: unsuccessful ones may give 1, 2, 3, 4 or 5. See the help for Inno Setup (`http://jrsoftware.org/`) for details.

We have some facilities for building a customized installer, in particular to add packages to the installer. See the 'R Installation and Administration' manual in the subsection 'Building the installers'.

2.5 How do I run it?

Just double-click on the shortcut you prepared at installation.

If you want to set up another project, make a new shortcut or use the existing one and change the 'Start in' field of the Properties.

You may if you prefer run R from the command line of any shell you use, for example a 'Command Prompt' or a port of a Unix shell such as `tcsh` or `bash`. (The command line can be anything you would put in the Target field of a shortcut, and the starting directory will

be the current working directory of the shell. Note that the R executables are not by default added to the `PATH`.) People running from a terminal usually prefer to run `Rterm.exe` and not `Rgui.exe`.

2.6 Can I run R from a CD or USB drive?

Yes, with care. A basic R installation is relocatable, so you can burn an image of the R installation on your hard disc or install directly onto a removable storage device such as a flash-memory USB drive.

Running R does need access to a writable temporary directory and to a home directory, and in the last resort these are taken to be the current directory. This should be no problem on a properly configured version of Windows, but otherwise does mean that it may not be possible to run R without creating a shortcut starting in a writable folder.

2.7 How do I UNinstall R?

Normally you can do this from the '`Programs and Features`' group in the Control Panel. If it does not appear there, run `unins000.exe` in the top-level installation directory. On recent versions of Windows you may be asked to confirm that you wish to run a program from an 'unknown' or 'unidentified' publisher.

Uninstalling R only removes files from the initial installation, not (for example) packages you have installed or updated.

If all else fails, you can just delete the whole directory in which R was installed.

2.8 What's the best way to upgrade?

That's a matter of taste. For most people the best thing to do is to uninstall R (see the previous Q), install the new version, copy any installed packages to the library folder in the new installation, run `update.packages(checkBuilt=TRUE, ask=FALSE)` in the new R and then delete anything left of the old installation. Different versions of R are quite deliberately installed in parallel folders so you can keep old versions around if you wish.

For those with a personal library (folder `R\win-library\`*`x.y`* of your home directory, `R\win64-library\`*`x.y`* on 64-bit builds), you will need to update that too when the minor version of R changes (e.g. from 3.0.2 to 3.1.0). A simple way to do so is to copy (say) `R\win-library\3.0` to `R\win-library\3.1` before running `update.packages(checkBuilt=TRUE, ask=FALSE)`.

2.9 There seems to be a limit on the memory it uses!

Indeed there is. It is set by the command-line flag `--max-mem-size` (see Section 2.2 [How do I install R for Windows?], page 2) or by environment variable `R_MAX_MEM_SIZE`.

For a 64-bit build of R it defaults to the amount of RAM.

For a 32-bit build of R it defaults to the smaller of the amount of physical RAM in the machine and 0.5GB less than the limit on user virtual memory for a process (most often 2GB when using a 32-bit edition of Windows).

Use `?Memory` and `?memory.size` for information about memory usage. The limit can be raised (if possible) by calling `memory.limit` within a running R session.

The 32-bit executables support up to 3GB of user address space per process under suitably enabled versions of 32-bit Windows (see http://www.microsoft.com/whdc/system/platform/server/PAE/PAEmem.mspx, http://msdn.microsoft.com/en-us/library/bb613473%28VS.85%29.aspx; this is not enabled by default). On such systems, the default for `--max-mem-size` is the smaller of the amount of RAM and 2.5GB. On all but the earliest 64-bit versions of Windows the user address space for a 32-bit process is 4GB, and there the default for `--max-mem-size` is the smaller of the amount of RAM and 3.5GB.

2.10 How can I keep workspaces for different projects in different directories?

Create a separate shortcut for each project: see Q2.5. All the paths to files used by R are relative to the starting directory, so setting the 'Start in' field automatically helps separate projects.

Alternatively, start R by double-clicking on a saved `.RData` file in the directory for the project you want to use, or drag-and-drop a file with extension `.RData` onto an R shortcut. In either case, the working directory will be set to that containing the file.

2.11 How do I print from R?

It depends what you want to print.

- You can print the graphics window from its menu or by using `dev.print` with suitable arguments (see its help page: most likely `dev.print(win.graph)` will work).
- You can print from the R console or pager by '`File | Print`'. (This will print the selection if there is one, otherwise the whole console or pager contents.)
- You can print help files from the pager or HTML browser.
- If you have LaTeX installed and a PDF printing system you can print help files by `help(fn_name, help_type="PDF")`.

2.12 Can I use `R CMD BATCH`?

Yes: use `R CMD BATCH --help` or `?BATCH` for full details.

You can also set up a batch file using `Rterm.exe`. A sample batch file might contain (as one line)

```
path_to_R\bin\x64\Rterm.exe --no-restore --no-save < %1 > %1.out 2>&1
```

The purpose of `2>&1` is to redirect warnings and errors to the same file as normal output.

2.13 Can I use 3.3.0 with ESS and Emacs?

Yes. ESS has for a long time supported R under Windows: it does so by running `Rterm.exe` without a visible console.

For help with ESS, please send email to ESS-help@stat.ethz.ch, not the R mailing lists.

2.14 What are HOME and working directories?

Several places in the documentation use these terms.

The working directory is the directory from which `Rgui` or `Rterm` was launched, unless a shortcut was used when it is given by the 'Start in' field of the shortcut's properties. You can find this from R code by the call `getwd()`.

The home directory is set as follows: If environment variable `R_USER` is set, its value is used. Otherwise if environment variable `HOME` is set, its value is used. After those two user-controllable settings, R tries to find system-defined home directories. It first tries to use the Windows "personal" directory (typically `C:\Users\username\Documents`). If that fails, if both environment variables `HOMEDRIVE` and `HOMEPATH` are set (and they normally are), the value is `${HOMEDRIVE}${HOMEPATH}`. If all of these fail, the current working directory is used.

You can find this from R code by `Sys.getenv("R_USER")` or `normalizePath("~")`, '~' being Unix notation for the home directory.

