project_quickstart Documentation

Release 0.3.7

Antonio J Berlanga-Taylor

Jun 11, 2019

Contents:

[1] [2] [3]

1 project_quickstart

Boilerplate tools and templates for setting up a data analysis project.

Create a new directory, subfolders and files that will help quickstart your data science project with packaging, testing, scripts, reporting and other templates.

Quickstart:

```
pip install project_quickstart
project_quickstart --help
project_quickstart -n my_super_project
```

[1] https://travis-ci.org/AntonioJBT/project_quickstart

[2] http://project-quickstart.readthedocs.io/en/latest/?badge=latest

[3] https://zenodo.org/badge/latestdoi/79537885

This tool was produced with the following in mind:

- Reproducibility concepts and best practice implementation
- Use of Ruffus[4] as a pipeline tool and CGAT tools[5] for support
- Python[6] programming and packaging[7]
- restructuredText[8] and Sphinx[9] for reporting
- Travis[10] and tox[11] for testing
- Conda[12] and Docker[13] for management and development
- GitHub[14] for version control

I've additionally put some basic instructions/reminders to link GitHub with:

- ReadtheDocs[15] (to easily render your documentation online)
- Zenodo[16] (for archiving your code and generating a DOI)
- Travis CI (to integrate code testing)

1.1 Some of the reasoning

- Analyses are rarely only run once even within the same project. Automating as much as possible saves time and errors. The setup can be costly initially but over time this pays off, particularly when needing to track errors, confirming results, handing over or reconstructing the history and reasoning (even to yourself months later).
- Usually a project is built around one data set/experiment/question but even in this case it's easy to see potential gains from automating and packaging.
- Packaging your project can help with reproducibility, freezing code, version control, collaboration and general mental sanity (while managing a project at least).
- Later on the code or parts of it could be extracted to work on the general case as a separate entity.
- This package is based on Python but the same applies to other languages. See discussions on writing your projects as packages in R (R. Flight[17], H. Parker[18] (also here[19]),

[4] http://www.ruffus.org.uk/

[5] http://www.cgat.org/cgat/Tools/the-cgat-code-collection

[6] https://www.python.org/

[7] https://packaging.python.org/

[8] http://docutils.sourceforge.net/rst.html

[9] http://www.sphinx-doc.org/en/stable/

[10] https://travis-ci.org/

[11] https://tox.readthedocs.io/en/latest/

[12] http://conda.pydata.org/docs/#

[13] https://www.docker.com/

[14] https://github.com/

[15] https://readthedocs.org/

[16] https://guides.github.com/activities/citable-code/

[17] http://rmflight.github.io/posts/2014/07/analyses_as_packages.html

[18] https://hilaryparker.com/2014/04/29/writing-an-r-package-from-scratch/

[19] https://hilaryparker.com/2013/04/03/personal-r-packages/

H. Wickham[20] and others[21] for example). See Hadley Wickham's R[22] ecosystem[23] and book[24].

1.2 Installation

Tested on macOS, Windows and Linux.

Please raise an issue if you have problems.

Dependencies

- Python >3.5
- If you run the examples option you will need many more tools. See the Dockerfiles included for specific instructions.

From GitHub

To download and install from GitHub (you need git installed), at the command line do:

```
pip install git+git://github.com/AntonioJBT/project_quickstart.git
```

or clone from GitHub (https example, you may need ssh):

```
git clone https://github.com/AntonioJBT/project_quickstart.git
cd project_quickstart
python setup.py install
```

See stackoverflow[25] example and pip docs[26] for further help and explanations pip and git installs.

pip

```
pip install project_quickstart
```

1.3 Usage

Create a project directory skeleton. From the command line do:

```
project_quickstart --help
project_quickstart -n my_super_project
project_quickstart --script-R my_super_script # which will create an R script␣
↪template called my_super_script.R
project_quickstart --script-python my_super_script # which will create a Python␣
↪script template called my_super_script.py
```

[20] http://r-pkgs.had.co.nz/intro.html
[21] https://github.com/kbroman/broman
[22] https://www.r-project.org/
[23] http://hadley.nz/
[24] http://r-pkgs.had.co.nz/
[25] http://stackoverflow.com/questions/8247605/configuring-so-that-pip-install-can-work-from-github
[26] https://pip.pypa.io/en/stable/reference/pip_install/#vcs-support/pip_install.html#vcs-support

This will create data, code, manuscript and results directories along with Python and R template scripts and the necessary skeleton files for Python packaging, Docker, Travis CI, Sphinx, etc.

The –script options will create additional copies of script templates in the current working directory.

