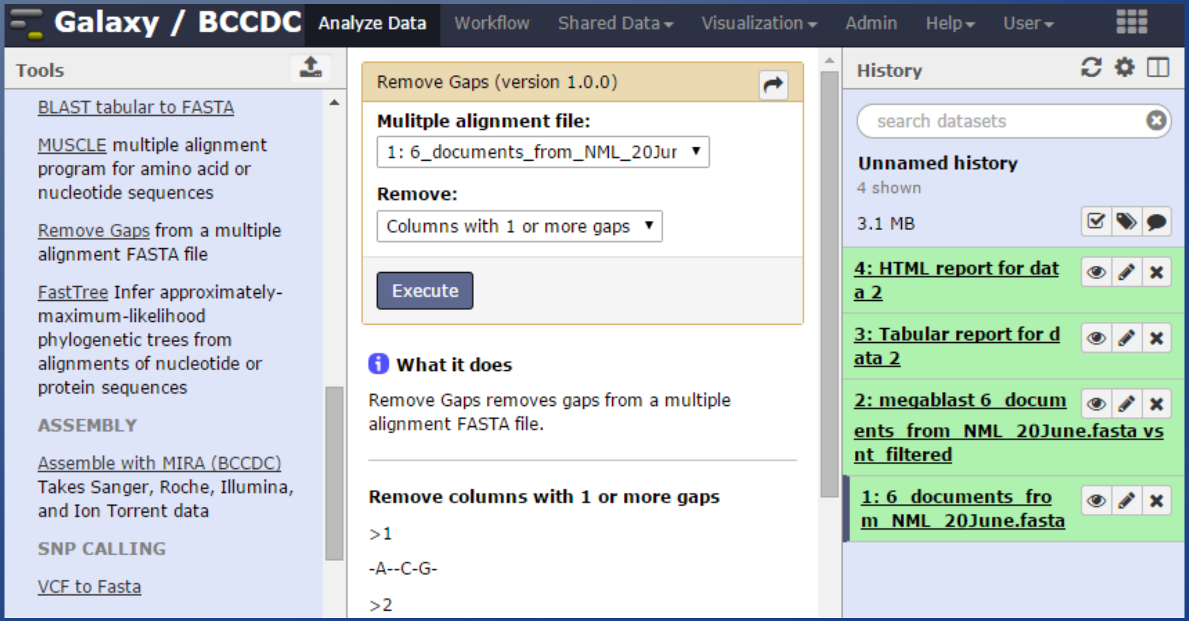
Building a Galaxy Tool

Benefits of the Galaxy way:

- Galaxy helps you get organized: Histories
- Experimental reproducibility: Record-keeping
- Tool Sharing, e.g. to support publication



BLAST tabular to FASTA

MUSCLE multiple alignment program for amino acid or nucleotide sequences

Remove Gaps from a multiple alignment FASTA file

FastTree Infer approximately-maximum-likelihood phylogenetic trees from alignments of nucleotide or protein sequences

ASSEMBLY

Assemble with MIRA (BCCDC) Takes Sanger, Roche, Illumina, and Ion Torrent data

SNP CALLING

VCF to Fasta

Remove Gaps (version 1.0.0)

Mulitple alignment file: 1: 6_documents_from_NML_20Jur

Remove: Columns with 1 or more gaps

What it does

Remove Gaps removes gaps from a multiple alignment FASTA file.

