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 daily life of R programmers on GNU/Linux systems without removing the possibility of specifying configurations that they deem convenient. Through the `ropenblas()` and `rcompiler()` functions, the package user will be able to compile and link the R language in his GNU/Linux distribution and link it to the OpenBLAS library, all within R and in a very simple way. All functions work without being influenced by the GNU/Linux distribution and its repositories, that is, it doesn't matter which GNU/Linux distribution is being used. Linking to the OpenBLAS library to R will bring a better computational performance of the language in the most diverse algebraic operations that are commonly used in areas such as statistics, data science and machine learning.

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 functions of the `ropenblas` package can help the R language user to make this link without leaving the R promprt, as well as allowing the user to choose the version of OpenBLAS to be considered, the last stable version being considered by default . Everything is accomplished by functions without many arguments and without major complications to be used. These functions can be very comforting for users of R on GNU/Linux systems who do not feel safe to run several codes in a terminal that runs Shell Script codes for the language configuration.

It is very common to see users of R in distributions with repositories not prone to immediate updates of programs using several tutorials found in communities on the internet suggesting several lines of code and steps that can be potentially dangerous or easy to be misunderstood by many users of R language. Being able to compile, enable resources if necessary and switch between versions of R using simple functions to be used and without leaving the command pompt of the R language is attractive.

General package information

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.

The `ropenblas` package is already available on the Comprehensive R Archive Network - CRAN, currently in version 0.2.8 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.

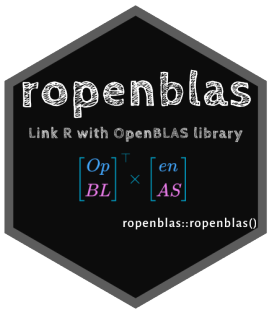

Figure 1: Computer library logo.

You can also specify older versions of the OpenBLAS library. Automatically, if no version is specified, the `ropenblas` package will consider the latest version of the library OpenBLAS.

Considering using the OpenBLAS library rather than the BLAS may bring extra optimizations for your code and improved computational performance for your simulations, since OpenBLAS is an optimized implementation of the library BLAS.

Some of the reasons why it is convenient to link R language to the use of BLAS optimized alternatives can be found here. Several other benchmarks that point to improved computing performance by considering the library OpenBLAS can be found on the internet.

The `ropenblas` package, by `rcompiler()` function is also useful if you want to install different versions of the R language. The different versions, specified by the user of the R language, will be compiled and will also be linked to the OpenBLAS library. If you want to switch between compiled versions of the R language, no compilation is needed anymore. This allows you to avoid having to get your hands dirty with tedious operating system settings, regardless of your GNU/Linux distribution. Another great use of the `rcompiler()` function is that you will not be dependent on updating your GNU/Linux distribution repositories and you can always have the latest version of the R language.

The use of the `ropenblas` package will return warnings that help you proceed with the use of the functions. If your internet is not working or if any dependency on the operating system is not present, the package will let you know. Enumerating some advantages of the package:

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.

Dependencies

In addition to dependencies in the form of other packages deposited with CRAN, the `ropenblas` package depends on external dependencies that are normally installed or are easily installed on any GNU/Linux distribution. Are they:

1. **GNU Make**: GNU Make utility to maintain groups of programs; 2. **GNU GCC Compiler (C and Fortran)**: The GNU Compiler Collection - C and Fortran frontends.

These programs that are described in `SystemRequirements` in the package's `DESCRIPTION` file are essential for compiling the OpenBLAS library and the R language. The functions of the `ropenblas` package are designed to identify the lack of these dependencies external to CRAN, informing the package user which dependencies are missing and suggesting

that they should be installed. The other dependencies indexed to CRAN are described in `Imports` in the file `DESCRIPTION`. These will be installed automatically.

Other warnings can also be suggested, such as, for example, a problem with the internet connection. All warnings are given very clearly so that the user has no doubts about the problem that may be occurring.

Installation

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. All code kept in the master branch of the package project on GitHub can be installed, since there will only be codes that are working properly and ready to use. The two forms of installation follow:

1. `install.packages("ropenblas")`: for installing the package available at CRAN;
2. `devtools::install_github(repo = "prdm0/ropenblas, ref = "master", force = TRUE)`: for installing the package from the project development directory on GitHub.