A pipeline example

To run an example of a project with scripts, pipeline and report, you'll need to install several additional tools. See the Dockerfiles on how to do this for Linux.

To create and run within a conda environment you can try the following bash script. You may need to run commands manually if it fails though and there are other dependencies which need manual installation. You'll need Cairo graphics libraries, Inkscape and latex for the plotting scripts and report.

Note that if you intend to run the pipeline example below, you may want to install cgat-core[27] before and within that environment install the additional tools required.

```
wget https://raw.githubusercontent.com/AntonioJBT/project_quickstart/master/
↪requirements_pq_example.sh
bash requirements_pq_example.sh # provided as an example, you probably want to␣
↪inspect it first and run commands manually
# If you're on Mac OS X you'll also need:
conda install python.app
```

Once you have everything installed, run:

```
conda activate pq_test
project_quickstart --example # will create a project with runnable scripts and␣
↪pipeline
cd pq_example/results
python ../code/pq_example/pq_example.py --createDF -O ../data/raw/my_dataframe #␣
↪Generate some start-up data in the raw data folder
ln -s ../data/raw/my_dataframe.tsv . # Create a symbolic link to your results␣
↪directory
Rscript ../code/pq_example/pq_example.R -I my_dataframe.tsv
Rscript ../code/pq_example/plot_pq_example_pandas.R -I my_dataframe.tsv
# You'll need pythonw for matplotlib if on a Mac:
python ../code/pq_example/svgutils_pq_example.py \
                     --plotA=my_dataframe_gender_glucose_boxplot.svg \
                     --plotB=my_dataframe_age_histogram.svg \
                     -O F1_my_dataframe
```

You can also try:

```
Rscript ../code/pq_example/pq_example_mtcars.R
Rscript ../code/pq_example/plot_pq_example_mtcars.R
python ../code/pq_example/svgutils_pq_example.py --plotA=mtcars_cyl_wt_boxplot_2.svg \
                                                 --plotB=mtcars_hp_qsec_scatterplot.
↪svg \
                                                 -O F1_mtcars
python ../code/pq_example/svgutils_pq_example.py --plotA=mtcars_wt_histogram.svg  \
                                                 --plotB=mtcars_boxplot_lm.svg \
                                                 -O F2_mtcars
```

[27] https://github.com/cgat-developers/cgat-core

svgutils_pq_example.py is a simple wrapper for the python package svgutils, don't expect too much. You can modify the script, play around with scale(), move(), Grid(), etc.

You can get a simple example of a report, based on sphinx-quickstart[28], by doing:

```
cp -r ../code/pq_example/pipeline_pq_example/pipeline_report .
cd pipeline_report
ln -s ../../code/pq_example/pipeline_pq_example/configuration/pipeline.yml .
make html
ln -s _build/html/report_pipeline_pq_example.html .
make latexpdf
ln -s _build/latex/pq_example.pdf .
```

You can run most of this with a bash script:

```
project_quickstart --example # will create a project with runnable scripts and␣
↪pipeline
cd pq_example/results
# Use pythonw if on a Mac, otherwise python:
bash ../code/pq_example/examples.sh python > examples.log
open pipeline_report/pq_example.pdf pipeline_report/report_pipeline_pq_example.html #␣
↪in a Mac
```

If you have cgat-core[29] installed you can try the following:

```
project_quickstart --example # will create a project with runnable scripts and␣
↪pipeline
cd pq_example/results
python ../code/pq_example/pipeline_pq_example/pipeline_pq_example.py --help
# Get a copy of the configuration files, you need to modify the yml file manually:
python ../code/pq_example/pipeline_pq_example/pipeline_pq_example.py config
python ../code/pq_example/pipeline_pq_example/pipeline_pq_example.py show full
python ../code/pq_example/pipeline_pq_example/pipeline_pq_example.py printconfig
python ../code/pq_example/pipeline_pq_example/pipeline_pq_example.py plot full
python ../code/pq_example/pipeline_pq_example/pipeline_pq_example.py make full --local
python ../code/pq_example/pipeline_pq_example/pipeline_pq_example.py make make_report␣
↪--local
open pipeline_report/_build/latex/pq_example.pdf pipeline_report/_build/html/report_
↪pipeline_pq_example.html
```

If submitting to a cluster consider using a ~/.cgat.yml file (see an example[30]) for configuration and submitting like:

You can also see this tutorial on pipelines[31] with Ruffus and CGAT tools from Ian Sudbery's lab. I've also created a repository to keep some tests and minimal setup for Ruffus, DRMAA and cgat-core which may be helpful (pipeline_example[32]).