2.15 How do I set environment variables?

Environment variables can be set for `Rgui.exe` and `Rterm.exe` in three different ways.

1. On the command line as name=value pairs. For example in the shortcut to `Rgui` you could have

   ```
   "path_to_R\bin\x64\Rgui.exe" HOME=p:/ R_LIBS=p:/myRlib
   ```

2. In an environment file `.Renviron` in the working directory or in your home directory, for example containing the line

   ```
   R_LIBS=p:/myRlib
   ```

   If you have permission to do so, you can also create an environment file `etc\Renviron.site` and set environmental variables in that file in the same way. This is useful for variables which should be set for all users and all usages of this R installation. (Their values can be overridden in a `.Renviron` file or on the command line.)

   See `?Startup` for more details of environment files.

3. For all applications via Windows. How you set an environment variable is system-specific: under recent versions of Windows, go to 'User Accounts' in the Control Panel, and select your account and then 'Change my environment variables'.

The order of precedence for environmental variables is the order in which these options are listed, that is the command line then `.Renviron` then the inherited environment.

2.16 R can't find my file, but I know it is there!

How did you specify it? Backslashes have to be doubled in R character strings, so for example one needs '`"d:\\3.3.0 \\library\\xgobi\\scripts\\xgobi.bat"`'. You can make life easier for yourself by using forward slashes as path separators: they do work under Windows. You should include the file extension (e.g. '`"xgobi.bat"`' rather than just '`"xgobi"`'); sometimes this isn't shown in Windows Explorer, but it is necessary in R.

A simple way to avoid these problems is to use the function `file.choose()` to invoke the standard Windows file selection dialog. If you select a file there, the name will be passed to R in the correct format.

Another possible source of grief is spaces in folder names. We have tried to make R work on paths with spaces in, but many people writing packages for Unix do not bother. So it is worth trying the alternative short name (something like '`PROGRA~1`'; you can get it as the 'MS-DOS name' from the Properties of the file on some versions of Windows, and from `dir /X` in a '`Command Prompt`' window), and using the function `shortPathName` from R code.

2.17 Does R use the Registry?

Not when R itself is running.

When you run the R installer, there are options (under '`Select Additional Tasks`') to '`Save version number in registry`' and (for Administrator installs) '`Associate R with .RData files`'.

If you tick the first option, the following string entries are added to the Windows registry:

- `HKEY_LOCAL_MACHINE\Software\R-core\R\Current Version` contains the version number, currently 3.3.0 .
- `HKEY_LOCAL_MACHINE\Software\R-core\R\[version]\InstallPath` (where `[version]` is currently 3.3.0) contains the path to the R home directory.

If you do not have administrative privileges on the machine while running the installer, then the entries are created under `HKEY_CURRENT_USER`. The same entries are also created under `Software\R-core\R32` or `Software\R-core\R64`, for 32- and 64-bit R respectively.

If you tick the second option (shown with administrative privileges only) ('`Associate R with .RData files`') then entries are created under `HKEY_CLASSES_ROOT\.RData` and `HKEY_CLASSES_ROOT\RWorkspace`.

After installation you can add the Registry entries by running `RSetReg.exe` in a sub-folder of the `bin` folder, and remove them by running this with argument `/U`. Note that this requires administrative privileges unless run with argument `/Personal` and neither sets up nor removes the file associations.

2.18 Does R support automation (OLE, COM)?

Directly, no. See packages such as `RDCOMClient` from http://www.omegahat.org/ and the non-Free project at http://www.statconn.com/.

2.19 The Internet download functions fail.

for example `update.packages()` and the menu items on the Packages menu.

We have had several reports of this, although they do work for us on *all* of our machines. There are two known possible causes.

(a) A proxy needs to be set up: see `?download.file`.

(b) Firewall settings are blocking the R executables from contacting the Internet (but this should result in informative error messages from the firewall program).

2.20 Entering certain characters crashes Rgui.

This has not been reported for a few years, but used to happen regularly. All the occurrences we have solved have been traced to faulty versions of '`msvcrt.dll`': we have installed a

workaround that seems to avoid this. A few other people have discovered this was caused by desktop switcher and keyboard macro programs, for example 'Macro Magic' and 'JS Pager'.

2.21 What does 'DLL attempted to change FPU control word' mean?

This is a *warning* which indicates that R has taken action to correct the action of some (non-R) DLL which has just been loaded and has changed the floating point control word (in its initialization code) to a setting incompatible with that needed for R. This is not good practice on the part of the DLL, and often indicates that it needs to be updated.

Unfortunately, because DLLs may themselves load other DLLs it is not possible for R to track which DLL caused the problem.

See also `?dyn.load`.

2.22 Other strange crashes.

Some users have found that `Rgui.exe` fails to start, exiting with a "Floating-point invalid operation" or other low level error. This error may also happen in the middle of a session. In some cases where we have tracked this down, it was due to bugs in the video driver on the system in question: it makes changes to the floating point control word which are incompatible with R. (Good practice would restore the control word to the state it was in when the driver code was called, and R tries hard to correct this before running its own code.) For example, one user reported that the virtual screen manager JSP2 caused this crash.

These errors are essentially impossible for us to fix or work around beyond the measures already taken. The only solution we know of is for the user to replace the buggy system component that is causing the error.

2.23 Why does R never use more than 50% of my CPU?

This is a misreading of Windows' confusing Task Manager. R's computation is single-threaded, and so it cannot use more than one CPU. What the task manager shows is not the usage in CPUs but the usage as a percentage of the apparent total number of CPUs. We say 'apparent' as it treats so-called 'hyper-threaded' CPUs such as two CPUs per core, and most modern CPUs have at least two cores.

You can see how many 'CPU's are assumed by looking at the number of graphs of 'CPU Usage History' on the 'Performance' tab of the Windows Task manager.

2.24 Does R run under Windows Vista/7/8/Server 2008?

It does. A few issues have been reported that are related to the way accounts and file permissions work. (These are not specifically R issues, but changes in user experiences.)

Earlier versions of Windows had user and Administrator accounts, and user accounts could be give administrative privileges (by being added to the local Administrators group) and so write permission over system areas such as `c:\Program Files`. R would be installed either by a user in his own file space or by an account with administrator privileges into

a system area. Sysadmins could set policies for user accounts, and you might for example have needed to be a 'Power User' to install software at all.

