Biopython Installation

Brad Chapman (chapmanb@uga.edu)

September 26, 2006

Contents

1 Purpose and Assumptions **2**

2 Installing Python **2**
2.1 Installation on UNIX systems and Mac OS X . . . 2
2.1.1 RPM and other Package Manager Installation . . . 3
2.2 Installation on Windows . . . 3
2.3 Installation on older Macintoshes . . . 3

3 Installing Biopython dependencies **3**
3.1 mxTextTools . . . 3
3.1.1 UNIX and Mac OS X systems . . . 4
3.1.2 Windows systems . . . 4
3.1.3 Making sure it installed correctly . . . 4
3.2 Numerical Python . . . 4
3.2.1 UNIX and Mac OS X systems . . . 5
3.2.2 Windows systems . . . 5
3.2.3 Making sure it installed correctly . . . 5
3.3 ReportLab (optional) . . . 5
3.3.1 UNIX and Mac OS X systems . . . 5
3.3.2 Windows systems . . . 6
3.3.3 Making sure it installed correctly . . . 6
3.4 Database Access (MySQLdb, ...) (optional) . . . 6

4 Installing Biopython **7**
4.1 Obtaining Biopython . . . 7
4.2 Installing on UNIX and Mac OS X . . . 7
4.2.1 Installation from source on UNIX and Mac OS X . . . 7
4.2.2 Installation on FreeBSD . . . 8
4.2.3 Installation on Mac OS X using the fink package manager . . . 8
4.2.4 Installation on UNIX systems using RPMs . . . 8
4.3 Installing with a Windows Installer . . . 9
4.4 Installing from source on Windows . . . 9
4.5 Installation on older Macintoshes . . . 9
4.5.1 Pre-Built Install . . . 10
4.5.2 Non Pre-Built Installation . . . 10

5 Making sure everything worked **10**

6 Notes for installing with non-administrator permissions **11**

1 Purpose and Assumptions

This document describes installing Biopython on your computer. To make things as simple as possible, it basically assumes you have nothing related to Python or Biopython on your computer and want to end up with a working installation of Biopython when you are finished following through this documentation.

Biopython should work on just any operating system where Python works, so these instructions contain directions for installation on UNIX/Linux , Windows and Macintosh machines. The directions assume that you have permission to install programs on the machine (root access on UNIX and Administrator privileges on Windows or Mac machines). While it is certainly possible to install things without these privileges, this is a serious pain and all the tedious workarounds aren't something that I'll go into very much in this documentation.

With all this said, hopefully these directions will make it straightforward to get Biopython installed on your machine so you can begin using it as quick as possible.

2 Installing Python

Python is a interpreting, interactive object-oriented programming language and the home for all things python is http://www.python.org. Presumedly you have some idea of python and what it can do if you are interested in Biopython, but if not the python website contains tons of documentation and reasons to learn to program in python.

Biopython is designed to work with Python 2.3 or later. With python, the general rule of thumb is to keep yourself using the latest version, as the development process is very clean and new releases are quite stable. Upgrading bug-fix releases (for example. 2.3.1 to 2.3.2) is incredibly easy and won't require any re-installation of libraries. Upgrading between versions (2.3 to 2.4) is more time consuming since you need to re-install all libraries you have added to python.

Let's get started with installation on various platforms.

2.1 Installation on UNIX systems and Mac OS X

First, you should go the main python web site and head over to the information page for the latest python release. At the time of this writing the latest stable python release is 2.4, which is available from http://www.python.org/2.4/. This page contains links to all released files for the given release. For UNIX, we'll want to use the tarred and gzipped file, which is called `Python-2.4.tgz` at the time of this writing.

