ADMB Installation Guide for a Windows PC

Jon T. Schnute, Chris Grandin, and Rowan Haigh
Fisheries and Oceans Canada
Pacific Biological Station, 3190 Hammond Bay Road, Nanaimo, BC, V9T 6N7, Canada
October, 2018

1. Background

On Unix[1] systems, such as Ubuntu or Mac OS, the ADMB package uses native software for compiling source code and building libraries. Unfortunately, Windows computers lack the required tools, and users must obtain them from other sources. We have taken advantage of open source software to assemble toolkits that extend Windows (32-bit or 64-bit) to a system capable of running ADMB and building binary libraries from ADMB source code.

At the time of writing, the current version of ADMB is 12.0. Section 3 outlines how one can install the necessary software to build ADMB on a Windows operating system (OS). Hopefully, the installation procedure will remain useful as ADMB is developed further. Originally, we pre-packaged toolkits for ADMB 11, which are still available on the ADMB Project web site. Section 0 explains precisely how to install ADMB 11.1 on a Windows platform with the aid of these toolkits. Sections 5 provides checks on testing your installation and Section 6 provides general, perhaps redundant, guidance on installing ADMB from source code. Finally, Appendix A gives a short primer on terminology associated with operating systems, where items A1-A8 in this report refer to notes in this appendix.

Before continuing further, please make sure that you know which version of Windows (32-bit or 64-bit) is installed on your computer. You can find this information in the Control Panel under the 'System' category, although the precise location may vary. (We call this the *Microsoft uncertainty principle*[2].) Alternatively, right-click a desktop icon for 'Computer', 'My Computer', or 'This PC', and select 'Properties'.

We emphasize that the guidance provided here is not the only way for using ADMB on Windows. For example, some people have used Microsoft Visual Studio; however, the Microsoft uncertainly principle almost guarantees that these instructions will soon go out of date. We have designed the guidance and tools below with three criteria in mind: (1) procedural success, (2) ease of implementation, and (3) clear documentation. We also restrict ourselves to free, open source software available from the GNU Project.

2. Support software to facilitate ADMB installation

1. We use the '7-Zip' utility to compress software for efficient transmission on the Internet. To unzip our files, you must install this package following instructions at https://www.7-zip.org/. Choose the appropriate version (32- or 64-bit).
2. You'll need to make small changes to your system environment variables, including the `Path`. (See Appendix A for a complete description of this terminology.) We find it helpful to use the 'Rapid Environment Editor'. Choose 'RapidEE_setup.exe - 32-bit & 64-bit in one package'. After installation, if you run the software from a shortcut on the desktop or elsewhere, you need to give the shortcut administrator privileges. Right-click the shortcut, select <Properties>, then the <Shortcut> tab, then the <Advanced> button, and click a check in the box <Run as administrator>.

[1] In 1970, a group from Bell Labs coined the name 'Unics' for Uniplexed Information and Computing Service.
[2] The act of learning to do something useful with Microsoft software increases the probability that Microsoft will move this feature to a different location in the interface or drop it altogether.

3. If you wish to obtain and compile the latest ADMB source code, you'll need software that can download it. These days, 'Git' is the versioning software of choice, and GitHub houses numerous repositories. We suggest using TortoiseGit, which provides a Windows shell interface to Git.

   Alternatively, it may be possible to access the source repository using TortoiseSVN. As usual, choose the appropriate version (32- or 64-bit). Incidentally, the Linux platform has a package 'RabbitVCS' designed to function like 'TortoiseSVN'. The developers of this package promise to make the rabbit capable of doing anything that the tortoise can.

4. Every Windows system can benefit from a decent set of Unix utilities. Gone are the days of amber screens and steering wheels; however, the modern Unix operating systems offer some useful functionality in file manipulation. There are various ways to get these utilities, from simple subsets (e.g., wintools, UnxUtils) to large (+100 MB) collections (e.g., Cygwin, Rtools). The smaller offerings are tempting but they just cannot handle the complexities of compiling ADMB, at least in our experience. The Rtools package seems to work best for us, and so we recommend using the latest version of Rtools, especially if you are heavily involved in creating and maintaining R packages.

3. ADMB 12.0 + GCC >5

The ADMB Project website provides a number of binary installers for ADMB 12. This is the easiest way to get a package containing pre-built ADMB 12 with a compatible GNU Compiler Collection (GCC 5.3.0, at the time of writing), specifically the installer '`admb-12.0-windows10-mingw64.exe`' for a 64-bit Windows system ('`admb-12.0-windows10-mingw32.exe`' for 32-bit systems). Download the executable file and run it as administrator. Install to a directory without spaces (e.g., '`C:\Apps\admb12`'), which will be termed 'ADMB Home'. Assuming this location, the GCC directory occurs at '`C:\Apps\admb12\utilities\mingw64`', which will be termed the 'MingGW Home'. Directories used here are for illustration; users can install the software to wherever they like (**caveat**: avoid directory names that contain spaces).

