Linux for Biologists

A Cookbook

Vimalkumar Velayudhan

Linux for Biologists

A Cookbook

Vimalkumar Velayudhan

First edition
July 14, 2021

Contents

1

Getting software on Linux

In this section, I will discuss how you can get software on your Linux desktop.

For any software you would like to install, you can start with the quick and easy method. It will install software available in Linux distribution repositories.

If your software of interest is not available or if you need a different version, you can consider using the other methods.

1.1 Python packages

If your software of interest is available as a Python package on *PyPI*, you can install it using *pip* — Python package installer.

Note: Your software might already be available using ../quick-and-easy/index. Search for package names starting with `python-`.

In certain cases, installing packages in a virtual environment might be a better option. Read:

-> *When should I use a virtual environment?*

1.1.1 Requirements

Python package installer (pip)

To install pip, follow ../quick-and-easy/index for installing software.

Search and install the `python3-pip` package.

$HOME/.local/bin added to $PATH

Packages that include commands will install them in the `$HOME/.local/bin` directory (What is $HOME?).

To be able to run these commands easily, you will need to add this directory to your `$PATH` variable. You can do so by following the steps in adding-directories-to-path.

Attention: *You should not add sudo in the commands below.*

This method only installs files in your home directory and so does not require administrator privileges.

1.1.2 Searching for a package on PyPI

Open PyPI website[1] in a web browser.

In the Search projects field, enter the name of the software you would like to install and press the ENTER key or click on the search button (Fig. 1).

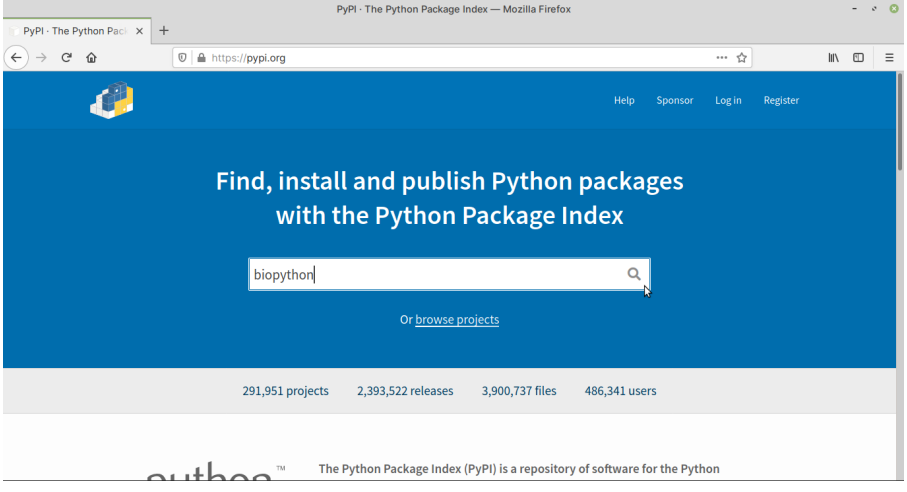

Fig. 1: Searching for a package on PyPI

A list of packages matching the search term will be displayed (Fig. 2).