Vista and later normally disable the Administrator account and expect software installation to be done by an account which is in the local Administrator group with 'admin approval mode' turned on. (The Administrator account by default has it turned off.) Unlike (say) Windows XP, such accounts do not run programs with full administrator privileges, and this is where the issues arise. These OSes have the concept of 'over-the-shoulder' credentials: if you are running without full administrator privileges and do something which needs them you may be prompted with one or more security-check dialog boxes, and may be required to provide administrator credentials or confirm that you really want to take that action.

Vista and later will report that the R installer has an 'unidentified publisher' or 'unknown publisher' and ask if it should be run. System administrators can disable installing applications from non-trusted sources, in which case you will have to persuade them that R is trustworthy, or digitally sign the R installer yourself, or (unless this is also disabled) run the installer from a standard account and install into your own file area. (The same issues apply to the `.msi` version of the installer.)

If you install R as a standard user into your own file space and use it under the same account, there are no known permission issues.

If you use the default Administrator account (without 'admin approval mode' being turned on) and install/update packages (in the system area or elsewhere), no issues are known.

If you use an account in the local Administrators group in 'admin approval mode' (which is the intended norm under these OSes), installation will make use of 'over-the-shoulder' credentials. You will run into problems if you try installing (including updating) packages in the main R library. (It would be nice if at that point R could use over-the-shoulder credentials, but they apply to processes as a whole. Vista and later disallow creating `.dll` files in the system area without credentials.) There are several ways around this.

- Run R with Administrator privileges in sessions where you want to install packages. (Do so by right-clicking on the R shortcut and selecting 'Run as Administrator'.)
- Transfer ownership of the R installation to the user which installed R. To do so, use the security tab on the '`Properties`' of the top-level R folder and give 'Full Control' over this directory to the user (not just the Administrator group).
- Install packages into a different library tree owned by the account used to install R.

  For an installation to be used by a single user, the simplest way is to make use of a 'personal library': See Section 4.2 [I don't have permission to write to the 3.3.0 \library directory], page 15.

  For a site installation, you can create a site-wide library directory anywhere convenient, and add it to the default package search path for all users via `R_LIBS_SITE` in `etc\Renviron.site`. See Section 2.14 [What are HOME and working directories?], page 5. There is a standard location for a site library, the `site-library` directory in the top-level R folder (which you would need to create with full control for the R installation account). This will be used for installation in preference to the main library folder if it exists.

  This approach will not allow you to update the recommended packages unless you 'Run as administrator': we suggest you use an R session running under Administrator privileges when updating those.

Another issue with Vista was that the standard POSIX ways that R uses (e.g. in `file.info` and `file.access`) to look at file permissions no longer work reliably. `file.access` was re-written to work with Windows NT-based security and the new version seems much more reliable with these OSes (but still not 100% correct).

On suitably recent hardware Vista and later can prevent the execution of code from data areas via 'Data Execution Prevention' (from a tab in System Properties -> Advanced -> Performance), and sysadmins can turn this on for all programs. R runs correctly with DEP enabled.

2.25 Quotes don't come out right on the console/terminal/pager.

R may make use of directional quotes that are not always rendered correctly by Windows: these are used by default only by `Rgui` in suitable locales (not Chinese/Japanese/Korean).

Whether these are used in R output (from functions `sQuote` and `dQuote`) is controlled by `getOption("useFancyQuotes")` whose default is `FALSE` except for the `Rgui` console. There are two potential problems with rendering directional quotes. The first is with running `Rterm`: in European locales the 'Windows Command Prompt' is by default set up to use MS-DOS and not Windows default encodings: this can be changed via `chcp`, with `chcp 1252` being appropriate for Western European (including English) locales. The other is that the default raster fonts only include directional single quotes and not directional double quotes (which will probably be rendered as a filled rectangle).

Directional quotes will also be used in text help which is normally displayed in R's internal pager: these may not be rendered correctly in an external pager. They are also used in HTML help, where most browsers use fonts which render them correctly.

The font used can affect whether quotes are rendered correctly. The default font in the `Rgui` console and internal pager is `Courier New`, which has directional quotes on all the systems we tried. `Lucida Console` which has elegant glyphs for directional quotes (but seems rather light unless ClearType is in use): `Consolas` is another font which we often select when ClearType is in use. Non-TrueType fonts such as `Courier` and `FixedSys` lack directional double quotes on the systems we tried.

There is a related problem with using `Sweave` output in `Rgui`, for LaTeX needs to be told about the encoding of directional quotes by including in the LaTeX preamble e.g. (for a Western European locale)

```
\usepackage[cp1252]{inputenc}
```

or their use suppressed by `options(useFancyQuotes=FALSE)`.

2.26 There is no tilde on my keyboard!

Where tilde does not appear on the main keyboard, it can normally be accessed by pressing AltGr (the right Alt key) plus some other key. This is `]` in Canadian (multilingual), German and Scandinavian layouts, `1` in Eastern Europe, `[` in Portuguese, `4` or `5` in Spanish, `/` in Francophone Belgian, and so on. You can explore those for your keyboard via the 'On-Screen Keyboard' (under Ease of access on Windows 7).

On all Windows versions you should be able to get tilde by holding the down the left Alt key and typing 0126 on the numeric keypad (if you have one), then releasing the Alt key.

2.27 Can I use R on 64-bit Windows?

The 32-bit build of R for Windows will run on both 32-bit and 64-bit[2] versions of Windows. 64-bit versions of Windows run 32-bit executables under the WOW (Windows on Windows) subsystem: they run in almost exactly the same way as on a 32-bit version of Windows, except that the address limit for the R process is 4GB (rather than 2GB or perhaps 3GB).

When R is installed on 64-bit Windows there is the option of installing 32- and/or 64-bit builds: the default is to install both. If you are using the 32-bit build, replace '`x64`' by '`i386`' in the examples in this FAQ.

2.28 Should I run 32-bit or 64-bit R?

Obviously, only relevant if you are using 64-bit Windows.

For most users we would recommend using the 'native' build, that is the 32-bit version on 32-bit Windows and the 64-bit version of 64-bit Windows.

