Building R for 64-bit Windows

Supplementary Notes
John W. Emerson and Michael J. Kane
Yale University
April 2, 2010

I provide this as a supplement to Section 3.3 of the R Administration and Installation guide development version. I'll do this simply by editing the 4/2/2010 version of that section with highlighting to mark my additions. I used Windows 7 Enterprise for this exercise, with:

- **R-latest.tar.gz (version 2.11.0 downloaded on 4/2/2010)**
- **basic-miktex-2.8.3582.exe (see Appendix D.2)**
- **mingw-w64-1.0-bin_i686-mingw_20100322.zip (see below)**
- **R64_Tcl_8-5-8.zip (see below)**
- **Rtools211.exe (see Appendix D)**

I've found that forgetting to create `c:\tmp` was the most common "gotcha" for me, both here and when building the traditional 32-bit Windows R. After finishing the build, I successfully created a 5 GB matrix (I have 8 GB of RAM), something that would not be possible in 32-bit R.

3.3 Building R for 64-bit Windows

Overview

This section is a supplement to earlier sections of **Installing R under Windows**; there is some duplication of material, but readers should start with Sections 3.1 and 3.2. In a nutshell, building R for 64-bit Windows requires a 64-bit toolchain that is a modification of the 32-bit toolchain, described in the earlier sections.

Note that the standard 32-bit build of R runs perfectly well on 64-bit versions of Windows, with a limit on the total memory allocation (and address space) of 4GB. Thus a native 64-bit build will be attractive to those wanting to work with large R objects, or many medium-sized ones.

MinGW-w64

To build a 64-bit version of R you need a 64-bit toolchain: the only one discussed here is based on the work of the MinGW-w64 project (http://sourceforge.net/projects/mingw-w64/, a port of GNU `binutils` and `gcc`), but commercial compilers such as those from Intel and PGI could be used (and have been by R redistributors).

Support for MinGW-w64 was developed in the R sources over the period 2008–10 and was first released as part of R 2.11.0. The binary version available from CRAN was built with the MinGW-w64 toolchain described in this section, using static linking.

Several versions of the MinGW-w64 toolchain are available: use one of them to replace the `MinGW/bin` directory of `Rtools` in your path (but keep the Rtools `bin` and Perl `bin` directories in your path). So, for example, the relevant part of your path may be something like: `c:\Rtools\bin;c:\Rtools\perl\bin;c:\MinGW-x64\bin`. Then in the R source tree, copy `src/gnuwin32/MkRules.dist` to `src/gnuwin32/MkRules.local` and edit it to set `WIN=64` and `BINPREF64` appropriate to your toolchain. Then R is built in the usual way (see below). If you are using the recommended MinGW-w64 toolchain (see next paragraph), it's likely that no change will be needed to `BINPREF64`. To check this, examine the prefix used inside the `bin` directory of your MinGW-w64 directory (it will be something like `x86_64-w64-mingw32-`, and needs to match the prefix specified by `BINPREF64`).

The toolchain we use is technically a cross-compiler: the tools run under 32-bit Windows but produce code to run under 64-bit Windows.[6] This comes from http://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win64/Automated%20Builds/ and has a name like `mingw-w64-1.0-bin_i686-mingw_20100322.zip`. (Care is needed: that site also has compilers targeting 32-bit Windows and running on other host OSes. You do want the `release' version with `-1.0-` in the name. There was a native x64 toolchain under `Personal builds'.) Some versions of that toolchain by default use dynamic linking to the compiler runtimes, which causes problems (see http://www.stats.ox.ac.uk/~ripley/Win64/W64porting.html), and can be converted to static linking by removing the files like

```
.../x86_64-w64-mingw32/lib64/*.dll.a
.../bin/*.dll
```

where `...` is the top-level directory of the toolchain. If do you use a dynamically-linking toolchain be sure to arrange to distribute the required run-time DLLs (and note the requirements that their GPL license imposes on re-distribution).

The recommended toolchain `mingw-w64-1.0-bin_i686-mingw_20100322.zip` has no such `.../x86_64-w64-mingw32/lib64` directory; there is a single DLL `.../bin/libgcc_s_sjls-1.dll` which caused no apparent problem in Emerson's building of R based on preliminary usage. If the `libgcc_s_sjls-1.dll` file needs to be deleted, relocated, or referenced, this could be clarified here and/or on Ripley's page.

Package tcltk

To build package **tcltk** you need to unpack http://www.stats.ox.ac.uk/pub/Rtools/R64_Tcl_8-5-8.zip at the top of the source tree and *rename the top-level directory* as `Tcl64`: otherwise edit `src/library/tcltk/Makefile.win` to make a stub package by setting `BUILD_TCLTK = no`. If you're building **tcltk**, edit `src/gnuwin32/MkRules.local` so that `TCL_HOME = $(RHOME)/Tcl64`.

The default is `options(pkgType="win64.binary")`, supported for R 2.11.x on both CRAN and CRANextras. Packages using Gtk+ (**Cairo**, **RGtk2**, **cairoDevice** and those that depend on them) need the `bin` directory of the distribution at http://www.gtk.org/download-windows-64bit.html in the path: note that this conflicts with the directory of DLLs needed for 32-bit Gtk+ so you need the correct one first in your path if using both.

Building R

After creating the 64-bit toolchain and setting up (or turning off) the build of package **tcltk**, make R in the usual way as described in Section 3.1: fundamentally, `make all recommended` inside of *`R_HOME`*`/src/gnuwin32`, paying attention to all the other recommendations not specific to the 64-bit toolchain.

Final notes

A package-building service is available at http://win-builder.r-project.org/, and some notes on porting packages can be found at http://www.stats.ox.ac.uk/~ripley/Win64/W64porting.html. A port of Bioconductor package **Rgraphviz** is available on CRANextras (see the previous URL for sources for GraphViz), but other Bioconductor will at present need to be installed as source packages.

The default memory limit will be the amount of installed RAM: it can be changed by using (preferably) `--max-mem-size=` at startup or `memory.limit` in a running R session.

The personal library is (by default) folder `R\win64-library\`*`x.y`* of your home directory.

The assistance of Yu Gong at a crucial step in porting R to MinGW-w64 is gratefully acknowledged, as well as help from Kai Tietz, the lead developer of the MinGW-w64 project.