Download this file and then unpack it with the following commands:

```
$ gunzip Python-2.4.tgz
$ tar -xvpf Python-2.4.tar
```

Then enter into the created directory:

```
$ cd Python-2.4
```

Now, start the build process by configuring everything to your system:

```
$ ./configure
```

Build all of the files with:

```
$ make
```

Finally, you'll need to have root permissions on the system and then install everything:

```
# make install
```

If there were no errors and everything worked correctly, you should now be able to type `python` at a command prompt and enter into the python interpreter:

```
$ python
Python 2.4 (#1, Dec  5 2004, 20:47:03)
[GCC 3.3.3] on cygwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

2.1.1 RPM and other Package Manager Installation

There are a multitude of package manager systems out there for which python is available. One popular one is the RPM (RedHat Package Manager) system. Each of these package managing systems has its own quirks and tricks and I certainly can't pretend to understand them all so I won't try to describe them all here.

However, there is one general point which it is important to remember when installing from any of these systems: you need to download and install the development packages for python. A number of distributions contain a "basic" python which contains libraries and enough stuff to run simple python programs. However, they do not contain the python libraries necessary to build third-party python applications (like Biopython and it's dependencies). You'll need to install these libraries and header files, which are often found in a separate package called `python-devel` or something similar.

2.2 Installation on Windows

Installation on Windows is most easily done using handy windows installers. As described above in the UNIX section, you should go to the webpage for the current stable version of Python to download this installer. At the current time, you'd go to http://www.python.org/2.4/ and download `Python-2.4.msi`.

The installer is an executable program, so you only need to double click it to run it. Then just follow the friendly instructions. On all newer Windows machines you'll need to have Administrator privileges to do this installation.

2.3 Installation on older Macintoshes

Mac OS X can readily compile the python distribution in the way described above for UNIX systems, but earlier versions of the Macintosh operating system aren't nearly as easy to work with. For these versions, the best place to go is the Macintosh page on the python site: http://www.python.org/download/download_mac.html. This site links to the MacPython pages, which contain installers for Macintosh.

The Macintosh installer is as simple to use at the Windows installers. You download the appropriate file (`MacPython222active.bin` at the current time), and then unpack it with StuffIt Expander. You then just need to double click on the resulting graphical installer and follow the easy instructions.

3 Installing Biopython dependencies

Once python is installed, the next step is getting the dependencies for Biopython installed. Since not all functionality is included in the main python installation, Biopython needs some support libraries to save us a lot of work re-writing code that already exists. We try to keep as few dependencies as possible to make installation as easy as possible.

3.1 mxTextTools

This is the most important Biopython dependency as it is used extensively in the internals of a number of parsers. You absolutely want to install this if you want to get any sort of serious use out of Biopython.

mxTextTools is available along with the entire mx-base system (which contains a number of other useful utilities as well) and is available for download at: http://www.lemburg.com/files/python/mxTextTools.html.

3.1.1 UNIX and Mac OS X systems

For UNIX and UNIX-like systems you should download the `tar.gz` file from the page listed above. At the current time, this is `egenix-mx-base-2.0.6.tar.gz`.

Once you download this, unpack it and change into the created directory:

```
$ gunzip egenix-mx-base-2.0.6.tar.gz
$ tar -xvpf egenix-mx-base-2.0.6.tar
$ cd egenix-mx-base-2.0.6
```

To build it, use the standard python build procedure:

```
$ python setup.py build
```

Then become root, and install it, again using the standard python mechanism:

```
$ python setup.py install
```

3.1.2 Windows systems

For Windows operating systems, you should download the Windows installer for the version of python you are running. At the current time, this would be: `egenix-mx-base-2.0.6.win32-py2.4.exe`.

This is a standard graphical installer, so after download double click it and follow the instructions and it should install with no problem. You'll have to have Administrator privileges to do this install, as with python itself.

3.1.3 Making sure it installed correctly

If you've installed mxTextTools correctly, you should be able to fire up your python interpreter and import it with no errors:

```
$ python
Python 2.4 (#1, Dec  5 2004, 20:47:03)
[GCC 3.3.3] on cygwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from mx import TextTools
>>>
```

3.2 Numerical Python

The Numerical Python distribution (also known an Numeric or Numpy) is a fast implementation of arrays and associated array functionality. This is important for a number of Biopython modules that deal with number processing. The main web site for Numeric is: http://sourceforge.net/projects/numpy and downloads are available from: http://sourceforge.net/project/showfiles.php?group_id=1369&package_id=1351.

3.2.1 UNIX and Mac OS X systems

As with mxTextTools, you should download the `tar.gz` file. Version 23.8 of Numerical Python (`Numeric-23.8.tar.gz`) compiles out of the box. The build process is exactly the same as with mxTextTools:

```
$ gunzip Numeric-23.8.tar.gz
$ tar -xvpf Numeric-23.8.tar
$ cd Numeric-23.8
$ python setup.py build
```

Once it is built, you should become root, and then install it:

```
$ python setup.py install
```

One important note if you use an RPM-based system and not installing from source as described above: you need to also install the Numeric header files which are not included with some Numeric packages. As with the main python distribution, this means you'll need to look for something like `python-numeric-devel` and make sure to install this as well as the basic Numeric package.

3.2.2 Windows systems

Once again, Windows installers are available for Numeric so you should follow the now-standard procedure of downloading the installer (`Numeric-23.8.win32-py2.4.exe` at the current time), double clicking it and then following the installation instructions. As before, you will need to have administrator permissions to do this.

3.2.3 Making sure it installed correctly

To make sure everything went okay during the install, fire up the python interpreter and ensure you can import Numeric without any errors:

```
$ python
Python 2.4 (#1, Dec  5 2004, 20:47:03)
[GCC 3.3.3] on cygwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from Numeric import *
>>>
```

3.3 ReportLab (optional)

The ReportLab package is a library for generating PDF documents. It is used in the Biopython Graphics modules, which contains basic functionality for drawing biological objects like chromosomes. If you are not planning on using this, installing ReportLab is not necessary. ReportLab in itself is very useful for a number of tasks besides just Biopython, so you may want to check out http://www.reportlab.org before making your decision.

The main download page for ReportLab is http://www.reportlab.org/downloads.html. The ReportLab company has some commercial products as well, but just scroll down their page to the Open Source software section for the base ReportLab downloads.

3.3.1 UNIX and Mac OS X systems

For UNIX installs, you should download the tarred and gzipped version of the ReportLab distribution. At the time of this writing, this is called `ReportLab_1_20.tgz`. First, unpack the distribution and change into the created directory:

```
$ gunzip ReportLab_1_20.tgz
$ tar -xvpf ReportLab_1_20.tar
$ cd reportlab_1_20/
```

Once again, ReportLab uses the standard python installation system which you are probably feeling really comfortable with by now. So, first build the package:

```
$ python setup.py build
```

Now become root, and install it:

```
$ python setup.py install
```

3.3.2 Windows systems

ReportLab does not have a graphical windows installer like the other Biopython requirements. Luckily, it doesn't require any compilation steps to work properly, so the installation is still quite easy.

First, download the zipped distribution from the download site listed above. At the current time this is called `ReportLab_1_20.zip`. You can also download the tarred/gzipped file (with a `.tgz` extension), but Windows handles zipped files better.

Secondly, unzip the downloaded file. WinZip is a common freely available program for doing this (http://www.winzip.com/ddchomea.htm). The unzip process should create a `reportlab` directory.

Finally, drag the created `reportlab` directory to the standard directory for Python extensions. On current versions of python with a standard installation this would be something like `C:/Python24/Lib/site-packages`. All you have to do is drag it over and you should be all set. Nice and easy.

3.3.3 Making sure it installed correctly

If reportlab is installed correctly, you should be able to do the following:

