NIMBLE User Manual

NIMBLE Development Team

Version 0.8.0

https://r-nimble.org
https://github.com/nimble-dev/nimble

Contents

Part I

Introduction

Chapter 1

Welcome to NIMBLE

NIMBLE is a system for building and sharing analysis methods for statistical models from R, especially for hierarchical models and computationally-intensive methods. While NIMBLE is embedded in R, it goes beyond R by supporting separate programming of models and algorithms along with compilation for fast execution.

As of version 0.9.0, NIMBLE has been around for a while and is reasonably stable, but we have a lot of plans to expand and improve it. The algorithm library provides MCMC with a lot of user control and ability to write new samplers easily. Other algorithms include particle filtering (sequential Monte Carlo) and Monte Carlo Expectation Maximization (MCEM).

But NIMBLE is about much more than providing an algorithm library. It provides a language for writing model-generic algorithms. We hope you will program in NIMBLE and make an R package providing your method. Of course, NIMBLE is open source, so we also hope you'll contribute to its development.

Please join the mailing lists (see R-nimble.org/more/issues-and-groups) and help improve NIMBLE by telling us what you want to do with it, what you like, and what could be better. We have a lot of ideas for how to improve it, but we want your help and ideas too. You can also follow and contribute to developer discussions on the wiki of our GitHub repository.

If you use NIMBLE in your work, please cite us, as this helps justify past and future funding for the development of NIMBLE. For more information, please call `citation('nimble')` in R.

1.1 What does NIMBLE do?

NIMBLE makes it easier to program statistical algorithms that will run efficiently and work on many different models from R.

You can think of NIMBLE as comprising four pieces:

1. A system for writing statistical models flexibly, which is an extension of the BUGS language[1].
2. A library of algorithms such as MCMC.
3. A language, called NIMBLE, embedded within and similar in style to R, for writing algorithms that operate on models written in BUGS.
4. A compiler that generates C++ for your models and algorithms, compiles that C++, and lets you use it seamlessly from R without knowing anything about C++.

NIMBLE stands for Numerical Inference for statistical Models for Bayesian and Likelihood Estimation.

Although NIMBLE was motivated by algorithms for hierarchical statistical models, it's useful for other goals too. You could use it for simpler models. And since NIMBLE can automatically compile R-like functions into C++ that use the Eigen library for fast linear algebra, you can use it to program fast numerical functions without any model involved[2].

One of the beauties of R is that many of the high-level analysis functions are themselves written in R, so it is easy to see their code and modify them. The same is true for NIMBLE: the algorithms are themselves written in the NIMBLE language.

1.2 How to use this manual

We suggest everyone start with the Lightning Introduction in Chapter **??**.

Then, if you want to jump into using NIMBLE's algorithms without learning about NIMBLE's programming system, go to Part II to learn how to build

[1] See Chapter **??** for information about NIMBLE's version of BUGS.

[2] The packages Rcpp and RcppEigen provide different ways of connecting C++, the Eigen library and R. In those packages you program directly in C++, while in NIMBLE you program in R in a nimbleFunction and the NIMBLE compiler turns it into C++.

your model and Part III to learn how to apply NIMBLE's built-in algorithms to your model.

If you want to learn about NIMBLE programming (nimbleFunctions), go to Part IV. This teaches how to program user-defined function or distributions to use in BUGS code, compile your R code for faster operations, and write algorithms with NIMBLE. These algorithms could be specific algorithms for your particular model (such as a user-defined MCMC sampler for a parameter in your model) or general algorithms you can distribute to others. In fact the algorithms provided as part of NIMBLE and described in Part III are written as nimbleFunctions.

Chapter 2

Installing NIMBLE

2.1 Requirements to run NIMBLE

You can run NIMBLE on any of the three common operating systems: Linux, MacOS, or Windows.

The following are required to run NIMBLE.

1. R, of course.
2. The igraph, coda and R6 R packages.
3. A working C++ compiler that NIMBLE can use from R on your system. There are standard open-source C++ compilers that the R community has already made easy to install. See Section 2.2 for instructions. You don't need to know anything about C++ to use NIMBLE. This must be done before installing NIMBLE.

