Macport help

Justin Erwin

updated: January 9, 2017

Abstract

This is my guide to installing gfortran, MPI, and NumPy/SciPy/-Matplotlib on my Mac. I will use the package manager Macports for this since it will download, compile, and link any necessary software for my needs. It also installs the software in the in the */opt* folder and leaves */usr* alone so that the default Mac compilers and Python are unaffected.

For post-processing and plotting data, I now use Python and its suite of tools. In particular Matplotlib is an excellent replacement for MatLab-like plotting (but in my humble opinion better and more customizable), and NumPy and SciPy offer a suite of numerical procedures and special math function. IPython is a more user-friendly command line interface for Python, with nice feature like tab suggestions. Spyder is an IDE (like Matlab or IDL) that uses IPython as the command prompt and with Numpy, SciPy and MaplotLib loaded automatically for numerics and plotting.

Lastly I found Macports can be used to install Latex with a much smaller footprint then MacTex, and rolling the update into Macports update is convenient.

Contents

1 Install MacPort

This section may change slightly depending on your Mac version. I am now happily on macOS Sierra10.12 Mavericks and therefor the instruction below represent that install.

1.1 Xcode

First we need to get Xcode up an running. This is the development IDL for Apple's computer and mobile platforms. Before we can use the commandline for development we need:

1. Install Xcode from the Mac App Store.
2. Accept the Xcode license agreement. Simply open Xcode and click through the menus for it to be happy.
3. Install command line tools. This lets us use some of the packaged tools inside of Xcode from the command line, including clang, llvm, and probably others. New to Mavericks, this is done by entering the following into the terminal:

```
$ xcode-select --install
```

This last step will differ depending on you OS. In some previous versions on Mac, the command line tools were install via "Preferences→Downloads", so figure out how to work with your install.

1.2 MacPorts

Again just a couple simple steps here

1. Get the MacPorts installer (`http://www.macports.org/install.php`) and run it. This installs in the directory "opt/local/bin" well out of the way of macOS's parts. It also modifies our PATH variable, which allows us to access the "port" command as well as the other programs we will install soon.
2. Then run an update to get the most current list of ports (called the tree). Enter the following into the command line:

```
$ sudo port selfupdate
```

Notice the use of ”sudo”, and that it will require you to enter your password. This is because we are modifying files in the ”/opt/loca/” folder, which is not in out home directory. This is ”super user do”, and runs the following command with higher privileges (or root). Be careful with ”sudo”, and use only when you know what you’re doing since it can modify anything on your computer. Proceeding, we trust MacPorts has the kinks worked out.

Now your ready to get some ports. It should have automatically updated your path in your *.profile* file. You might also set some aliases here. For example, I use AquaMacs as a text editor, and I set up some common calls to ls, so I have the following lines in my *.profile*

```
# some aliases
alias ls='ls -GFh'
alias ll='ls -l'
alias la='ls -la'
alias amacs='open -a /Applications/Aquamacs.app/'
```

2 Get your compiler(s)

2.1 Install gcc

There are many versions of gcc available, and I see no reason not to just get the current one. But depending on your projects and code, you may want to download a specific one that is known to be supported for your code. Also consider that many other ports may depend on gcc, and may have many variants to support different versions of *gcc*. For example many Python modules depend on gcc, so you may want to look at the many python modules that you want and find that latest gcc version that they will support. Read the optional section below to set your configuration to have a default variant to avoid accidentally downloading multiple version of *gcc*.

I will proceed by installing gcc 4.9. Even though gcc version 5 or 6 might work for most of my packages, I stick with the tried and tested 4 series as the gains of these newer version (namely build time) are not so important for me.

```
$ sudo port install gcc49
```

Be prepared to sit back and wait. Depending on your computer this can take several hours of using 90 percent CPU. Just let it go and don't let the battery die.

Now select this compiler. First try:

```
$ gcc -v
    ->shows using Xcode's gcc version 4.2.1
$ gfortran -v
    ->doesn't work
```

So we are not yet using the new *gcc*. Lets see that its installed

```
$ port select --list gcc
Available versions for gcc:
        mp-gcc49
        none (active)
```

We see that we have two options, but haven't told MacPorts which one we want to use. The *none* option is the Apple supplied *gcc*, and the *mp-gcc48* is the new MacPorts *gcc*. To select the new one, type in

```
$ sudo port select --set gcc mp-gcc49
```

Test this

```
$ gcc -v
    ->shows using Macports' gcc 4.9.4
$ gfortran -v
    ->the same
```

Now your ready.

