Pymote Documentation

Release 0.2.2

Damir Arbula

May 11, 2018

Contents

1 Installation **3**
- 1.1 Requirements . . . 3
- 1.2 Windows . . . 5
- 1.3 Linux (Ubuntu) . . . 5

2 Starting Pymote **7**
- 2.1 Interactive console (IPython) . . . 8
- 2.2 Simulation GUI . . . 8

3 Tutorials **9**
- 3.1 Hello distributed world . . . 9

4 Reference **11**
- 4.1 Sensors . . . 11
- 4.2 Network Generator . . . 11
- 4.3 Pymote configuration . . . 12

5 Developer Guide **13**
- 5.1 Distributing to PyPI . . . 13
- 5.2 Running tests . . . 14
- 5.3 Writing documentation . . . 14

6 Indices and tables **17**

This document refers to the Pymote version 0.2.2

CHAPTER 1

Installation

This document assumes you are familiar with using command prompt or shell. It should outline the necessary steps to install software needed for using Pymote.

1.1 Requirements

- Python 2.7
- Setuptools
- NumPy
- SciPy
- Matplotlib 1.2
- PySide (for gui)
- IPython
- NetworkX
- PyPNG

If you don't have all required packages already installed and/or want them installed in an isolated environment (see note below) please follow the instructions for your OS in the following sections.

Note: Since there can be only one version of any package installed systemwide in some cases this can result in situation where two programs need different versions of the same package. This is resolved by using isolated virtual environments.

Alternatively, if none of the above is your concern, although not recommended, all required packages can be installed systemwide using their respective instructions for appropriate OS and then Pymote can be installed by using:

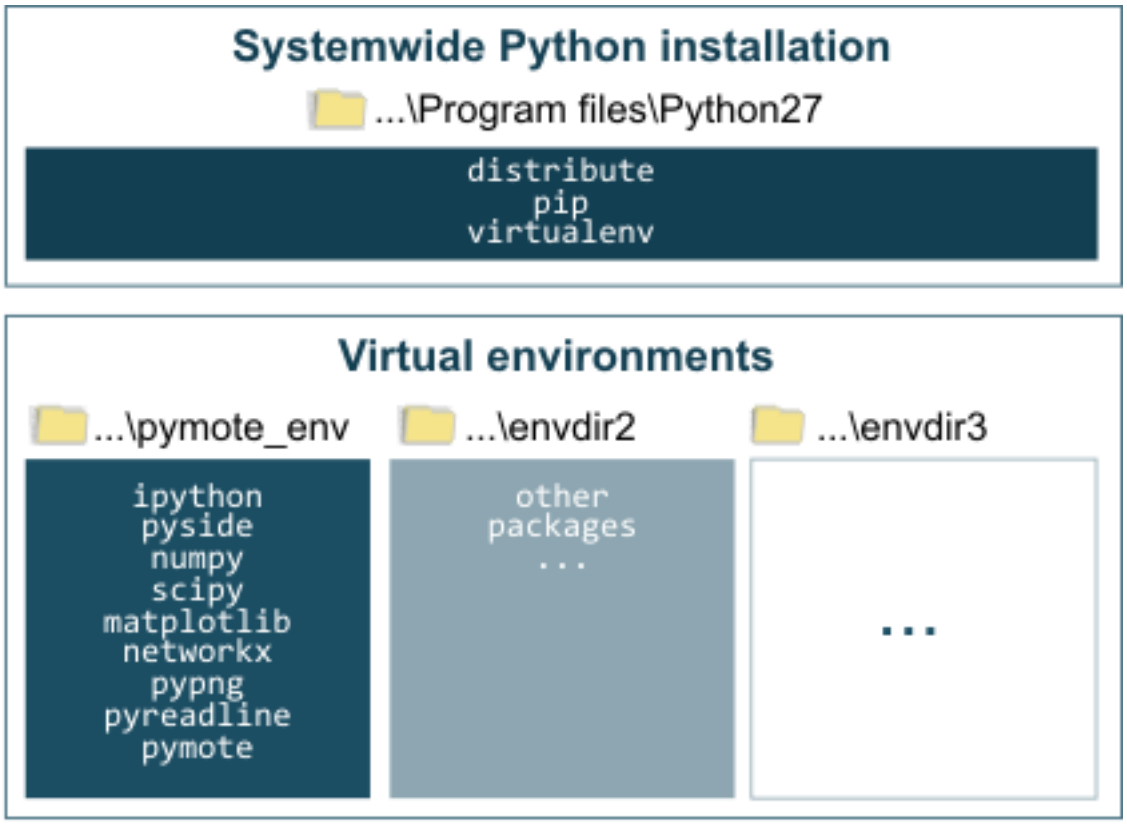

Fig. 1: Virtual environments *live* in a separate directories and they are independent form systemwide Python installation.

```
> pip install pymote
```

1.2 Windows

Windows version can be installed in an isolated environment very easy:

1. Install WinPython 2.7. WinPython has included python and all required packages inside simple exe installer and they are all installed one relocatable*[0] directory.