Exported functions and usage

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.

The following subsections are intended to explain and exemplify the use of these functions focused on the main problems that can be solved using the `ropenblas` package. Some functions may take a few minutes to run, as they are responsible for compiling code that is considerably larges and complicateds. This is the case with the `rcompiler()` and `ropenblas()` functions.

'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. Consider the example:

```
> last_version_r(major = 3L)
```

```
## $last_version
## [1] "3.6.3"
##
## $versions
##  [1] "3.0.0"         "3.0.1"         "3.0.2"         "3.0.3"
##  [5] "3.1.0"         "3.1.1"         "3.1.2"         "3.1.3"
##  [9] "3.2.0"         "3.2.1"         "3.2.2"         "3.2.3"
## [13] "3.2.4-revised" "3.2.4"         "3.2.5"         "3.3.0"
## [17] "3.3.1"         "3.3.2"         "3.3.3"         "3.4.0"
## [21] "3.4.1"         "3.4.2"         "3.4.3"         "3.4.4"
## [25] "3.5.0"         "3.5.1"         "3.5.2"         "3.5.3"
## [29] "3.6.0"         "3.6.1"         "3.6.2"         "3.6.3"
##
## $n
## [1] 32
```

Note that the `last_version_r()` function returns a list of three elements, where the first element called `last_version` returns the latest version of the R language based on the argument `major`. Thus, the latest version of the R language since the largest version is 3 is version 3.6.3. The second element of name `versions` of the return list is a vector containing all versions of the language R for `major = 3L`. The third and last element of the return list is named `n` and returns an integer value referring to the number of versions of R given `major = 3L`. By default, the `major` argument of the `last_version_r()` function is `NULL`. Considering the standard, the function will always consider the largest current version of the language, as in the example below:

```
> last_version_r(major = NULL)
```

```
## $last_version
## [1] "4.0.2"
##
## $versions
## [1] "4.0.0" "4.0.1" "4.0.2"
##
## $n
## [1] 3
```

The function is always adapted to search for new versions and will understand the new language updates since it has been implemented in a general way. Therefore, `last_version_r(major = NULL)$last_version` will always return the latest version of R and not the latest version of R available in the repositories of your GNU/Linux distribution. Therefore, it may be that, depending on when this function is performed, different results may occur.

'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. Unlike the `last_version_r()` function that searches for the versions of R on

the official language website, the `last_version_openblas()` function will search for information on the OpenBLAS library versions on official language development repository on GitHub. In addition, due to the way that the versions of the OpenBLAS library are numbered, the function will always consider the universe of all versions of the OpenBLAS library, therefore, being a function without arguments. The latest version of the OpenBLAS library can be returned by doing `last_version_openblas()$last_version`. Just like the `last_version_r()` function, the `last_version_openblas()` function does not depend on which version of OpenBLAS is available in the repositories of your GNU/Linux distribution.

```
> last_version_openblas()
```

```
## $last_version
## [1] "0.3.10"
##
## $versions
##  [1] "0.1alpha1"   "0.1alpha2"   "0.1alpha2.1" "0.1alpha2.2" "0.1alpha2.3"
##  [6] "0.1alpha2.4" "0.1alpha2.5" "0.1.0"       "0.1.1"       "0.2.0"
## [11] "0.2.1"       "0.2.2"       "0.2.3"       "0.2.4"       "0.2.5"
## [16] "0.2.6"       "0.2.7"       "0.2.8"       "0.2.9"       "0.2.9.rc1"
## [21] "0.2.9.rc2"   "0.2.10"      "0.2.10.rc1"  "0.2.10.rc2"  "0.2.11"
## [26] "0.2.12"      "0.2.13"      "0.2.14"      "0.2.15"      "0.2.16"
## [31] "0.2.16.rc1"  "0.2.17"      "0.2.18"      "0.2.19"      "0.2.20"
## [36] "0.3.0"       "0.3.1"       "0.3.2"       "0.3.3"       "0.3.4"
## [41] "0.3.5"       "0.3.6"       "0.3.7"       "0.3.8"       "0.3.9"
## [46] "0.3.10"
##
## $n
## [1] 46
```

'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 = "4.0.2"` will compile and link R-4.0.2 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.