An alternative to this fixed-point build (perhaps you want the latest revision of ADMB from GitHub), ADMB 12 can be built from scratch using GCC 5.1.0 from Twilight Dragon Media (TDM). Providing and maintaining pre-packaged ADMB is not the primary focus of the authors, who would rather understand and communicate the methods so that others can follow suit and provide feedback for improvement.

Following are the steps that have been successful for us on a Windows 64-bit operating system, herein referred to as 'the OS':

1. Unix utilities – download '`Rtools35.exe`' from CRAN's website 'Building R for Windows'. Run this executable as administrator, choose a location (e.g., '`C:\Apps\Rtools`') and add the directory '`C:\Apps\Rtools\bin`' to your OS Path using Rapid Environment Editor (REE) or via the Control Panel (e.g., <Control Panel><System><Advanced system settings><Environment Variables>). Place the Rtools' `bin` directory ahead of any other Unix utility directories. We have successfully used an isolated copy of the utilities in `C:\Apps\Rtools\bin`, so you may wish to delete the other subdirectories if you do not use the R statistical platform.

2. GCC – go to Twilight Dragon Media (TDM) and download the latest installer. For the OS, we used '`tdm64-gcc-5.1.0-2.exe`' (TDM64 MinGW-w64 edition).

3. Run the GCC installer executable as administrator to install the TDM files. On the OS, assume that TDM's GCC was installed to '`C:\Apps\TDM-GCC-64`'.

4. ADMB – download the latest ADMB 12 source package. At the time of writing, ADMB's website offered a Dec 2017 release. Download the source code archive and unzip the contents to some working directory on your computer (e.g., 'C:\Dev\admb'). Alternatively, If you are familiar with GitHub, clone the admb GitHub repository to your computer and copy the contents to 'C:\Dev\admb' for compilation. Remove the subdirectory '.git' (or '.svn').
5. Make sure that the binary directory containing 'g++.exe' (C:\Apps\TDM-GCC-64\bin) occurs on the OS path before any other g++ installations. As above, you can use REE for this if you are not familiar with changing the path through Windows' Control Panel.
6. Open a command line console (called 'DOS' for brevity) and re-check that the g++ from TDM occurs first on your path by typing in DOS: 'where g++'. This will provide a list of all g++ locations on your path. If the TDM version does not occur in first place, you will have to close DOS, return to Step 5 to fix the path, and repeat Step 6. Do this Step for 'make' also to ensure that the directory of your preferred Unix utilities occurs before others.
7. In DOS, navigate to the ADMB source directory by typing: 'cd C:\Dev\admb' and start the ADMB build by typing 'make'. This will start a cascade of calls to g++ and will take ~1 hour to complete, depending on the speed of the OS. We encountered clean-up errors on completion (which is worrisome) but the compiled ADMB 12 software seems to work.
8. Extract the ADMB 12 binary build from the ADMB working directory. In Explorer (or Explorer++), navigate to 'C:\Dev\admb\build' and copy the entire directory called 'dist' to your preferred location for programs (e.g., 'C:\Apps'). Once the directory is copied, rename 'C:\Apps\dist' to 'C:\Apps\admb12' (or whatever you like).
9. If you wish, edit the VERSION file to include more info on the version. For example, at the time of writing, the GitHub revision number was 'f931dc9' (a truncated SHA[3] number) so one might change the version from '12.0' to '12.0.f931dc9', but this is not strictly necessary.
10. Add a new system environment variable (or modify an existing one): ADMB_HOME=C:\Apps\admb12. using REE or see A6.

4. ADMB 11.1 + GCC 4.8.1

In 2013, the authors made available pre-packaged zip archives on ADMB Tools for Windows, including binary builds of ADMB 11.1 revision 1483, both 32-bit and 64-bit, and mingw C++ executables and Unix tools (collectively called 'ADMB Tools'). These files are static in that ADMB 11.1 was built using MinGW's g++ 4.8.1. That is, the ADMB code was compatible with the C++ at the time.

Installation Summary

Download the ADMB Windows toolkit and the ADMB binary library for your computer. Unzip these files, and put their contents into suitable directories. Then alter the Path to reflect your choices. This shouldn't take very long, although it may require notable time to download and unzip the toolkit, depending on the speed of your processor and Internet connection.

