How to organize efficient bioinformatics data analysis workflows

Alina Frolova

Institute of molecular biology and genetics,
Kyiv, Ukraine
a.o.frolova@imbg.org.ua

12 Oct 2017, Warsaw, Poland
A scientific workflow system is a specialized form of a workflow management system designed specifically to compose and execute a series of computational or data manipulation steps, or workflow, in a scientific application.
Why do we need specific tools to organize our data analysis?

- Dozens of dependencies (binary tools, compilers, libraries, system tools, etc)
- Experimental nature of scientific workflows tends to be difficult to install, configure and deploy
- Heterogeneous executing platforms and system architecture (laptop → supercomputer)
Reproducibility layers

- Code (github, bitbucket)
- Data (datproject, git LFS)
- Workflow (SnakeMake, NextFlow)
- Environment (conda, docker, vagrant)
CONDA
package manager

Some slides are adapted from Travis Oliphant and Kale Franz presentation
Complex things are built out of simple things

- Fundamental principle of software engineering is “separation of concerns” (modularity)
- Reusability is enhanced when you “do one thing and do it well”
- To deploy you need to bring the pieces back together
- This all means you need a good packaging system
Packaging is a critical part of software

A package manager or package management system is a collection of software tools that automates the process of installing, upgrading, configuring, and removing computer programs for a computer's operating system in a consistent manner.

Poor packaging and deployment solutions are everywhere in open source and industry and lead to software engineering mistakes with poorly factored code:

- hard to test
- hard to debug
- hard to maintain
Package manager functions

- Extracting package archives
- Ensuring the integrity and authenticity of the package by verifying their digital certificates and checksums
- Looking up, downloading, installing or updating existing software from a software repository
- Grouping packages by function to reduce user confusion
- Managing dependencies to ensure a package is installed with all packages it requires, thus avoiding "dependency hell"
Packages typical metadata

- the software's name
- description of its purpose
- version number
- vendor
- checksum
- list of dependencies necessary for the software to run properly
Typical software managing problems for data scientists

- I work on a server, I don’t have root access, how to compile/install locally (python, R packages)?!
- I need both python 2 and 3!
- I need to replace standard lib by compiling custom lib!
- I compiled and installed custom lib and everything stopped working!
- I updated the soft and custom lib was replaced again!
- I somehow configured working environment, but now I need to switch to another server!
- I try to install R Bioconductor package, but it says the package is missing for my R version!
- What the heck is JAVA_HOME, I don’t understand update-alternatives utility!
- Friend recommended me to try Gentoo linux, he said it is fun...
- Your story?
ANAconda NAVIGATOR
Desktop Portal to Data Science

ANAconda PROJECT
Portable Data Science Encapsulation

DATA SCIENCE LIBRARIES

Data Science IDEs
- jupyter
- jupyterlab
- spyder
- R Studio

Analytics & Scientific Computing
- NumPy
- SciPy
- Numba
- pandas
- DASK

Visualization
- Bokeh
- HoloViews
- Dash
- Dash reflector
- matplotlib

Machine Learning
- TensorFlow
- Scikit-learn
- H2O.ai
- Theano

CONDa
Data Science Package & Environment Manager

...and many more!
Anaconda Distribution

- **Anaconda Navigator**: GUI that allows you to launch applications and easily manage conda packages, environments and channels without using a command prompt or terminal program.

- **Anaconda Project**: automates setup steps such as installing the right packages, downloading files, setting environment variables and running commands; simplifies deployment to servers.

- **Conda**: cross-platform package manager, works from console.
Conda allows to

- Manage project dependencies, including programming languages and libraries
- Isolate development and production environments with channels
- Share environments with a minimal footprint
- Support multiple versions of languages, storage systems, and packages
- Provide a common interface for building, installing, and sharing packages
Enabling Environments

- **portability**
  system-level package management that’s not tied to hard-coded system paths

- **multiple, composable environments**
  multiple instances of otherwise-colliding software, functionally isolated on the same system

- **preferential use of hard links**
  soft links are slower and problematic when working with linked shared libraries; hard links (or copies) more reliably keep the compiled-in relative paths intact)
Power and Flexibility for Users and Sysadmins Alike

• natively multi-user
  fully functional within the limited privileges of a non-privileged user

  admin/root users enabled with extensive configuration
  and enforcement capabilities
Enforcer of Safety and Correctness

• **pre-compiled packages only**
  will never require a compiler in production, or unexpectedly invoke one on you

• **environment integrity**
  disk-mutating operations are wrapped in a transaction, and rolled back in the event of errors

• **environment correctness**
  conda enforces compatibility of packages within environments
Channels for User Empowerment

- the channel is a component of a package’s identity first-class citizen in package specifications

- easy package building with conda-build, a dedicated tool with engaged and dedicated code contributors

- channels enable community independent contributors, independent packaging communities, and corporations
channel for the conda package manager specializing in bioinformatics software

a repository of > 2700 bioinformatics packages ready to use with conda install
Conda: Cross-Platform Package Manager

Linux
- yum (rpm)
- apt-get (dpkg)

macOS
- macports
- homebrew
- fink

Windows
- chocolatey
- npackd

Cross-Platform
- conda

http://conda.pydata.org

Sophisticated light-weight environments included!
Cconda + Docker: better together