(NOTE: the 'port select –set' make symbolic links to the correct gcc executable. Sometimes these don't get updated immediately. If you see the result of 'gcc -v' doesn't change, try opening a new terminal and see if it works. It's amazing what turning a computer off and on again can fix.)

2.1.1 Set default MacPorts variant (optional)

Most ports in MacPorts depend on other ports, and further there are different variants we can add/choose from to set a particular dependent. The great thing about MacPorts is it keeps track of all these dependents and variants and will install exactly what you need. Unfortunately, when there are many different variants, there is also a default variant that has no clue about what we have already installed. For our purposes, I am concerned about which variant of *gcc4x* the python ports are going to use. (For instance, *py27-scipy* needs *gfortran* so it depends on a *gcc* port. The default variant for *py27-scipy* could be *gcc45*, and will download and build *gcc45* if your not careful.)

In order to get subsequent ports to use the *gcc49* port we installed we can add the *+gcc49* variant flag during the install. But checking every port that we want to install for variants, while probably good practice, can be cumbersome. There is a file */opt/local/etc/MacPorts/variants.config* that is the place to put the default variants we want to use, and will be applied to any port we try to install. If we add a variant and it doesn't exist for a particular port, then it will just be ignored.

But this file is protected by system privileges. You can use the *Finder*→*Get Info* to change the privileges and then modify, or you can just run the following command to add to end of the file.

```
$ echo "+gcc49" | sudo tee -a /opt/local/etc/
   macports/variants.conf
```

I also put "+openblas" to get the accelerate linear algebra routines, but it not necessary.

2.2 Install MPI

I like to have the ability to develop and test MPI codes on my laptop. There are two popular options, MPICH and OPENMPI, and for most users the differences are not apparent. So I choose OPENMPI because this is the one installed on my universities clusters, so this is the one I choose:

```
$ sudo port install openmpi
```

Since I set "+gcc49" in my variants.conf, I know it is compiling towards the correct compiler. Now you should have *mpif90* and *mpiexec* installed (look at */opt/local/bin* for other mpi function you might need).

3 Python

3.1 Install Python and modules

There are two parts the to install, first is the *python* port. I am choosing Python 2.7 since some scientific packages do not support Python 3.x yet (n.b. Python 3.4 is latest Python 3 version available via Macports and support all the components I list here. Know that the *print* syntax has change between Python 2 and Python 2→ frustrating, but this can be dealt with by using the __future__ module). So I choose the port *python27*.

Second are all the module add-ons for Python, and these are named *py27-** so that you know that they run on certain version of Python. To get everything we can just run

```
$ sudo port install python27 py27-numpy py27-
  scipy py27-matplotlib py27-ipython py27-
  notebook
```

Note that this will install many other python modules that these ones depend on, but don't worry the glory of MacPorts is that is handles all of this for you, it will just take some time. Be careful, since SciPy requires a *gcc* version for *gfortran*. So if you haven't set your *variants.config* file, you may download and install another compiler (this can increase the full build time considerably, and having two versions of gcc requires extra space and increases the update times). Notice that py27-ipython and py27-notebook were split into separate ports some time ago.

Like with gcc we need to select the new python installs as the default:

```
$ sudo port select --set python python27
$ sudo port select --set ipython ipython27
```

to set the Python/IPython versions to the new ones. Now you can run your python scripts using "python myscipt.py", which will invoke your freshly installed Macports python with all your installed packages.

Notes:

- If you enjoy the excellent Python notebooks and wish to share them, you may want to install "pandoc" to be able to save these a pdfs to print and share. "pandoc" is also very good to convert between many document types, including rst, doc, tex, etc.
- Numpy and Scipy use Blas and Lapack, in addition to gcc and gfortran, so they can take advantage of an accelerated framework. Atlas and

OpenBlas are two excellent options. Remember to add one to you default variants list.

- Matplotlib is excellent for plotting, but can also be use with different backends and to build gui applications. Look into the variants of this package. But don't worry, you can uninstall and reinstall with addition variants at a later time if you decide to take advantage of those.

3.2 Installing modules not in Macports

Some package we may want are not available in the Macports repository (for example some scientific packages made by collaborators). But there are other ways to install packages so that our Macports Python can use them, but we need to be careful about the installation path. We do not want to install packages in the same folder that Macports uses, namely:

```
/opt/local/Library/Frameworks/Python.framework
   /Versions/2.7/lib/python2.7/site-packages
```

Rather, we should install packages in our User directory in a place Python will look:

```
~/Library/Python/2.7/lib/python/site-packages
```

This way Macports will not mess with these installations, and vice versa.