The steps below should be detailed enough that new users will understand precisely what to do.

[3] Secure Hash Algorithm – cryptographic hash function for data security.

Installation Details

1. Go to the site ADMB Tools for Windows and select 'ADMB Tools for nn-bit Windows', where nn (32 or 64) corresponds to your computer. This link takes you to a file ADMBtoolsnn-vv.7z, where vv is a version number. Download this file. As mentioned above, the process may take a while because the file has size in the range 120 MB to 140 MB.
2. Return to the site ADMB Tools for Windows and select the appropriate 'binary library for nn-bit Windows'. Follow the link and download a file ADMBlibnn.vv.v.rrrr.7z, where vv.v (currently 11.1) is a version number and rrrr is a revision number for ADMB. (We explain more about the revision number in the section below on building ADMB from the latest source code.)
3. Open the file explorer in the directory (normally C:\Users\<User>\Downloads, where <User> refers to your user name) that contains the two .7z files that you just downloaded. Right-click the ADMBtools file, then select 7-Zip and Extract Here. (This might take a few minutes, depending on the speed of your computer.) When completed, the result is a new subdirectory ADMBtoolsnn-vv that contains a further subdirectory mingw.

   Copy (or cut and paste) mingw to a location of your choice. You may first have to remove or rename any previous version of this directory. In this report, we assume that you use the standard location C:\mingw. As a check, verify that your installation includes the two files C:\mingw\bin\g++.exe and C:\mingw\msys\bin\make.exe.
4. Similarly, right-click the ADMBlib file, then select 7-Zip and Extract Here. This gives a new subdirectory admblibrrrr with the revision number rrrr. Rename this subdirectory to admb, and copy (or cut and paste) it to a location of your choice. Again, you might first need to remove or rename a previous version of this directory. In this report, we assume that you use the standard location C:\admb. As a check, verify that your installation includes the two files C:\admb\bin\admb.cmd and C:\admb\bin\tpl2cpp.exe.
5. You now have to tell Windows where you've put all these files. Add the following three directories to the Path system environment variable: C:\admb\bin, C:\mingw\bin, and C:\mingw\msys\bin. Usually, it's best to put these directories before all others on the path. (Strictly speaking, the third directory is needed only if you intend to build ADMB libraries from source.) Also, add a new system environment variable ADMB_HOME=C:\admb. Of course, *if you choose other directories for the software, then these variables must reflect your choices*. Furthermore, if you make choices different from the standard, ***make sure that your directory names do not include spaces.***

   We recommend using the Rapid Environment Editor (cited above) to make these changes. You may need a little time to figure how this software works, but for us the effort was worth the trouble. (Alternatively, see A6.) If you use the R package PBSadmb to run ADMB, then you don't need to alter the Windows environment variables at all. Paths are handled by the package itself.

   Warning: Avoid conflicts with other software on your computer. For example, if you have installed Rtools, you might have multiple versions of 'make.exe' on your Path. The one you want for ADMB 11.1 should be in C:\mingw\msys\bin (if you installed the toolkit in its standard location). See A7 and A8.

5. ADMB software check

To test your installation, navigate to the example in C:\admb\examples\admb\simple. Copy the files simple.tpl and simple.dat to a test directory of your choice and open a DOS command window in this directory.

If you navigate to a directory in the Windows file explorer, click on its name in the scroll bar on the left. Then press and hold the <Shift> key, right-click, and select <Open command window here> from the menu. This opens a console in the chosen directory. Windows 8 users may need to ensure that the command prompt is elevated to administrator status.
See this tutorial.

In the command window, type the command:

```
admb simple
```

This should produce output that includes the lines:

```
*** Parsing simple.tpl:
*** Compile simple.cpp:
*** Linking simple.obj:
Successfully built 'simple.exe'
```

Running `simple` (i.e. typing `simple` on the command line) should give this output:

```
Initial statistics: 2 variables; iteration 0; function evaluation 0; phase 1
Function value  3.6493579e+001; maximum gradient component mag -3.6127e+000
Var   Value    Gradient   |Var   Value    Gradient   |Var   Value    Gradient
  1  0.00000 -3.61269e+000 |  2  0.00000 -7.27814e-001 |

 - final statistics:
2 variables; iteration 7; function evaluation 19
Function value  1.4964e+001; maximum gradient component mag -7.0014e-005
Exit code = 1;  converg criter  1.0000e-004
Var   Value    Gradient   |Var   Value    Gradient   |Var   Value    Gradient
  1  1.90909 -7.00140e-005 |  2  4.07818 -2.08982e-005 |

Estimating row 1 out of 2 for hessian
Estimating row 2 out of 2 for hessian
```

Hopefully, everything works for you! If not, check all your steps carefully and follow the suggestions in A8.

