



# ENABLE MULTITHREAD WITH DATA.TABLE IN MAC/INTEL CHIPS

Multithreading offers faster and more efficient data processing, allowing for quicker analysis of complex datasets.

Jorge Roa, Fernando Alarid-Escudero



This document shows you how we can enable the use of multiple cores on Macs with Intel/Apple silicon chips (M1 and M2).

---

If we load data.table library in <fa brands r-project> studio, this message will appear in your console:

```
> library(data.table)
data.table 1.14.6 using 1 threads (see ?getDTthreads). Latest news: r-ddatatable.com
*****
This installation of data.table has not detected OpenMP support. It should still work but in single-threaded mode.
This is a Mac. Please read https://mac.r-project.org/openmp/. Please engage with Apple and ask them for support. Check r-ddatatable.com for updates, and our Mac instructions here: https://github.com/Rdatatable/data.table/wiki/Installation. After several years of many reports of installation problems on Mac, it's time to gingerly point out that there have been no similar problems on Windows or Linux.
*****
```

As you can see, OpenMP support is needed to use multiple cores in Macs. Therefore, we must install those packages through the terminal and set the required paths to run OpenMP.

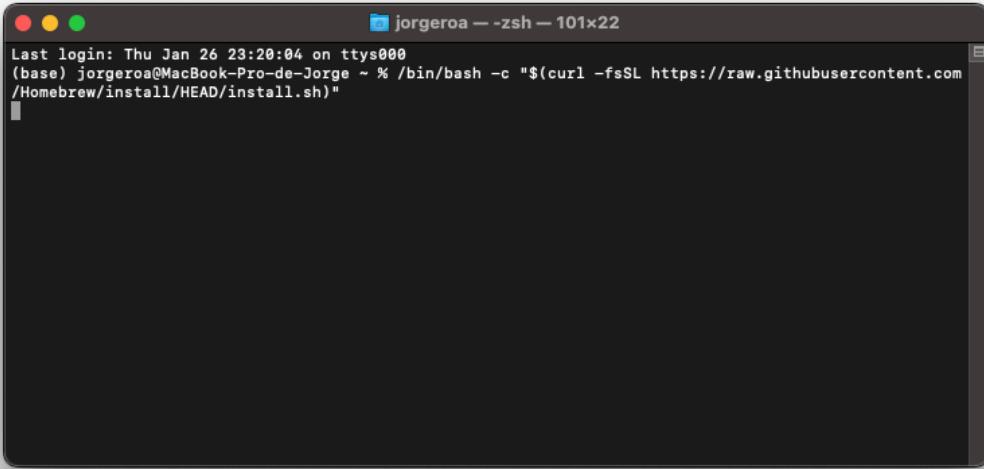
## Prerequisites

- 1) Have the latest version of <fa brands r-project> studio.
- 2) Have the latest version of Mac Ventura



## First Step

- 1) Open your terminal. It should see like this:



```
Last login: Thu Jan 26 23:20:04 on ttys000
(base) jorgeroea@MacBook-Pro-de-Jorge ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

## Second Step (Install homebrew)



**Homebrew**  
The missing package manager  
for macOS (or Linux)

is an open-source software package management system that makes installing applications on Linux and Apple's macOS operating systems easier. Homebrew will help us install OpenMP. Open Multi-Processing allows us to run applications in parallel to efficient processes. In this case, we want to employ the multiple cores that Mac has for `data.table` wrangling.

As the Homebrew page states, Homebrew installs the stuff you need that Apple (or your Linux system) didn't. So, once we open our terminal, we need to paste this command on our terminal.



