The Journal of Open Source Software

ropenblas: Download, Compile and Link OpenBLAS Library with R

Pedro Rafael Diniz Marinho[1]

1 Department of Statistics, Federal University of Paraíba, João Pessoa, Paraíba - PB, Brazil

DOI:

Software

- Review
- Repository
- Archive

Submitted:
Published:

Summary

The `ropenblas` package aims to facilitate the day-to-day life of R programmers who want a little more performance on GNU/Linux systems, without removing the possibility that specific configurations are made, if they deem convenient. Through the package's `ropenblas()` and `rcompiler()` functions, the library user will be able to compile and link the R language in his GNU/Linux distribution with the OpenBLAS library, all within R and in a very simple. All functions work without being influenced by the GNU/Linux distribution and are independent of their repositories, that is, it does not matter which GNU/Linux distribution is being used. Linking the OpenBLAS library to R will bring better computational performance to the language in the most diverse algebraic operations commonly used in areas such as statistics, data science and machine learning.

Statement of Need

The `ropenblas` package aims to allow algebraic computing of R to be performed using the OpenBLAS library, that is, it allows easy linking of R to the OpenBLAS library on GNU/Linux systems without depending on the distribution repositories. This will allow several researches in the areas of statistics, data science and machine learning to take advantage of a more efficient performance in algebraic calculations, for example, multiplication, factorization and matrix inversion. The `ropenblas` library, version 0.2.9 will also allow the R programmer to have, in his GNU/Linux distribution, several compiled versions of the R language giving the possibility to easily switch between these versions, this being an Open Source functionality that is only possible in some commercial IDEs of R. All this is done within the R language, minimizing the chance of less experienced users to break their operating system by running several instructions that are not perfectly understood.

The fact that the `ropenblas` package does not depend on the repositories of the GNU/Linux distribution will allow that in more stable distributions the R programmer will have at his disposal the most recent version of the OpenBLAS libraries and the R language. Everything is done in a safe way, a since the `ropenblas` package uses the stable versions of the official development repositories of the OpenBLAS library and the R programming language, respectively. Until the present version, the package has more than 1000 downloads, having an average of more than 1000 downloads in the month prior to the date of this submission, based on the official R language repositories.

Introduction

The term "computational efficiency" is very common for those who program statistical methods, in which a large part of them involve algebraic operations that are often reproduced in computationally intensive simulations, such as Monte-Carlo simulations - MC

and resampling methods, as is the case with bootstrap resampling. Statistics is just one example within so many other areas that need performance and uses the R language.

In addition to the adoption of good programming practices and the maximum, efficient and adequate use of available computational resources, such as code parallelization, through multicore parallelism procedures allowed by most current processors and operating systems, small adjustments and linkage of libraries can provide useful benefits.

The `ropenblas` package aims to provide useful and simple experiences to R (R Core Team, 2016) programmers who develop their activities on GNU/Linux operating systems, these many developers around the world producing codes of great impact for the community. These experiences consist of being able to link any version of the OpenBLAS (Xianyi, Zhang, Wang Qian, and Werner Saar, 2016) library to the R language, as well as allowing the programmer to install and link various versions of R and make them available on his operating system as well as switch between these versions as they see fit.

Linking the R language to the OpenBLAS library can bring several benefits to algebraic computing in R. OpenBLAS is an Open-Source implementation of the Basic Linear Algebra Subprograms - BLAS library that is often the first library option for algebraic computing to be linked in the installation of R on many GNU/Linux distributions. The OpenBLAS library is available at https://github.com/xianyi/OpenBLAS and adds optimized implementations of linear algebra kernels that can run optimized on various processor architectures. OpenBLAS is based on the GotoBLAS2 project code in version 1.13 (Goto, 2010), code available under the terms of the BSD license.

