Package Management with Anaconda

TTS skills clinic

Rebecca Batorsky

March 2019
Python versions and Packages

You probably start using python with whatever version of python came installed on your laptop.

```
python --version
```

## Python 2.7.10

Next, you install a bunch of packages and start making beautiful plots. Eventually, a new package version comes along that requires an updated version of python in order to install:

```
pip install matplotlib==3.0.3
```

Collecting matplotlib==3.0.3
  Could not find a version that satisfies the requirement matplotlib==3.0.3 (from versions)
No matching distribution found for matplotlib==3.0.3
Python versions and Packages

You may try a number of options to upgrade your python installation...

Image credit: biostars
Conda Environments

Conda environments allow you to maintain package versions together

Image credit: https://medium.freecodecamp.org
Pip, Conda and Miniconda

Pip : package manager for python packages only
Conda : package and environment manager for any language

Image credit: https://medium.freecodecamp.org
Choosing a Miniconda Installation

Image credit: https://medium.freecodecamp.org
Installing Miniconda on your system

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

Follow installation instructions for your OS
## Root environments

Cconda creates a root environment that contains a version Python and some basic packages

<table>
<thead>
<tr>
<th>active environment</th>
<th>base</th>
</tr>
</thead>
<tbody>
<tr>
<td>active env location</td>
<td><code>/anaconda3</code></td>
</tr>
<tr>
<td>shell level</td>
<td>1</td>
</tr>
<tr>
<td>user config file</td>
<td><code>/Users/rbator01/.condarc</code></td>
</tr>
<tr>
<td>populated config files</td>
<td></td>
</tr>
<tr>
<td>conda version</td>
<td>4.5.11</td>
</tr>
<tr>
<td>conda-build version</td>
<td>3.17.1</td>
</tr>
<tr>
<td>python version</td>
<td>3.6.6.final.0</td>
</tr>
<tr>
<td>base environment</td>
<td><code>/anaconda3</code> (writable)</td>
</tr>
<tr>
<td>channel URLs</td>
<td></td>
</tr>
<tr>
<td></td>
<td><a href="https://repo.anaconda.com/pkgs/main/osx-64">https://repo.anaconda.com/pkgs/main/osx-64</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://repo.anaconda.com/pkgs/main/noarch">https://repo.anaconda.com/pkgs/main/noarch</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://repo.anaconda.com/pkgs/free/osx-64">https://repo.anaconda.com/pkgs/free/osx-64</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://repo.anaconda.com/pkgs/free/noarch">https://repo.anaconda.com/pkgs/free/noarch</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://repo.anaconda.com/pkgs/r/osx-64">https://repo.anaconda.com/pkgs/r/osx-64</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://repo.anaconda.com/pkgs/r/noarch">https://repo.anaconda.com/pkgs/r/noarch</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://repo.anaconda.com/pkgs/pro/osx-64">https://repo.anaconda.com/pkgs/pro/osx-64</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://repo.anaconda.com/pkgs/pro/noarch">https://repo.anaconda.com/pkgs/pro/noarch</a></td>
</tr>
<tr>
<td>package cache</td>
<td><code>/anaconda3/pkgs</code></td>
</tr>
<tr>
<td></td>
<td><code>/Users/rbator01/.conda/pkgs</code></td>
</tr>
<tr>
<td>envs directories</td>
<td><code>/anaconda3/envs</code></td>
</tr>
<tr>
<td></td>
<td><code>/Users/rbator01/.conda/envs</code></td>
</tr>
<tr>
<td>platform</td>
<td><code>osx-64</code></td>
</tr>
</tbody>
</table>
Create and Activate an environment

conda create --name testenv python=3.4

Activate your new environment:

```bash
source activate testenv
```

Leave your environment:

```bash
source deactivate testenv
```

You can see your environment

```bash
conda env list
```

```
# conda environments:
#
base                  * /Users/rbator01/miniconda3
/anaconda3
/anaconda3/envs/testenv
```
# Installing packages

```bash
source activate testenv
conda install pandas
```