```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

To install homebrew, you need to enter your password and press ENTER. After installing homebrew, you

```
[jorgeroa_macbookair@MacBook-Air-de-Jorge ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
==> Checking for `sudo` access (which may request your password)...
[Password:
==> This script will install:
/opt/homebrew/bin/brew
/opt/homebrew/share/doc/homebrew
/opt/homebrew/share/man/man1/brew.1
/opt/homebrew/share/zsh/site-functions/_brew
/opt/homebrew/etc/bash_completion.d/brew
/opt/homebrew

Press RETURN/ENTER to continue or any other key to abort:
==> /usr/bin/sudo /usr/sbin/chown -R jorgeroa_macbookair:admin /opt/homebrew
==> Downloading and installing Homebrew...
remote: Enumerating objects: 1854, done.
remote: Counting objects: 100% (1084/1084), done.
remote: Compressing objects: 100% (171/171), done.
remote: Total 1854 (delta 929), reused 1033 (delta 903), pack-reused 770
Receiving objects: 100% (1854/1854), 1.08 MiB | 7.29 MiB/s, done.
Resolving deltas: 100% (1261/1261), completed with 247 local objects.
From https://github.com/Homebrew/brew
 * [new branch]           dawidd6-patch-1 -> origin/dawidd6-patch-1
 * [new branch]           dependabot/bundler/Library/Homebrew/tzinfo-2.0.6 ->
origin/dependabot/bundler/Library/Homebrew/tzinfo-2.0.6
 * [new branch]           dependabot/bundler/Library/Homebrew/webrick-1.8.1 ->
origin/dependabot/bundler/Library/Homebrew/webrick-1.8.1
   9e9b71bea..4b3d64d30  master       -> origin/master
 * [new tag]              3.6.18        -> 3.6.18
 * [new tag]              3.6.19        -> 3.6.19
 * [new tag]              3.6.20        -> 3.6.20
HEAD is now at 4b3d64d30 Merge pull request #14453 from Rylan12/languages-in-cask-api
Updated 1 tap (homebrew/core).
Warning: /opt/homebrew/bin is not in your PATH.
Instructions on how to configure your shell for Homebrew
can be found in the 'Next steps' section below.
==> Installation successful!

==> Homebrew has enabled anonymous aggregate formulae and cask analytics.
Read the analytics documentation (and how to opt-out) here:
  https://docs.brew.sh/Analytics
```

## Third Step (Check package and change paths)

We need to be sure that homebrew was installed in the correct path. For this, we can type in the terminal:

```
brew help
```



We need to set our paths properly if the command is not recognized.

---

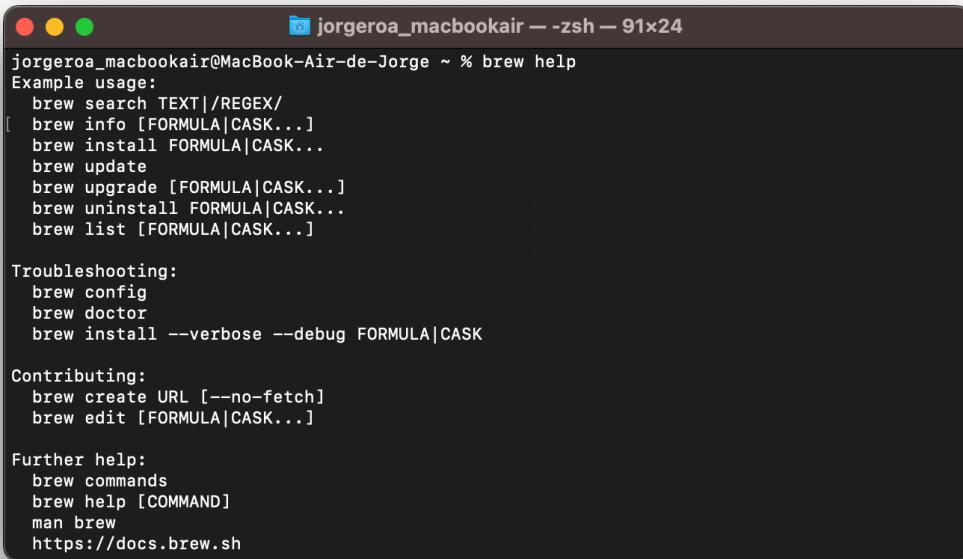
Apple chips 

---

```
export PATH=/opt/homebrew/bin:$PATH  
Intel chips intel.  
export PATH=/usr/local/opt/homebrew/bin:$PATH
```