2. Run `WinPython Command Prompt.exe` located in WinPython installation dir and install latest official version of Pymote with `pip install pymote`

Note: For latest development version instead of `pip intall pymote` use `pip install -e git+https://github.com/darbula/pymote.git#egg=Pymote` and for upgrade after git pull use run `python setup.py develop` inside pymote dir.

For starting Pymote interactive console just run `ipymote` from the WinPython Command Prompt, and for simulation GUI run `pymote-simgui`. For more details refer to *Starting Pymote*.

Alternatively, installation can be done using windows_virtualenv but it is much more involved and not recommended.

1.3 Linux (Ubuntu)

Python 2.7 should already be installed on all new releases of Linux.

Install packages required for getting and compiling the source:

```
$ sudo apt-get install git libatlas-dev libpng12-dev libfreetype6 libfreetype6-dev␣
↪g++ libzmq-dev liblapack-dev gfortran python-dev build-essential
```

1.3.1 Virtualenv

Install pip and virtualenv:

```
$ sudo apt-get install python-pip python-virtualenv
```

Create virtual environment:

```
$ virtualenv pymote_env --system-site-packages
```

Warning: If you want to avoid using *–system-site-packages* then PySide package has to be installed in virtualenv which is slightly involved or time and disk-space consuming. Both procedures are described below in PySide section.

Activate virtual environment:

[0] After directory reloacation change links in headers of `Scripts/ipymote` and `Scripts/pymote-simgui` and paths in `Lib/site-packages/easy_install.pth` and `Lib/site-packages/Pymote.egg-link`.

```
$ source pymote_env/bin/activate
```

1.3.2 Required packages

Install required python packages into virtual environment:

```
(pymote_env)$ pip install numpy scipy ipython matplotlib networkx pypng
```

IPython notebook

Optionally for IPython notebook install these:

```
(pymote_env)$ pip install tornado pyzmq pygments jinja2
```

PySide

Installing PySide into virtual environment can take some skill or time and disk space. To avoid this, it can be installed systemwide (if *–system-site-packages* option is used when creating virtualenv, as noted above) using these instructions or simply like this:

```
$ sudo add-apt-repository ppa:pyside
$ sudo apt-get update
$ sudo apt-get install python-pyside
```

If you really want to install PySide into virtual environment quick option is to follow this solution or simmilar and even better solution

More time consuming option is to use pip. In this case you'll have to install packages needed for compilation using following commands:

```
$ sudo apt-get install cmake qt4-qmake qt-sdk
(pymote_env)$ pip install pyside
```

1.3.3 Pymote

Finally, in order to download and install Pymote and all other required packages there are two aviliable options, use one of them:

1. *Stable*: for latest stable version use package from PyPI:

   ```
   (pymote_env)> pip install pymote
   ```

2. *Development*: to install latest development version of the Pymote use source from github repo:

   ```
   (pymote_env)> pip install -e git+https://github.com/darbula/pymote.git#egg=Pymote
   ```

1.3.4 Starting Pymote

Before starting, make sure that virtual environment is *activated* and run `ipymote` for interactive console or `pymote-simgui` for simulation GUI. For more details refer to *Starting Pymote*.

CHAPTER 2

Starting Pymote

Pymote features interactive console based on IPython and simulation GUI.

