Bioinformatics Project Organization

Keith Hughitt
NIH BYOB
Apr 14, 2022

Preface

- *There is not a single “best” approach to “bioinformatics project organization”...*
- I will share some things that I have found useful & adopted into my workflow, but,..
- It’s still an evolving process (..and that’s okay!)
- I’m equally curious to hear what others have found helpful :)

Topics

- Directory structure
- Version control
- Versioning
- Configuration
- Documentation
- Testing
- Packaging
- Virtual environments
- Containers
- Web deployment

This talk overlaps with & builds on an earlier BYOB presentation I gave a few years ago..

(Feel free to check it out for other related ideas..)

master | bioinformatics-best-practices / Keith_Hughitt.md | Go to file | ···

khughitt Added my best practice notes | Latest commit f5fda7a on May 24, 2019 | History

1 contributor

487 lines (407 sloc) | 21.4 KB | Raw | Blame

Bioinformatics advice I wish I learned 10 years ago

V. Keith Hughitt (May 23, 2019)

Overview

Below is a list of some of things I've learned over the past ten or so years working in bioinformatics, and, more generally, scientific data analysis.

Over the years my overall approach, preferred tools, and software environment has changed many times based on what works and what doesn't work for me.

This is not to say that my approach the only way to do things, or even the best way to do things, but hopefully some of the ideas or tools mentioned below will be useful to others in their own path towards bioinformatics bliss.

TL;DR

1. Don't rely on intuition
2. Subsample your data during development to allow for quick iteration

Project directory structure

1. top-level “`proj/`” folder
   a. Previously, tried using subdirs for *year* and *institution*, but both approaches tended to be overly constrained..
2. for larger projects, top-level “`proj/foo/`” folder may contain multiple *sub-folders*, each corresponding to different sub-projects or analyses..

```
~/d/r/proj/mm25
❯ l
drwxr-xr-x keith users 4 KB Wed Apr 13 18:59:06 2022 .
drwxr-xr-x keith users 4 KB Wed Mar 23 10:57:21 2022 ..
drwxr-xr-x keith keith 4 KB Wed Apr 13 18:59:06 2022 archive
drwxr-xr-x keith keith 4 KB Fri Nov  5 18:00:30 2021 assoc-direction
drwxr-xr-x keith users 4 KB Mon Jun 14 14:25:16 2021 cancermine-ongogene-vs-tsp-count
drwxr-xr-x keith users 4 KB Wed Jul  7 07:00:20 2021 coex
drwxr-xr-x keith users 4 KB Mon Jul  5 15:37:01 2021 cytogenetic-events
drwxr-xr-x keith users 4 KB Wed Jul 21 13:36:27 2021 dataset-properties
drwxr-xr-x keith users 4 KB Thu Sep 23 18:54:40 2021 disease-stage-evolution
drwxr-xr-x keith users 4 KB Mon Jul  5 17:39:09 2021 docker
drwxr-xr-x keith users 4 KB Mon Jul 19 15:16:02 2021 fmap
drwxr-xr-x keith users 4 KB Tue Aug 11 16:18:30 2020 gene-ids
drwxr-xr-x keith users 4 KB Mon Jun 28 09:41:40 2021 gene-score-correlatons
drwxr-xr-x keith users 4 KB Mon Feb 21 14:42:05 2022 geo
drwxr-xr-x keith keith 4 KB Thu Feb 24 06:17:58 2022 increasing-expr-bias
drwxr-xr-x keith users 4 KB Mon Jul 19 18:15:54 2021 manuscript
drwxr-xr-x keith keith 4 KB Sat Sep 25 19:42:36 2021 meta-integrator
drwxr-xr-x keith users 4 KB Mon Jul 26 17:19:52 2021 pipeline
drwxr-xr-x keith users 4 KB Fri Sep 11 10:08:17 2020 pubmed
drwxr-xr-x keith users 4 KB Sat Jul 17 17:12:03 2021 pubtator-co-citation
drwxr-xr-x keith users 4 KB Wed Jul  7 06:18:57 2021 pubtator-gene-disease
drwxr-xr-x keith users 4 KB Mon Sep  6 19:34:13 2021 pubtator-kif14
drwxr-xr-x keith users 4 KB Tue Jun 15 15:42:58 2021 sample-pca
drwxr-xr-x keith users 4 KB Wed Jul 21 14:12:51 2021 sig
drwxr-xr-x keith keith 4 KB Fri Apr  1 21:13:19 2022 sig-surv
drwxr-xr-x keith users 4 KB Sat Jun 19 18:20:19 2021 stringdb
drwxr-xr-x keith users 4 KB Wed Mar 24 09:20:15 2021 study-top-gene-heatmap
drwxr-xr-x keith keith 4 KB Wed Apr  6 10:23:20 2022 surv-os-vs-pfs
drwxr-xr-x keith keith 4 KB Thu Oct  7 21:51:47 2021 survival-vs-gene-expr
drwxr-xr-x keith users 4 KB Tue Nov 24 15:21:47 2020 top-genes
drwxr-xr-x keith users 4 KB Wed Jul 21 14:02:38 2021 unused
```