---

With these changes, we can rerun homebrew help, and this information should appear in our terminal.



jorgeroa\_macbookair@MacBook-Air-de-Jorge ~ % brew help

Example usage:

```
brew search TEXT|/REGEX/  
[ brew info [FORMULA|CASK...]  
brew install FORMULA|CASK...  
brew update  
brew upgrade [FORMULA|CASK...]  
brew uninstall FORMULA|CASK...  
brew list [FORMULA|CASK...]
```

Troubleshooting:

```
brew config  
brew doctor  
brew install --verbose --debug FORMULA|CASK
```

Contributing:

```
brew create URL [--no-fetch]  
brew edit [FORMULA|CASK...]
```

Further help:

```
brew commands  
brew help [COMMAND]  
man brew  
https://docs.brew.sh
```

## Fourth Step: Installing required packages (libopenmp, libopenmpt, llvm, cask, ggc)

### Install llvm

---

We need to install the llvm package to access the clang compiler, which helps us to set multi-threading.

`brew install llvm`  
Once the installation is finished, we should see this screen in the terminal.



```
jorgeroa_macbookair ~ % brew install llvm
cd /opt/homebrew/opt/libomp/lib
==> Fetching dependencies for llvm: mpdecimal, ca-certificates, openssl@1.1, readline, sqlite, xz, python@3.11, six, z3, lz4 and zstd
[==> Fetching mpdecimal
==> Downloading https://ghcr.io/v2/homebrew/core/mpdecimal/manifests/2.5.1
#####
==> Downloading https://ghcr.io/v2/homebrew/core/mpdecimal/blobs/sha256:5b1c62c0
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:5b1c62c0
#####
==> Fetching ca-certificates
==> Downloading https://ghcr.io/v2/homebrew/core/ca-certificates/manifests/2023-05-10T12:00:00Z
#####
==> Downloading https://ghcr.io/v2/homebrew/core/ca-certificates/blobs/sha256:11
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:11
#####
==> Fetching openssl@1.1
==> Downloading https://ghcr.io/v2/homebrew/core/openssl@1.1/manifests/1.1.1s
#####
==> Downloading https://ghcr.io/v2/homebrew/core/openssl@1.1/blobs/sha256:3a7812
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:3a7812
#####
==> Fetching readline
==> Downloading https://ghcr.io/v2/homebrew/core/readline/manifests/8.2.1
```

```
==> Installing llvm
==> Pouring llvm--15.0.7_1.arm64_ventura.bottle.tar.gz
==> Caveats
To use the bundled libc++ please add the following LDFLAGS:
  LDFLAGS="-L/opt/homebrew/opt/llvm/lib/c++ -Wl,-rpath,/opt/homebrew/opt/llvm/lib/c++"

llvm is keg-only, which means it was not symlinked into /opt/homebrew,
because macOS already provides this software and installing another version in
parallel can cause all kinds of trouble.

If you need to have llvm first in your PATH, run:
  echo 'export PATH="/opt/homebrew/opt/llvm/bin:$PATH"' >> ~/.zshrc

For compilers to find llvm you may need to set:
  export LDFLAGS="-L/opt/homebrew/opt/llvm/lib"
  export CPPFLAGS="-I/opt/homebrew/opt/llvm/include"

==> Summary
  /opt/homebrew/Cellar/llvm/15.0.7_1: 6,411 files, 1.3GB
==> Running `brew cleanup llvm`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
==> Caveats
==> llvm
```



