Nicola Soranzo

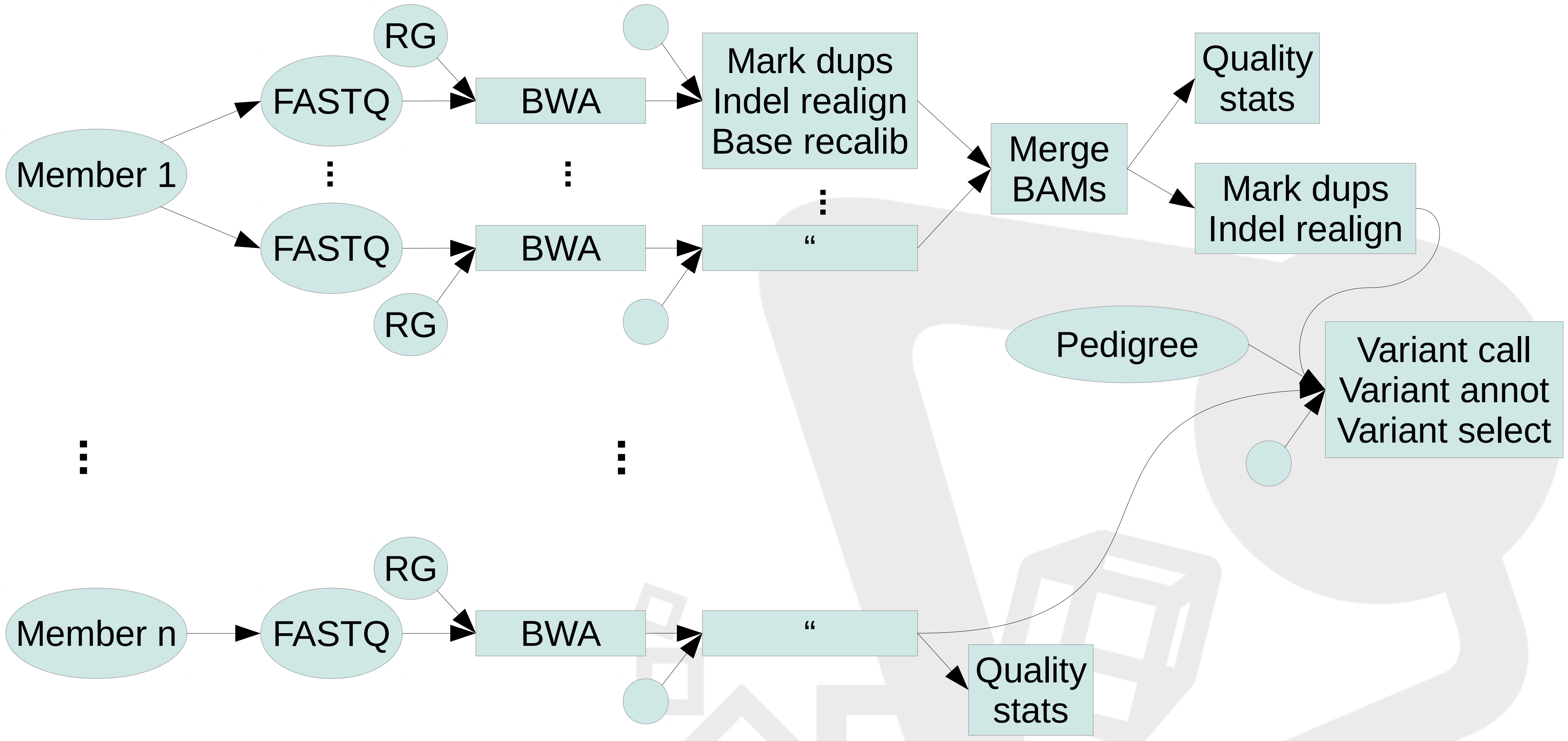
CRS4 Bioinformatica, Pula, Italy

BioBlend and BioBlend.objects: high level API for scripting Galaxy

1st German Galaxy Developer workshop, Freiburg im Breisgau, October 2nd 2014

Motivation

- Exome sequencing of families (GATK2-based)



Galaxy API

- Interact programmatically with a Galaxy server
  - Automate repetitive jobs
  - Enable complex control: branching and looping
  - Continue to use Galaxy to design workflows and to visualize outputs
- **RESTful**: client–server, stateless, cacheable, URIs, JSON data type, standard HTTP methods (GET, PUT, POST, DELETE) and status codes

Galaxy API functionality

- You can do almost everything:
  - Manage histories and datasets
  - Upload and download data
  - Run tools and workflows, ...
- Most functions require authentication
  - API key: alphanumeric string (32 chars) identifying a registered user
- Also admin functions can be performed:
  - manage data libraries, tool shed repositories, users, quotas, roles...