NIMBLE also uses a couple of C++ libraries that you don't need to install, as they will already be on your system or are provided by NIMBLE.

1. The Eigen C++ library for linear algebra. This comes with NIMBLE, or you can use your own copy.
2. The BLAS and LAPACK numerical libraries. These come with R, but see Section 2.5.3 for how to use a faster version of the BLAS.

Most fairly recent versions of these requirements should work.

2.2 Installing a C++ compiler for NIMBLE to use

NIMBLE needs a C++ compiler and the standard utility *make* in order to generate and compile C++ for models and algorithms.[1]

2.2.1 MacOS

On MacOS, you should install *Xcode*. The command-line tools, which are available as a smaller installation, should be sufficient. This is freely available from the Apple developer site and the App Store.

In the somewhat unlikely event you want to install from the source package rather than the CRAN binary package, the easiest approach is to use the source package provided at R-nimble.org. If you do want to install from the source package provided by CRAN, you'll need to install this gfortran package.

2.2.2 Linux

On Linux, you can install the GNU compiler suite (*gcc*/*g++*). You can use the package manager to install pre-built binaries. On Ubuntu, the following command will install or update *make*, *gcc* and *libc*.

```
sudo apt-get install build-essential
```

2.2.3 Windows

On Windows, you should download and install `Rtools.exe` available from https://cran.r-project.org/bin/windows/Rtools/. Select the appropriate executable corresponding to your version of R (and follow the urge to update your version of R if you notice it is not the most recent). This installer leads you through several 'pages'. We think you can accept the defaults with one exception: check the PATH checkbox (page 5) so that the installer will add the location of the C++ compiler and related tools to your system's PATH, ensuring that R can find them. After you click 'Next', you will get a page

[1] This differs from most packages, which might need a C++ compiler only when the package is built. If you normally install R packages using `install.packages` on Windows or MacOS, the package arrives already built to your system.

with a window for customizing the new PATH variable. You shouldn't need to do anything there, so you can simply click 'Next' again.

The checkbox for the 'R 2.15+ toolchain' (page 4) must be checked (in order to have *gcc/g++*, *make*, etc. installed). This should be checked by default.

2.3 Installing the NIMBLE package

Since NIMBLE is an R package, you can install it in the usual way, via `install.packages("nimble")` in R or using the `R CMD INSTALL` method if you download the package source directly.

NIMBLE can also be obtained from the NIMBLE website. To install from our website, please see our Download page for the specific invocation of `install.packages`.

2.4 Troubleshooting installation problems

We have tested the installation on the three commonly used platforms – MacOS, Linux, Windows[2]. We don't anticipate problems with installation, but we want to hear about any and help resolve them.

The following are some troubleshooting tips that have helped users in some situations.

For Windows:

- Be sure to check box to set `PATH` when installing Rtools.exe. Alternatively, one can set the `PATH` manually using syntax similar to this (after changing `C:\\Rtools\\bin;C:\\Rtools\\mingw_64\\bin` to be appropriate for your system):

```
path <- Sys.getenv('PATH')
newPath <- paste("C:\\Rtools\\bin;C:\\Rtools\\mingw_64\\bin;",
                 path, sep = "")
Sys.setenv(PATH = newPath)
```

- Be sure the `Rtools.exe` version matches the R version.
- Try re-installing `Rtools` followed by re-installing NIMBLE.

[2] We've tested NIMBLE on Windows 7, 8 and 10.

- If there are filesystem or permissions issues, it is possible to install NIMBLE in a local directory using the `lib` argument to `install.packages`.
- In the past we've heard reports from Windows users of problems when their filesystem involved a space in a directory name in the path to `RHOME`. We've also heard reports from Windows users of problems when R is installed on a network drive. We think both of these issues have been resolved.

For MacOS:

- Newly installed Xcode/command line tools may need to be started once manually to provide a one-time permission before they will work from NIMBLE.
- If multiple C++ compilers are present on a system, be sure the `PATH` will find the right one.