## Install libopenmp

---

```
brew install libopenmp
```

This is the screen that you should see once the installation is done.

```
jorgeroa_macbookair ~ % brew install libopenmp
==> Tapping homebrew/cask
[Cloning into '/opt/homebrew/Library/Taps/homebrew/homebrew-cask'...]
remote: Enumerating objects: 705392, done.
[remote: Counting objects: 100% (76/76), done.
[remote: Compressing objects: 100% (55/55), done.
[remote: Total 705392 (delta 38), reused 56 (delta 21), pack-reused 705316
Receiving objects: 100% (705392/705392), 332.00 MiB | 3.66 MiB/s, done.
Resolving deltas: 100% (501569/501569), done.
Tapped 4152 casks (4,224 files, 354.4MB).
Warning: No available formula with the name "libopenmp". Did you mean libopenmpt, libomp, libopenpnet, libupnp or liboping?
==> Searching for similarly named formulae and casks...
==> Formulae
libopenmpt      libomp       libopenpnet     libupnp      liboping

To install libopenmpt, run:
  brew install libopenmpt

==> Casks
openmtp

To install openmtp, run:
  brew install --cask openmtp
```

## Install libopenmpt

---

```
brew install libopenmpt
```

This is the screen that you should see once the installation is done.



```
jorgeroa_macbookair ~ % brew install libopenmpt

==> Fetching dependencies for libopenmpt: libogg, flac, lame, libvorbis, mpg123, opus, libsndfile and portaudio
==> Fetching libogg
==> Downloading https://ghcr.io/v2/homebrew/core/libogg/manifests/1.3.5
#####
==> Downloading https://ghcr.io/v2/homebrew/core/libogg/blobs/sha256:f18fefb04d1
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:4e8f1c6298d84
#####
==> Fetching flac
==> Downloading https://ghcr.io/v2/homebrew/core/flac/manifests/1.4.2
#####
==> Downloading https://ghcr.io/v2/homebrew/core/flac/blobs/sha256:4e8f1c6298d84
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:4e8f1c6298d84
#####
==> Fetching lame
==> Downloading https://ghcr.io/v2/homebrew/core/lame/manifests/3.100
#####
==> Downloading https://ghcr.io/v2/homebrew/core/lame/blobs/sha256:dde2fd627f564
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:dde2fd627f564
#####
==> Fetching libvorbis
==> Downloading https://ghcr.io/v2/homebrew/core/libvorbis/manifests/1.3.7
```

⋮

## Install gcc

---

```
brew install gcc
```

This is the screen that you should see once the installation is done.

```
jorgeroa_macbookair ~ % brew install gcc

==> Fetching dependencies for gcc: gmp, isl, mpfr and libmpc
==> Fetching gmp
==> Downloading https://ghcr.io/v2/homebrew/core/gmp/manifests/6.2.1_1
#####
==> Downloading https://ghcr.io/v2/homebrew/core/gmp/blobs/sha256:2436cd120e5678
[==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:93e737d38d526b
#####
==> Fetching isl
==> Downloading https://ghcr.io/v2/homebrew/core/isl/manifests/0.25
#####
==> Downloading https://ghcr.io/v2/homebrew/core/isl/blobs/sha256:93e737d38d526b
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:93e737d38d526b
#####
==> Fetching mpfr
==> Downloading https://ghcr.io/v2/homebrew/core/mpfr/manifests/4.2.0
#####
==> Downloading https://ghcr.io/v2/homebrew/core/mpfr/blobs/sha256:e8d923fe4d128
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:e8d923fe4d128
#####
==> Fetching libmpc
==> Downloading https://ghcr.io/v2/homebrew/core/libmpc/manifests/1.3.1
#####
==> Downloading https://ghcr.io/v2/homebrew/core/libmpc/blobs/sha256:da4ff781bc4
```