API controllers

```
$ cd galaxy-central/lib/galaxy/webapps/galaxy/api/
$ grep -l expose *.py
annotations.py         groups.py             provenance.py
authenticate.py        group_users.py        quotas.py
configuration.py       histories.py          requests.py
dataset_collections.py history_contents.py   request_types.py
datasets.py            item_tags.py          roles.py
datatypes.py           job_files.py          samples.py
extended_metadata.py   jobs.py               search.py
folder_contents.py     lda_datasets.py       tool_shed_repositories.py
folders.py             libraries.py          tools.py
forms.py               library_contents.py   users.py
ftp_files.py           metrics.py            visualizations.py
genomes.py             page_revisions.py     workflows.py
group_roles.py         pages.py
```

GET history collection

```
>>> import json, urllib2
>>> url = 'http://orione.crs4.it/api/histories?
key=2f8808a73f047652caf826aa03f22ca4'
>>> json.loads(urllib2.urlopen(url).read())
[{u'deleted': False,
  u'id': u'842de33dd31748f2',
  u'model_class': u'History',
  u'name': u'Saved history',
  u'published': False,
  u'tags': [],
  u'url': u'/api/histories/842de33dd31748f2'},
 {u'deleted': False,
  u'id': u'0c6638b87851b996',
  u'model_class': u'History',
  u'name': u'Unnamed history',
  u'published': False,
  u'tags': [],
  u'url': u'/api/histories/0c6638b87851b996'}]
```

compose URL

perform GET request, read response, deserialize JSON

returns a list of dictionaries describing the histories owned by the user

GET history by id

```
>>> url =
'http://orione.crs4.it/api/histories/842de33dd31748f2?
key=2f8808a73f047652caf826aa03f22ca4'
>>> json.loads(urllib2.urlopen(url).read())
{u'annotation': u'',
 u'contents_url': u'/api/histories/842de33dd31748f2/contents',
 u'deleted': False,
 u'empty': False,
 u'genome_build': u'?',
 u'id': u'842de33dd31748f2',
 u'importable': False,
 u'model_class': u'History',
 u'name': u'Saved history',
 u'nice_size': u'282 bytes',
 u'published': False,
 u'purged': False,
 u'slug': None,
 u'state': u'ok',
 u'state_details': {...},
 u'state_ids': {...},
 u'tags': [],
 u'user_id': u'4442cc8424317lbb'}
```

history state based on the states of its datasets

GET history contents

```
>>> url =
'http://orione.crs4.it/api/histories/842de33dd31748f2/contents?
key=2f8808a73f047652caf826aa03f22ca4'
>>> json.loads(urllib2.urlopen(url).read())
[{u'deleted': False,
  u'hid': 1,
  u'history_content_type': u'dataset',
  u'history_id': u'842de33dd31748f2',
  u'id': u'09e153364ff6e397',
  u'name': u'1.fastqsanger',
  u'purged': False,
  u'state': u'ok',
  u'type': u'file',
  u'url':
u'/api/histories/842de33dd31748f2/contents/datasets/09e153364ff
6e397',
  u'visible': True},
 {u'deleted': False,
  ...,
  u'visible': True}]
```

list of dictionaries describing the datasets contained in the history

GET history dataset by id

```
>>> url =
'http://orione.crs4.it/api/histories/842de33dd31748f2/contents/datasets/09e153364ff6e397?key=2f8808a73f047652caf826aa03f22ca4'
{u'annotation': None,
 u'deleted': False,
 u'download_url':                  may be used to download the dataset
u'/api/histories/842de33dd31748f2/contents/09e153364ff6e397/display',
 u'file_ext': u'fastqsanger',      Galaxy data type
 u'file_name': u'/opt/galaxy/database/files/000/119/dataset_115.dat',
 u'file_size': 178,
 u'genome_build': u'hg19',      dbkey
 u'hda_ldda': u'hda',
 u'hid': 1,
 u'history_id': u'842de33dd31748f2',
 u'id': u'09e153364ff6e397',
 u'model_class': u'HistoryDatasetAssociation',
 u'name': u'1.fastqsanger',
 u'purged': False,
 u'state': u'ok',
 u'tags': [],
 u'update_time': u'2014-09-28T23:06:41.907533',
 u'visible': True}
```

Create new history with POST

```
>>> data = {'name': 'New history'}
>>> url = 'http://orione.crs4.it/api/histories?key=2f8808a73f047652caf826aa03f22ca4'
>>> req = urllib2.Request(url, headers={'Content-Type': 'application/json'}, data=json.dumps(data))
>>> json.loads(urllib2.urlopen(req).read())
{u'annotation': u'',
 u'deleted': False,
 u'empty': True,
 u'genome_build': None,
 u'id': u'ae0eda04c1b1ee65',
 u'importable': False,
 u'model_class': u'History',
 u'name': u'New history',
 u'nice_size': u'0 bytes',
 u'published': False,
 u'purged': False,
 u'slug': None,
 u'state': u'new',
 ...}
```

prepare data

create a POST HTTP request, serialize data

returns the dictionary of the created history

Galaxy API pros and cons

- Pros:
  - Distributed with Galaxy
  - Well tested
  - Language-agnostic
- Cons:
  - Too low-level

