Bioinformatics Advance Access published July 1, 2014

BIOINFORMATICS APPLICATIONS NOTE

2014, pages 1–2
doi:10.1093/bioinformatics/btu386

Sequence analysis

Advance Access publication June 12, 2014

BioBlend.objects: metacomputing with Galaxy

Simone Leo[1,2,*], Luca Pireddu[1,2], Gianmauro Cuccuru[1], Luca Lianas[1], Nicola Soranzo[1], Enis Afgan[3] and Gianluigi Zanetti[1]

[1]CRS4, Polaris, 09010 Pula (CA), [2]Università degli Studi di Cagliari, via Università 40, 09124 Cagliari, Italy and [3]Ruđer Bošković Institute, 10000 Zagreb, Croatia

Associate Editor: Alfonso Valencia

ABSTRACT

Summary: BioBlend.objects is a new component of the BioBlend package, adding an object-oriented interface for the Galaxy REST-based application programming interface. It improves support for metacomputing on Galaxy entities by providing higher-level functionality and allowing users to more easily create programs to explore, query and create Galaxy datasets and workflows.

Availability and implementation: BioBlend.objects is available online at https://github.com/afgane/bioblend. The new object-oriented API is implemented by the `galaxy/objects` subpackage.

Contact: simone.leo@crs4.it

Received on April 9, 2014; revised on May 23, 2014; accepted on June 9, 2014

1 INTRODUCTION

In recent times, the massive increase in the amount of data produced by genomic sequencers and other data-intensive acquisition devices used in the life sciences has led to a continuous intensification of the effort required for biological data analysis. Huge and numerous datasets must be processed by complex analysis workflows, articulated in a large number of steps, most of which are highly dependent on many configuration parameters. Data processing frameworks can help mitigate the complexity by simplifying the pipeline execution. An example of such a framework is Galaxy (Goecks *et al.*, 2010), an extremely popular Web application for bioinformatics analysis. It provides a simple way to encapsulate computational tools and datasets in a graphical user interface (GUI), together with a mechanism to keep track of the execution history in a reproducible manner.

However convenient and user-friendly, though, GUIs are ill-suited to automated analysis and bulk processing. For instance, consider a situation that happens regularly with each release of a new reference genome for resequencing, or with the update of sequence alignment software: to ensure that analysis results stay relevant, such events require that the full set of experimental results (e.g. single nucleotide polymorphism discovery) be reevaluated from scratch using the new model data or software. This laborious task requires better support from the computational framework being used, in the form of reliable ways to automate operations, process datasets in bulk and document the analysis performed on any of them. More generally, studies tend to handle a growing numbers of samples; they also tend to last longer than the relatively frequent update cycles for model data and software. Both these conditions pose requirements for such automated bulk data operations that are currently not handled well by GUIs.

To facilitate this sort of processing, Galaxy includes a RESTful (Richardson and Ruby, 2007) application programming interface (API) that allows other programs to control it automatically. However, this API is fairly low level, as it requires users to construct and issue HTTP requests, explicitly handle the standard error cases that occur in such distributed scenarios and take care of data serialization and deserialization in exchanges between the client and the server. This gap in functionality motivated the development of BioBlend (Sloggett *et al.*, 2013), a Python package that hides HTTP communication, error handling and JSON (de)serialization from the user, providing a dictionary-based API that greatly simplifies interaction with the Galaxy server.

However, despite its significant enhancements over the raw low-level interface, BioBlend still leaves room for improvement. For instance, most of the BioBlend API still offers a one-to-one mapping of generic Python dictionaries to the Galaxy REST resources, with no explicit modeling of Galaxy entities and their relationships. Also, the interface fails to isolate client code from changes in the Galaxy API, as it passes to the caller the same dictionary structures that the server sends. Finally, BioBlend does not provide much in the way of ‘rich’ functionality to perform higher-level, sophisticated yet generic tasks, despite being positioned in a prime location in the software stack where it is potentially shared by all the user’s client applications.

In this work we present BioBlend.objects, a Galaxy interface implemented as a new layer above BioBlend. The new API addresses the aforementioned issues with two main features: an object-oriented (OO) programming model, which simplifies development and isolates client code from changes in the Galaxy API and a high-level component that simplifies complex operations and supports *metacomputing* on the information describing the various Galaxy entities. With BioBlend.objects, running a Galaxy workflow requires just a few lines of simple code:

```
from bioblend.galaxy.objects import GalaxyInstance

gi = GalaxyInstance(“URL", “API_KEY”)
wf = gi.workflows.list()[0]
hist = gi.histories.list()[0]
inputs = hist.get_datasets()[:2]
input_map = dict(zip(wf.input_labels, inputs))
params = {"Paste1": {"delimiter": “U"}}
wf.run(input_map, “wf_output", params=params)
```

The new API is described in more detail in Section 2.

*To whom correspondence should be addressed.