All operating systems:

- If problems arise from generating and compiling C++ files from the default location in R's `tempdir()`, one can use the `dirName` argument to `compileNimble` to put such files elsewhere, such as in a local working directory.

If those suggestions don't help, please post about installation problems to the nimble-users Google group or email nimble.stats@gmail.com.

2.5 Customizing your installation

For most installations, you can ignore low-level details. However, there are some options that some users may want to utilize.

2.5.1 Using your own copy of Eigen

NIMBLE uses the Eigen C++ template library for linear algebra. Version 3.2.1 of Eigen is included in the NIMBLE package and that version will be used unless the package's configuration script finds another version on the machine. This works well, and the following is only relevant if you want to use a different (e.g., newer) version.

The configuration script looks in the standard include directories, e.g. `/usr/include` and `/usr/local/include` for the header file `Eigen/Dense`. You can specify a particular location in either of two ways:

1. Set the environment variable `EIGEN_DIR` before installing the R package, for example: `export EIGEN_DIR=/usr/include/eigen3` in the bash shell.
2. Use `R CMD INSTALL --configure-args='--with-eigen=/path/to/eigen' nimble_VERSION.tar.gz` or `install.packages("nimble", configure.args = "--with-eigen=/path/to/eigen")`

In these cases, the directory should be the full path to the directory that contains the Eigen directory, e.g., `/usr/include/eigen3`. It is not the full path to the Eigen directory itself, i.e., NOT `/usr/include/eigen3/Eigen`.

2.5.2 Using libnimble

NIMBLE generates specialized C++ code for user-specified models and nimbleFunctions. This code uses some NIMBLE C++ library classes and functions. By default, on Linux the library code is compiled once as a linkable library - *libnimble.so*. This single instance of the library is then linked with the code for each generated model. In contrast, the default for Windows and MacOS is to compile the library code as a static library - *libnimble.a* - that is compiled into each model's and each algorithm's own dynamically loadable library (DLL). This does repeat the same code across models and so occupies more memory. There may be a marginal speed advantage. If one would like to enable the linkable library in place of the static library (do this only on MacOS and other UNIX variants and not on Windows), one can install the source package with the configuration argument `--enable-dylib` set to true. First obtain the NIMBLE source package (which will have the extension `.tar.gz` from our website and then install as follows, replacing `VERSION` with the appropriate version number:

```
R CMD INSTALL --configure-args='--enable-dylib=true' nimble_VERSION.tar.gz
```

2.5.3 BLAS and LAPACK

NIMBLE also uses BLAS and LAPACK for some of its linear algebra (in particular calculating density values and generating random samples from multivariate distributions). NIMBLE will use the same BLAS and LAPACK

installed on your system that R uses. Note that a fast (and where appropriate, threaded) BLAS can greatly increase the speed of linear algebra calculations. See Section A.3.1 of the R Installation and Administration manual available on CRAN for more details on providing a fast BLAS for your R installation.

2.5.4 Customizing compilation of the NIMBLE-generated C++

For each model or nimbleFunction, NIMBLE can generate and compile C++. To compile generated C++, NIMBLE makes system calls starting with `R CMD SHLIB` and therefore uses the regular R configuration in `${R_HOME}/etc/${R_ARCH}/Makeconf`. NIMBLE places a `Makevars` file in the directory in which the code is generated, and `R CMD SHLIB` uses this file as usual.

In all but specialized cases, the general compilation mechanism will suffice. However, one can customize this. One can specify the location of an alternative `Makevars` (or `Makevars.win`) file to use. Such an alternative file should define the variables `PKG_CPPFLAGS` and `PKG_LIBS`. These should contain, respectively, the pre-processor flag to locate the NIMBLE include directory, and the necessary libraries to link against (and their location as necessary), e.g., *Rlapack* and *Rblas* on Windows, and *libnimble*. Advanced users can also change their default compilers by editing the *Makevars* file, see Section 1.2.1 of the Writing R Extensions manual available on CRAN.

Use of this file allows users to specify additional compilation and linking flags. See the Writing R Extensions manual for more details of how this can be used and what it can contain.