Remove columns with 1 or more gaps

```
>1
-A--C-G-
>2
```

Unnamed history
4 shown
3.1 MB

4: HTML report for data 2

3: Tabular report for data 2

2: megablast 6_documents_from_NML_20June.fasta vs nt_filtered

1: 6_documents_from_NML_20June.fasta

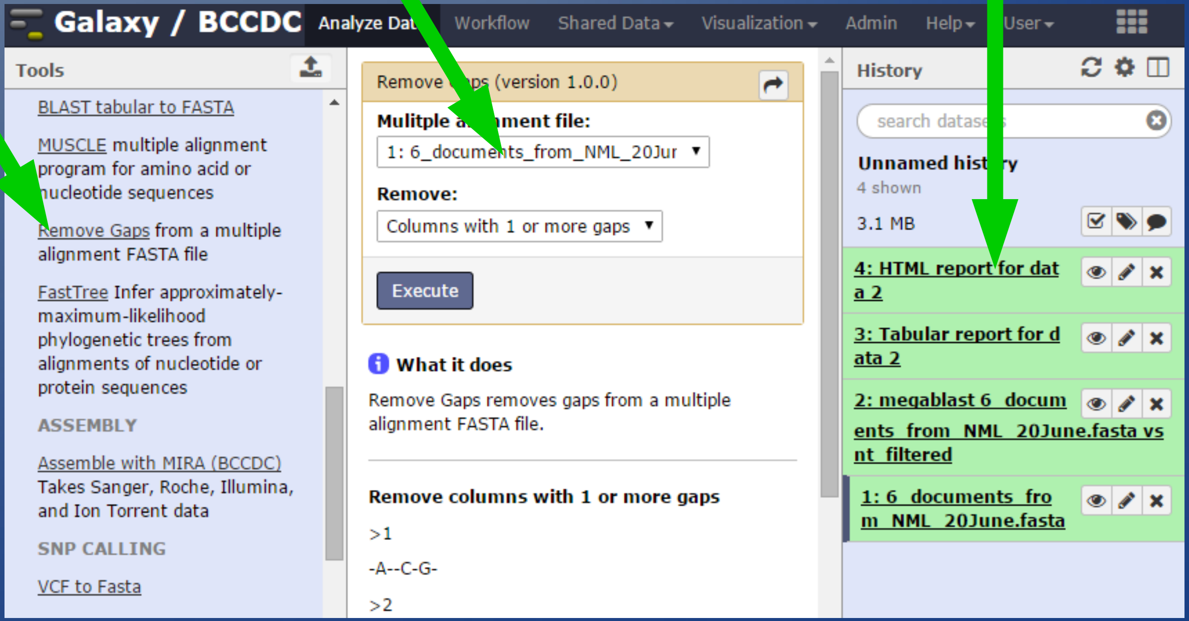
Created by Damion Dooley, Hsiao Lab, BC Public Health Microbiology & Reference Laboratory, BC Centre for Disease Control, https://github.com/Public-Health-Bioinformatics

Galaxy Intro

List of available tools

Tool **web form** that a user can fill in and submit.

Results put into a user's history as a **dataset** that can be viewed, downloaded or used as part of a larger workflow.



BLAST tabular to FASTA

MUSCLE multiple alignment program for amino acid or nucleotide sequences

Remove Gaps from a multiple alignment FASTA file

FastTree Infer approximately-maximum-likelihood phylogenetic trees from alignments of nucleotide or protein sequences

ASSEMBLY

Assemble with MIRA (BCCDC) Takes Sanger, Roche, Illumina, and Ion Torrent data

SNP CALLING

VCF to Fasta

Remove Gaps (version 1.0.0)

Mulitple alignment file: 1: 6_documents_from_NML_20Jur

Remove: Columns with 1 or more gaps

What it does

Remove Gaps removes gaps from a multiple alignment FASTA file.

