Managing your software environment with Conda

Oxford Biomedical Data Science Training Programme

Managing scientific software

- The problem: dependency hell
- Solutions
- Why Conda?
- Conda tutorial

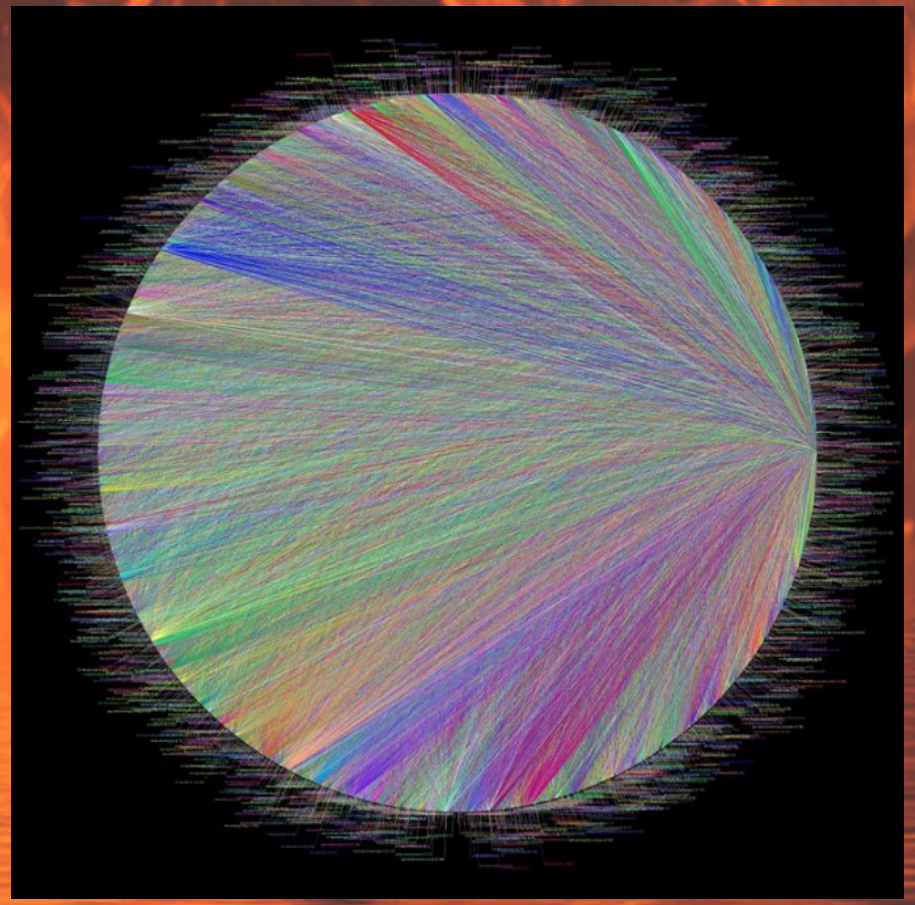
Salmon —Don't count . . . quantify!

The dependency hell

- Code reuse = best practice
- Instead of reinventing the wheel:
  - Build new packages using code/functions from pre-existing packages
  - Only write new code where desired functionality not implemented in an existing package
- Problems:
  - Any given package is dependent on specific versions of other packages
  - The bigger the project, the more the dependencies
  - You may end up with: conflicting dependencies, long chains of dependencies

The dependency hell

- Gentoo Linux (Linux distribution)
- Packages: 14,319
- Dependencies: 63,988



http://disfunksioneel.blogspot.co.uk/2011/04/linux-software-dependencies.html

The dependency hell

External dependencies for the CGAT genomics code (as of May 2017):

- R packages: 58 dependencies
  - DESeq2
  - Sleuth…
- Python packages: 45 dependencies
  - Numpy
  - Scipy…
- Bioinformatics packages: 205 dependencies
  - Bedtools
  - Samtools…

Historical solutions

1. Distribute software as a tarball (compressed archive)
   - Dependencies listed in README file, install yourself

2. Package managers
   - Tarballs plus metadata about dependencies – package manager will download package and all of its dependencies
   - Package managers started as core part of Linux distributions: YUM (Red Hat), APT (Debian)
   - Also package managers specific for programming languages: PyPi (Python), CRAN (R), CPAN (Perl)

https://cgatoxford.wordpress.com/2017/05/12/the-dependency-hell-in-software-development/

Historical solutions

3. General purpose package managers e.g. Conda
   - Manage packages in multiple programming languages
   - Can be installed on all major operating systems

4. Virtual machines
   - Entire operating system with all required software installed
   - e.g. run Linux programs on Windows

5. Containers e.g. Docker, Singularity
   - More lightweight than a virtual machine (uses host OS)

https://cgatoxford.wordpress.com/2017/05/12/the-dependency-hell-in-software-development/

General advice

- Minimise number of dependencies
- Keep list up-to-date
- Be selective when adding new dependencies:
  - Well tested
  - Well documented
  - Well packaged

Why Conda?

- General purpose packaging system:
  - Python, R, Java, JavaScript, Fortran etc.
- Cross platform
  - Windows, macOS, Linux
- Does not require administrator privileges
  - Can install anywhere on your system
- Install binaries
  - No compilation needed (make sure the binaries can be trusted)

Why Conda?

- Lets you create many separate software environments
  - Necessary if software has conflicting dependencies
  - Use different versions of software for different projects
  - Isolated environment to test new software
- Share your software environment with others

Conda environments

- Base
  - Default environment
  - Conda environments do not inherit from base
- Multiple environments using the same software binaries
  - Conda uses hard links to point to same software in Conda cache (only if version is identical)
  - Save disk space
- Pinned versions
  - Specify exact version of package – conda-meta/pinned file
  - Delete package name from pinned list to update it

Tracking environments

- Conda tracks what you have added:
  - Complete history of all packages added
  - Revert back to any environment

`conda list --revisions` # tells you what packages were added/updated/downgraded and when

```
2018-07-13 19:28:25  (rev 7)
pysam  {0.13.0 (bioconda) -> 0.14.0 (bioconda)}
samtools  {1.6 (bioconda) -> 1.7 (bioconda)}
scipy  {0.19.1 (conda-forge) -> 1.1.0}
+plotly-3.0.0 (conda-forge)
+retrying-1.3.3 (conda-forge)
```

Conda channels

- Conda packages can come from different repositories – channels
- Defaults – from Conda team
- Conda-forge – community-led channel
- Bioconda – channel specialising in bioinformatics packages (we add this channel last, giving it the highest priority)
- Be careful when specifying channels using --channel
  - This will override the hierarchy
  - Can lead to lots of unwanted updates in an existing environment

Anaconda vs. Miniconda

Anaconda

- Large base environment
  - Conda + dependencies
  - Over 150 data science software packages + dependencies
- Graphical user interface
- Large amount of disk space
- Easy and automated

Miniconda

- Lightweight installation
  - Conda + dependencies
- Uses less disk space
- More control

Conda tutorial

1. Download latest version of Conda (Miniconda)
2. Set up base Conda environment
3. Add packages to an environment
4. Export software environments to file (YAML format)
5. Create new environments from YAML files:
   - Python/genomics
   - R/Bioconductor

Conda tutorial

https://github.com/OBDS-Training/Conda_Workshops/blob/master/1_Conda_intro.md

Help:

- Conda docs: https://docs.conda.io/en/latest/index.html
- Conda cheat sheet: https://docs.conda.io/projects/conda/en/latest/_downloads/843d9e0198f2a193a3484886fa28163c/conda-cheatsheet.pdf