6. Building ADMB from source – general guidance

The section on building ADMB 12 above outlines how to compile an ADMB library from source code. Obviously, this involves two steps: getting the source code and building the library.

To get the latest source, you can use TortoiseGit (cited above). Make a new directory for this purpose on your computer. Right-click this directory icon in the Windows file explorer, select 'Git Clone', enter the address '`https://github.com/admb-project/admb.git`' in the 'URL' entry box, and press `OK`. A sequence of files should download automatically, corresponding to the main 'trunk' of the code tree on the ADMB server. This may take some time. Once completed, the dialogue box near the bottom will show a message something like this:
'`Showing 4702 revision(s), from revision 6f165a1 to revision f931dc9`'
This 7-character revision `rrrrrrr` is a truncation of the Secure Hash Algorithm number:
`f931dc955601771a4b736a8824ffb9b7afcb6489`

Copy the contents of your download directory to directory `admb_rrrrrrr` that can serve as an archival copy of this revision. Remove the subdirectory `.git` and any files in the root starting with '`.git`' (e.g. '`.gitignore`'). Use a text editor on the `VERSION` file to include the revision number as part of the version number. (For example, at the time of writing, we changed '`12.0`'

to '12.0.f931dc9'.) We consider it important to keep track of the ADMB revision number in our binary libraries because significant changes to the source code can occur at any time.

You now have the latest code archived in its own directory (e.g., admb_f931dc9). To build the ADMB binary library, open a command window in this directory (following a hint above), and enter the simple command:

```
make
```

A seemingly endless stream of output now scrolls by, as various commands compile and link all the source code in ADMB. If all goes well, everything should end after about an hour with the encouraging message:

```
ADMB build completed.
```

In case something goes wrong, you can clean your archive directory with the command

```
make clean
```

that removes everything but the original contents downloaded from the ADMB server.

Assuming that you have successfully built ADMB, the actual binary library resides in the subdirectory build\dist of your archive directory. Copy the dist subdirectory to your intended location for the ADMB binary library (by default on C:\) and rename dist to admb. If you already have an existing library (by default C:\admb), you first need to rename it (e.g., C:\admb_old) or delete it. In the end, the contents of \build\dist should precisely match the contents of <your_chosen_location>\admb. As in step 4 under ADMB 11.1, check that your installation includes the two files …\admb\bin\admb.cmd and …\admb\bin\tpl2cpp.exe.

Hint 1: Some users prefer to maintain a permanent ADMB repository that gets updated and rebuilt regularly. For example, they might use the directory C:\ADMBrepo to hold the trunk contents of the remote server. Then they might use the following path to the ADMB library: C:\ADMBrepo\build\dist\bin in place of the standard C:\admb\bin.

Hint 2: If you have problems compiling in a DOS command shell, try using an MSYS shell, as described in A4. Assume that you have the directory ADMBrepo mentioned above. In the MSYS command window, type:
`cd /c/ADMBrepo`
to get to the source repository's directory; then type
`make`

Hint 3: If you type
`make 1> buildLog.txt 2>&1`
instead of just make, a file called buildLog.txt will contain all the outputs, including errors, from the build process. You will see nothing on the screen during the build. This is useful if you are having problems and need to ask for help on the ADMB developers mailing list. If you attach the file to your message, you will have a much higher chance of getting help.

Hint 4: You can use the MSYS command objdump to verify that your ADMB build has the intended 32-bit or 64-bit specifications. After a successful build, navigate to the directory C:\ADMBRepo\build\objects\dist, where you should see many *.obj files. Open a DOS command prompt in this directory and type the command:
`objdump -a optlp-linad99-adglobl.obj`
If the output includes the phrase 'pe-i386', the object is 32-bit. Similarly the phrase 'pe-x86-64' indicates a 64-bit object. You might wish to check other objects in this directory to confirm that the results are consistent with your intended specifications.

Tip for R users: The R package PBSadmb gives complete R support to ADMB. In a standard ADMB installation, users interact with the program via a DOS command shell. The package PBSadmb makes it possible to interact entirely from an R console. A single R script can encapsulate commands to ADMB, as well as all analyses that follow, including the production of graphs. The package also includes protocols for writing code that makes the integration between R and ADMB almost seamless, with the same variable names in both environments. An optional graphical user interface (GUI) in PBSadmb can guide users through the development of ADMB applications.

