Environment Modules

Software and library versioning the easy way

Piotr Dworzynski, Pers Lab, NNF Center for Basic Metabolic Research
piotr@sund.ku.dk

The problem

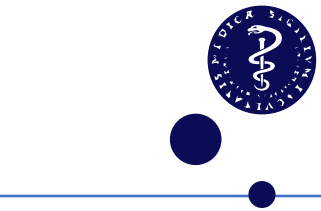
Everyday struggle:

- Tool A requires R-3.2
- Tool B requires R-3.5 and Python 2.7
- Tool C-v1 requires R-3.4 and SomeLibrary v2.3
- Tool C-v2 requires R-3.5 and SomeLibrary v2.5

Environment modules address software:

- version dependencies
- version conflicts
- library conflicts
- reproducibility

Environment modules



Available at http://modules.sourceforge.net/

Ubuntu/RedHat/CentOS pkg: environment-modules

Manipulates environmental variables

- e.g. $PATH, $LD_LIBRARY_PATH, $CPLUS_INCLUDE_PATH
- Handles dependencies and conflicts between modules

Basic use-case

By default R points to system R v3.5.0

```
$ which R
> /usr/local/bin/R
$ R
> R version 3.5.0 (2018-04-23) -- "Joy in Playing"
```

With module load command we make R point to a custom R installation

```
$ module load R/3.5.3
$ which R
> /tools/R/3.5.3/bin/R
$ R
> R version 3.5.3 (2019-03-11) -- "Great Truth”
```

Example setup

`/tools`

- overarching directory for software not coming from yum/apt

`/tools/src`

- downloaded software sources

`/tools/modules/`**`XXX`**`/`**`YYY`**

- environment modules live here
- **`XXX`** – name of the tool (R, anaconda, bwatools)
- **`YYY`** – plain text module file named as tool's version/flag (3.4, 3.5-openblas)

`/tools/`**`XXX`**`/`**`YYY`**`/`**`ZZZ`**

- software (tools, libraries) lives here
- **`XXX`** – name of the tool (R, anaconda, bwatools)
- **`YYY`** – directory named as tool's version/flag (3.4, 3.5-openblas)
- **`ZZZ`** – software executables/libs (--prefix-path would point here) e.g. /bin /lib /include

Anatomy of an environment module

Contents of example env module (text file; /tools/modules/R/3.5.3)

```
#%Module1.0
module-whatis "R is a free software environment for statistical computing and graphics."

conflict R

prepend-path      PATH                /tools/R/3.5.3/bin/
prepend-path      MANPATH             /tools/R/3.5.3/share/man
prepend-path      LD_LIBRARY_PATH     /tools/R/3.5.3/lib64/
prepend-path      LD_LIBRARY_PATH     /tools/R/3.5.3/lib64/R/library/
prepend-path      C_INCLUDE_PATH      /tools/R/3.5.3/lib64/R/include/
prepend-path      CPLUS_INCLUDE_PATH  /tools/R/3.5.3/lib64/R/include/
prepend-path      R_ROOT              /tools/R/3.5.3/lib64

setenv      RINSIDE_LIBRARY          /tools/R/3.5.3/lib64/R/library
setenv      RINSIDE_INCLUDE_DIR      /tools/R/3.5.3/lib64/R/include
```

Use cases & Limitations

Use cases:

- Basic tool/library version management
- Data version management (e.g. reference genomes)
- Pipeline tool/library/data version management
- Ansible & env-modules powered software collections akin to bio-ansible

Limitations:

- Python/R packages management
- Not meant for very complex situations
- Supports only specific shells

Thanks!

Learn more:

- official webpage
- official documentation
- tutorial 1 (admin-magazine.com)
- tutorial 2 (geoghegan.me)

Followup discussion:

A common software repository for NNF centers?

Ansible, EnvModules, Docker/Singularity images

Stay in touch:

- Piotr Dworzynski
- p.dworzynski@gmail.com
- piotr@sund.ku.dk