[1] https://pypi.org

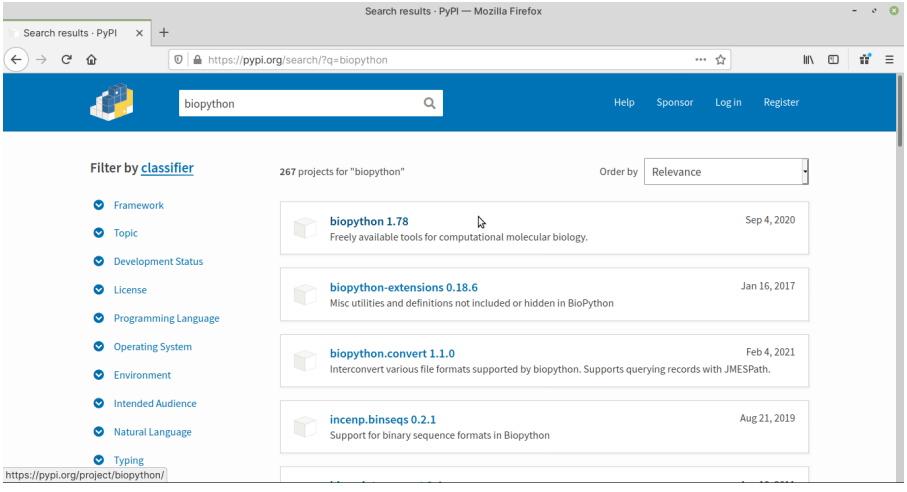

Fig. 2: Search results for biopython. 1.78 is the version number.

Click on the result to proceed to the project description page.

In the project description page, find the package name (Fig. 3) in the `pip install` command.

In this case, it is `biopython`.

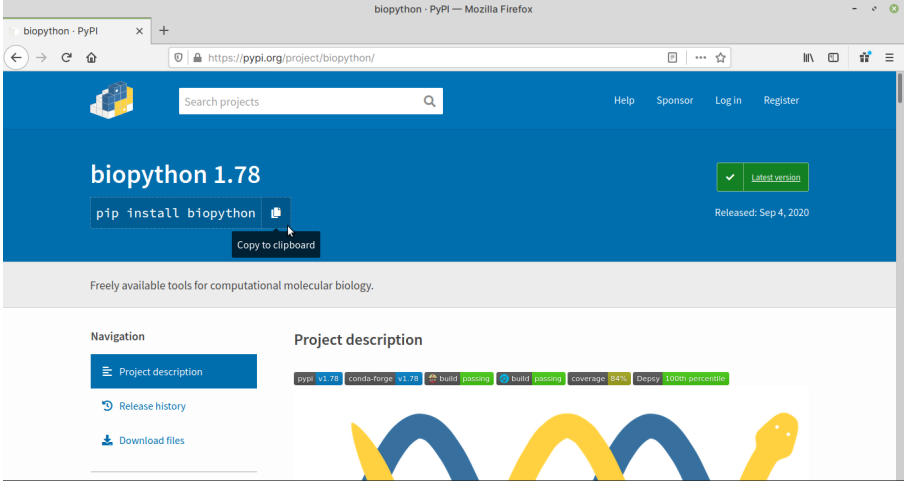

Fig. 3: Project description page for Biopython

Note: You will need to use `pip3` instead of the `pip` command in the steps below. *Why?*

You can now proceed towards installing the package.

1.1.3 Installing a Python package

Open a terminal.

Use the `pip3 install` command with by the name of the package, you would like to install.

Using biopython as an example:

```
pip3 install biopython
```

Output:

```
 Collecting biopython
  Downloading biopython-1.78-cp38-cp38-manylinux1_x86_64.
↪whl (2.3 MB)
     |********************************| 2.3 MB 2.1 MB/s
Collecting numpy
  Downloading numpy-1.20.1-cp38-cp38-manylinux2010_x86_64.
↪whl (15.4 MB)
     |********************************| 15.4 MB 90 kB/s
Installing collected packages: numpy, biopython
Successfully installed biopython-1.78 numpy-1.20.1
```

Note: One limitation of this method is that, you will not be able to install *multiple versions* of a package. This can be solved using a *virtual environment*.

1.1.4 Updating an installed package

When you notice an update is available for the package, you can use the `install` command with the `-U` option, and the name of the package:

```
pip3 install -U biopython
```

1.1.5 Removing an installed package

Use the `uninstall` command with the name of the package you would like to remove:

```
pip3 uninstall biopython
```

1.1.6 Using installed packages

Where are the files installed?

You can use the `pip3 show` command with the name of the package to identify the path where it is installed:

```
pip3 show biopython
```

Output:

```
Name: biopython
Version: 1.78
Summary: Freely available tools for computational
↪molecular biology.
Home-page: https://biopython.org/
Author: The Biopython Contributors
Author-email: biopython@biopython.org
License: UNKNOWN
Location: /home/user/.local/lib/python3.8/site-packages
Requires: numpy
Required-by:
```

The installation path will be listed next to the `Location` keyword.