7. Acknowledgments

We thank our colleague Wade Cooper for finding the first official GNU 64-bit library to work with ADMB 11.1. We have adopted his suggestions here. The ADMB developer team, particularly Dave Fournier and Johnoel Ancheta, extensively revised the various scripts and make files that support the Windows distribution. Finally, a big thank-you to Yuri (Ellie) Jung whose passion for learning how to build ADMB led to a significant revision of this document and to the R package PBSadmb.

Appendix A. Notes about operating systems for ADMB users

A1. Operating systems have scripting languages that can be run in text-based windows called 'terminals' or 'consoles'. In a Windows OS, the terminal is often called a 'command prompt'. The scripting language implements standard procedures like listing, copying, moving, renaming, or deleting files. Similar to most computer languages, scripting languages support variables with names and values; for example, the variable named `ADMB_HOME` might have the value `C:\admb`. Such variables are called *environment variables*. A program's behaviour might be influenced by the value of an environment variable.

A2. In a Windows OS, the native scripting language is historically called DOS, for the Disk Operating System in the earliest Microsoft computers. For a list of possible DOS commands, see https://ss64.com/nt/. (Many other web sites are available.) Text files with the extensions `.bat` (for batch file) or `.cmd` (for command file) can contain programs written as a sequence of DOS commands. The main such script for ADMB on Windows is (in a standard installation) `C:\admb\bin\admb.cmd`, mentioned in step 4 of the ADBM 11.1 installation procedure above. If you open this file with a text editor, you can see a long sequence of DOS commands.

A3. A Unix-based operating system often has several scripting languages available. They have a history of colourful names, such as the C shell (somewhat like the C language) csh, the Bourne shell (revised by Bourne) sh, and the Bourne-again shell (further revised by Fox) bash. Unlike Windows, Unix variables and scripts are case-sensitive. You can view a summary of bash commands at https://ss64.com/bash/. The main such script for ADMB on Unix-based systems also appears in the Windows version as `C:\admb\bin\admb.sh` (in a standard installation). If you open this file with a text editor, you can see a long sequence of bash commands.

A4. The ADMB toolkits for Windows give you access to bash. For example, assuming a standard installation, you can navigate to the file `C:\mingw\msys\msys.bat` and double-click on its name. You should see a Unix terminal pop up on your Windows screen. It can accept bash commands, just as Windows command prompt can accept DOS commands. Furthermore, the embedded MSYS installation makes numerous bash commands available in a Windows command prompt. To see this, open a command prompt and type the usual command `dir` to see a listing of all files in the current directory. Similarly, type `ls` to execute a similar bash

command. This works because the ADMB toolkit includes the file `C:\mingw\msys\bin\ls.exe`, which lies on the path if you installed everything correctly.

A5. Windows and Unix-based computers allow for multiple users of the same system. This concept carries through to environment variables, which exist on two levels. The system level applies to the entire system (all users). The user level applies only to the current user, so that different users can have distinctly different user environment variables. If you use the Rapid Environment Editor, you'll see the system environment on the left and the user environment on the right. Changes to the environment for ADMB described here always apply to the *system* environment.

A6. If you prefer to alter the environment with standard Windows tools, you can use the sequence: <Control Panel>, <System>, <Advanced system settings>, <Environment Variables>. You can also use the `setx` command in a command prompt; e.g.:

```
setx ADMB_HOME C:\admb
```

Variables defined by `setx` are available in future command windows only, not in the current command window. If you specify a path containing spaces, you must delimit the path using quotes. For example:

```
setx ADMB_HOME "C:\Program Files\My ADMB Home"
```

However, in the context of ADMB, spaces are not allowed in directory names, so *don't follow this example*.

A7. The Windows `Path` variable specifies a sequence of directories in which to search for programs. Directories on the path are separated by semicolons, with no space. They apply sequentially from first on the left to last on the right. (In Unix, which is case-sensitive, directories on the `PATH` are separated by colons.) If multiple directories on the `Path` include a program with the same name, the first on the list will be used by default.

A8. Of course, 'path' is also an English word that conjures images like the trail of bread crumbs left by Hansel and Gretel. So, with this metaphor in mind, we sometimes just speak of the path, rather than the `Path` or `PATH`. In this case, the bread crumbs lead to executable programs, like `make.exe` or `admb.cmd`.

The DOS command '`where`' (comparable to the Unix `which`, available only in a bash shell) can be used to discover precisely where the path leads. To test this, open a DOS command prompt and type:

```
where make
```

If you have a standard installation, the result should be:

```
C:\mingw\msys\bin\make.exe
```

Similarly, where admb should give the result `C:\admb\bin\admb.cmd`. If you have multiple instances of either of these files on your path, you need to make sure that the intended results occur first on the path.

Even better, try to clean up old breadcrumbs so that they aren't scattered in multiple directions.