## Install --cask openmtp

```
brew install --cask openmtp
```

This is the screen that you should see once the installation is done.

```
jorgeroa_macbookair ~ % brew install --cask openmtp
==> Downloading https://github.com/ganeshrvel/openmtp/releases/download/v3.2.10/
==> Downloading from https://objects.githubusercontent.com/github-production-releas...
#####
# 100.0%
==> Installing Cask openmtp
==> Moving App 'OpenMTP.app' to '/Applications/OpenMTP.app'
[!] openmtp was successfully installed!
jorgeroa_macbookair ~ % brew install gcc
==> Fetching dependencies for gcc: gmp, isl, mpfr and libmpc
==> Fetching gmp
==> Downloading https://ghcr.io/v2/homebrew/core/gmp/manifests/6.2.1_1
#####
# 100.0%
==> Downloading https://ghcr.io/v2/homebrew/core/gmp/blobs/sha256:2436cd120e5678
[==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:...
#####
# 100.0%
==> Fetching isl
==> Downloading https://ghcr.io/v2/homebrew/core/isl/manifests/0.25
#####
# 100.0%
==> Downloading https://ghcr.io/v2/homebrew/core/isl/blobs/sha256:93e737d38d526b
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:...
#####
# 100.0%
==> Fetching mpfr
==> Downloading https://ghcr.io/v2/homebrew/core/mpfr/manifests/4.2.0
#####
# 100.0%
```

}

## Fifth Step: create .R folder and Makevars file.

We need to create a text file called `Makevars`. This file is necessary because we need to set a file with multiple paths where we will retrieve the various packages we have installed in our R environment. We need to open a new script or open our terminal to execute the next commands:

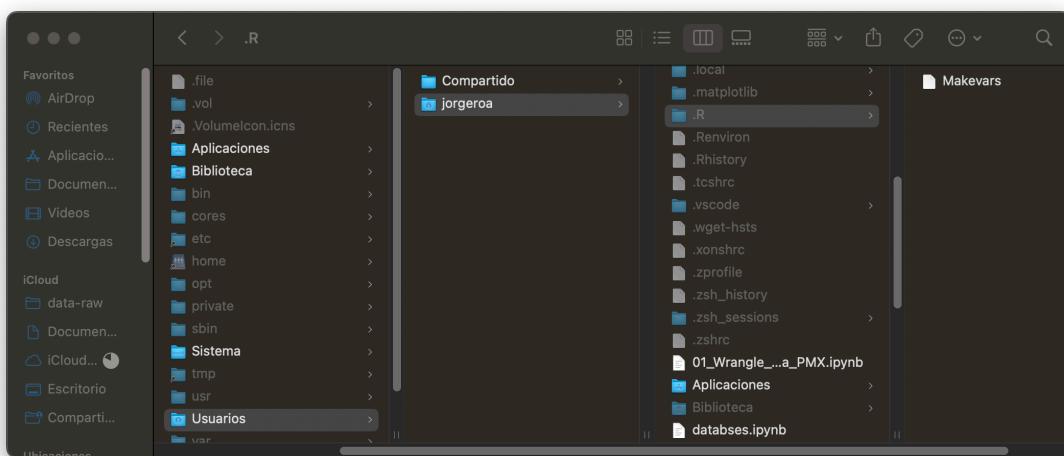
```
dir.create('~/ .R')
file.create('~/ .R/Makevars')
```



`dir.create` creates a hidden folder in our environment, and `file.create` creates a text file in the folder we just created. This folder is located in our `usr`s folder. You can access your user folder and your hidden folders with the next shortcuts:

Open Finder	Shortcuts
Go to your home folder:	<code>CMD + Shift + H</code>
Show your hidden folders:	<code>CMD + Shift + .</code>

You should have this path (according to your chip).