Using Python packages and modules

Python packages and modules will be installed in `$HOME/.local/lib/python3.8/site-packages`.

Note: This is the path for Python 3.8.

If you have a different version of Python installed, this value will change.

This directory will be automatically included in `$PYTHONPATH`. So, you can import and use installed packages and modules in your scripts, without any extra effort.

For example, here is a simple Python script to test BioPython installed using pip:

```
from Bio.Seq import Seq

seq = Seq('ATGC')
comp = seq.complement()

print(f'The complement of {seq} is {comp}')
```

The `Bio.Seq` package is part of BioPython. Copy the code sample above and save it as `biopy_test.py`. Then run it from the terminal like so:

```
python3 biopy_test.py
```

Output:

```
The complement of ATGC is TACG
```

Using included commands

If the package includes commands, those will be installed in `$HOME/.local/bin` directory. For these commands to be easily accessible, you will need to add this directory to `PATH`, as mentioned under *requirements*.

As an example, when you install the Python package of *cutadapt*, the `cutadapt` command will be installed in `$HOME/.local/bin`, which you can then run from a terminal like this:

```
cutadapt --version
```

Output:

```
3.1
```

1.1.7 Python virtual environments

A virtual environment is a *self-contained* directory tree containing Python and some additional packages.

Advantages

These are *some* advantages of using virtual environments.

No administrator privileges

You do not need administrator privileges to create a virtual environment or install packages in an environment.

Multiple environments

Multiple virtual environments can be created with each containing their own sets of packages.

These are isolated and packages in an environment can be installed, updated or removed without affecting other environments.

Share your environment

You can share your configuration with others. They will be able to reproduce your environment, with the exact versions of packages.

Creating a virtual environment

First, using ../quick-and-easy/index, search and install the `python3-venv` package. This includes the `venv` module necessary for creating virtual environments.

You can create a virtual environment in any directory where you have write privileges, for example, your home directory.

To demonstrate, I will create a virtual environment called `py3env` in my home directory.

```
python3 -m venv py3env
```

If successful, you will find a directory named `py3env` in the current directory. No messages will be displayed.

Note: `python3` is the command to run the Python 3 interpreter. Its complete path is `/usr/bin/python3`.

`venv` is the Python module to create virtual environments.

The `-m` option runs the `venv` module as a script.

Before you can use a virtual environment, you will need to activate it.

Activating a virtual environment

You will need to activate a virtual environment before you can start using it. To do so, use the `source` command with the path to the virtual environment's `activate` script.

For example, to activate `py3env` created in the *previous step*, use:

```
source py3env/bin/activate
```

Your shell prompt will now change to indicate that the virtual environment is now active. Note the `(py3env)` label at the beginning of the prompt:

```
(py3env) user@cookbook:~$
```

You can now start using this virtual environment.

Note: *Before you start installing packages...*

It is a good idea to install (or upgrade) Python build tools — *pip*, *setuptools*, and *wheel* in a new virtual environment.

These build tools are necessary for building and installing packages from *PyPI* and other sources.

Installing them will ensure that additional packages will build and install without errors.

Installing Python build tools

Use `pip3 install` to install or upgrade the required packages:

```
pip3 install -U pip setuptools wheel
```

The `-U` option of `pip3 install`, will upgrade listed packages, if newer versions are available.

Output:

```
Collecting pip
Downloading pip-21.0.1-py3-none-any.whl (1.5 MB)
...
Installing collected packages: pip, setuptools, wheel
...
Successfully installed pip-21.0.1 setuptools-54.2.0 wheel-
↪0.36.2
```

Deactivating a virtual environment

To exit a virtual environment, use the command:

```
deactivate
```

Your shell prompt will change to its original appearance:

```
user@cookbook:~$
```

1.1.8 Notes

You will need to update these packages manually

Packages installed in this manner should also be updated manually.

When you notice there is an update for the package, for example, from the project's website or from their source code repository, follow the steps in *Updating an installed package* to install the latest version.