The advantage of a native 64-bit application is that it gets a 64-bit address space and hence can address far more than 4GB (how much depends on the version of Windows, but in principle 8TB). This allows a single process to take advantage of more than 4GB of RAM (if available) and for R's memory manager to more easily handle large objects (in particular those of 1GB or more). The disadvantages are that all the pointers are 8 rather than 4 bytes and so small objects are larger and more data has to be moved around, and that less external software is available for 64-bit versions of the OS. The 64-bit compilers are able to take advantage of extra features of all x86-64 chips (more registers, SSE2/3 instructions, ...) and so the code may run faster despite using larger pointers. The 64-bit build is nowadays usually slightly faster than the 32-bit build on a recent CPU (Intel Core 2 or later or AMD equivalent).

For advanced users the choice may be dictated by whether the contributed packages needed are available in 64-bit builds (although CRAN only offers 32/64-bit builds). The considerations can be more complex: for example 32/64-bit `RODBC` need 32/64-bit ODBC drivers respectively, and where both exist they may not be able to be installed together. An extreme example is the Microsoft Access/Excel ODBC drivers: if you have installed 64-bit Microsoft Office you can only install the 64-bit drivers and so need to use 64-bit `RODBC` and hence R. (And similarly for 32-bit Microsoft Office.)

2.29 Can both 32- and 64-bit R be installed on the same machine?

Obviously, only relevant if the machine is running a 64-bit version of Windows – simply select both when using the installer. You can also go back and add 64-bit components to a 32-bit install, or *vice versa*.

For many Registry items, 32- and 64-bit programs have different views of the Registry, but clashes can occur. The most obvious problem is the file association for `.RData` files, which will use the last installation for which this option is selected, and if that was for an installation of both, will use 64-bit R. To change the association the safest way is to edit the Registry entry '`HKEY_CLASSES_ROOT\RWorkspace\shell\open\command`' and replace '`x64`' by '`i386`' or *vice versa*.

[2] what Windows calls x64 for x86-64 CPUs, not the very rare ia64 Windows for Itanium CPUs.

2.30 Rcmd is not found in my PATH!

This has often been reported after an upgrade.

The R installer does not put `Rcmd.exe` (nor any other R executable) on your `PATH`. What seems to have happened is that people did this for themselves in the past, upgraded R (which by default will install to a different location) and un-installed the old version of R. If you do that (or install R for the first time), you need to edit the `PATH`.

The element you want to add to the path is something like

```
c:\Program Files\R\R-3.1.0\bin\x64
```

for 64-bit `Rcmd.exe`, replacing `x64` by `i386` for 32-bit.

How you set the path depends on your OS version. Under recent versions, go to 'User Accounts' in the Control Panel, and select your account and then 'Change my environment variables'. (System policies can prevent end users making changes.)

An alternative is to set the `PATH` in the shell you are running (`Rcmd.exe` is a command-line program). For those using the standard Windows 'Command Prompt' Duncan Murdoch suggested:

The simple way to do it just for the command prompt is to write a little batch file `setpath.bat` containing

```
set PATH=newstuff;%PATH%
```

and then run `cmd` with

```
CMD /K setpath.bat
```

3 Languages and Internationalization

3.1 The installer does not offer my language.

Only a limited range of languages is supported, currently Catalan, both Simplified and Traditional Chinese, Czech, Danish, Dutch, Finnish, French, German, Greek, Hebrew, Hungarian, Italian, Japanese, Korean, Norwegian, Polish, Portuguese (Brazil), Portuguese (Portugal), Russian, Slovenian, Spanish (Spain) and Ukranian.

3.2 I want R in English (and not in French/Chinese/...)!

The default behaviour of R is to try to run in the language you run Windows in.

Apparently some users want[1] Windows in their native language, but not R. To do so, set `LANGUAGE=en` as discussed in Q2.2 and Q2.15, or in the `Rconsole` file.

3.3 I want to run R in Chinese/Japanese/Korean.

Suitable versions of Windows support what it calls 'East Asian' languages, but e.g. Western installations of Windows often do not have such support. So we need to assume that your copy of Windows does.

Both `Rterm.exe` and `Rgui.exe` support single- and double-width characters. It will be necessary to select suitable fonts in files `Rconsole` and `Rdevga` (see `?Rconsole` or the comments in the files: the system versions are in the `etc` folder); in the latter you can replace `Arial` by `Arial Unicode MS`, and we tried `FixedSys` and `MS Mincho` in `Rconsole`. (Note that `Rdevga` only applies to Windows graphics devices and not, say, to `pdf`.)

Note that it is important that the console font uses double-width characters for all CJK characters (as that is what the width table used assumes): this is true for the fonts intended for CJK locales but not for example for `Lucida Console` or `Consolas`.

You do need to ensure that R is running in a suitable locale: use `Sys.getlocale()` to find out. (CJK users may be used to their language characters always being available, which is the case for so-called 'Unicode' Windows applications. However, R is primarily written for Unix-alikes and is not therefore 'Unicode' in the Windows sense.) You can find suitable locale names from https://msdn.microsoft.com/en-us/library/39cwe7zf%28v=vs.80%29.aspx and https://msdn.microsoft.com/en-us/library/cdax410z%28v=vs.80%29.aspx beware that `"Chinese"` is Traditional Chinese (code page 950, Big5) and `"chs"` is needed for Simplified Chinese (code page 936, GB2312).

When using `Rterm` the window in which it is run has to be set up to use a suitable font (e.g. `Lucida Console` or `Consolas`, not the OEM raster fonts) and a suitable codepage (which for the Windows Cmd shell can be done using `chcp`).

3.4 I selected English for installation but R runs in Chinese.

Precisely, you selected English **for installation**! The language of the installer has nothing to do with the language used to run R: this is completely standard Windows practice (and necessary as different users of the computer may use different languages).

[1] or they may have no choice: apparently some Windows editions are tied to a specific language.

The language R uses for menus and messages is determined by the *locale*: please read the appropriate manual ('R Installation and Administration') for the details. You can ensure that R uses English messages by appending `LANGUAGE=en` to the shortcut you use to start R, or setting it in the `Rconsole` file.

