**Sequence analysis**

**wft4galaxy: a workflow testing tool for galaxy**

Marco Enrico Piras*, Luca Pireddu and Gianluigi Zanetti

Data Intensive Computing, CRS4 (Center of Advanced Studies, Research and Development in Sardinia), Pula 09010, Italy

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on May 2, 2017; revised on June 9, 2017; editorial decision on July 12, 2017; accepted on July 21, 2017

**Abstract**

**Motivation:** Workflow managers for scientific analysis provide a high-level programming platform facilitating standardization, automation, collaboration and access to sophisticated computing resources. The Galaxy workflow manager provides a prime example of this type of platform. As compositions of simpler tools, workflows effectively comprise specialized computer programs implementing often very complex analysis procedures. To date, no simple way to automatically test Galaxy workflows and ensure their correctness has appeared in the literature.

**Results:** With wft4galaxy we offer a tool to bring automated testing to Galaxy workflows, making it feasible to bring continuous integration to their development and ensuring that defects are detected promptly.

**Availability and implementation:** Available at https://github.com/phnmnl/wft4galaxy under the Academic Free License v3.0.

**Contact:** marcoenrico.piras@crs4.it

1 Introduction

Typical bioinformatics analyses involve a number of steps to extract information from various forms of raw data; these analysis procedures are often referred to as workflows or pipelines. The pattern is so common that a number of workflow managers have been created (Leipzig, 2016) to provide high-level platforms on which to implement these procedures, supporting simpler and more robust implementations than would be reasonably feasible with simple shell scripting. Thus, with the help of workflow managers it becomes practical to implement ever more complex workflows—in fact, workflows with tens of steps are not uncommon. The increase in complexity is accompanied by an increased risk of defects. At best, these will crash and interrupt an analysis procedure; at worst, they will produce subtly wrong results which may only be detected much later. Therefore, given the risks, it seems wise to adopt a mitigation strategy: it is the authors’ opinion the workflow development should be as rigorous as any other kind of software development, especially in light of the growing trend to release and share standard workflows. Automated workflow testing then should become an important part of the development process—one which as of yet has not received a lot of attention.

In this work, we present wft4galaxy, the WorkFlow Testing tool for the Galaxy data analysis platform (Afgan et al., 2016). To the best of the authors’ knowledge, wft4galaxy is the first published automatic workflow testing tool for Galaxy. wft4galaxy works based on the unit testing model: a test case is specified as a set of input datasets and parameters, expected output datasets and the workflow itself; the workflow is run and the actual and expected outputs are compared. The testing tool uses Galaxy’s RESTful API through the object-oriented interface of the BioBlend package (Leo et al., 2014) to automate the entire test execution operation as well as much of the work required to compose the test cases. Of note, our tool is currently used in production within the PhenoMeNal project (http://phenomenal-h2020.eu) to continuously test the workflows integrated in the platform.

2 Materials and methods

The testing model provided by wft4galaxy is centered around test cases. Each test case defines a workflow and a specific scenario which is to be tested. It contains: the path of the workflow definition file; optionally, the parameters of the various workflow steps; the
datasets to be used as workflow inputs; and, finally, expected output datasets. Any number of test cases are collected in a YAML file such as the one shown in Listing 1.

The test definition file is the input for the wft4galaxy test runner, which automatically executes the entire collection of tests. For each test, the runner connects to an available Galaxy instance provided by the user and then, through the Galaxy API: (i) uploads the workflow; (ii) creates a new Galaxy history; (iii) uploads all the input datasets; (iv) runs the workflow; and (v) downloads output datasets. The runner then compares the output to the expected datasets using a comparator function (by default, simple file equality). Finally, all test results are collected and reported.