Why pip3 and not pip?

The *pip* package includes the following commands:

- `pip3` — Python 3 version
- `pip` — Python 2 version

Support for Python 2 ended in January 2020.

Since there is a possibility for both commands to exist on a system, it is safer to use `pip3` when installing packages using this method.

What about programs written in Python 2?

Support for Python 2 ended in January 2020.

If you do need to use a program written only in Python 2, you can create an isolated environment — either using Python venv or Conda and then install the package there.

Related sections:

- *Python virtual environments*
- ../conda/index

When should I use a virtual environment?

The method described here will not work if the programs you are installing require two different versions of the same package from PyPI.

In that case, you can consider creating an isolated environment — either using Python virtualenv or Conda and then installing the packages there.

Related sections:

- *Python virtual environments*
- ../conda/index

Older versions of pip

If the version of pip installed in your system is older than 20.0, it will attempt to install packages in system paths by default, resulting in *permission denied* errors.

To avoid that, you will need to add the `--user` option to the `install` and `uninstall` commands, for example:

```
pip3 install --user biopython
```

A better approach is to *upgrade* your local version of pip. Once upgraded, you will no longer need to user `--user`.

To upgrade pip, do:

```
pip3 install --user -U pip
```

You can check the installed version of pip using:

```
pip3 -V
```

2 Glossary

Anaconda A free and open-source (Anaconda Individual Edition) Python and R distribution. It includes *Conda* and more than 250 open-source scientific packages. Additional packages can be installed from *Anaconda Cloud* repositories.

Website: https://www.anaconda.com/products/individual

Anaconda Cloud Repository for Python and R packages and notebooks.

Website: https://anaconda.org

apt A command-line program that handles the installation and removal of software on Debian, Ubuntu and other Linux distributions.

Related section:

-> apt-install-software

bioconda A channel for *Conda* package manager with over 7000

packages of bioinformatics software.

Website: https://bioconda.github.io/

build-essential A package that installs software build tools like `make` and compilers like `gcc`, which are required for building modules written in languages like C.

Bash *GNU* project's shell program. The name stands for The Bourne-Again Shell. It is the most commonly used shell in Linux distributions.

Website: https://www.gnu.org/software/bash/

Bioconductor Tools written in the R programming language for the analysis and comprehension of high-throughput genomic data.

Website: https://www.bioconductor.org/

BiocManager An R package to install and update packages from the *Bioconductor* project repository.

Website: https://cran.r-project.org/package=BiocManager

BioPython Python tools for computational molecular biology.

Website: http://biopython.org/

Bio::Phylo Perl package for phylogenetic analysis.

Website: https://metacpan.org/pod/Bio::Phylo

commands Also called as binaries or executables.

Conda Conda is a package and environment manager.

As a *package manager*, it can be used to install software from *Anaconda Cloud*. Software *dependencies* will be installed automatically.

As an *environment manager*, it can be used to create and manage environments containing different sets of packages. You can activate and use these environments as necessary.

cpanminus A Perl script to get, build and install modules from *CPAN*. It provides the `cpanm` command.

Website: https://metacpan.org/pod/App::cpanminus

cutadapt A Python program that finds and removes adapter sequences, primers, poly-A tails and other unwanted sequences from high-throughput sequencing reads.

Website: https://cutadapt.readthedocs.io

CPAN Abbreviation for Comprehensive Perl Archive Network. It provides additional Perl modules for installation (196,752 as of Dec 2020).

Website: https://www.cpan.org/

CRAN Abbreviation for Comprehensive R Archive Network. A network of FTP and web servers providing up-to-date versions of R, packages and documentation.

Website: https://cran.r-project.org/

Debian A Linux distribution made of free and open source software. It is free for anyone to download, use, modify and distribute.

Website: https://www.debian.org/

Debian package An archive of executable files, libraries, and documentation of software. It can be installed on Debian Linux and Debian-based systems like Ubuntu and Linux Mint. These files have the `.deb` file extension.

dependencies Additional programs or libraries that are needed for a program to work.

edgeR An R package for empirical analysis of digital gene expression data. It is available from the *Bioconductor* project repository.

