Building More Powerful Galaxy Workflows with Dataset Collections

Progress and Plans

John Chilton and the Galaxy Team


Arbitrary # Inputs

Applications run in parallel (once per input)

Merged into one output for subsequent steps.
More Powerful Workflows

Arbitrary # of Inputs (... paired).

Run applications in parallel (one per input).

Merged output for subsequent processing.
API First Development

Initial work focused on building an API for creating and using dataset collections.

Upshot - API is richer than UI currently (especially in stable).

bioblend contains high-level functionality for creating and “viewing” collections in different ways.
Collection Types

Currently two supported type pseudo-plugins - “list” and “paired”.

- Lists can contain arbitrary number of named elements
- Pairs contain a “forward” and “reverse” element.

Types can be combined to build nested types - for instance “list:paired” describes a list of paired datasets.
Upload Some Data...

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Type</th>
<th>Genome</th>
<th>Settings</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>M23C6-ch_1.fq</td>
<td>45.4 MB</td>
<td>fastq-sanger</td>
<td>unspecified (?)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M23C6-ch_2.fq</td>
<td>45.4 MB</td>
<td>fastq-sanger</td>
<td>unspecified (?)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M48C2-ch_1.fq</td>
<td>46.9 MB</td>
<td>fastq-sanger</td>
<td>unspecified (?)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M48C2-ch_2.fq</td>
<td>46.9 MB</td>
<td>fastq-sanger</td>
<td>unspecified (?)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SC14-ch_1.fq</td>
<td>74.4 MB</td>
<td>fastq-sanger</td>
<td>unspecified (?)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SC14-ch_2.fq</td>
<td>74.4 MB</td>
<td>fastq-sanger</td>
<td>unspecified (?)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sequence.fasta</td>
<td>16.9 KB</td>
<td>fastq-sanger</td>
<td>specified (?)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Download data directly from web or upload files from your disk.
Select the Pairs

- 7: sequence.fasta
- 6: SC14-ch_2.fq
- 5: SC14-ch_1.fq
- 4: M486C2-ch_2.fq
- 3: M486C2-ch_1.fq
- 2: M236C4-ch_2.fq
- 1: M236C4-ch_1.fq

Operations on multiple datasets

- Hide datasets
- Unhide datasets
- Delete datasets
- Undelete datasets
- Build Dataset List (Experimental)
- Build Dataset Pair (Experimental)

Build List of Dataset Pairs (Experimental)
Create a Collection...

Create a collection of paired datasets

Collections of paired datasets are ordered lists of dataset pairs (often forward and reverse reads) that can be passed to tools and workflows in order to have analyses done on the entire group. This interface allows you to...

Choose filters
Clear filters

0 unpaired forward - 0 filtered out

Choose filters
Clear filters

0 unpaired reverse - 0 filtered out

3 paired

M230G4-ch_1.fq ➔ M230G4
M486C2-ch_1.fq ➔ M486C2
SC14-ch_1.fq ➔ SC14

Name: Paired miDatasets

Create list

Cancel
Collection Mapping (1 / 3)

Tool consumes a FASTQ file.

- List of Paired Datasets
- Individual FASTQ datasets.

This tool offers several conversion options relating to the FASTQ format.

When using Basic options, the output will be sanger formatted or csanger formatted (when the input is Color Space Sanger).

When converting, if a quality score falls outside of the target score range, it will be coerced to the closest available value (i.e. the minimum or maximum).
Collection Mapping (2 / 3)

Collection map icon replaces input options with valid collections.

Runs tool over every dataset in list of pairs and produces groomed list of pairs.
Like hiding workflow datasets, they are visible initially and hidden after completion (only collection remains visible).

Collection always green regardless of contents (stateless).