Remove columns with 1 or more gaps

```
>1
-A--C-G-
>2
```

search datasets

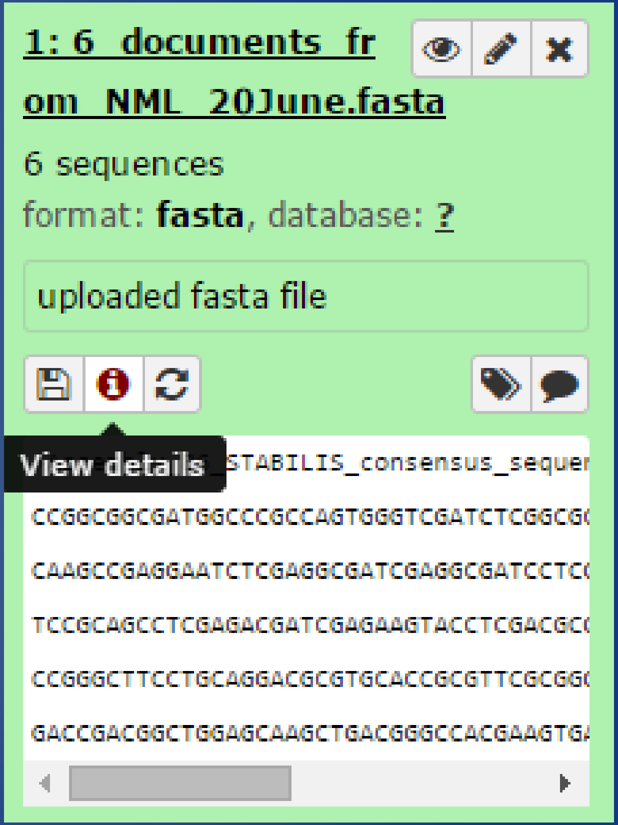
Unnamed history
4 shown
3.1 MB

4: HTML report for data 2
3: Tabular report for data 2
2: megablast 6_documents_from_NML_20June.fasta vs nt_filtered
1: 6_documents_from_NML_20June.fasta

See: http://salk.bccdc.med.ubc.ca/galaxylab

Galaxy Datasets

- History tab's dataset items are tool inputs and outputs.
- Galaxy's data type system specifies structured data formats (fasta, sam, bam, etc.), including ones that involve several files.
- Galaxy guesses the type of file a user has uploaded.
- You can control what (standard) type of input data your tool takes.


1: 6_documents_from_NML_20June.fasta

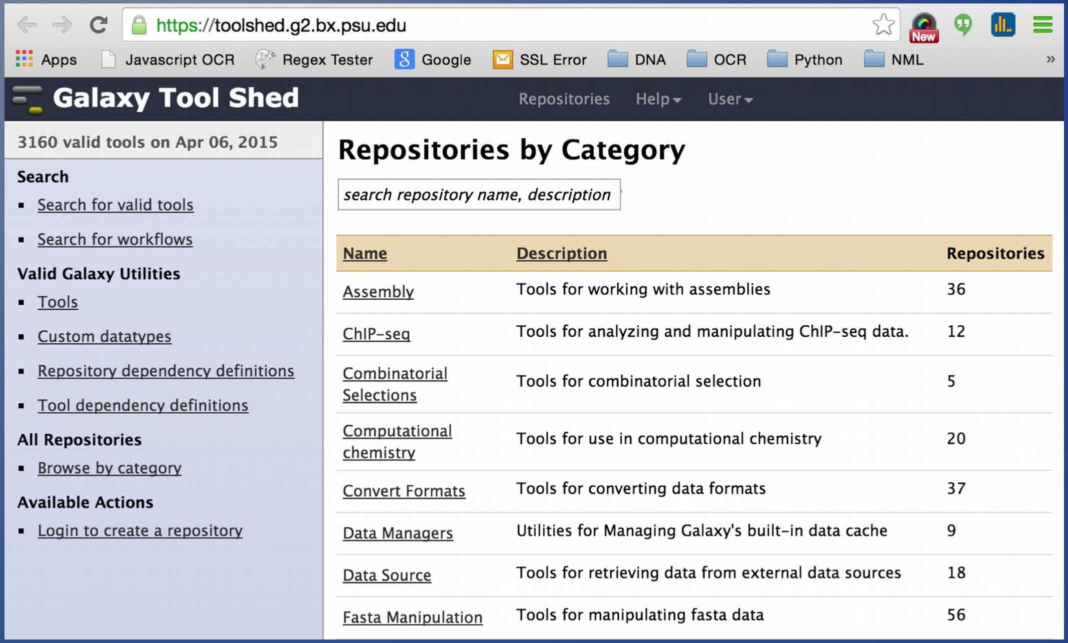
6 sequences
format: **fasta**, database: ?

uploaded fasta file

```
STABILIS_consensus_sequer
CCGGCGGCGATGGCCCGCCAGTGGGTCGATCTCGGCG
CAAGCCGAGGAATCTCGAGGCGATCGAGGCGATCCTC
TCCGCAGCCTCGAGACGATCGAGAAGTACCTCGACGC
CCGGGCTTCCTGCAGGACGCGTGCACCGCGTTCGCGG
GACCGACGGCTGGAGCAAGCTGACGGGCCACGAAGTG
```