3.2.1 Pip

Pip is a program written to install packages from the Python Package Index (PyPI) using *easy_install*. It has the ability to install necessary dependencies to build a package, much like Macports.

To install pip

```
$ sudo port install py27-pip
$ sudo port select --set pip pip27
```

Like many Python packages in Macports we need to select the active version.

Now we can install packages using *pip*:

```
$ pip install --user coveralls
```

Here *coveralls* is a package that is not available in Macports. The "--user" flag tells *pip* (and transitively *easy_install*) to install to the User *site-packages* folder. As far as dependencies, it will search for any dependencies to see if its installed already, and then it will download and install any needed packages.

In this case, it found the port *coverage* was already installed (by Macports into the /opt/local site-packages folder), and then it installed *docopt* and *PyYAML* packages needed by *coveralls* before installing *coveralls* itself (all of these to the User site-packages folder).

3.2.2 From source

We can install even more packages that may not be in the Macports or PyPI repositories if we have there source package and it has a setup.py script that uses *distutils* or *setuptools* (which themselves are used by pip). Neither of these will install dependencies for you, you must install them yourself before you try to install a package from source.

For example, I want to uses the package SpiceyPy available on Github. I get the source

```
$ cd ~/lib/python
$ git clone https://github.com/AndrewAnnex/
  SpiceyPy.git
$ cd SpiceyPy
```

Now when I check the *setup.py* file, we see it requires *numpy*, *pytest*, *coveralls*, *coverage*, and *six*. I check and I already have *numpy* and *six* (via Macports. I search Macports and it also has *pytest* and *coverage*, so I install these

```
$ sudo port install py27-pytest
$ sudo port install py27-coverage
```

Finally I install *coveralls* package via pip (like in the previous session)

```
$ pip install --user coveralls
```

At this point we have all the dependencies necessary to install SpicePy (and are still in its folder), and we install it to the User site-package folder as:

```
$ python setup.py install --user
```

Again the "--user" option tell the the python package installing to send the package to the user location.

Be sure to check dependencies, and be careful upgrading.

3.3 Spyder

Spyder is an IDE for python that uses python for the shell window and lets you watch variable grow and change. This can be very useful in development.

```
$ sudo port install py27-spyder
```

Now you can start up spyder from the by entering *spyder* at the command line. I am working on creating a *.app* to link to the Macports install.

3.3.1 Creating Spyder.app bundle

So some Macports packages create a ".app" package that is put in "/Applications/Macports/", which is then available in Launchpad (for instance TexShop and LaTeXit). We can create a simple .app package using Applescript editor.

1. Open *AppleScript Editor* (Application→Other→AppleScript Editor), and click "New Document".

2. Enter the following lines into the editor:

```
tell application "Terminal"
     do script "/opt/local/bin/spyder;exit"
end tell
```

3. Save this script, and remember to:

   - Give it a name, ex. "Spyder"
   - Put it in a temporary place, ex "∼/Desktop"
   - Change *File Format* to *Application*

4. (Optional) Change the icon

   - I got the spyder icon (a .icns file has icon of several sizes for OS X to use) from `https://spyderlib.googlecode.com/hg/img_src/`
   - Control click on your new *Spyder.app*, and click *Get Info*.
   - Drag your new icon to the icon in the top left of the info window (you should see a green plus sign as to hover over to show you are about to change the image).

5. Move *Spyder.app* to the */Application/Macports* folder, you will need to enter you password when trial to modify this folder. Now *Spyder.app* should appear in Launchpad. You can move it to where you like.

6. (optional) By default, the terminal will stay open after the "exit" call from the script. You can change this in "Terminal → Preferences → Settings → Shell" under "When shell exits" change this to "close"

So that's it. You should be able to adjust the above instructions for any other executable (from Macports or elsewhere). I'm not sure how Macports will handle me modifying the *Application/Macports* folder. Hopefully it will just ignore the new .app as it updates.

4 LaTeX

So the MacTex package installs everything one really needs to compile a latex file, including most available packages. Well I thought I might be clever and use MacPorts to trim some size off the install. Like Python there is the metaport called *texlive* and then all the packaged add ons *texlive-**. These add ons are generally collection of packages (and associated docs), so some will be needed to compile a document.