Need to do better on both points... not scalable enough.
Sample Tracking: Identifiers + Indices

Paired mt Datasets

list:paired collection

Element - 0:M236C4 (paired collection)
  Element - 0:forward
    hda - M236C4-ch_1.fq
  Element - 1:reverse
    hda - M236C4-ch_2.fq

Element - 1:M486C2 (paired collection)
  Element - 0:forward (hda)
    hda - M486C2-ch_1.fq
  Element - 1:reverse (hda)
    hda - M486C2-ch_2.fq

FASTQ Groomer across collection 8

list:paired collection

Element - 0:M236C4 (paired collection)
  Element - 0:forward
    hda - FASTQ Groomer on data 1
  Element - 1:reverse
    hda - FASTQ Groomer on data 2

Element - 1:M486C2 (paired collection)
  Element - 0:forward (hda)
    hda - FASTQ Groomer on data 3
  Element - 1:reverse (hda)
    hda - FASTQ Groomer on data 4

Mapping over collections - dataset naming is normal, but new collection created with identical tree structure and element identifiers preserved.
Subcollection Mapping

Bowtie2 wrapper modified with option to take in a paired dataset instead of two separate datasets.
Tool Parameters - Tool XML

```xml
<param name="collect_param1" type="data_collection" format="bam" collection_type="paired" />
```

Optional - filter collections by contained formats.

Optional - filter collections by collection_type.
Tool Parameters - Cheetah-isms

Common paired data idiom:

```bash
bowtie $collect_param.forward $collect_param.reverse
```

Common list data idiom:

```bash
#for $f in $collect_param# $f #end for#
-or-

#for $name in $collect_param.keys()# $f[$name] #end for#
```

Nested data:

```bash
#for $f in $collect_param# $f.is_collection ...
```
Tool Parameters - Testing

<test>
  <param name="collect_param">
    <collection type="paired">
      <element name="forward" value="simple_line.txt" />
      <element name="reverse" value="simple_line_alternative.txt" />
    </collection>
  </param>
</test>

...
Subcollection Mapping
Subcollection Mapping (Identifiers)

Paired mt Datasets

list: paired collection
Element - 0: M236C4 (paired collection)
  hda - M236C4-ch_1.fq
Element - 1: reverse
  hda - M236C4-ch_2.fq
Element - 1: M486C2 (paired collection)
  Element - 0: forward (hda)
  hda - M486C2-ch_1.fq
  Element - 1: reverse (hda)
  hda - M486C2-ch_2.fq
...

Bowtie 2 across collection 13

list collection
Element - 0: M236C4
  hda - Bowtie 2 on data 9 and data 10
Element - 1: M486C2
  hda - Bowtie 2 on data 11 and data 12
...

19: Bowtie2 across collection 15
15: FASTQ Groomer across collection 8
8: Paired mt Datasets
7: sequence.fasta
6: SC14-ch_2.fq
Reducing Collections

Modified “Merge BAM Files” tool to use multiple input data parameter instead of two input parameters and a repeat block.
Reducing Collections

Can dynamically substitute collection for the multiple selection of datasets.
A handful of reduction tools need to be updated (so will tools consuming pairs). Using multiple input data parameters instead of repeat parameters will still allow these tools to work with uncollected dataset.

repeat blocks - while cumbersome - allow duplicated entries & control of order. Multiple input data parameters should be enhanced to have same control.
Plan: Multiple-Data Improvements

Enhance multiple input data parameters to allow control of order and repeated entries.

All the ease of multiple data inputs with actually greater versatility than placing simple data inputs into repeat blocks.

An advanced “add to selection” modal would provide interesting room to grow - options for importing library datasets, digging into collections, etc....
Extract a Workflow

Dataset Collection Creation
- Dataset collection created in a way not compatible with workflows

Dataset Collection Creation
- Dataset collection created in a way not compatible with workflows

FASTQ Groomer
- Include "FASTQ Groomer" in workflow

Bowtie2
- Include "Bowtie2" in workflow

Merge BAM Files
- Include "Merge BAM Files" in workflow

flagstat
- Include "flagstat" in workflow

7: M486C2
- Treat as input dataset

8: Paired mt Datasets
- Treat as input dataset

13: FASTQ Groomer across collection 8

16: Bowtie2 across collection 13

17: NewBam.bam

18: NewBam_Merge BAM Files.log

19: flagstat on data 17
More Powerful Workflows

- Run applications in parallel (one per input).
- Merged output for subsequent processing.

Arbitrary # of Inputs (... paired).
More workflows...