## Sixth Step: Create our paths in our Makevars file.

We need to open our `Makevars` file with `TextEdit`, and we must paste the following paths according to our chip (Apple Silicon or Intel)

### Apple Silicon

```
HOMEBREW_LOC=/opt/homebrew
LLVM_LOC=$(HOMEBREW_LOC)/opt/llvm
CC=$(LLVM_LOC)/bin/clang -fopenmp
CXX=$(LLVM_LOC)/bin/clang++ -fopenmp
CFLAGS=-g -O3 -Wall -pedantic -std=gnu99 -mtune=native -pipe
CXXFLAGS=-g -O3 -Wall -pedantic -std=c++11 -mtune=native -pipe
LDFLAGS=-L$(HOMEBREW_LOC)/opt/gettext/lib -L$(LLVM_LOC)/lib
    -Wl,-rpath,$(LLVM_LOC)/lib
CPPFLAGS=-I$(HOMEBREW_LOC)/opt/gettext/include -I$(LLVM_LOC)/include
```



```
HOMEBREW_LOC=/opt/homebrew  
LLVM_LOC=$(HOMEBREW_LOC)/opt/llvm  
CC=$(LLVM_LOC)/bin/clang -fopenmp  
CXX=$(LLVM_LOC)/bin/clang++ -fopenmp  
CFLAGS=-g -O3 -Wall -pedantic -std=gnu99 -mtune=native -pipe  
CXXFLAGS=-g -O3 -Wall -pedantic -std=c++11 -mtune=native -pipe  
LDFLAGS=-L$(HOMEBREW_LOC)/opt/gettext/lib -L$(LLVM_LOC)/lib -Wl,-rpath,$(LLVM_LOC)/lib  
CPPFLAGS=-I$(HOMEBREW_LOC)/opt/gettext/include -I$(LLVM_LOC)/include
```

Intel

```
HOMEBREW_LOC=/usr/local
LLVM_LOC=$(HOMEBREW_LOC)/opt/llvm
CC=$(LLVM_LOC)/bin/clang -fopenmp
CXX=$(LLVM_LOC)/bin/clang++ -fopenmp
CFLAGS=-g -O3 -Wall -pedantic -std=gnu99 -mtune=native -pipe
CXXFLAGS=-g -O3 -Wall -pedantic -std=c++11 -mtune=native -pipe
LDFLAGS=-L$(HOMEBREW_LOC)/opt/gettext/lib -L$(LLVM_LOC)/lib
    -Wl,-rpath,$(LLVM_LOC)/lib
CPPFLAGS=-I$(HOMEBREW_LOC)/opt/gettext/include -I$(LLVM_LOC)/include
```

```
Makevars — Modificado

HOMEBREW_LOC=/usr/local
LLVM_LOC=$(HOMEBREW_LOC)/opt/llvm
CC=$(LLVM_LOC)/bin/clang -fopenmp
CXX=$(LLVM_LOC)/bin/clang++ -fopenmp
CFLAGS=-g -O3 -Wall -pedantic -std=gnu99 -mtune=native -pipe
CXXFLAGS=-g -O3 -Wall -pedantic -std=c+11 -mtune=native -pipe
LDFLAGS=-L$(HOMEBREW_LOC)/opt/gettext/lib -L$(LLVM_LOC)/lib -Wl,-rpath,$(LLVM_LOC)/lib
CPPFLAGS=-I$(HOMEBREW_LOC)/opt/gettext/include -I$(LLVM_LOC)/include
```

Once you put the paths, save the text file and close it. The difference between Apple Silicon and Intel is just the path; everything else remains similar. For Apple Silicon, the path is `/opt/homebrew` and for Intel is `/usr/local`.

## **Seventh Step: Reinstall data.table and set your cores.**

Finally, we need to remove `data.table` and reinstall it via `source`.



```
remove.packages("data.table")
install.packages('data.table', type='source')
```

If everything was done correctly, you should load the package with library(data.table) and see the following message.

```
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
** checking absolute paths in shared objects and dynamic libraries
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (data.table)

The downloaded source packages are in
  '/private/var/folders/7r/cc5ch4n1yn8ny0c_nq8bny8000gn/T/RtmpY8sEUe/downloaded_packages'
> library(data.table)
data.table 1.14.6 using 4 threads (see ?getDTthreads). Latest news: r-ddatatable.com
```

Remember that you can set more threads with `setDTthreads` (according to the specifications of your computer). Happy coding!