Now the *texlive* package has a few variants: *minimal*, *basic*, *medium*, and *full*. With *minimal* you only get the Tex engine, and with *basic* you get the Tex and Latex engines. The basic idea of *medium* is to give you suggested packages, and then *full* installs all the available package including documentation and is quite large. I find the *medium* variant is ∼1.4GB, and the *full* variant is ∼4.6GB. But the *medium* port could not compile my latest article that needed the els-article package, this was found in *texlive-publishers* port that was an additional 1GB (mostly due to its dependencies on *textlive-latex-extraa*). So depending on how precious space is to you, you can decide which one to use.

I chose the default (*medium*) and then installing necessary packages that I ended up needing. The website (`http://trac.macports.org/wiki/TeXLivePackages`) has a list of the packages included in the various *texlive-** ports that I used as a reference to find with port I needed. So my install went

```
$ sudo port install texlive
        :
        :
$ sudo port install texlive-publishers
```

Now you have the executables *tex*, *latex*, *pdflatex*, *gs*, among others (including *latex2ht* if you have found that to work for you).

4.1 Leave out documentation

Try compiling with *-doc* variant and the install will be a good deal smaller. This leaves out all the documentation stuff which I never seen before, so I guess I don't use it.

```
$ sudo port install texlive -doc
```

This install is now only ∼1.1GB (this is smaller than the MacTex installer!). I am happy...

In the end I added "-doc" to my *variants.conf* file as in Section 2.1.1, in this way ensuring documentation is left out by default, and I install everything I need using:

```
$ sudo port install texlive texlive-publishers
    texlive-science
```

4.2 Latex utilities

Now MacPorts even has TexShop (but not BibDesk, that's OK, use Papers).

```
$ sudo port install TeXShop3
```

This application then appear in LaunchPad (and in */Applications/MacPorts/* btw). Next you should open TexShop's preferences to tell TexShop to look in */opt/local/bin* for both *latex* and *gs*, but then you're good.

Also if you need a package or style files that is not in a port, or don't want to install a whole port, the */Library/texmf* tree is there for you.

5 Maintenance

5.1 Updating

Update is quite simple with MacPorts, just two steps. First update MacPorts so it knows the current set of ports. And second, update all of your installed ports (this may take some time depending on how often you do an update and what needs to be updated):

```
$ sudo port selfupdate
$ sudo port upgrade outdated
```

5.2 Basic cleaning

MacPorts tends not to uninstall old versions of ports that have been upgraded. So we can occasionally free up some space. First lets check to see how much space

```
$ du -sh /opt
6.8G    /opt/local
```

Now we can execute the following commands to clean up our ports

```
$ sudo port clean --all installed
$ sudo port -f uninstall inactive
```

Now check to see how much space you freed up.

```
$ du -sh /opt
6.5G    /opt
```

5.3 Checking Ports

If you want to see a list of ports that are installed

```
$ port echo installed
```

Well that was a whole bunch of junk. If you want to see the list of ports that you asked MacPorts to install, try

```
$ port echo installed and requested
```

Now if you want to check to see if any of your ports need to be update

```
$ port echo outdated
```

You can search, get info, and get detailed variant info on a port via

```
$ port search matplotlib
$ port info py27-matplotlib
$ port variants py27-matplotlib
```

5.4 Deleting a Port

Lets say your done with a port, or are upgrading to a newer version, or like me tried some variants that added a whole bunch of other ports to the installs. The basic command to uninstall a port is

```
$ port uninstall vile
```

Here *vile* is the name of the port we wish to uninstall. But this may leave a bunch of ports that were installed for the *vile* port that are not used by any other ports around. These are called *leaves* in MacPorts lingo. We can tell MacPorts to do a better by calling

```
$ sudo port uninstall --follow-dependencies
   vile
```

This will follow the port that *vile* depends on and uninstall them unless the port is used by another port. But this may not do a perfect job, maybe since some dependencies are used by other dependencies that weren't deleted first. We can check for more leaves by invoking

```
$ port echo leaves
```

If there are some leaves left, the go to advanced cleaning.

5.5 Advanced cleaning

If we have some leaves left around through updating or uninstalling, see a list of them.

```
$ port echo leaves
```

These are the ports that are not dependencies of any installed ports, but are not requested. Some of these ports are required for installation or updating, so we don't want to delete them, just to reinstall next time we update. So set these as requested with

```
$ sudo port setrequested autoconf automake
   pkgconfig
```

Now get to uninstalling the leaves, invoke:

```
$ sudo port uninstall leaves
```

Check for new leaves after this cleanup, and keep pruning until there are no leaves left.