The `ropenblas` is a package designed to facilitate the linking of the library OpenBLAS with the language R. The package, which works only for Linux systems, will automatically download the latest source code from the OpenBLAS library and compile the code. The package will automatically bind the language R, through the `ropenblas()` function, to use the OpenBLAS library. Everything will be done automatically regardless of the Linux distribution you are using. Enumerating some advantages of the package:

The `ropenblas` package is already available on the Comprehensive R Archive Network - CRAN, currently in version 0.2.9 and the project is maintained on GitHub at https://github.com/prdm0/ropenblas where contributors can find others details of the code, information, as well as being able to contribute with the development of the project. Information can also be found on the project website. On the website it is also possible to read the `NEWS.md` file with details of the versions and the focus of the current development. The site is deposited at https://prdm0.github.io/ropenblas/. Suggestions for improvements and bug reports can be sent via the link https://github.com/prdm0/ropenblas/issues. You can find out how to contribute to the package by accessing the `CONTRIBUTING.md` file at https://github.com/prdm0/ropenblas/blob/master/CONTRIBUTING.md

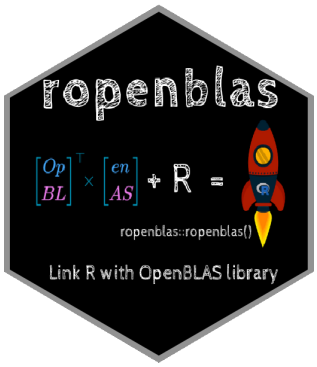

Figure 1: Computer library logo.

1. Everything is done within the R language;
2. The procedure (use of functions) will be the same for any Linux distribution;
3. The OpenBLAS library will be compiled and you will choose which build version to bind to R, regardless of your Linux distribution;
4. The package allows you to install R $\geq$ 3.1.0, also allowing you to install one more version, in addition to allowing you to easily switch between those versions;
5. The linked versions of R will continue to be recognized by their Integrated Development Environment - IDE and nothing will have to be adjusted in your GNU/Linux distribution after using any function of the package;
6. Unnecessary builds will be avoided. Therefore, if you need to switch between compiled versions of the R language, the use of binaries compiled in previous times will be suggested;
7. If any errors occur, the functions of the package will not damage the previous installation of the language;
8. If something better can be done or if a newer version of what you want to install (R or OpenBLAS) exists, the functions will automatically suggest that you consider installing newer versions.

Brief explanation

The `ropenblas` package can be installed in two ways. The first is using the `install.packages()` function of the `utils` package which is available in any basic language installation and the second is using the `devtools` package which will allow the package to be installed directly from the development directory on GitHub.

The `ropenblas` library exports six functions for use which are the `rcompiler()`, `ropenblas()`, `last_version_r()`, `last_version_openblas()`, `link_again()` and `rnews()`. All of them are very simple to use and have few arguments that are sufficient to maintain flexibility of use. Any example that follows will consider that the installation of the `ropenblas` package has been carried out and the package has been loaded (`library(ropenblas)`). In addition, functions like `rcompiler()` and `ropenblas()` do not return content or data structures that are of any practical use. What these functions do is configure the GNU/Linux system to use R, configure different versions of the language, switch between versions and link with the OpenBLAS library. It is also possible to obtain a summary of the versions of R and the OpenBLAS library that are available.

'last_version_r' function

The function `last_version_r()` automatically searches, in the official repositories of language R, for information about versions of language R. Its general use is `last_version_r(major = NULL)` , where the argument `major` indicates which is the largest version of R that you want to search for the version list. Therefore, for the argument `major` a number must be passed, preferably an integer that indicates which is the largest version to be considered.

'last_version_openblas' function

The `last_version_openblas()` function works similarly to the `last_version_r()` function, returning a list of three elements named in the same way with the information from the latest version, all the versions and the number of versions of the OpenBLAS library, respectively.