Example sub-project folders for a larger project..

Project directory structure

1. For individual projects / sub-projects, the appropriate file/directory structure depends on the specific *type* of project being created, e.g.:
   a. Snakemake
   b. RMarkdown
   c. Jupyter notebook
   d. Software package
   e. Shiny app
   f. Web app
   g. "lose code"
2. Useful idioms:
   a. "`archive/`" – earlier efforts (decluttering)
   b. "`eda/`" – exploratory analyses
   c. "`doc/`" – notes & refs
3. Related: `cookiecutter`

```
~/d/r/proj/mm25/sig-surv
❯ l
drwxr-xr-x keith keith    4 KB Fri Apr  1 21:13:19 2022 .
drwxr-xr-x keith users    4 KB Wed Apr 13 18:59:06 2022 ..
drwxr-xr-x keith keith    4 KB Fri Apr  1 20:47:36 2022 config
drwxr-xr-x keith keith    4 KB Fri Apr  1 20:51:15 2022 data
drwxr-xr-x keith keith    4 KB Fri Apr  1 20:43:20 2022 output
drwxr-xr-x keith keith    4 KB Fri Apr  1 20:41:41 2022 renv
.rw-r--r-- keith keith   25 B  Fri Apr  1 20:41:41 2022 .gitignore
.rw-r--r-- keith keith  4.4 MB Fri Apr  1 21:13:19 2022 README.html
.rw-r--r-- keith keith 15.4 KB Fri Apr  1 21:09:34 2022 README.Rmd
.rw-r--r-- keith keith  220 B  Fri Apr  1 20:41:41 2022 requirements.txt
```

Example RMarkdown project folder

Data/configuration

```
~/d/r/proj/mm25/pipeline/config main*
❯ l
drwxr-xr-x keith users   4 KB Wed Nov 25 10:49:47 2020 .
drwxr-xr-x keith users   4 KB Mon Jul 26 17:19:52 2021 ..
drwxr-xr-x keith users   4 KB Tue Jul  7 13:05:02 2020 archive
.rw-r--r-- keith users 370 B  Thu Apr 30 20:21:57 2020 config-v1.0.yml
.rw-r--r-- keith users 681 B  Thu Apr 30 20:21:57 2020 config-v1.1.yml
.rw-r--r-- keith users 999 B  Thu Apr 30 20:21:57 2020 config-v1.2.yml
.rw-r--r-- keith users 999 B  Sun May 17 14:56:33 2020 config-v1.3.yml
.rw-r--r-- keith users 893 B  Mon Jun  1 19:16:20 2020 config-v2.0.yml
.rw-r--r-- keith users 893 B  Wed Jul  8 07:59:18 2020 config-v2.1.yml
.rw-r--r-- keith users 893 B  Thu Aug 13 12:32:53 2020 config-v3.0.yml
.rw-r--r-- keith users 893 B  Tue Aug 25 14:23:30 2020 config-v3.1.yml
.rw-r--r-- keith users 1.3 KB Wed Nov 25 11:03:36 2020 config-v3.2.yml
```

Sometimes it's nice to keep multiple explicitly versioned YAML config files for a project

- Almost all of my projects contain a "`config/`" folder with one or more **YAML config files**..
  - (June 9: Caroline Esnault talk :))
- **Data** is a bit trickier…
  - "input" data
    - `data/` sub-folder
    - `/data/xx/` system folder
    - *reproducibility*: have pipeline/analysis fetch any relevant data not included in vcs..
  - "output" data
    - `data/` sub-folder
    - `/data/proj/foo/xx/`
    - make it easy for users to use your data
  - *suggestion*: for externally-downloaded data that you download by hand, include a `README.md` with info on the data source, version, steps to retrieve, etc.
  - *Data package* provide a nice way to package data & metadata together