```
$ python
Python 2.4 (#1, Dec  5 2004, 20:47:03)
[GCC 3.3.3] on cygwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from reportlab.graphics import renderPDF
>>>
```

Depending on your version of python and what you have installed, you may get the following warning message: `Warn: Python Imaging Library not available`. This isn't anything to worry about since the Biopython parts that use ReportLab will work just fine without it.

3.4 Database Access (MySQLdb, ...) (optional)

The MySQLdb package is a library for accessing MySQL databases. It is used in the Biopython GFF module, which allows access to databases created from GFF feature tables. Biopython also includes an accessory module, DocSQL, which provides a convenient interface to MySQLdb. If you are not planning on using Bio.GFF or Bio.DocSQL, installing MySQLdb is not necessary.

Additionally, both MySQLdb and psycopg (a PostgreSQL database adaptor) can be used for accessing BioSQL databases through Biopython. Again if you are not going to use BioSQL, there shouldn't be any need to install these modules.

Installation instructions for MySQLdb and psycopg are included in the BioSQL documentation, which is available from http://www.biopython.org/docs/biosql/python_biosql_basic.html and http://www.biopython.org/docs/biosql/python_biosql_basic.pdf.

4 Installing Biopython

4.1 Obtaining Biopython

Biopython's internet home is at, naturally enough, http://www.biopython.org. This is the home of all things Biopython, so it is the best place to start looking around. You have two choices for obtaining Biopython:

1. Release code – We made available releases on the download page (http://www.biopython.org/download/). The releases are also available both as source and as installers (windows installers right now), so you have some choices to pick from on releases if you prefer not to deal with source code directly.

2. CVS – The current working copy of the Biopython sources is always available via CVS (Concurrent Versions Systems – http://www.cvshome.org/). Concise instructions for accessing this copy are available at http://cvs.biopython.org. CVS is normally quite stable but there is always the caveat that the code there is under development.

Based on which way you choose, you'll need to follow one of the following installation options. Read on for the platform you are working on.

4.2 Installing on UNIX and Mac OS X

4.2.1 Installation from source on UNIX and Mac OS X

Biopython uses Distutils, the standard python installation package, for its installation. If you read the install instructions above you are already quite familiar with its workings. Distutils comes standard with Python 1.6 and beyond.

Now that we've got what we need, let's get into the installation:

1. First you need to unpack the distribution. If you got the CVS version, you are all set to go and can skip on ahead. Otherwise, you'll need to unpack it. On UN*X machines, a tar.gz package is provided, which you can unpack with `tar -xzvpf biopython-X.X.tar.gz`. A zip file is also provided for other platforms.

2. Now that everything is unpacked, move into the `biopython*` directory (this will just be `biopython` for CVS users, and will be `biopython-X.X` for those using a packaged download).

3. Now you are ready for your one step install – `python setup.py install`. This performs the default install, and will put Biopython into the `site-packages` directory of your python library tree (on my machine this is `/usr/local/lib/python2.4/site-packages`). You will have to have permissions to write to this directory, so you'll need to have root access on the machine.

   (a) This install requires that you have the python source available. You can check this by looking for `Python.h` and `config.h` in some place like `/usr/local/include/python2.4`. If you installed python with RPMs or some other packaging system, this means you'll also have to install the header files. This requires installing the python development libraries as well (normally called something like `python-devel-2.4.rpm`).

   (b) The distutils setup process allows you to do some customization of your install so you don't have to stick everything in the default location (in case you don't have write permissions there, or just want to test Biopython out). You have quite a few choices, which are covered in detail in the distutils installation manual (http://www.python.org/sigs/distutils-sig/doc/inst/inst.html), specifically in the Alternative installation section. For example, to install Biopython into your home directory, you need to type `python setup.py install --home=$HOME`. This will

install the package into someplace like `$HOME/lib/python2.4/site-packages`. You'll need to subsequently modify the `PYTHONPATH` environmental variable to include this directory so python will be able to find the installation.

4. That's it! Biopython is installed. Wasn't that easy? Now let's check and make sure it worked properly. Skip on ahead to section 5.

4.2.2 Installation on FreeBSD

Johann Visagie has been kind enough to create (and keep updated) a FreeBSD port of Biopython. Thanks to the wonders of the ports system, this means that all you need to do to install Biopython on FreeBSD is do the following as root:

```
cd /usr/ports/biology/py-biopython
make install
```

And voila! It's installed.

If you want more information on FreeBSD and things, Johann has written a nice primer for his FreeBSD EMBOSS port. This has lots of generally useful information, such as how to keep your ports tree up to date. If you are new to FreeBSD, you should definitely check it out at `ftp://ftp.no.embnet.org/pub/EMBOSS-extras/EMBOSS-FreeBSD-HOWTO.txt`.

4.2.3 Installation on Mac OS X using the fink package manager

Instead of installing from source, on Mac OS X you can also use the fink package manager, see `http://fink.sf.net`. Fink will take care of downloading the source code and installing all needed packages for biopython, including python itself. Once you have installed fink, you can install biopython using:

```
fink install biopython-pyXX
```

where XX is the python version you would like to use. Currently, python 2.3, 2.4, and 2.5 are available through fink on Mac OS X 10.4, so you would have to replace XX with 23, 24, or 25, respectively. Most likely, you will have to enable the unstable tree of fink in order to install the most recent versions of the package, see also this item in the Fink FAQ: `http://fink.sourceforge.net/faq/usage-fink.php#unstable`. Note that 'unstable' doesn't mean that a package won't work, but only that there has not been feedback to the fink team from users.

4.2.4 Installation on UNIX systems using RPMs

Warning. Right now we're not making RPMs for biopython (because I stopped using an RPM system, basically). If anyone wants to pick this up, or feels especially strongly that they'd like RPMs, please let us know.

To simplify things for people running RPM-based systems, biopython can also be installed via the RPM system. Additionally, this saves the necessity of having a C compiler to install biopython.

Installing Biopython from a RPM package should be much the same process as used for other RPMs. If you need general information about how RPMs work, the best place to go is `http://www.rpm.org`.

To install it, you should just need to do:

```
rpm -i your_biopython.rpm
```

To see what you installed try doing `rpm -qpl your_biopython.rpm` which will list all of the installed files.

RPMs do not install the documentation, tests, or example code, so you might want to also grab a source distribution, so you can use these resources (and also look at the source code if you want to).

4.3 Installing with a Windows Installer

Installing things on Windows with the installer should be really easy (hey, that's why they've got graphical installers, right?). You should just need to download the `Biopython-version.exe` installer from biopython web site. Then you just need to double click and voila, a nice little installer will come up and you can stick the libraries where you need to. No need for a C compiler or anything fancy. You will need to have Administrator privileges on the machine to do the installation.

This does not install the documentation, tests, example code or source code, so it is probably also a good idea to download the zip file containing this so you can test your installation and learn how to use it.

4.4 Installing from source on Windows

This section deals with installing the source (i. e. from CVS or from a source zip file) on a Windows machine. Much of the information from the UNIX install applies here, so it would be good to read section 4.2 before starting. Also, a little warning – I (Brad) am writing these instructions based on very limited experience with Windows; I am basically a UNIX geek. So if you know more about Windows and want to add/correct things in this section, please feel let us know!

I have successfully managed to use distutils to compile Biopython with Borland's free C++ compiler (available from http://www.inprise.com/bcppbuilder/freecompiler/). It should also be possible with other Distutils supported compilers (please provide info if you've done this!).

1. Borland C++ compiler

   - First you need to install and setup Borland C++. There are instructions on the Borland page and on the web, so you should follow these.
   - Now you have to get python, which is compiled with a Microsoft compiler, able to live happily with Borland compiled extensions. Gordon Williams has an excellent page describing doing this (http://www.cyberus.ca/~g_will/pyExtenDL.shtml), which is where I learned everything I know. Basically, what I did was run the Borland supplied tool `COFF2OMF` on the python20.lib file:

     ```
     > COFF2OMF python20.lib python20-borland.lib
     ```

     Then I just renamed `python20.lib` to `python20-orig.lib` and renamed `python20-borland.lib` to `python20.lib`, so that distutils will link against this "Borland friendly" python library.
   - Now that that is ready, we are at the easy part – using distutils to build it. All I did was:

     ```
     > python setup.py build --compiler=bcpp
     > python setup.py install
     ```

     and voila!, it's installed.

Now that you've got everything installed, skip on ahead to section 5 to make sure everything worked.

4.5 Installation on older Macintoshes

This section describes installation on pre-OS X machines. On OS X Biopython can be installed using the UNIX instructions.

Biopython code should work on Pre-OS X Macintoshes, using the MacPython distribution. I (Brad) am not a big Mac user, but have had good luck using several on the modules on the Macintosh.

4.5.1 Pre-Built Install

Yair Benita has been kind enough prepare pre-compiled and ready to go binaries for Macintosh machines. These distributions also contain the required Biopython libraries, so you can get everything installed all at once.

These builds are available from the Biopython download page at `.sit` files. You need to download this file and unpack it with StuffIt Expander. This will create a `MacBiopython-version` directory. You then just need to drag the contents of this directory to a standard location python searches (something like `Macintosh HD::Python2.2::Lib::site-packages`) and it's all installed.

4.5.2 Non Pre-Built Installation

If you don't want to use the pre-built releases, you can get some basic functionality from Biopython without compiling anything. You need to download either the `biopython-version.tar.gz` or `biopython-version.zip` file from the download page, and unpack these. This can be done with tools such as Aladdin's Stuff-It expander. It will unpack into a directory called `biopython-version`. If you open up this directory, you will find the main directory of modules, called `Bio`. You should then open up your python installation (which should be in some place like `Macintosh HD::Python2.2`) to the directory `Lib::site-packages`, and copy the `Bio` directory there by dragging it. Bam! You're done! By default, `site-packages` is included in your `PYTHONPATH`, so you should be ready to use it.

Some notes: Obviously this will not compile any of the C extensions in biopython. There are pure python implementations of all of these extensions, though, so you shouldn't need to worry about lack of functionality, only lack of speed. Jack Jansen (the MacPython god) has made patches to distutils which allow it to work on the Mac with the Metrowerks CodeWarrior compiler. I don't have this compiler (it costs money, oh no!), so I can't speak of how well it works. If anyone who codes more on the Mac has more information, I would be very happy to include it here.

5 Making sure everything worked

First, we'll just do a quick test to make sure Biopython is installed correctly. The most important thing is that python can find the biopython installation. Biopython installs into top level `Bio`, `Martel` and `BioSQL` directories, so you'll want to make sure these directories are located in a directory specified in your `$PYTHONPATH` environmental variable. If you used the default install, this shouldn't be a problem, but if not, you'll need to set the `PYTHONPATH` with something like `export PYTHONPATH = $PYTHONPATH':/directory/where/you/put/Biopython'` (on UNIX). Now that we think we are ready, fire up your python interpreter and follow along with the following code:

```
$ python
Python 2.4 (#1, Dec  5 2004, 20:47:03)
[GCC 3.3.3] on cygwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from Bio.Seq import Seq
>>> from Bio.Alphabet.IUPAC import unambiguous_dna
>>> new_seq = Seq('GATCAGAAG', unambiguous_dna)
>>> new_seq[0:2]
Seq('GA', IUPACUnambiguousDNA())
>>> from Bio import Translate
>>> translator = Translate.unambiguous_dna_by_name["Standard"]
>>> translator.translate(new_seq)
Seq('DQK', HasStopCodon(IUPACProtein(), '*'))
>>>
```

If this worked properly, then it looks like Biopython is in a happy place where python can find it, so now you might want to do some more rigorous tests. The `Tests` directory inside the distribution contains a number of tests you can run to make sure all of the different parts of biopython are working. These should all work just by running `python test_WhateverTheTestIs.py`.

You can also run all of the tests using a nice graphical interface supplied by using PyUnit. To do this, you just need to be in the installation directory and type:

```
python setup.py test
```

This should start up a Tk based graphical user interface (or default to the command line if you don't have Tkinter installed), which you can run the tests from. You can also run them by typing `python run_tests.py` in the Tests directory.

If you've made it this far, you've gotten Biopython installed and running. Congratulations!

6 Notes for installing with non-administrator permissions

Although I mentioned above that I wouldn't go much into installing in non-root directories, if you are stuck installing Biopython and it's dependencies into your home directory here are a few notes and tricks to keep you going:

- The Numeric package will take a little trickery to get it to be able to import. It will install a `Numeric` directory and `Numeric.pth` file into `your_dir/lib/python`. The problem is that the `Numeric.pth` file only works when located in the main python `site-packages` directory. The solution, fortunately, is simple – you need to make sure `your_dir/lib/python` is in the `PYTHONPATH` and then need to make a `__init__.py` file in the `Numeric` directory (an empty file or a file with just a comment is fine, as long as it exists). Then `import Numeric` should work.
- Building some C modules, such as `Bio.Cluster` require that the Numeric include files (normally installed in `your_dir/include/python/Numeric`) be available. If the compiler can't find these directories you'll normally get an error like:

  ```
  Bio/Cluster/clustermodule.c:2: Numeric/arrayobject.h: No such file or directory
  ```

  Followed by a long messy list of syntax errors. To fix this, you'll have to edit the `setup.py` file to let it know where the include directories are located. Look for the line in `setup.py` that looks like:

  ```
  include_dirs=["Bio/Cluster"]
  ```

  and adjust it so that it includes the include directory where the numeric libraries were installed:

  ```
  include_dirs=["Bio/Cluster", "your_dir/include/python"]
  ```

  Then you should be able to install everything happily.

Yes, it's a bit of a mess installing lots of packages in non-standard locations. The best solution is to talk with your friendly system administrator and get them to assist with the installation of at least the required packages (they are generally quite useful for any python install) before going ahead with Biopython installation.