'rcompiler' function

This function is responsible for compiling a version of the R language. The x argument is the version of R that you want to compile. For example, `x = "4.0.4"` will compile and link R-4.0.4 version as the major version on your system. By default (`x = NULL`) will be compiled the latest stable version of the R. For example, to compile the latest stable version of the R language, run `rcompiler()`. The `rcompiler()` function can only be used if the user is an administrator of the GNU/Linux distribution. If the user is using programming IDE's, a screen similar to the image below will be displayed requesting the entry of the system administrator password.

```
> rcompiler(x = NULL, with_blas = NULL, complementary_flags = NULL)
```

- `x`: String with a valid R language version. A list valid of the latest language versions can be obtained using the `last_version_r()` function. You can move to `x` any of the returned versions. This is the best way to choose a valid argument for `x`. By default, `x = NULL` is equivalent to pass `last_version_r()$last_version`, that is, it will be considered the last stable version of the R language;
- `with_blas`: This argument sets the `--with-blas` flag in the R language compilation process and must be passed as a string. Details on the use of this flag can be found here;
- `complementary_flags`: String with complementary flags to be used in the R language compilation process.

If the goal is to install the R language, switch between versions of R, and link the installed versions of the language with the OpenBLAS library, you shouldn't have to worry about the `with_blas` and `complementary_flags` arguments, respectively. These arguments are useful for a minority of programmers who feel very specific needs to pass complementary flags to be considered in the R language compilation process, inclusion of complementary library directories, among other arguments that can be found in the official language manuals. By default, if nothing is passed to the `with_blas` and `complementary_flags` arguments, the compilation will be performed as follows:

```
./configure --prefix=/opt/R/version_r --enable-memory-profiling
--enable-R-shlib --enable-threads=posix
--with-blas="-L/opt/OpenBLAS/lib -I/opt/OpenBLAS/include -lpthread -lm"
```

As a suggestion and if you want to use the complementary arguments, that is, use `with_blas` and `complementary_flags`, do not change the directory `include` and `lib` of the OpenBLAS library in argument `with_blas`, considering that the `rcompiler()` function will always install the OpenBLAS library in the `/opt/OpenBLAS` directory, thus avoiding problems with breaking important links on your system and in the configuration of the initial version of R. If you try to pass a different installation directory to the OpenBLAS library, the `rcompiler()` function will disregard this installation of OpenBLAS and perform a safe installation in the `/opt/OpenBLAS` directory. Everything is built in this directory and symbolic links are used so that the initial state of R configuration in the GNU/Linux distribution is not changed. This allows no errors to be made and unforeseen bugs to occur. If there are reasons to consider complementary strings for the arguments `with_blas` and `complementary_flags`, the R language will be compiled as follows:

```
--prefix=/opt/R/version_r --enable-memory-profiling --enable-R-shlib
--enable-threads=posix --with-blas="..." complementary_flags
```

The `rcompiler()` function will also avoid unnecessary compilations. Therefore, if your initial version of R is already linked to some version of the OpenBLAS library, only the R language will be compiled, with no need to compile the OpenBLAS library. Although the initial description of the `rcompiler()` function may seem a little complicated, its use is very simple. Below I will exemplify some situations in which the use of the `rcompiler()`

function may be convenient.

'ropenblas' function

The `ropenblas()` function, a function of the same name in the package, links the main version of the R language installed on your operating system with the OpenBLAS library. As in the `rcompiler()` function, the `ropenblas()` function requires the operating system administration password, that is, it must be executed by the system administrator.

```
> ropenblas(x = NULL, restart_r = TRUE)
```

The `ropenblas()` function is made up of two arguments. Are they:

- `x`: String with the version of the OpenBLAS library to be compiled, installed and linked with the main R installation (by default it is considered the latest version);
- `restart_r`: Logical value (default `restart_r = TRUE`) to update the R section after compiling, installing and linking the OpenBLAS library.