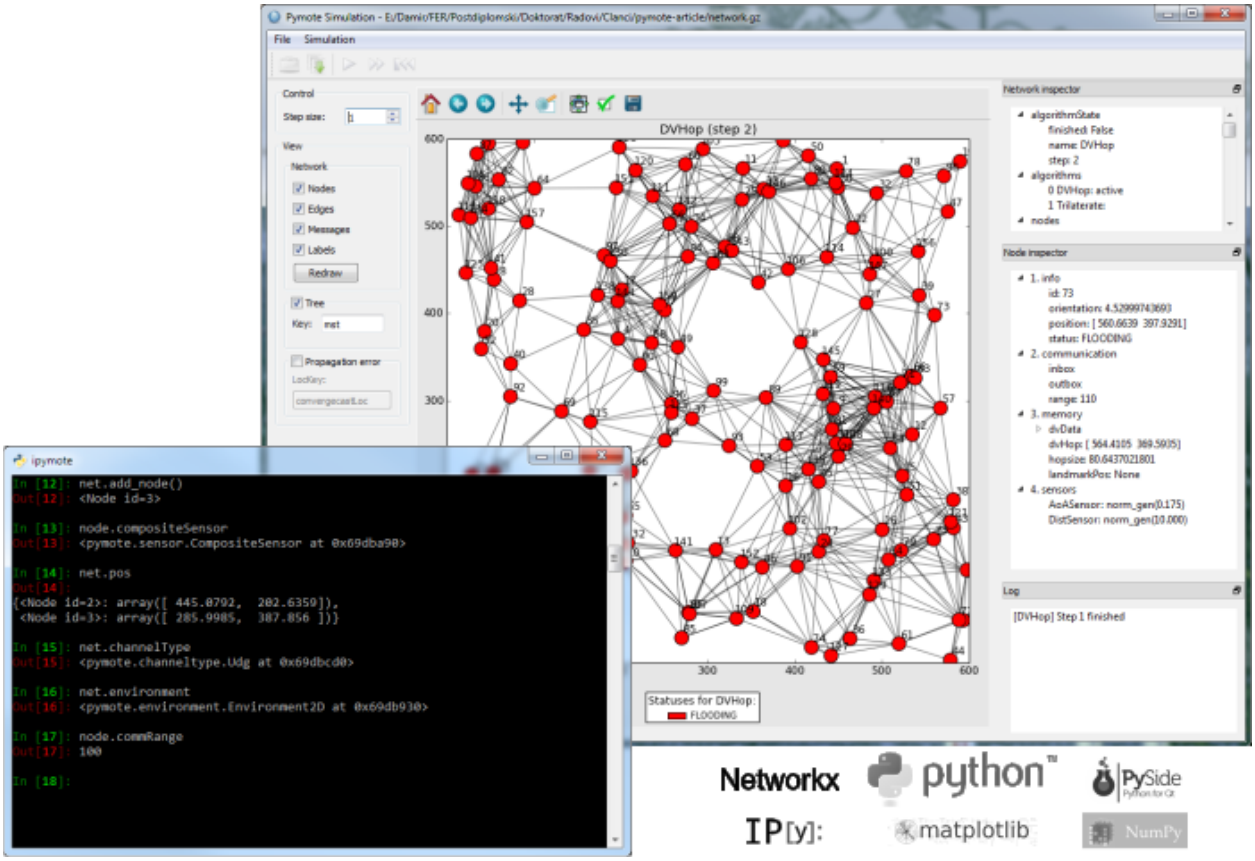

Fig. 1: Pymote console and GUI

2.1 Interactive console (IPython)

To use Pymote from the interactive console (IPython) start provided program `ipymote` with previously activated virtual environment:

```
> ipymote
```

Note: If virtualenv is used make sure that virtual environment is activated (*linux*, windows) and if WinPython is used then run `ipymote` from WinPython Command Prompt.

Pymote can also be started by starting IPython directly and using dedicated `pymote` profile:

```
> ipython --profile=pymote
```

Note: Pymote profile files should be present inside `pymote_env/.ipython/profile_pymote/ipython_config.py` or `~/.ipython/profile_pymote/ipython_config.py` file created during Pymote installation.

2.2 Simulation GUI

Pymote features simulation GUI which can be started as standalone application using `pymote-simgui` (in Windows `pymote-simgui.exe`).

Note: If pymote is installed in virtual environment then *pymote-simgui* starts inside this environment. When network pickle is opened in simulator all algorithms this network is referencing must be importable from virtual environment. The easy and proper way to ensure that the algorithms are importable is to use bootstrap algorithms package that can be found in pymote-algorithms-bootstrap and follow the instructions found there.

2.2.1 Simulation GUI running from the interactive console

Very convenient way of starting and working with the GUI is from the interactive console by running `simulationgui.py` like this:

```
In [1]: %run path/to/pymote/gui/simulationgui.py
```

The gui event loop is separated from the console. Simulation window can be accessed by using `simgui` and network in the simulator window by using `simgui.net` so all simulation objects (network, nodes, messages...) are fully inspectable and usable via console.

CHAPTER 3

Tutorials

Tutorials assume that the Pymote and all required packages are installed. If not, please refer to the *Installation* section of this documentation.

3.1 Hello distributed world

This tutorial demonstrates a distributed version of the classic *Hello world* example. Tutorial is currently available in form of IPython notebook which can be viewed and downloaded.