Core phylogenomics SNP pipeline by Aaron Petkau, Gary Van Domselaar, Philip Mabon, and Lee Katz. Worked 208 single end reads producing 1469 datasets. Galaxy took 10 minutes to schedule workflow.
Plans - Improvements to Builder

Iteration 2 - [https://trello.com/c/8hEO00xj](https://trello.com/c/8hEO00xj)
  Regex filters, more assistance, allow reordering

Iteration 3 - [https://trello.com/c/LLk9ICvM](https://trello.com/c/LLk9ICvM)
  Batch renaming, dataset info on click, hide original datasets.
Plans - More Options in History Panel

https://trello.com/c/hnmWWKlB

Currently can hide, delete, and see name.

Cannot rename, rerun, see type, see contents, see/add annotations, see/add tags, download, etc...
Incorporate collection builder when uploading files (or vise versa).
Plans - UI for Viewing Collections

https://trello.com/c/PVdbbpQS
Plans - Store Collections in Data Libraries

https://trello.com/c/3axmjaxE
Plans - Improved Reductions

https://trello.com/c/lp5YmA1O

Improvements to multiple data parameters described earlier and/or ability to reduce across repeat statements.
Main Goal: Filter out the failed datasets and keep going.

Would like more general filters - filter on metadata (file size, number of sequences, etc...)

Needs to be trackable so can extract and execute in workflows. May require delayed workflow evaluation.
Plans - Output Collections

https://trello.com/c/KXjp6lIn

Use Cases:

- 1→N (metagenomics, splitting)
- N→N (normalization across files)

Progress on tool running was made at hackathon (thanks JJ and Carrie) - workflows will challenging (ever more bookkeeping for editor).
Plans - Rerun Tools / Resuming Workflows

https://trello.com/c/lxVJy7fs
Plans - Update and Add New Tools

https://trello.com/c/lxVJy7fs

- Paired-end mappers (bowtie, etc...)
- Concatenate Datasets
- Merge Bam
- Many sorts of interesting tabular operations to merge datasets (also using element identifiers).
- etc...
Toward 10,000 samples (beyond collections)

- **Optimize database** interactions, tool execution.
- **Move workflow scheduling** into own process, optimize.
- Differentiate between cluster failures and tool failures.
  - Retry later on cluster failures.
  - Retry on different cluster or with different resource params on failures.
- **Optimize disk usage - streaming**
- **More diverse and bigger compute and storage**
  - Separate metadata calculation out into its own "job"
  - XSEDE
  - More portable dependency management (docker, nix, tool shed installs without galaxy)
Docker... Docker... Docker...

https://github.com/jmchilton/gcc2014_demo
The Galaxy Team

Thanks!

The Galaxy-P grant, team, and the Minnesota Supercomputing Institute for funding development of multiple file datasets (a precursor) - with special thanks to Tim Griffin, Pratik Jagtap, Benjamin Lynch, and Anne-Françoise Lamblin.

The Galaxy Community for building awesome stuff with Galaxy and pushing the platform forward - especially Philip Mabon and Bjoern Gruening.

With special thanks to Carl Eberhard - for building UI powering this work, Jeremy and Dannon for scoping out initial plans, and Nick, James, Dan, and Anton for ongoing feedback.
Lots of changes but the yellow boxes are the core additions.
Extra Content
Plans - Other

- [https://trello.com/c/WodW2sLb](https://trello.com/c/WodW2sLb)
- Subcollection mapping over multiple data parameters.
- Fix history import/export for data collections.
- Implicit conversion
- Allow batch input of collections to workflows
- HID s of copied collections are wrong - either always copy HDAs also or reconsider naming in context of collections.
TODO:

- Screenshots of building up workflow from scratch?
  Extra Slides (post presentation)…
- Comparison with multiple file datasets.

REDO Initial Screenshots with Correct History Name on Bigger Monitor.
Building Collections...

```python
>>> from bioblend import galaxy

>>> gi = galaxy.GalaxyInstance(url="localhost:8080",
   key="db53bb4500dfaeda25ceb378069b722b")

>>> hist = gi.histories.get_histories(name="Map/Reduce Test")[0]

>>> gi.histories.show_history(hist["id"], contents=True, deleted=False)

>>> pair1_id = [d for d in gi.histories.show_history(hist["id"], contents=True)
                 if d["hid"] == 5][0]["id"]

>>> pair2_id = [d for d in gi.histories.show_history(hist["id"], contents=True)
                 if d["hid"] == 6][0]["id"]

>>> gi.histories.update_dataset_collection(hist["id"], pair1_id, name="M236C4")

>>> gi.histories.update_dataset_collection(hist["id"], pair2_id, name="M486C2")
```

bioblend contains support for creating, reading, updating (name, annotations, etc…), and deleting history dataset collections.

https://github.com/afgane/bioblend/commit/f8d40b687be4c699d608e930c59726793922fa0a
Collection Mapping (1 / 3)

Tool consumes a FASTQ file.