5.6 Uninstalling

So maybe your unhappy with MacPorts, or, like me, you were playing around and want to restart clean. The following should clean up just about everything that MacPorts has put on your computer

```
$ sudo port -fp uninstall installed
$ sudo rm -rf \
    /opt/local \
    /Applications/DarwinPorts \
    /Applications/MacPorts \
    /Library/LaunchDaemons/org.MacPorts.* \
    /Library/Receipts/DarwinPorts*.pkg \
    /Library/Receipts/MacPorts*.pkg \
    /Library/StartupItems/DarwinPortsStartup \
    /Library/Tcl/darwinports1.0 \
    /Library/Tcl/MacPorts1.0 \
    ~/.MacPorts
```

A My installed ports

In case you interesting in all the packages that I have installed here they are:

```
$ port echo installed and requested
doxygen                         @1.8.13_1
gcc49                           @4.9.4_0
ghc                             @7.8.3_5
git                             @2.11.0_0+
   credential_osxkeychain+gitweb+pcre+perl5_24+
   svn
graphviz                        @2.40.1_0+
   pangocairo+x11
hdf5                            @1.10.0-
   patch1_0+cxx+gcc49+hl
libcerf                         @1.5_0
nmap                            @7.40_0+pcre+
   ssl+subversion
OpenBLAS                        @0.2.19_0+gcc49
   +lapack
openmpi                         @1.10.3_0
pandoc                          @1.12.4.2_1
py27-h5py                       @2.6.0_0+gcc49
py27-ipython                    @5.1.0_0
py27-matplotlib                 @1.5.3_0+latex+
   pyside+qt4
py27-notebook                   @4.3.1_0
py27-numpy                      @1.11.3_0+gcc49
   +openblas
py27-pip                        @9.0.1_0
py27-scipy                      @0.18.1_0+gcc49
   +openblas
py27-sphinx                     @1.5.1_0
python27                        @2.7.13_0
rsync                           @3.1.2_0
texlive                         @2016_0+medium
texlive-publishers              @41243_0
texlive-science                 @41041_0
TeXShop3                        @3.75_0
```

```
xorg-server                          @1.18.4_1
```

This represents the ports that I asked MacPorts to install. In all I have 392 ports installed to make these ones work, and such it the power of a package management system like MacPorts.

Ports not explained above:

As you can see I have a few more ports installed that I didn't explain above. I tried to give instruction for a minimalist development environment. After this, you should be able to find and install additional packages necessary for your needs.

doxygen This is a great tool to create documentation for your coding projects. For my large projects I can create interactive webpages with dependency graphs and detailed explanations for each function/subroutine and its inputs and outputs. The **graphviz** package is needed to create call graphs.

git I have fallen in love with this revision system. I use it took keep track of changes to my code, keep code updated between my computer and the computer clusters, and to share code development with collaborators.

hdf5 This installs the library to read and write to hdf5 files, a binary file format for storing large arrays, images, and other things in a compact binary form. To compile with this library I add to the following to my compilation: "-L/opt/local/lib" to show where the library is installed, "-I/opt/local/include" to show where the ".mod" files are installed (for Fortran), and "-lhdf5" to link to library file libhdf5.a in the before mentioned folder. The **py27-h5py** package is a python API to hdf5. Alternately, **py27-tables** also gives the ability to access HDF files and then some.

OpenBlas This is an accelerate tuned library for BLAS and LAPACK routines. They can be a bit faster for large numerical problems involving linear systems (think matrices). Another option is **ATLAS**, but after some. admittedly brief, googling I settled on this one.

pandoc This is a capable document conversion tool. It can be used by **py27-notebook** to save python notebooks as pdf's for printing and sharing. It can also be used to precess .md or .tex files into other formats, like .html or .docx.

py27-sphinx The standard documentation tool for python projects. Like **doxygen** it can create nice documentation pages for your python projects. Once upon a time I tried to use this for my Fortran projects as well, but doxygen has better support (even if it is meant for c).

rsync This was installed as a dependent of git, therefore not in the list above, but is “a fast, versatile, remote (and local) file-copying tool”. It’s a very nice tool to learn to get data back and forth between a local and remote machine.

xorg-server This installs the XQuartz package, which include X11 support for terminal. This is important if you plan to make a graphical connection to a remote machine (ie ”ssh -X”).