Data Science Development

Desktop / Laptop

Data Science Deployment

Server

conda env 1  conda env 2  conda env 3

Analysis 1  Analysis 2  Analysis 3

conda env 1  conda env 2  conda env 3

Analysis 1  Analysis 2  Analysis 3

Docker container
Layers of Process Isolation

- bare metal
- hypervisor
- virtual machine
- container (e.g. Docker)
- chroot
- conda env
- python virtualenv
- application

isolation in-depth

- pure-python isolation
- functional isolation
- more secure isolation
How to start

DOWNLOAD Anaconda or miniconda and do test-drive
https://www.anaconda.com/download/
http://conda.pydata.org/docs/test-drive.html

GET the conda cheatsheet
https://conda.io/docs/_downloads/conda-cheatsheet.pdf

READ more about conda in the developer blog
https://www.anaconda.com/blog/developer-blog/
What is conda?
- packages
- environments

What is conda *not*?
- configuration management
- process management
SnakeMake
workflow manager

Some slides are adapted from Johannes Köster presentation
Advantages of workflows

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>efficiency</td>
<td>Being automatic make researchers free from repetitive tasks and support “good practices”</td>
</tr>
<tr>
<td>reproducibility</td>
<td>Analysis may be replicated over time, easily and effectively</td>
</tr>
<tr>
<td>reuse</td>
<td>Both intermediate results and workflows can be reused</td>
</tr>
<tr>
<td>traceability</td>
<td>The workflow is enacted in an environment that allows tracing back results</td>
</tr>
<tr>
<td>separation of concerns</td>
<td>Modulatization of the sub-tasks</td>
</tr>
<tr>
<td>multi-tasking</td>
<td>Involving many pieces with a single command in a parallel manner</td>
</tr>
<tr>
<td>scaling</td>
<td>Across nodes, clusters with heterogeneous environments</td>
</tr>
<tr>
<td>graphical user interfaces</td>
<td>Some workflows automatically generate GUI (Galaxy)</td>
</tr>
</tbody>
</table>
Workflow systems examples

- Apache Taverna
- Galaxy
- Pegasus
- KNIME
- ExTasy
- MyExperiment
- SnakeMake
SnakeMake Concept

• Snakemake is a workflow management system that aims to reduce the complexity of creating workflows by providing a fast and comfortable execution environment.

• Snakemake workflows are essentially Python scripts extended by declarative code to define rules.

• Rules describe how to create output files from input files.
Rules

rule myrule:
  input:
    "path/to/{sample}.txt"
  output:
    "path/to/{sample}.column1.txt"
  shell:
    "cut -f1 < {input} > {output}"
Rules

```python
rule myrule:
    input:
        a="path/to/{sample}.txt"
    output:
        b="path/to/{sample}.column1.txt"
    shell:
        "cut -f1 < {input.a} > {output.b}""
Rules

```python
rule myrule:
    input:
        a = "path/to/{sample}.txt"
    output:
        b = "path/to/{sample}.column1.txt"
    run:
        with open(output.b, "w") as out:
            for l in csv.reader(open(input.a)):
                print(l[0], file=out)
```
SAMPLES = "500 501 502 503".split()

# require a bam for each sample
rule all:
    input:
        expand("{sample}.bam", sample=SAMPLES)

# map reads
rule map:
    input:
        ref="reference.fasta",
        index="reference.bwt",
        reads="{sample}.fastq"
    output:
        "{sample}.bam"
    threads: 8
    shell:
        "bwa mem -t {threads} {input.ref} {input.reads} | "
        "samtools view -Sbh - > {output}" # refer to threads and input

# create an index
rule index:
    input:
        "reference.fasta"
    output:
        "reference.bwt"
    shell:
        "bwa index {input}"
Workflow execution

- dependencies between rules are determined automatically
- directed acyclic graph (DAG) of jobs

# visualize the DAG of jobs

```
$ snakemake --dag | dot | display
```
Workflow execution

- disjoint paths in the DAG can be parallelized
- only outdated or missing files are created

```bash
# perform a dry-run
$ snakemake -n

# execute the workflow using 8 cores
$ snakemake --cores 8

# execute the workflow on a cluster (with up to 20 jobs)
$ snakemake --jobs 20 --cluster "qsub -pe threaded {threads}"

# execute the workflow on a cluster using the DRMAA API
$ snakemake --jobs 20 --drmaa
```
Workflows interoperability

- Common format for bioinformatics tool & workflow execution
- Community based standards effort
- Designed for clusters & clouds
- Supports the use of containers (e.g. Docker)
- Specify data dependencies between steps
- Scatter/gather on steps
- Nest workflows in steps

- Develop your pipeline on your local computer (optionally with Docker)
- Execute on your research cluster or in the cloud
- Deliver to users via workbenches

by Carole Goble
More to read about workflows

- https://www.biostars.org/p/91301/
- https://github.com/pditommaso/awesome-pipeline
Integrative Biology & Medicine, 2-7 Oct, 2017, Kyiv, Ukraine
https://integrativebio.com.ua/