Conda & Friends

- *Use conda for everything..*
  - it's really easy to do so..
  - it will improve the reproducibility of your projects significantly
  - Snakemake has built-in support for it
- `requirements.txt`
  - means of specifying *software dependencies*
  - used by conda & pip
- Other tools for managing virtual environments
  - virtualenv (python)
  - renv (R)
  - `activate` (julia)

```
❯ cat geo/requirements.txt
File: geo/requirements.txt
Size: 120 B

python ≥ 3.9
snakemake-minimal
r-annotables
r-devtools
r-jsonlite
r-tidyverse
bioconductor-biomart
bioconductor-geoquery
```

requirements.txt example

Versioning & Version control

- Use version control for all of your projects
  - Git + github is a popular choice, but others work well, too
- Some reasons why:
  - reproducibility
  - collaboration
  - versioning
  - trying out “sandboxed” ideas
  - saving future-you significant heartache
- Versioning
  - Useful way to keep track of changes / alternate versions of analyses
  - Software vs. config versions
  - `/data/proj/foo/v0.1/`

Art: Anthony Pucelle

Documentation

A few possible strategies for project docs:

1. README.md
2. “`proj/foo/docs/`”
3. “`notes/proj/foo/`”

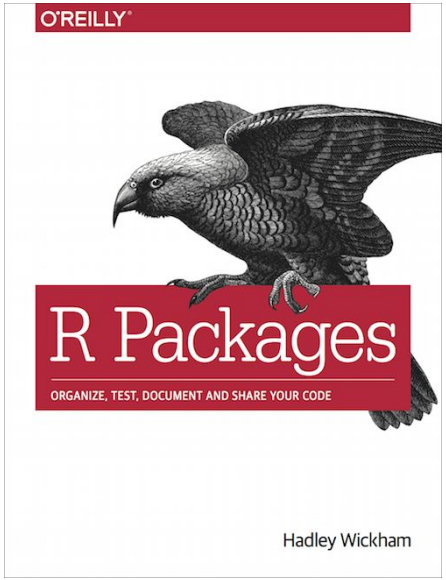
Useful notes to consider writing:

- `next-steps.md`
- `ideas.md` (or, `ideas/xx.md`)
- `questions.md`
- `changelog.md`

Testing

- “Testing” ~ writing small bits of code to check that your code does what you think it does..
- Useful for *robustness & reproducibility*
- Used by all major software applications you use (or else, chances are, you wouldn’t be using them..)
- *Testing frameworks* simply & speed up the process of writing test code
  - Pytest (Python)
  - testthat (R)
  - `@testset` (Julia)

Packaging



https://r-pkgs.org/index.html

1. “Packaging” ~> making it easy for others to install & use your code
2. Language-specific approaches
   a. `setuptools` (Python)
   b. `devtools` (R)
   c. `PkgTemplates.jl` (Julia)
3. Conda
   a. `conda-build`
   b. `conda_r_skeleton_helper`
4. OS
   a. Brew (OS X)
   b. PKGBUILD, .deb, etc. (Linux)

Containers 🐳

- If you want to go one step further, with respect to **reproducibility**, *containers* are a great option.
- Containers ~ sandboxed environments, down the the level of the OS.
- **Docker** is commonly used (Singularity is another example..)
- Easy to work with (`Dockerfile`)
- Also very useful for web/cloud deployment.

Web Deployment

- Shiny (R)
- Dash (Python)
- `docker-compose`
  - control multiple containers/services
  - YAML config
- Example web app structure:
  - Back-end: FastAPI + gunicorn
  - Front-end: Node.js + React
    - `create-react-app`
- Deploying to a new server / cloud: `docker-compose up`

docker-compose.yml

```yaml
version: '3'

services:
  fastapi:
    build:
      context: api/
      dockerfile: Dockerfile
    command: sh -c "gunicorn main:app --timeout 6000
    environment:
      - "PYTHONUNBUFFERED=1"
    ports:
      - "5000:5000"
    volumes:
      - "/data/proj/pubtator/2022-01-08/:/data"
  node:
    build:
      context: node/
      dockerfile: Dockerfile
    command: "yarn start"
    user: "node"
    working_dir: /home/node/app
    ports:
      - "81:81"
```

example `docker-compose.yml` for a FastAPI + React web app

Thank you!