View installed packages

```bash
conda list -n testenv python=3.4
```

```bash
# packages in environment at /anaconda3/envs/testenv:
#
# Name          Version     Build    Channel
# blas          1.0         mkl
# ca-certificates 2019.1.23 0
# intel-openmp  2019.1     144
# mkl           2017.0.4   h1fae6ae_0
# numpy         1.11.3      py34_0
# openssl       1.0.2r      h1de35cc_0
# pandas        0.19.2      np111py34_1
# pip           9.0.1       py34_1
# python        3.4.5       0
# python-dateutil 2.6.1     py34_0
# pytz          2017.2      py34_0
# readline      6.2         2
# setuptools    27.2.0      py34_0
# six           1.10.0      py34_0
# sqlite        3.13.0      0
```

`/anaconda3/envs/testenv`
Channels

Channels are like storages...

CONDA IS LOOKING FOR THIS PACKAGE

CHANNEL 1
FIRST, IT IS LOOKING FOR THE PACKAGE IN THIS CHANNEL (THIS HAS THE HIGHEST PRIORITY)

CHANNEL 2
IT WAS NOT IN THE FIRST CHANNEL, SO IT MOVES TO THE STORAGE WITH THE 2ND HIGHEST PRIORITY

CHANNEL 3
YAY! CONDA HAS FOUND THE PACKAGE! NOW IT IS ADDED TO YOUR ENVIRONMENT.

CHANNEL 4

WHAT HAPPENS IF CONDA DID NOT FIND THE PACKAGE?

BY DEFAULT, CONDA LOOKS FOR PACKAGES IN THE OFFICIAL STORAGES OF CONTINUUM.
WHY? BECAUSE BY DEFAULT THESE HAVE THE HIGHEST PRIORITIES.

HOWEVER, YOU HAVE THE POWER TO

ADD NEW CHANNELS (STORAGES) THAT CONTAIN THE PACKAGES YOU NEED!

SO LET'S SAY YOU WANT TO INSTALL THIS PACKAGE:

\[ \rightarrow \text{conda does not find it in the first 3 channels} \]
\[ \Rightarrow \text{you need to add a 4th one!} \]

Image credit: https://medium.freecodecamp.org
Finding packages

If you're not sure if your package is available from conda, just google it!
E.g. "conda IGV"
Adding channel

You could either install it by specifying the channel

```bash
cconda install -c bioconda igv
```

Or by adding the channel to the default list of channels to search:

```bash
cconda config --add channels bioconda
cat ~/.condarc
```

```
channels:
  - bioconda
  - defaults
```

You can now install packages from bioconda without specifying:

```bash
cconda install igv
```
Using conda on the Tufts HPC

... is easy! Anaconda is installed system wide, and loadable as a module

```bash
module load anaconda/3
conda create -n testenv python=3.4
```

Note that this will both store package downloads and environments in your home directory. This is Not Good on an HPC, where home directories are small.

You can change this by specifying the path of the environment in a space where you have more storage

```bash
conda create -p /cluster/tufts/bio/tools/conda_envs/testenv python=3.4
```

And by explicitly setting the package cache location in your ~/.condarc

```yaml
channels:
  - conda-forge
  - bioconda
  - defaults
envs_dirs:
  - /cluster/tufts/bio/tools/conda_envs/
pkgs_dirs:
  - /cluster/tufts/bio/tools/conda_envs/pkgs/
```
Sharing environments with others

```bash
source activate testenv
cconda env export > testenv.yml
```

In order to create it on a new system:

```bash
cconda env create -f environment.yml
```

The YAML file will look like this:

```yaml
name: testenv
channels:
  - bioconda
  - defaults
dependencies:
  - igv=2.4.17=0
  - blas=1.0=mkl
  - ca-certificates=2019.1.23=0
  - intel-openmp=2019.1=144
  - mkl=2017.0.4=h1fae6ae_0
  - numpy=1.11.3=py34_0
  - openjdk=8.0.152=h393ad39_1
  - openssl=1.0.2r=h1de35cc_0
  - pandas=0.19.2=np111py34_1
  - pip=9.0.1=py34_1
  - python=3.4.5=0
```
Removing an environment

conda remove -n testenv --all

For more information see this blog post: https://medium.freecodecamp.org/why-you-need-python-environments-and-how-to-manage-them-with-conda-85f155f4353c

Documentation on Anaconda: https://conda.readthedocs.io/en/latest/

Thanks!