- List of Paired Datasets
- Paired Datasets
- Individual FASTQ datasets.
Like hiding datasets in workflow execution, datasets are visible running or queued and they are hidden after (and only collection is visible).

Collection is always green regardless of contents - is currently stateless.

Need to do a better job on both points - this is not too scalable - but it was an easy quick win.
# Plans - UI for Creating Collections

[https://trello.com/c/C1Idaxl2](https://trello.com/c/C1Idaxl2)  [Mockup @ mybalsamiq](https://mybalsamiq.com)

## Create a list of paired datasets

**Help Text:** Create a list of paired datasets by...

### Forward

<table>
<thead>
<tr>
<th>Forward</th>
<th>2 unpaired</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 Pairs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>MRX3348_1.fastq</td>
<td>MRX3348</td>
</tr>
<tr>
<td>MRX3348_1.fastq</td>
<td>MRX3348</td>
</tr>
<tr>
<td>MRX3348_1.fastq</td>
<td>MRX3348</td>
</tr>
<tr>
<td>MRX3348_1.fastq</td>
<td>MRX3348</td>
</tr>
<tr>
<td>MRX3348_1.fastq</td>
<td>MRX3348</td>
</tr>
<tr>
<td>MRX3348_1.fastq</td>
<td>MRX3348</td>
</tr>
<tr>
<td>MRX3348_1.fastq</td>
<td>MRX3348</td>
</tr>
<tr>
<td>MRX3348_1.fastq</td>
<td>MRX3348</td>
</tr>
</tbody>
</table>

### Reverse

<table>
<thead>
<tr>
<th>Reverse</th>
<th>1 unpaired</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>MRX3348_2.fastq</td>
<td></td>
</tr>
<tr>
<td>MRX3348_2.fastq</td>
<td></td>
</tr>
<tr>
<td>MRX3348_2.fastq</td>
<td></td>
</tr>
<tr>
<td>MRX3348_2.fastq</td>
<td></td>
</tr>
<tr>
<td>MRX3348_2.fastq</td>
<td></td>
</tr>
<tr>
<td>MRX3348_2.fastq</td>
<td></td>
</tr>
<tr>
<td>MRX3348_2.fastq</td>
<td></td>
</tr>
<tr>
<td>MRX3348_2.fastq</td>
<td></td>
</tr>
</tbody>
</table>

**Name of new list:** My List

- [Cancel](#)
- [Create a different kind of collection](#)
- [Create list](#)

**Middle Section:** A scrollable table divided into two parts: the upper, paired section and the lower, unpaired section. Filtering only affects the unpaired section.

- **A:** Color background color, font, and justification can all be used to differentiate paired/unpaired.

**When the user clicks on an unpaired forward then an unpaired reverse (or vice versa) a pair is created. That pair is moved to the bottom of the paired section of the table.**

**Each row in the ‘Pairs’ section of the list will have some control to unpair that pair. When clicked, the row disappears and the two files go back to the unpaired/lower section of the table in the appropriate, sorted order.**

**Alternately, we can send the user to a second pane (2nd ‘Wizard’ step) to review and re-order the final list. (An option to move back to this step should also be there).**
Why not repeat replacements?

In its most simple form - allowing replacement of one repeat block with a collection - this feature would be gross to implement - it would add a lot of complexity to already complex parts of Galaxy.

... and it would not work with any tools.
Concatenate (Easiest Reduction)

Not just a repeat, would need to be able to dynamically replace input + repeat to work with this. That will be ugly and will have implications all over.
Merging Bams

Second most common reduction - has two inputs and a repeat. So we need to be able to dynamically replace any number inputs and a repeat. Hmm....
Merging BedGraph

Found another reduction tool on main. Multiple inputs, multiple extra options. How could this reasonably allow collection replacement at the infrastructure level.