3.5 I would like to be able to use Japanese fonts.

for example, in the console and to annotate graphs. Similar comments apply to any non-Western-European language.

With suitable fonts, this should just work. You will need to set MS Mincho or MS Gothic as the console font to ensure that single- and double-width characters are handled correctly. The default graphics fonts for the `windows()` graphics device can handle most common Japanese characters, but more specialized fonts may need to be set. (See Q5.2 for how to set fonts: the console font can also be set from the 'GUI preferences' menu item.) The help for `windowsFonts` has examples of selecting Japanese fonts for the `windows()` family of devices.

In addition, the Hershey vector fonts (see `?Hershey`, `?Japanese` and `demo(Japanese)`) can be used on any graphics device to display Japanese characters.

To use non-Latin-1 characters in the `postscript` graphics device, see its help page (which also applies to `pdf`).

3.6 I don't see characters with accents at the R console, for example in ?text.

You need to specify a font in `Rconsole` (see Q5.2) that supports the encoding in use. This used to be a problem in earlier versions of Windows, but now it is hard to find a font which does not.

Support for these characters within `Rterm` depends on the environment (the terminal window and shell, including locale and codepage settings) within which it is run as well as the font used by the terminal window. Those are usually on legacy DOS settings and need to altered.

3.7 The dialog buttons are not translated.

In most cases they actually are, but by Windows. Setting the locale or the `LANGUAGE` environment variable does not change the Windows setting of its 'UI language'. Vista and later talk about the 'UI language' and the 'system locale' for setting the language used for 'non-Unicode' programs (on the 'Administrative' tab in Windows 7).

If you have Windows running completely in say French or Chinese these settings are likely to be consistent. However, if you try to run Windows in one language and R in another, you may find the way Windows handles internationalization slightly odd.

4 Packages

4.1 Can I install packages into libraries in this version?

Yes, but you will need a lot of tools to do so, unless the author or the maintainers of the `bin/windows/contrib` section on CRAN have been kind enough to provide a binary version for Windows as a `.zip` file, or the package is a simple one involving no compiled code (and binary versions are usually available for simple packages).

You can install binary packages either from a repository such as CRAN or from a local `.zip` file by using `install.packages`: see its help page. There are menu items on the `Packages` menu to provide a point-and-click interface to package installation. The packages for each minor (3.x.?) version will be stored in a separate area, so for R 3.1.? the files are in `bin/windows/contrib/3.1`.

Note that the binary versions on CRAN are unsupported: see `https://cran.r-project.org/bin/windows/contrib/3.1/ReadMe`, which also gives the locations of a few other binary packages.

If there is no binary package or that is not up-to-date or you prefer compiling from source, read the 'R Installation and Administration' manual section on 'Add-on Packages'. Source packages which contain no C/C++/Fortran code which needs compilation can simply be installed by `install.packages(type = "source")` or `R CMD INSTALL pkgname` at a Windows command prompt. For packages with code that needs compilation you will need to collect and install several tools: you can download them via the portal at `http://www.murdoch-sutherland.com/Rtools/`. Once you have done so, just run `R CMD INSTALL pkgname` at a Windows command prompt. To check the package (including running all the examples on its help pages and in its test suite, if any) use `R CMD check pkgname`: see the *'Writing R Extensions'* manual.

Note that setting up Windows to install a source package that needs compilation is rather tricky; please do ensure that you have followed the instructions **exactly**. At least 90% of the questions asked are because people have not done so.

If you have a source package that is known to work on a Unix-alike system, you can try the automated Windows binary package builder documented at `http://win-builder.r-project.org`.

4.2 I don't have permission to write to the `3.3.0 \library` directory.

You can install packages anywhere and use the environment variable `R_LIBS` (see Section 2.15 [How do I set environment variables?], page 6) to point to the library location(s).

Suppose your packages are installed in `p:\myRlib`. Then you can EITHER

```
set the environment variable R_LIBS to p:/myRlib before starting R
```

OR use a package by, e.g.

```
library(mypkg, lib.loc="p:/myRlib")
```

You can also have a personal library, which defaults to the directory `R\win-library\x.y` of your home directory for versions *x.y.z* of R. This location can be changed by setting the environment variable `R_LIBS_USER`, and can be found from inside R by running `Sys.getenv("R_LIBS_USER")`. This will only be used if it exists so you may need to create it: you can use

```
dir.create(Sys.getenv("R_LIBS_USER"), recursive = TRUE)
```

to do so. If you use `install.packages` and do not have permission to write to the main or site library, it should offer to create a personal library for you and install the packages there. This will also happen if `update.packages` offers to update packages for you in a library where you do not have write permission.

There can be additional security issues under Windows Vista and later: See Section 2.24 [Does R run under Windows Vista?], page 8. In particular, the detection that a standard user has suitable permissions appears to be unreliable under Vista, so we recommend that you do create a personal directory yourself.

4.3 The packages I installed do not appear in the HTML help system.

This question applied to the pre-2.10.0 HTML help system, which has been replaced.

4.4 My functions are not found by the HTML help search system.

This question applied to the pre-2.10.0 search system, which has been replaced.

4.5 Loading a package fails.

Is the package installed for this version of R? Packages need to have prepared for R 2.10.0 or later, and packages containing compiled code for R 2.12.0 or later.

You can tell the version the package was compiled for by looking at the '`Built:`' line in its `DESCRIPTION` file.

For a small number of binary packages you need to install additional software and have its DLLs in your `PATH`. Windows will normally give an informative message about a certain DLL not being found. See `https://cran.r-project.org/bin/windows/contrib/3.1/ReadMe` for a listing of some of these packages (notably `RGtk2`, `cairoDevice`, `rggobi`, `rJava`, `rjags` and some of the packages connecting to databases).

4.6 Package TclTk does not work.

For package `tcltk` to work (try `demo(tkdensity)` or `demo(tkttest)` after `library(tcltk)`) you need to have Tcl/Tk installed. This part of the R installation, so it should be there.