Listing 1. Example of “Test definition file”

```python
dataflows:
  test_case:
    file: “workflow.ga”
    params:
      3:
        “respC”: “gender”
    inputs:
      “DataMatrix”: “input/dataMatrix.tsv”
    expected:
      output:
        file: “expected/Univariate_variableMetadata.tsv”
        comparator: “comparators.csv_src”
```

As an aid to users having to write test definitions, wft4galaxy provides a template generator: this tool creates a blank definition and a further simplifies the creation of workflow test cases through the wft4galaxy-wizard, which generates ‘ready-to-run’ workflow test cases from existing Galaxy histories. With the wizard, the steps to create a working test case are reduced to the following. First, the user creates a new history with the required input datasets. Then, the user runs the workflow, after setting any required tool parameters. The workflow should produce a set of new output datasets within the same history. Now, assuming that the workflow has produced correct results, the history can be transformed into a test case by running the wft4galaxy-wizard. The wizard will inspect the history to extract and store the underlying workflow (i.e. its .ga definition file) and all its datasets (both input and output) in a new test directory. The suite definition file is then automatically generated: it will contain a single test case configured to execute the extracted workflow on the input datasets and compare the generated datasets to the outputs of the recorded workflow run.

Programmatic Usage. To integrate wft4galaxy with third-party tools or for elaborate automation requirements, it can also be used programmatically. Its API is organized around two main packages: wft4galaxy.core and wft4galaxy.wrapper. The former contains the core logic of the test framework, exposing an Object-Oriented (OO) API for defining and running test cases and test suites programmatically (Listing 3 shows an example of its usage). On the other hand, the latter package contains an OO representation of Galaxy workflows and histories providing a simplified way to inspect inputs, parameters and outputs of tested Galaxy workflows and histories.

Listing 3. Programmatic test case definition and execution.

```python
from wft4galaxy.core import WorkflowTestCase
workflow_filename = “workflow.ga”
inputs = [“InputText”: (“file”: “input”)]
expected_outputs = [“OutputText”: (“file”: “expected_outputs”)]
test_case = WorkflowTestCase(workflow_filename, inputs, expected_outputs)
test_result = test_case.run(enable_logger=True)
test_result.check_output(”OutputText”)```

Docker integration. wft4galaxy can easily run within a Docker container, completely avoiding any installation hassles. This feature is particularly useful when using continuous integration (CI) services such as Travis CI and Jenkins, where users benefit from not using root privileges for installing new software packages. To simplify the usage of the wft4galaxy Docker image, we provide the wft4galaxy-docker script, which configures the container execution to use wft4galaxy as if it were locally installed. The script can be run standalone, after simply downloading it from the wft4galaxy GitHub repository.

3 Conclusion

wft4galaxy is a tool to simplify and automate workflow tests. It supports the adoption of ‘unit testing’ and continuous integration into the workflow development and maintenance process. Its native support for Docker enables easy integration with specialized CI systems, such as Jenkins. Indeed, within the PhenoMeNal project, Jenkins with wft4galaxy are used to test complex workflows such as the ones described by De Atauri et al. (De Atauri et al., 2016). Although in its current version wft4galaxy is tied to the Galaxy platform, in the future we would like to investigate the feasibility of extending it to work with other workflow management systems and, in particular, implementations of the Common Workflow Language (Amstutz et al., 2016). The full documentation for wft4galaxy is available at http://wft4galaxy.readthedocs.io.
Acknowledgement

The authors would like to thank the fellow members of the PhenoMeNal team for their valuable feedback.

Funding

This work was partially supported by the European Commission’s Horizon2020 programme under the PhenoMeNal project (grant agreement number 654241) and by the Region of Sardinia under project ABLE.

Conflict of Interest: none declared.

References

Afgan, E. et al. (2016) The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. *Nucleic Acids Res.*, 44, gkw343.

Amstutz, P. et al. (2016) Common Workflow Language, v1. 0. figshare.

De Atauri, P. et al. (2016) Workflows For Fluxomics In The Framework Of Phenomenal Project.

Leipzig, J. (2016) A review of bioinformatic pipeline frameworks. *Brief. Bioinf.*, 18, bbw020.

Leo, S. et al. (2014) BioBlend.objects: Metacomputing with galaxy. *Bioinformatics*, 30, 2816–2817.