1.4 Citation

This is a simple utility tool but if you find a way to cite it please do so (!):

[28] http://www.sphinx-doc.org/en/stable/index.html

[29] https://github.com/cgat-developers/cgat-core

[30] https://github.com/AntonioJBT/pipeline_example/blob/master/Docker_and_config_file_examples/cgat.yml

[31] https://github.com/sudlab/pipeline_tutorial

[32] https://github.com/AntonioJBT/pipeline_example

[33]

1.5 Contribute

Issue Tracker[34]

You are more than welcome to fork or submit pull requests (!).

1.6 Change log

v0.4 (future)

v0.3

- updated to cgat-core 0.5.6
- switched from ini to yml
- minor bugs in bash example
- included function to find path to R script being executed
- minor bug in the example report conf.py
- added ggthem template
- added scripts option in setup.py template to run package scripts from CLI
- added rsync example command and instructions for remote copies
- added Ruffus/CGAT simplified pipeline template script
- added example scripts and pipeline, option '–example'

v0.2

- Initial release

1.7 License

GPL-3

1.8 More details and suggestions

Project workflow

1. Run this package to setup folders, github repo structure, code testing, py package files, etc.
2. Download packages, tools, etc. Setup Docker, conda kaspel, or other form of tracking environment, packages and their versions.
3. Manually connect GitHub with integrated services (Travis CI, Zenodo, RTD).
4. Code and test code with tox, travis and py.test

[33] https://zenodo.org/badge/latestdoi/79537885
[34] https://github.com/AntonioJBT/project_quickstart/issues

5. Analyse
6. Create new scripts, new pipelines, test them
7. Document code as you go, update with sphinx autodoc
8. Generate internal report with plots, text, etc.
9. Freeze with release tag + zenodo archiving and/or tar ball with py sdist
10. Repeat cycle

Even if the code is project specific it can still be versioned, frozen and archived for reproducibility purposes later on.

You can later on build computational pipelines using for example a pipeline quickstart tool based on a Ruffus and CGAT framework[35].

You will need to install other software (e.g. R, Ruffus[36], Sphinx[37], etc.) to make full use depending on your preferences.

project_quickstart usage notes

project_quickstart.py creates a folder structure with file templates for:

- data
- code
- results
- manuscript (reports, general documents, references, etc.)

See this layout[38] for one explanation on organising Python projects

project_quickstart.py copies the contents of project_quickstart/templates/project_template/ so as to have all the skeleton files needed for:

- Github repository files (but not .git) like: .gitignore, README, THANKS, TODO, LICENCE, etc.
- Travis testing files, tests dir with skeleton files
- Tox python testing
- Python packaging files
- Dockerfile
- etc
- Zenodo, see Zenodo GitHub guide[39]. Allow permissions and then with each tag release Zenodo archives the repo and gives it a DOI. See also SSI blog[40] on Zenodo.

[35] https://github.com/cgat-developers/cgat-flow/blob/master/CGATPipelines/pipeline_quickstart.py

[36] http://www.ruffus.org.uk/

[37] http://www.sphinx-doc.org/en/stable/

[38] https://www.cgat.org/downloads/public/cgatpipelines/documentation/Reference.html#term-pipeline-scripts

[39] https://guides.github.com/activities/citable-code/

[40] https://www.software.ac.uk/blog/2016-09-26-making-code-citable-zenodo-and-github

These go into the code directory.

Make additional script template copies with project_quickstart.py (located in project_quickstart/templates/project_template/).

Testing

- See tox, travis and py.test for a proper setup of py virtualenv, CI and unit testing respectively.
- Check travis setup, add pep8 and flake8 to improve your code.
- See CGAT docs for an explanation on testing[41].

Upload code to GitHub

To create a repository after having already created files do the following:

> Manually create a blank (no files at all) repository online in your GitHub account

In your local machine, under my_project_xxx/code/ do:

```
git init
git add *
git commit -am 'great message'
git remote add origin https://github.com/user_xxx/my_project_xxx.git
git push -u origin master

# To copy on any other machine simply run:
git clone https://github.com/user_xxx/my_project_xxx.git
```

Documentation

After setting up a project, edit the INI and rst files so that variables that get repeated (such as project name, author, date, etc.) are automatically passed to the various files that need them (setup.py, Dockerfile, manuscript_template, etc.). These will get substituted when running python setup.py or rendering rst documents for instance.

Different renderers can give slightly different results (e.g. GitHub, RTD, Sphinx[42], rst2pdf, etc.)

rst2pdf can substitute rst variables but pandoc doesn't seem to do it properly.

See some notes in CGAT reports[43].