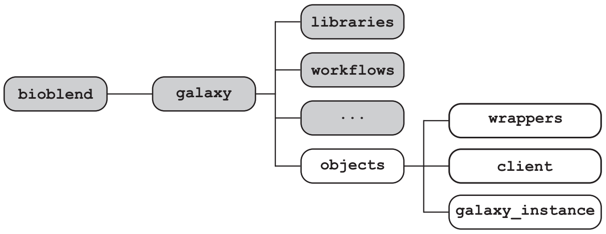

Fig. 1. BioBlend.objects location within BioBlend's logical structure. New modules are displayed in white background

2 METHODS

BioBlend.objects has been developed as a submodule of the original BioBlend library. Hierarchically, the code is currently located at the same level as BioBlend's Galaxy submodules (Fig. 1); in the future, the new API will be moved up to replace the current one. The library consists of two main components: the `wrappers` module, which defines the object structure that mirrors Galaxy's entities, and the `client` module, a high-level code layer built upon the original API to expose a simpler, more concise interface based on the object hierarchy defined in `wrappers`. The `client` module consists of three main classes that encapsulate interactions with Galaxy's most important entities: histories, workflows and libraries. The `galaxy_instance` module contains the `GalaxyInstance` class, which unifies the three clients, acting as a common entry point for all interactions with the Galaxy server.

OO interface: BioBlend.objects provides a compact OO interface for controlling operations performed with Galaxy. The new interface provides objects for the entities that are handled within Galaxy, explicitly modeling the underlying logical structure, and thus is arguably more intuitive than the older one. The OO interface also facilitates development by enabling programmer-friendly features such as code completion in modern development tools and in the IPython shell (Perez and Granger, 2007). Moreover, the new API defines specific objects as its method return values, thus effectively isolating client code from changes in the server-side Galaxy interface. This improvement should result in less painful upgrades of the Galaxy server since, at worst, client compatibility would require updating BioBlend.objects to the latest version.

Metacomputing library: The second principal contribution in BioBlend.objects consists of a set of high-level functions that simplify complex interactions with the Galaxy back end. These functions encapsulate sequences of common operations and implement functionality to support computing on the information describing the various Galaxy entities—i.e. *metacomputing*. Supported features range from running workflows to downloading Galaxy histories and querying for datasets with particular characteristics. This library is a key component of the automation mechanisms used at CRS4 to run its sequencing pipeline and acquire the details of the operations applied to generate each dataset so that they may be stored into OMERO.biobank, a 'computable biobank' that extends OMERO (Allan *et al.*, 2012) to handle data types produced in sequencing and microarray experiments (http://www.openmicroscopy.org/site/support/partner/omero.biobank).

Consider the example given in the introduction, where a workflow is retrieved and run on a set of input datasets, setting a tool parameter at run time. With the original BioBlend API, the same task requires writing the following code:

```
from bioblend.galaxy.objects import GalaxyInstance

gi = GalaxyInstance("URL", "API_KEY")
summaries = gi.workflows.get_workflows()
wf_id = summaries[0]["id"]
wf_info = gi.workflows.show_workflow(wf_id)
hist_infos = gi.histories.get_histories()
hist_id = hist_infos[0]["id"]
hist_dict = gi.histories.show_history(hist_id)
content_info = gi.histories.show_history(hist_id,
  contents=True)
datasets = [gi.histories.show_dataset (hist_id, _["id"])
  for _ in content_info]
inputs = datasets[:2]
input_slots = wf_info["inputs"].keys()
input_map = {
  input_slots[0]: {"id": inputs[0]["id"], "src": "hda"},
  input_slots[1]: {"id": inputs[1]["id"], "src": "hda"}
  }
params = {"Paste1": {"delimiter": "U"}}
gi.workflows.run_workflow(wf_id, input_map,
  history_name="wf_output", params=params)
```

A comparison of the two versions shows how the higher-level interface allows for much more compact code that is easier to read and write.

To keep the example as simple as possible, in the above code fragments we used a 'toy' workflow that merges columns from two input tabular files. However, the git repository (see 'Availability and implementation') includes three examples of interaction with real-world microbiology workflows (Cuccuru *et al.*, 2014) hosted by CRS4's Orione platform (http://orione.crs4.it): bacterial resequencing, bacterial *de novo* assembly and metagenomics. The examples are available under `docs/examples/objects` and can be run on Orione after registering and obtaining an API key (details are included in the examples directory itself).

3 DISCUSSION

BioBlend.objects is designed to model the relations between Galaxy entities. For instance, a History object can be used to retrieve its datasets through an instance method: this makes the API similar to an object-relational mapping library for Galaxy. The BioBlend.objects module has received the support of the original BioBlend team members, who are involved in its development. As such, it is expected to supplant, in the future, the original programming interface.

Funding: CRS4 work was partially supported by a Wellcome Trust Strategic Award [095931/Z/11/Z]. S.L. and L.P. have performed their activity within the context of the PhD program in Biomedical Engineering at the University of Cagliari, Italy.

Conflict of Interest: none declared.

REFERENCES

Allan,C. *et al.* (2012) OMERO: flexible, model-driven data management for experimental biology. *Nat. Methods*, **9**, 245–253.

Cuccuru,G. *et al.* (2014) Orione, a web-based framework for NGS analysis in microbiology. *Bioinformatics*, doi:10.1093/bioinformatics/btu135.

Goecks,J. *et al.* (2010) Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol.*, **11**, R86.

Perez,F. and Granger,B.E. (2007) IPython: a system for interactive scientific computing. *Comput. Sci. Eng.*, **9**, 21–29.

Richardson,L. and Ruby,S. (2007) *RESTful Web Services*. O'Reilly Media, Sebastopol, CA, USA.

Sloggett,C. *et al.* (2013) BioBlend: automating pipeline analyses within Galaxy and CloudMan. *Bioinformatics*, **29**, 1685–1686.