Galaxy Toolshed

- Basic Galaxy tools:http://salk.bccdc.med.ubc.ca/galaxylab
- Main Library: https://toolshed.g2.bx.psu.edu/



https://toolshed.g2.bx.psu.edu

3160 valid tools on Apr 06, 2015

Search
- Search for valid tools
- Search for workflows

Valid Galaxy Utilities
- Tools
- Custom datatypes
- Repository dependency definitions
- Tool dependency definitions

All Repositories
- Browse by category

Available Actions
- Login to create a repository

Repositories by Category

search repository name, description

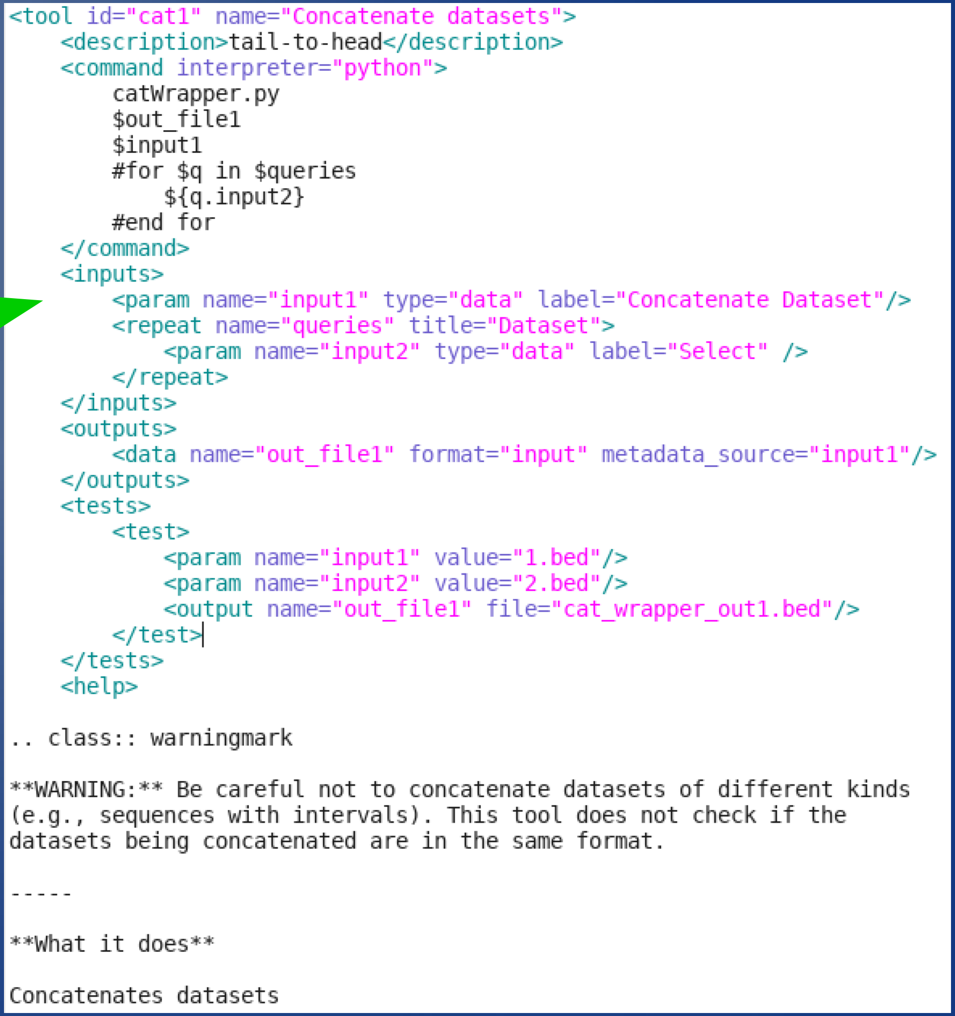
Name	Description	Repositories
Assembly	Tools for working with assemblies	36
ChIP-seq	Tools for analyzing and manipulating ChIP-seq data.	12
Combinatorial Selections	Tools for combinatorial selection	5
Computational chemistry	Tools for use in computational chemistry	20
Convert Formats	Tools for converting data formats	37
Data Managers	Utilities for Managing Galaxy's built-in data cache	9
Data Source	Tools for retrieving data from external data sources	18
Fasta Manipulation	Tools for manipulating fasta data	56