- Add Python docs with rst, Sphinx[44], quickstart[45]
- Check doctests[46]

[41] https://www.cgat.org/downloads/public/cgat/documentation/testing.html#testing

[42] http://www.sphinx-doc.org/en/stable/

[43] https://www.cgat.org/downloads/public/cgatpipelines/documentation/PipelineReports.html#writingreports

[44] http://www.sphinx-doc.org/en/stable/

[45] http://thomas-cokelaer.info/tutorials/sphinx/quickstart.html

[46] http://thomas-cokelaer.info/tutorials/sphinx/doctest.html

- See this tutorial[47] for Sphinx[48] and general python packaging/workflow
- See also Jeff Knupp's tutorial[49] and other similar blogs[50] on Python packaging.

Try to follow Python style guides. See projects where these have been slightly adapted as an example (CGAT style[51]).

Dependencies

These can become a nightmare as many programs are needed when running pipelines in biomedical research. Try to stick to one package manager, such as conda. Pip and conda usually play well and complement each other.

Docker images and testing can also make things easier for reproducible environments.

To run the example pipeline above see the Dockerfiles in this repository for installation instructions and images you can try.

Archiving and computing environment

You can use releases as code freezes. These can be public, remote, local, private, etc.

For example, you can greate tags for commits on GitHub, these create compressed files with versioning. See git tagging[52] on how to do this.

For example, if you want to tag and version a previous commit, do the following:

```
# Update version.py if needed
# Check the tag history:
git tag

# Check the commit log and copy the commit reference:
git log --pretty=oneline

# Create a tag, give it a version, internal message and point it to the commit you want to tag:
git tag -a v0.1 -m "code freeze for draft 1, 23 June 2017" 7c3c7b76e4e3b47016b4f899c3aa093a44c5e053

# Push the tag
# By default, the git push command does not transfer tags to remote servers, so run:
git push origin v0.1

# You'll then need to click around in the GitHub repository to formally publish the release.
```

[47] https://jeffknupp.com/blog/2013/08/16/open-sourcing-a-python-project-the-right-way/
[48] http://www.sphinx-doc.org/en/stable/
[49] https://www.jeffknupp.com/blog/2013/08/16/open-sourcing-a-python-project-the-right-way/
[50] https://www.pydanny.com/cookie-project-templates-made-easy.html
[51] https://www.cgat.org/downloads/public/cgat/documentation/styleguide.html#styleguide
[52] https://git-scm.com/book/en/v2/Git-Basics-Tagging

See bioconda[53], contributing a recipe[54] and guidelines[55] to help manage the project's dependencies and computational environment.

If your code is useful to others, you can make it available with PyPI, create a Dockerfile and/or Conda recipe.

Note: Many links are tutorials I've come across, if you know of other good ones please share them.

Make sure to check the official sites and follow their tutorials for each of the tools as a primary source however.

Feel free to fork, raise issues and send pull requests.

Similar packages

I discovered CookieCutter[56] while working on this. It probably does what I have setup here better, with useful features, flexibility and many templates for different types of projects.

See its data-science[57] and reproducibility[58] templates for example.

2 Thanks

Many thanks to Andreas Heger[59] and CGAT[60]. The project follows CGAT approaches and tools. The script pipeline_template.py is a simplified version of this template[61].

For the full experience, including powerful and flexible reporting, please install CGAT tools[62].

3 Things to do

Date 25 April 2017

- Conda recipe
- Zenodo hook
- Keep track of installations for Docker. Check conda kapsel[63] and here[64].

[53] https://bioconda.github.io/index.html
[54] https://bioconda.github.io/contribute-a-recipe.html
[55] https://bioconda.github.io/guidelines.html
[56] https://github.com/audreyr/cookiecutter-pypackage
[57] https://github.com/drivendata/cookiecutter-data-science
[58] https://github.com/mkrapp/cookiecutter-reproducible-science
[59] https://github.com/AndreasHeger
[60] http://www.cgat.org
[61] https://github.com/CGATOxford/CGATPipelines/blob/master/CGATPipelines/pipeline_template_data/pipeline_template_full.py
[62] https://github.com/CGATOxford
[63] https://conda.io/docs/kapsel/
[64] https://github.com/conda/kapsel

- Add a project checklist template (eg with comp bio best practice checklist, conda recipe, zenodo deposit, etc.)
- Automatically run pipeline_quickstart.py at project creation?
- Add R's sessionInfo()
- Add scan/ppt pipeline workflow plus notes
- Sphinx and docs, see here[65] for a different solution.
- Have methods and plot scripts output legend*.rst with parameters used, method*.rst and generic interpretation*.rst files output to disk to then glob into manuscript template.

4 Indices and tables

- genindex
- modindex
- search

[65] https://daler.github.io/sphinxdoc-test/includeme.html