The code below is a small example of the benefits of considering linking the R language to the OpenBLAS library, in which singular-value decomposition of a rectangular matrix is computed, codes that are executed on the same machine, version of R and section. The first part of the code was executed with R without being linked to a version of the OpenBLAS library and the second part was executed after using the `ropenblas()` function. Note that the code when executed in R linked to a version of the OpenBLAS library can be clearly more efficient:

```
# Using the BLAS + Lapack library:

# Matrix products: default

# BLAS:   /usr/lib/libblas.so.3.9.0

# LAPACK: /usr/lib/liblapack.so.3.9.0

n <- 1e3L

X <- matrix(rnorm(n * n), n, n)

system.time(svd(X))

## user      system      elapsed

## 7.246     0.029       7.286

# ----

# After linking the OpenBLAS library to R using the ropenblas() function:

# Using the OpenBLAS version 0.3.13 library:

# Matrix products: default

# BLAS/LAPACK: /opt/OpenBLAS/lib/libopenblas_haswellp-r0.3.13.so

n <- 1e3L

X <- matrix(rnorm(n * n), n, n)
```

```
system.time(svd(X))

## user      system    elapsed

## 3.024     2.004     0.677
```

Through a banchmark it is possible to better understand the performance gain that can be achieved by linking the R language to the OpenBLAS library. Figure 2 presents the banchmarks in the form of violin plot, in which 100 reproductions of the `svd(X)` expression were considered, in the form of the code above, with the R linked to the BLAS library and linked to the OpenBLAS library, respectively, on the same hardware. It was observed that the average time of execution of the routine `svd(X)` considering the OpenBLAS library was less than 10 times the time necessary to execute it in R linking to a non-optimized version of BLAS, being the average time of 0.64 and 6.52 seconds, respectively.

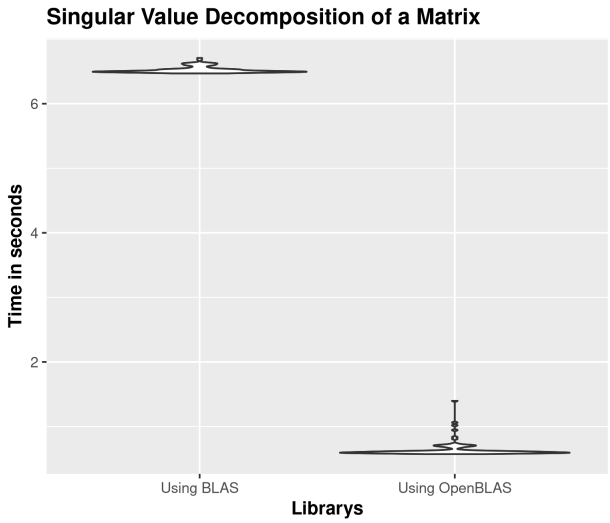

Figure 2: Benchmarks of a Singular Value Decomposition of a Matrix.

'link_again' function

The `link_again()` function links again the OpenBLAS library with the R language, being useful to correct problems of untying the OpenBLAS library that is common when the operating system is updated. The function be able to link again the R language with the OpenBLAS library.

Thus, `link_again()` will only make the relinkagem when in some previous section of R the `ropenblas()` function has been used for the initial binding of the R language with the OpenBLAS library.

The use of the function is quite simple, just by running the code `link_again()`, since the function has no arguments. It will automatically detect if there was a link break that will be rebuilt again without the need for any compilation. From time to time, after a major update of the operating system, it may be convenient to run the `link_again()` function. Link breakage rarely occurs, but if it does, it can be resolved quickly. The following code and image exemplify a possible reconstruction of symbolic links using the `link_again()` function:

```
> link_again()
```

'rnews function'

The `rnews()` function returns the contents of the `NEWS.html` file in the standard browser installed on the operating system. The `NEWS.html` file contains the main changes from the recently released versions of the R language. The goal is to facilitate the query by invoking it directly from the R command prompt.

References

Goto, K. (2010). GotoBLAS2 1.13 BSD version. *Texas Advanced Computing Center*. Retrieved from https://www.tacc.utexas.edu/research-development/tacc-software/gotoblas2

R Core Team. (2016). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

Xianyi, Zhang, Wang Qian, and Werner Saar. (2016). OpenBLAS: An optimized BLASLibrary. *Texas Advanced Computing Center*. Retrieved from http://www.openblas.net/