However, if you have the environment variable `MY_TCLTK` set to a non-empty value, it is assumed that you want to use a different Tcl/Tk 8.5.x installation with the path to its `bin` directory given by value of `MY_TCLTK`, and that this is set up correctly (with `TCL_LIBRARY` set if needed). Note that you do need 8.5.x and not 8.4.x nor 8.6.0, and you do need the architecture to match, that is a 32-bit or 64-bit build of Tcl/Tk to match the R build in use. (There is no guarantee that a 64-bit build will work: it depends on the layout it uses.)

In the past several package authors have suggested using ActiveTcl (https://www.activestate.com/Products/activetcl/) as a way to get Tcl/Tk extensions (but the support files do contain the most commonly used `TkTable` and `BWidget` extensions). This could be used by setting (for a default install)

```
MY_TCLTK=c:/Tcl/bin
```

but current versions do not by default contain any extra extensions (although they may be downloaded via the `Teacup` facility) and this only works for 32-bit R.

4.7 Hyperlinks in HTML sometimes do not work.

This question was much more relevant prior to version 2.10.0.

They may still not work between packages installed in different libraries if the HTTP server has been disabled: the remedy is not to do that!

4.8 `update.packages()` fails.

You may not be able to update a package which is in use: Windows 'locks' the package's DLL when it is loaded. So use `update.packages()` (or the menu equivalent) in a new session.

If you put `library(foo)` in your `.Rprofile` you will need to start R with `--vanilla` to be able to update package `foo`. If you set `R_DEFAULT_PACKAGES` to include `foo`, you will need to unset it temporarily.

It has been reported that some other software has interfered with the installation process by preventing the renaming of temporary files, *Google Desktop* being a known example.

4.9 How do I add to the list of repositories?

as shown in the `Select repositories...` item on the `Packages` menu?

This reads from the tab-delimited file `R_HOME\etc\repositories`, which you can edit, or put a modified copy at `.R\repositories` in your HOME directory (see Section 2.14 [What are HOME and working directories?], page 5).

4.10 Help is not shown for some of the packages

This was about Compiled HTML help, which has not been supported since R 2.10.0.

4.11 How do I get static HTML pages?

We presume you want to do this for some special purpose: R's help system will not make use of them, links across library directories will not work (unlike R < 2.10.0), ambiguous links will be resolved at install time and missing links will be broken (previous versions used JavaScript to look for them at run time). But if you still want them, here is how to do it.

Static HTML pages are not part of the binary distribution, so you will need to install R and/or packages from their sources. To install just a few packages with static HTML pages use

```
R CMD INSTALL --html pkg1 pkg2 ...
```

To install R itself with static HTML pages you need to build it from the sources for yourself. Change the following line in file `MkRules.local` (after copying `MkRules.dist` to `MkRules.local` if that has not already been done).

```
# set this to YES to build static HTML help
BUILD_HTML = NO
```

and them all packages installed by that build of R will (by default) be installed with static HTML pages.

4.12 How can I get a binary version of a package?

Presumably one not available on CRAN, BioC or a similar repository.

If you have a source package that is known to work on a Unix-alike system, you can try the automated Windows binary package builder documented at http://win-builder.r-project.org. If the package is not yours, please remember to change the maintainer address so the results go to you and not the author(s)!

However, if a CRAN package is not available in binary form, this usually means that there is a problem with some dependent package or external software (often mentioned in the `@ReadMe` file in the binary repository directory). You can email R-windows@r-project.org expressing a wish for such a package to be ported—the maintainers will take such wishes into account when prioritizing work on binary packages.

In many cases installing packages from the sources is not at all difficult (it is simple if the package contains no compiled code), so please attempt that for yourself before requesting help from the busy volunteers.

4.13 Package xxx is out of date for Windows

Here are three possible reasons:

You are simply impatient, and need to wait until the binary package has been built and propagated to the CRAN mirror you are using. This normally (but not always) happens within 24 hours. Sometimes mirrors do get behind (especially unofficial ones), so you could try another mirror. (Mirror statistics are linked near the top of the CRAN mirror page at https://cran.r-project.org/mirrors.html.)

The latest version of the package might require a later version of R than the one you are using. You can check on the package's HTML page on CRAN, and update your R if needed.

Your R might be too old. Binary packages for the 3.x series are built (if possible) whilst 3.(x+1) is current, but building stops once 3.(x+2) reaches alpha (pre-release, about a month before release). You can always try installing from the sources.

4.14 No binary packages appear to be available for my version of R

How old is it? The CRAN policy is to archive binary packages two years after the 2.x series is closed. Other repositories may do so sooner.

If you are using an R version that old we advise you to update your R, but you do also have the option of installing packages from their source.

4.15 How do I build my package for both 32- and 64-bit R?

Packages without compiled code nor a `configure.win` script will run on both 32- and 64-bit R.

Packages with compiled code but no `configure.win` nor `src/Makefile.win` file will be built for both when running on a 64-bit version of Windows if both versions of R are installed.

An empty `configure.win` is treated in the same way as if none existed. Also, there is a list of packages known to have an architecture-independent `configure.win` hardcoded into `R CMD INSTALL`, and for these packages, both architectures will be built under the above conditions. Other packages can be installed with `configure.win` run for just the first architecture by using option `--force-biarch`.

Any package can be installed for first one architecture and then the other with option `--merge-multiarch`, but the package source must be a tarball (and as before, running on a 64-bit version of Windows with both versions of R installed).

Finally, a package without a `src/Makefile.win` file and no or empty or architecture-independent `configure.win` file can be installed for both architectures from 32-bit Windows if the 64-bit components were selected when R was installed and option `--compile-both` is given. Obviously, only the 32-bit installation can be tested.

5 Windows Features

5.1 What should I expect to behave differently from the Unix version of R?

- R commands can be interrupted by `Esc` in `Rgui.exe` and `Ctrl-break` or `Ctrl-C` in `Rterm.exe`: `Ctrl-C` is used for copying in `Rgui.exe`.
- Command-line editing is always available, but is somewhat simpler than the readline-based editing on Unix. For `Rgui.exe`, the menu item 'Help | Console' will give details. For `Rterm.exe` see file `README.rterm`.
- Paths to files (e.g. in `source()`) can be specified with either `"/"` or `"\\"`.
- `system()` is slightly different: see its help page and that of `shell()`.