- BCCDC playground: http://salk.bccdc.med.ubc.ca/toolshed

Tool Development Process

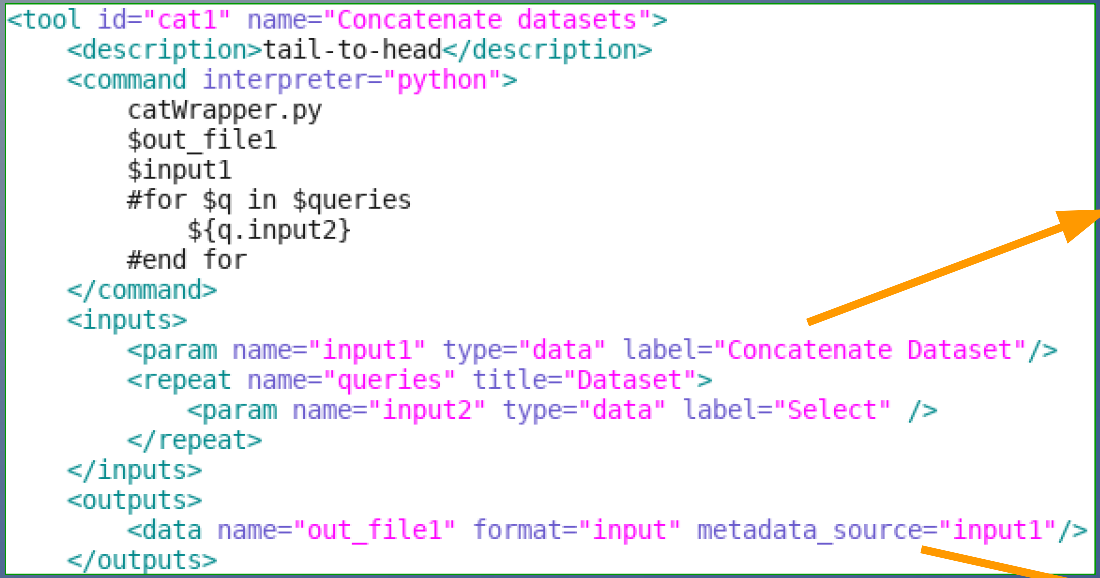
Stages

- Develop command line script
- Craft XML "tool definition file"
- View & run form in Galaxy
- Make and pass tests
- Add documentation
- Share

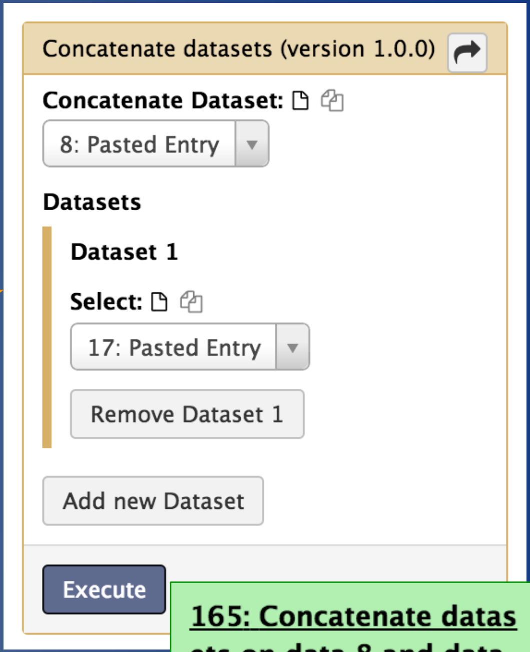


```
<tool id="cat1" name="Concatenate datasets">
    <description>tail-to-head</description>
    <command interpreter="python">
        catWrapper.py
        $out_file1
        $input1
        #for $q in $queries
            ${q.input2}
        #end for
    </command>
    <inputs>
        <param name="input1" type="data" label="Concatenate Dataset"/>
        <repeat name="queries" title="Dataset">
            <param name="input2" type="data" label="Select" />
        </repeat>
    </inputs>
    <outputs>
        <data name="out_file1" format="input" metadata_source="input1"/>
    </outputs>
    <tests>
        <test>
            <param name="input1" value="1.bed"/>
            <param name="input2" value="2.bed"/>
            <output name="out_file1" file="cat_wrapper_out1.bed"/>
        </test>
    </tests>
    <help>

.. class:: warningmark

**WARNING:** Be careful not to concatenate datasets of different kinds
(e.g., sequences with intervals). This tool does not check if the
datasets being concatenated are in the same format.

-----

**What it does**

Concatenates datasets
```