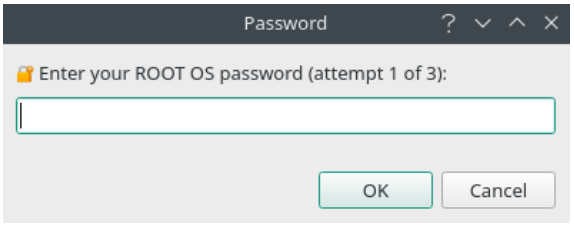

Figure 2: Window for entering the system administration password.

If the user is running the R language in a terminal of the GNU/Linux distribution, a Shell Bash type terminal, they will be asked to include the password by the terminal itself. The

function will not change the operation of other programs, nor even major changes will be made in the initial version of R, just summarizing the creation of symbolic links.

The general way of using the function is given by:

General use:

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

The function's arguments are described below:

- `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.

Example: In the hypothetical situation of an R user on any GNU/Linux distribution, if are interested in knowing the latest version of the R language, installing it and linking it to the OpenBLAS library, it should proceed as follows:

```
> rcompiler(x = "4.0.2") # or simply rcompiler () or rcompiler (x = NULL)
```

After executing the function, everything will be ready, and can be checked by executing the command `sessionInfo()` in the next execution of R, that is, in a new instance. Briefly, the `sessionInfo()` command will return something like:

```
> sessionInfo()
```

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Manjaro Linux
##
## Matrix products: default
## BLAS/LAPACK: /opt/OpenBLAS/lib/libopenblas_haswellp-r0.3.10.so
##
## locale:
##  [1] LC_CTYPE=pt_BR.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=pt_BR.UTF-8        LC_COLLATE=pt_BR.UTF-8
##  [5] LC_MONETARY=pt_BR.UTF-8    LC_MESSAGES=pt_BR.UTF-8
##  [7] LC_PAPER=pt_BR.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=pt_BR.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.0.2  magrittr_1.5    tools_4.0.2     htmltools_0.5.0
##  [5] rticles_0.15    yaml_2.2.1      stringi_1.5.3   rmarkdown_2.3
##  [9] highr_0.8       knitr_1.29      stringr_1.4.0   xfun_0.17
## [13] digest_0.6.25   rlang_0.4.7     evaluate_0.14
```

Thus the user will know that the compilation of the R library, and if necessary, of the OpenBLAS library was carried out as requested. The user can also use the command `extSoftVersion()["BLAS"]` if it is only necessary to return which version of the OpenBLAS library is being considered, according to the following code:

```
> extSoftVersion()["BLAS"]
```

```
##                                                    BLAS
## "/opt/OpenBLAS/lib/libopenblas_haswellp-r0.3.10.so"
```

Example: In a new hypothetical situation of a user of R having installed two different versions of the R language as presented above, using the function `rcompiler()`, for example, versions 3.6.3 and 4.0.0, so:

```
> # First installation of R performed by the rcompiler() function:
> rcompiler(x = "4.0.0")
> # Second installation of R performed by the rcompiler() function:
> rcompiler(x = "3.6.3")
> # Returning version 4.0.0 from R:
> rcompiler(x = "4.0.0")
```

Running the code `rcompiler(x = "4.0.0")` as the last line of the example above will allow the user to link the binaries previously created in the 4.0.0 compilation process when `rcompiler(x = "4.0.0")` was run for the first time. This will avoid a compilation so that switching between versions of R that have already been compiled will be quick and easy. Although this seems ideal and it is, the function will ask if the user wants to recompile an already compiled version.

Example: Assuming that the user of the package has already used the function `rcompiler(x = "4.0.0")` and is in a version other than 4.0.0 but intends to return to version 4.0.0, he can return without compile the language again, as shown in the image below. The output was placed in the form of an image since the return of the functions of the package use symbols that facilitate the user of the functions of the package to understand the suggestions, as well as understand if something went right or wrong.