5.2 I hear about some nifty features: please tell me about them!

You have read the file `README.3.3.0` ? There are file menus on the R console, pager and graphics windows. You can source and save from those menus, and copy the graphics to `png`, `jpeg`, `bmp`, `postscript`, `PDF` or `metafile`. There are right-click menus giving shortcuts to menu items, and optionally toolbars with buttons giving shortcuts to frequent operations.

If you resize the R console the `options(width=)` is automatically set to the console width (unless disabled in the configuration file).

The graphics has a history mechanism. As `README.3.3.0` says:

> 'The History menu allows the recording of plots. When plots have been recorded they can be reviewed by `PgUp` and `PgDn`, saved and replaced. Recording can be turned on automatically (the Recording item on the list) or individual plots can be added (Add or the `INS` key). The whole plot history can be saved to or retrieved from an R variable in the global environment. The format of recorded plots may change between R versions. Recorded plots should **not** be used as a permanent storage format for R plots.
>
> There is only one graphics history shared by all the windows devices.'

The R console and graphics windows have configuration files stored in the `RHOME\etc` directory called `Rconsole` and `Rdevga`; you can keep personal copies in your `HOME` directory. They contain comments which should suffice for you to edit them to your preferences. For more details see `?Rconsole`. There is a GUI preferences editor invoked from the `Edit` menu which can be used to edit the file `Rconsole`.

5.3 Circles appear as ovals on screen.

The graphics system asks Windows for the number of pixels per inch in the X and Y directions, and uses that to size graphics (which in R are in units of inches). Sometimes the answer is a complete invention, and in any case Windows will not know exactly how the horizontal and vertical size have been set on a monitor which allows them to be adjusted. You can specify correct values either in the call to `windows` or as options: see `?windows`. (Typically these are of the order of 100.)

On one of our systems, the screen height was reported as 240mm, and the width as 300mm in 1280 x 1024 mode and 320mm in 1280 x 960 and 1600 x 1200 modes. In fact it was a 21" monitor and 400mm x 300mm!

This is less common with LCD screens but not unknown, particularly if they are not running at their native resolution.

5.4 How do I move focus to a graphics window or the console?

You may want to do this from within a function, for example when calling '`identify`' or '`readline`'. Use the function '`bringToTop()`'. With its default argument it brings the active graphics window to the top and gives it focus. With argument '`-1`' it brings the console to the top and gives it focus.

This works for `Rgui.exe` in MDI and SDI modes, and can be used for graphics windows from `Rterm.exe` (although Windows may not always act on it).

5.5 What does TAB completion do?

Both `Rgui` and `Rterm` support *`TAB`* completion. Hitting *`TAB`* whilst entering a command line completes the current 'word' as far as is unambiguously possible. Hitting *`TAB`* a second time then shows a list of possible completions (or the first few if there are many): the user can then enter one or more characters and hit *`TAB`* again.

What is it 'completing'? There are two modes: within an unterminated (single- or double-) quoted expression it completes file paths.[1] Otherwise, it is completing R expressions: most obviously it will match visible R object names and keywords, so *`apr`* followed by *`TAB`* will (in a vanilla session) complete to `apropos`. After a function name and parenthesis (e.g. *`apropos(`*) it will complete argument names (and `=`), and after *`$`* or *`@`* it will complete list components or slot names respectively.

This feature can be turned off: `Rgui` has two menu items to do so, and setting the environment variable `R_COMPLETION` to `FALSE` turns it off completely for both `Rgui` and `Rterm`. Further, the behaviour can be fine-tuned: to see the settings available use

```
?rc.settings
```

which also explains how the various types of completion work.

This feature is very similar to the completion available in the `readline`-based command line interface on Unix-alikes: the OS X GUI `R.app` has a different completion scheme.

[1] It does not have a complete understanding of Windows file paths, but can complete most relative or absolute file paths, including drives and spaces. Relative paths on drives are not handled, for example.

6 Workspaces

6.1 My workspace gets saved in a strange place: how do I stop this?

Have you changed the working directory?: see Q6.2.

6.2 How do I store my workspace in a different place?

Use the 'File | Change Dir...' menu item to select a new working directory: this defaults to the last directory you loaded a file from. The workspace is saved in the working directory. You can also save a snapshot of the workspace from the 'Save Workspace...' menu item.

From the command line you can change the working directory by the function `setwd`: see its help page.

6.3 Can I load workspaces saved under Unix/GNU-Linux or Mac OS X?

Yes. All ports of R use the same format for saved workspaces, so they are interchangeable (for the same 3.x.? version of R, at least).

Note though that character data in a workspace will be in a particular encoding that is not recorded in the workspace, so workspaces containing non-ASCII character data may not be interchangeable even on the same OS. Since R marks character data when it knows it to be in UTF-8 or Latin-1 (including its Windows superset, CP1252), strings in those encodings are likely to be transferred correctly: fortunately this covers most of the common cases (OS X normally uses UTF-8, and Linux users are likely to use UTF-8 or perhaps Latin-1).

It is possible to save references to package namespaces when saving the workspace: if that happens the package will need to be installed on the machine loading the workspace.

7 The R Console

7.1 When using Rgui the output to the console seems to be delayed.

This is deliberate: the console output is buffered and re-written in chunks to be faster and less distracting. You can turn buffering off or on from the 'Misc' menu or the right-click menu: `Ctrl-W` toggles the setting.

If you are sourcing R code or writing from a function, there is another option. A call to the R function `flush.console()` will write out the buffer and so update the console.

7.2 Long lines in the console or pager are truncated.

They only **seem** to be truncated: that $ at the end indicates you can scroll the window to see the rest of the line. Use the horizontal scrollbar or the `CTRL + left/right arrow` keys to scroll horizontally. (The `left/right arrow` keys work in the pager too.)

8 Building from Source

8.1 How can I compile R from source?

See the 'R Installation and Administration' manual (for the version of R you want to install).

8.2 Can I use a fast BLAS?

Fast BLAS (Basic Linear Algebra Subprograms, http://www.netlib.org/blas/faq.html) routines are used to speed up numerical linear algebra. There is support in the R sources for the 'tuned' BLAS called ATLAS (http://math-atlas.sourceforge.net). The savings can be appreciable but because ATLAS is tuned to a particular chip we can't use it generally. However, linear algebra on large matrices is not often an important part of R computations, and more typical calculations on small matrices may run slower.