Anatomy of a tool



```
<tool id="cat1" name="Concatenate datasets">
    <description>tail-to-head</description>
    <command interpreter="python">
        catWrapper.py
        $out_file1
        $input1
        #for $q in $queries
            ${q.input2}
        #end for
    </command>
    <inputs>
        <param name="input1" type="data" label="Concatenate Dataset"/>
        <repeat name="queries" title="Dataset">
            <param name="input2" type="data" label="Select" />
        </repeat>
    </inputs>
    <outputs>
        <data name="out_file1" format="input" metadata_source="input1"/>
    </outputs>
```



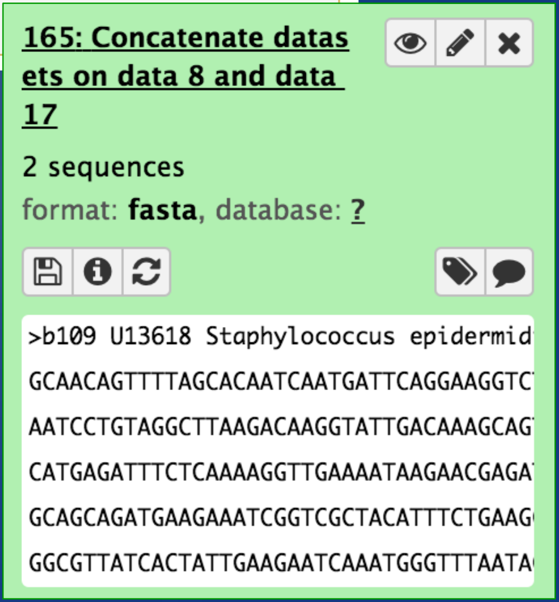
Concatenate datasets (version 1.0.0)

Concatenate Dataset: 8: Pasted Entry

Datasets

Dataset 1

Select: 17: Pasted Entry



165: Concatenate datasets on data 8 and data 17

2 sequences

format: **fasta**, database: **?**

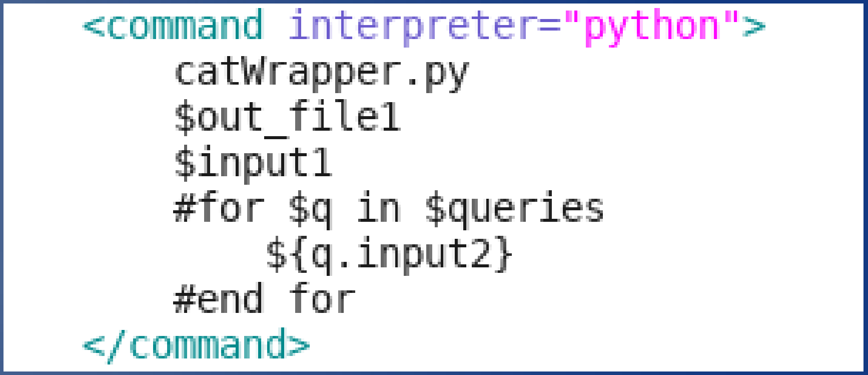
```
>b109 U13618 Staphylococcus epidermid
GCAACAGTTTTAGCACAATCAATGATTCAGGAAGGTC
AATCCTGTAGGCTTAAGACAAGGTATTGACAAAGCAG
CATGAGATTTCTCAAAAGGTTGAAAATAAGAACGAGA
GCAGCAGATGAAGAAATCGGTCGCTACATTTCTGAAG
GGCGTTATCACTATTGAAGAATCAAATGGGTTTAATA
```

An XML file controls form display, field validation and app execution. Tags specify the command to be run, fields for selecting input file(s), etc.