BioBlend

- BioBlend is a **Python2 library** that wraps the functionality of Galaxy and CloudMan APIs
- Development started by Enis Afgan, Nuwan Goonasekera and Clare Sloggett in June 2012. Contributions by other Galaxy Team members and various Galaxy users
- Stable **procedural** API
- **Open source** (MIT license)
- Available via PyPI and https://github.com/afgane/bioblend/

BioBlend modules

- Interaction with a Galaxy server through a GalaxyInstance object

```
>>> import bioblend.galaxy
>>> gi = bioblend.galaxy.GalaxyInstance(url='http://orione.crs4.it',
key='2f8808a73f047652caf826aa03f22ca4')
>>> sorted([_ for _ in gi.__dict__ if
type(gi.__dict__[_]).__name__.endswith('Client')])

['config', 'datasets', 'datatypes', 'forms', 'ftpfiles', 'genomes',
 'groups', 'histories', 'jobs', 'libraries', 'quotas',
 'toolShed', 'tools', 'users', 'visual', 'workflows']
```

list of gi attributes
which are client objects

- Only pages and roles are not wrapped

Get history collection in BioBlend

```
>>>  gi.histories.get_histories()
[{u'deleted': False,
  u'id': u'842de33dd31748f2',
  u'model_class': u'History',
  u'name': u'Saved history',
  u'published': False,
  u'tags': [],
  u'url': u'/api/histories/842de33dd31748f2'},
 {u'deleted': False,
  u'id': u'0c6638b87851b996',
  u'model_class': u'History',
  u'name': u'Unnamed history',
  u'published': False,
  u'tags': [],
  u'url': u'/api/histories/0c6638b87851b996'}]
```

same output of the GET API call, but already deserialized

- Filter by name:

```
>>> gi.histories.get_histories(name='Saved history')
```

Other previous API calls translated in BioBlend

- Get history by id:

```
>>> gi.histories.show_history('842de33dd31748f2')
```

- Get history contents:

```
>>> gi.histories.show_history('842de33dd31748f2', contents=True)
```

- Get history dataset by id:

```
>>> gi.histories.show_dataset('842de33dd31748f2', '09e153364ff6e397')
```

- Create new history:

```
>>> gi.histories.create_history(name='New history')
```

No need to serialize data

BioBlend limitations

- Python-only (but there's also blend4j)
- Functions just deserialize the JSON response
  - No isolation from changes in the Galaxy API
  - Need to extract the entity id for further processing
- No explicit modeling of Galaxy entities and their relationships
- Some complex but generic operations still need many function calls
  - Need for higher-level functionality

BioBlend.objects

- BioBlend.objects is a new module which adds an **object-oriented** interface for the Galaxy API
- Developed by Simone Leo, Luca Pireddu and Nicola Soranzo at CRS4 since October 2013
- Distributed together with BioBlend since v. 0.5.0
- For now limited to datasets, histories, libraries, tools and workflows

BioBlend.objects examples

- Create a GalaxyInstance for BioBlend.objects:

```
>>> import bioblend.galaxy.objects
>>> gi =
bioblend.galaxy.objects.GalaxyInstance(url='http://orione.crs4.it',
api_key='2f8808a73f047652caf826aa03f22ca4')
```

- Get previews of histories:

```
>>> [_.name for _ in gi.histories.get_previews()]
[u'Saved history',
 u'Unnamed history']
```

- Get full History object by name and display dataset names:

```
>>> h = gi.histories.list('Saved history')[0]
>>> [_.name for _ in h.content_infos]
[u'1.fastqsanger', u'FASTQ to FASTA on data 1']
```

BioBlend.objects examples

- Get dataset by name:

```
>>> hda = h.get_datasets('1.fastqsanger')[0]
>>> hda.__dict__
{'container': History(...),
 'deleted': False,
 'file_name':
u'/opt/galaxy/database/files/000/119/dataset_119805.dat',
 'file_size': 178,
 'gi': <bioblend.galaxy.objects.galaxy_instance.GalaxyInstance
at 0x2e74cd0>,
 'id': u'09e153364ff6e397',
 'name': u'1.fastqsanger',
 'state': u'ok',
 'tags': [],
 'wrapped': {...}}
```

- Create new history:

```
>>> new_h = gi.histories.create(name='New history')
```

References

- https://galaxy-central.readthedocs.org/
- http://bioblend.readthedocs.org/
- C. Sloggett, N. Goonasekera, E. Afgan. BioBlend: automating pipeline analyses within Galaxy and CloudMan. *Bioinformatics* 29(13), 1685-1686, 2013
- S. Leo, L. Pireddu, G. Cuccuru, L. Lianas, N. Soranzo, E. Afgan, G. Zanetti. BioBlend.objects: metacomputing with Galaxy. *Bioinformatics* 30 (19), 2816-2817, 2014

Acknowledgments

Simone Leo

Luca Pireddu

Enis Afgan

Gianmauro Cuccuru

Gianluigi Zanetti