BLAS support is supplied by the single DLL `R_HOME\bin\x64\Rblas.dll`, and you can add a fast BLAS just by replacing that. Replacements for 32-bit R and some of the older common chips are available on CRAN in directory `bin/windows/contrib/ATLAS`. See the R Installation and Administration' manual for how to build an ATLAS `Rblas.dll` tuned to your system using the R sources. Unfortunately the process has been less successful when tried for the common current chips such as Intel's Core 2.

Versions of Dr Kazushige Goto's BLAS (see https://en.wikipedia.org/wiki/Kazushige_Goto) for 64-bit Windows by Ei-Ji Nakama can be found at http://prs.ism.ac.jp/~nakama/SurviveGotoBLAS2/binary/windows/x64/. Just download the file `Rblas.dll` appropriate to your CPU and replace *`R_HOME`*`/bin/x64/Rblas.dll`. (There is also a generic version called '`DYNAMIC_ARCH`' that tries to adapt itself to the CPU found – however if you know the exact CPU used it is better to download the CPU-specific version. Note that development of that BLAS was frozen in 2010 so you will not find versions for recent CPUs.)

Note that fast BLAS implementations may give different (and often slightly less accurate) results than the reference BLAS included in R.

8.3 How do I include compiled C code?

We strongly encourage you to do this *via* building an R package: see the *'Writing R Extensions'* manual. In any event you should get and install the tools and toolchain mentioned in the 'R Installation and Administration' manual. Then you can use

```
...\bin\x64\R CMD SHLIB foo.c bar.f
```

to make `foo.dll`. Use `...\bin\x64\R CMD SHLIB --help` for further options, or see `?SHLIB`. (Replace `x64` by `i386` for 64-bit R.)

If you want to use Visual C++, Borland C++ or other compilers, see the appropriate section in `README.packages`.

8.4 How do I debug code that I have compiled and dyn.load-ed?

Debugging under Windows is often a fraught process, and sometimes does not work at all. If all you need is a *just-in-time* debugger to catch crashes, consider (32-bit) `Dr. Mingw` from

the `mingw-utils` bundle on http://www.mingw.org. That will be able to pinpoint the error, most effectively if you build a version of R with debugging information as described below.

First, build a version of the R system with debugging information by

```
make clean
make DEBUG=T
```

and make a debug version of your package by

```
Rcmd INSTALL --debug mypkg
```

You will need a suitable version of `gdb` which matches your compiler. Then you can debug by

```
gdb /path/to/3.3.0 /bin/x64/Rgui.exe
```

(or use `Rterm.exe`.) However, note

- `gdb` may only be able to find the source code if we run in the location where the source was compiled (`3.3.0 /src/gnuwin32` for the main system, `3.3.0 /src/library/mypkg/src` for a package), unless told otherwise by the `directory` command. It is most convenient to set a list of code locations via `directory` commands in the file `.gdbinit` in the directory from which `gdb` is run.
- It is only possible to set breakpoints in a DLL after the DLL has been loaded. So a way to examine the function `tukeyline` in package `stats` might be

  ```
  gdb ../../../../bin/i386/Rgui.exe
  (gdb) break WinMain
  (gdb) run
  [ stops with R.dll loaded ]
  (gdb) break R_ReadConsole
  (gdb) continue
  [ stops with console running ]
  (gdb) continue
  Rconsole> library(stats)
  (gdb) break tukeyline
  (gdb) clear R_ReadConsole
  (gdb) continue
  Rconsole> example(line)
  ...
  ```

  Alternatively, in `Rgui` you can use the 'Misc|Break to debugger' menu item after your DLL is loaded. The C function call `breaktodebugger()` will do the same thing.
- Fortran symbols need an underline appended.
- Windows has little support for signals, so the Unix idea of running a program under a debugger and sending it a signal to interrupt it and drop control back to the debugger does not work with a MinGW version of `gdb`. It does often work with the `cygwin` version.

See http://www.stats.uwo.ca/faculty/murdoch/software/debuggingR/gdb.shtml for some further details.

8.5 How do I include C++ code?

You need to do two things:

(a) Write a wrapper to export the symbols you want to call from R as `extern "C"`.

(b) Include the C++ libraries in the link to make the DLL. Suppose `X.cc` contains your C++ code, and `X_main.cc` is the wrapper, as in the example in *'Writing R Extensions'*. Then build the DLL by (`gcc`)

```
...\bin\x64\R CMD SHLIB X.cc X_main.cc
```

or (VC++, which requires extension `.cpp`)

```
cl /MT /c X.cpp X_main.cpp
link /dll /out:X.dll /export:X_main X.obj X_main.obj
```

and call the entry point(s) in `X_R`, such as `X_main`. Construction of static variables will occur when the DLL is loaded, and destruction when the DLL is unloaded, usually when R terminates.

Note that you will not see the messages from this example in the GUI console: see the next section.

8.6 The output from my C code disappears. Why?

The `Rgui.exe` console is a Windows application: writing to `stdout` or `stderr` will not produce output in the console. (This will work with `Rterm.exe`.) Use `Rprintf` or `REprintf` instead. These are declared in header file `R_ext/PrtUtil.h`.

Note that output from the console is delayed (see Section 7.1 [The output to the console seems to be delayed], page 23), so that you will not normally see any output before returning to the R prompt.

8.7 The output from my Fortran code disappears. Why?

Writing to Fortran output writes to a file, not the `Rgui` console. Use one of the subroutines `dblepr`, `intpr` or `realpr` documented in the *'Writing R Extensions'* manual.

Note that output from the console is delayed (see Section 7.1 [The output to the console seems to be delayed], page 23), so that you will not normally see any output before returning to the R prompt even when using the `xxxpr` subroutines.

8.8 The console freezes when my compiled code is running.

The console, pagers and graphics window all run in the same thread as the R engine. To allow the console etc to respond to Windows events, call `R_ProcessEvents()` periodically from your compiled code. If you want output to be updated on the console, call `R_FlushConsole()` and then `R_ProcessEvents()`.