Website: https://www.bioconductor.org/packages/edgeR/

FAST Abbreviation for FAST Analysis of Sequences Toolbox. A set of utilities written in Perl that extend the UNIX paradigm to bioinformatic sequence records.

Website: https://metacpan.org/pod/FAST

Files The default file manager in *Linux Mint* Cinnamon edition. Its original name is Nemo.

gdebi A simple tool to install *Debian package* files along with their *dependencies* (if any).

Website: https://launchpad.net/gdebi

GNU GNU is a recursive acronym for GNU's Not Unix. The goal of the project is to offer a Unix-compatible system that would be 100% free software.

Website: https://www.gnu.org/

GUI Abbreviation for graphical user interface. On a personal computer, it typically includes application windows with buttons to access their functions, icons for launching applications and widgets for managing devices and services.

IDE Abbreviation for integrated development environment. Some examples include PyCharm, RStudio and Eclipse.

local-lib A Perl module to create a local directory structure to install modules with their *dependencies* without requiring administrator privileges.

Website: https://metacpan.org/pod/local::lib

Linux Mint A desktop Linux distribution. It is based on *Debian* and *Ubuntu*.

Website: https://linuxmint.com/

MEGA Software for Molecular Evolutionary Genetics Analysis. Available as *GUI* and command-line versions.

Website: https://megasoftware.net/

MetaCPAN A search engine for Perl packages and modules available on *CPAN*.

Website: https://metacpan.org/

Miniconda A minimal distribution of *Conda*. It is faster to install and also uses less disk space, when compared to *Anaconda* — the alternative installer which comes bundled with additional packages.

Website: https://docs.conda.io/en/latest/miniconda.html

Modeller A program for comparative protein structure modelling by satisfaction of spatial restraints.

Website: https://salilab.org/modeller/

nano A simple command-line based text editor.

Website: https://www.nano-editor.org/

OVA A file format for distributing virtual machine images.

pip The Python package installer. It can be used to install packages from *PyPI* and other Python package indexes.

Website: https://pip.pypa.io/

PyMOL PyMOL is a molecular visualization system originally developed by Warren L. Delano. It is currently maintained by Schrödinger, Inc.

Website: https://pymol.org/2/

PyPI Abbreviation for Python Package Index. Repository for software written in Python (275,161 projects as of Dec 2020).

Website: https://pypi.org/

root The primary administrator account on a Linux system.

R Programming language and free software environment for statistical computing and graphics.

Website: https://r-project.org

RStudio Desktop An *IDE* for *R*. It includes an R console, an editor with syntax-highlighting and tools for plotting, debug-

ging and managing R packages. The open-source version can be downloaded for free from the project website.

Website: https://rstudio.com/products/rstudio/download/

Related sections:

-> installing-debian-package

setuptools Python program for building and installing Python packages.

shell The shell provides a command-line interface (CLI) to the operating system's services.

Software Manager The default application(*GUI*), for installing software on Linux Mint.

Applications with similar functionality are available on other Linux distributions. For example:

- Ubuntu Software on Ubuntu
- Software on Fedora and other GNOME-based distributions
- Discover on Kubuntu and other KDE-based distributions.

Related sections:

-> getting-software/quick-and-easy/index

Synaptic A *GUI* package manager for systems using *apt*.

Website: https://www.nongnu.org/synaptic/

text editor Program to edit text files.

Examples (*GUI*) - Text Editor (xed) on Linux Mint, Text Editor (gedit) on Ubuntu, Kwrite on KDE Plasma, Geany etc.,

Examples (command-line) — *nano*, VIM or vi, GNU Emacs etc.,

Related sections:

-> /using-linux/applications/text-editor

-> /command-line/editing-text/index

Ubuntu Linux distribution developed by Canonical and community of developers. It is based on *Debian*.

Website: https://ubuntu.com/

wheel Python program for installing packages distributed in Python wheel (`.whl`) format.

genindex

Index

M

N

O

P

R

S

T

U

W