CHAPTER 4

Reference

Release 0.2.2

Date May 11, 2018

4.1 Sensors

4.1.1 Real world sensors

AoASensor
DistSensor

4.1.2 Knowledge sensors

NeighborsSensor
TruePosSensor

4.1.3 Composite sensor

4.2 Network Generator

Implementation of different methods for automated network creation. It defines parameters (conditions) that generated network must satisfy.

4.2.1 Methods

4.2.2 Default procedure

For any generator method network attributes take default priorities which are defined like this:

- first network is created in given environment with *n_count* number of nodes and *comm_range* communication range
- **if *connected* is True it must be satisfied, if not satisfied initially:**
  - gradually increase number of nodes up to *n_max*
  - if comm_range is None gradually increase nodes commRange
  - if still not connected raise an exception
- **if *degree* condition is defined and current network degree is**
  - lower - repeat measures from the last step to increase current network degree
  - higher one degree or more - try countermeasures i.e. decrease number of nodes and commRange but without influencing other defined and already satisfied parameters (*connected*)

4.3 Pymote configuration

Pymote initial configuration is defined by its *global settings*

Implementation of different methods for automated network creation.

4.3.1 Configuration module

4.3.2 Global settings

Below is list of current Pymote global settings.

CHAPTER 5

Developer Guide

5.1 Distributing to PyPI

http://docs.python.org/2/distutils/index.html#distutils-index https://python-packaging-user-guide.readthedocs.org/en/latest/current/

5.1.1 Windows

Install required packages:

```
> pip install twine wheel
```

Create `C:\Users\<user>\.pypirc` file:

```
[distutils]
index-servers =
    pypi

[pypi]
repository: http://pypi.python.org/pypi
username: <username>
password: <password>

[server-login]
repository: http://pypi.python.org/pypi
username: <username>
password: <password>
```

Issue these commands:

```
> setx HOME C:\Users\<user>
> python setup.py sdist bdist_wheel
> twine upload dist/*
```

5.2 Running tests

To execute tests install nose `pip install nose` and run it inside pymote directory. All tests should be found recursively scanning directories. To run all tests run this from root pymote directory:

```
nosetests -v
```

To run selected test module:

```
nosetests -v pymote.tests.test_algorithm
```

5.2.1 Tests coverage

For tests coverage install Coverage package and run it:

```
pip install coverage
coverage run --source=pymote setup.py nosetests
```

Configuration file is in `.coveragerc`.

Make report in console or html:

```
coverage report -m
coverage html
```

For integration with coveralls refer to coverall readme.

5.3 Writing documentation

This section describes certain not obvious details in writing documentation for Pymote in sphinx.

5.3.1 Intersphinx

To auto-reference external document in with intersphinx:

1. set intersphinx_mapping in `conf.py`:

```
intersphinx_mapping = {
    'python': ('http://docs.python.org/', None),
    'numpy': ('http://docs.scipy.org/doc/numpy/', None),
    'scipy': ('http://docs.scipy.org/doc/scipy/reference/', None),
}
```

2. reference in docs with `` :py:<type>:`<ref>` `` i.e. `:py:class:`numpy.poly1d``. For finding reference manually read on.

Finding reference

If `<type>` is not explicitly known it can be found out from `objects.inv` files found in URLs in intersphinx_mapping above.

To get and decode objects.inv for numpy:

```
$ wget http://docs.scipy.org/doc/numpy/
$ ipython
```

In python:

```
import zlib
with open("objects.inv","r") as f:
    inv_lines = f.readlines()
lista = zlib.decompress(''.join(inv_lines[4:])).split('\n')
with open('objects_numpy.inv','w') as f:
    for line in lista:
        f.write(line+'\n')
```

To find reference for `numpy.poly1d` serach for it in decoded file `objects_numpy.inv`.

The line should include word `class`

In documentation include `:py:class:`numpy.poly1d``

For `scipy.stats.norm`:

```
find scipy.stats.norm -> data -> :py:data:`scipy.stats.norm`
```

5.3.2 Readthedocs.org

In order for readthedocs.org to make documentation it needs to have certain packages accessible.

1. On readthedocs.org admin page check option `Use virtualenv` And `Use system packages` and in `Requirements file` put the name to the requirements file in repo (i.e. `requirements.txt`).
2. Make `readthedocs.py` module and put it in `docs` folder with Mock class found here
3. In documentation `conf.py` put the following lines to import mock class for certain modules that are not present in virtual environment.

```
if os.environ.get('READTHEDOCS', None) == 'True':
    sys.path.insert(0,'.')
    from readthedocs import *
    sys.path.pop(0)
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search
- Glossary