```
rcompiler(x = "4.0.0")
```

```
🔵 R version already compiled: (yes - changes without recompiling) and (no - compiles again) (yes/no): yes

═══════════════════ Procedure Completed ═══════════════════
✅ R version 4.0.0.
🔄 The roles are active after terminating the current R session ...
```

Figure 3: A possible exit from the 'rcompiler' function. Suggestion of switching between versions of R without the need to recompile the language.

Due to the internal dependencies of the `ropenblas` package, it is advisable not to try to go to versions of R prior to 3.1.0, as these dependencies do not work on these versions of R. Therefore, in earlier versions of R, installing the `ropenblas` package will not be possible. Therefore, if the user wishes to return to a more current version of R, he must install the language R in the GNU/Linux distribution without the help of the package. Although it is allowed to install previous versions of R prior to 3.1.0, it will only be done after the user responds to some questions of an alert message, as shown in the image below:

```
rcompiler(x = "3.0.0")
```

```
═══════════════════════════════ VERY ATTENTION ═══════════════════════════════
The ropenblas package depends on versions of R ≥ 3.1.0. Therefore,you will not be able to
use this package to go back to a later version of R!
🔴 Do you understand? (1 of 3) (yes/no): yes
🔴 Do you understand? (2 of 3) (yes/no): yes
🔴 Do you understand? (3 of 3) (yes/no): no
Warning message:
In rcompiler(x = "3.0.0") :
🔵 Given the answers, it is not possible to continue ...
```

Figure 4: Attempted to install an R language version prior to version 3.1.0.

'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.

Unlike the `rcompiler()` function that will allow multiple versions of R to be available in your GNU/Linux distribution, the `ropenblas()` function will allow only one version of the OpenBLAS library to be installed. The function will allow versions of OpenBLAS $\geq$ 0.3.0 to be installed. Even if the function `rcompile()` responsible for compiling and installing the R language has not been used, the function `ropenblas()` may link a version of OpenBLAS to the main installation of R, that is, even if the R is installed by procedures external to the package.

General use:

```
> 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;
- `restart_r`: Logical value (default `restart_r = TRUE`) to update the R section after compiling, installing and linking the OpenBLAS library. \end{itemize}

By default, if `x = NULL` the latest version of the OpenBLAS library will be considered. This is equivalent to passing `last_version_openblas()$last_version` to `x`. There are no great reasons to consider an older version of the OpenBLAS library, although the `ropenblas` package will allow this to be done in a simple way, as in the following example:

Example: Regardless of how the R language was installed, suppose you want to compile, install and link the latest version of the OpenBLAS library to the R language. This can be done by doing:

```
> ropenblas()
```

```
═══════════════════════ Procedure Completed ═══════════════════════
✅ OpenBLAS version 0.3.9.

Restarting R session...
```

Figure 5: Output informing the end of the procedure for linking the OpenBLAS library to the R language.

Example: In situations where the latest version of the OpenBLAS library is already installed, if the user wishes to install a previous version of OpenBLAS, for example the version 0.3.8, he should make it clear by answering a question, as shown in the image below:

```
> ropenblas("0.3.8")
```

Figure 6: Asking if the user really wants to compile and link an older version of 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 0.3.10 library:
# Matrix products: default
# BLAS/LAPACK: /opt/OpenBLAS/lib/libopenblas_haswellp-r0.3.10.so
n <- 1e3L
X <- matrix(rnorm(n * n), n, n)
system.time(svd(X))
## user      system    elapsed
## 3.024     2.004     0.677
```

'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()
```

```
═══════════════════════ Procedure Completed ═══════════════════════
✅ OpenBLAS.

Restarting R session...
```

Figure 7: If an unlinking of the OpenBLAS library occurs, the function will re-link the library.

Running the `link_again()` function in a situation where there is no need will not generate problems. The function will return the message that everything is linked correctly, according to the code and image that follows:

```
🔵 Linking again is not necessary. R already uses the OpenBLAS library. You can stay calm.
```

Figure 8: If relinking is not required, the function will make it clear.

'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. The `rnews()` function is analogous to the news function of the `utils` package. However, using the news command in a terminal style bash shell is possible to receive a message like:

```
news()
## starting httpd help server ... done
## Error in browseURL(url): 'browser' must be a non-empty character string
```

If `pdf = FALSE` (default), the `NEWS.html` file will open in the browser, otherwise `NEWS.pdf` will be opened. If `dev = FALSE` (default), it will not show changes made to the language development version. To see changes in the development version, do `dev = TRUE`.

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/