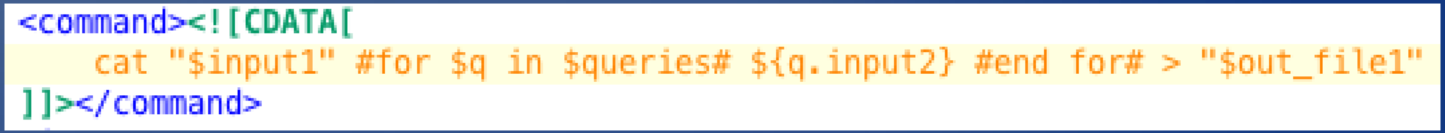
Examples are in the tools/ folder of a Galaxy installation.

<command interpreter=”...”>

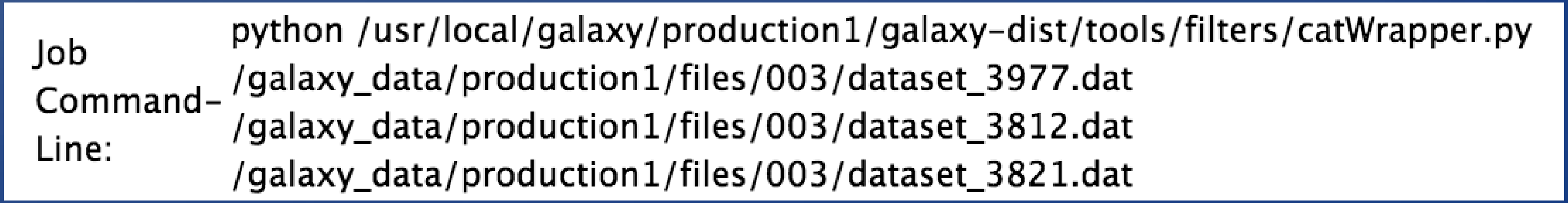
- The command is any linux shell command, or python, java, or other executable (that Galaxy knows of).
- It runs with the Galaxy user's permissions.
- Galaxy sets the tool's exact input and output file paths.
- Command tag allows Cheetah expressions.



```
<command interpreter="python">
    catWrapper.py
    $out_file1
    $input1
    #for $q in $queries
        ${q.input2}
    #end for
</command>
```



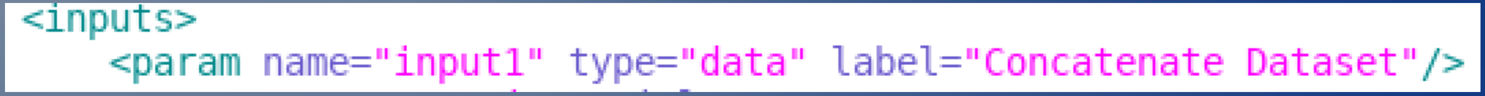
```
<command><![CDATA[
    cat "$input1" #for $q in $queries# ${q.input2} #end for# > "$out_file1"
]]></command>
```



```
Job           python /usr/local/galaxy/production1/galaxy-dist/tools/filters/catWrapper.py
Command-      /galaxy_data/production1/files/003/dataset_3977.dat
Line:         /galaxy_data/production1/files/003/dataset_3812.dat
              /galaxy_data/production1/files/003/dataset_3821.dat
```

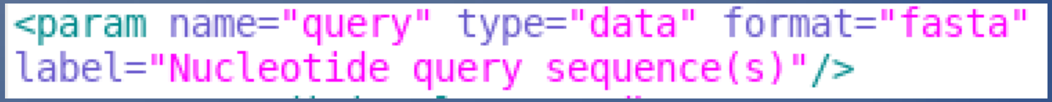
<inputs> <param type=”data”>

- Provides a file selector of one or more input files from current history.



```
<inputs>
    <param name="input1" type="data" label="Concatenate Dataset"/>
```

- You can constrain this to files of a particular data type.



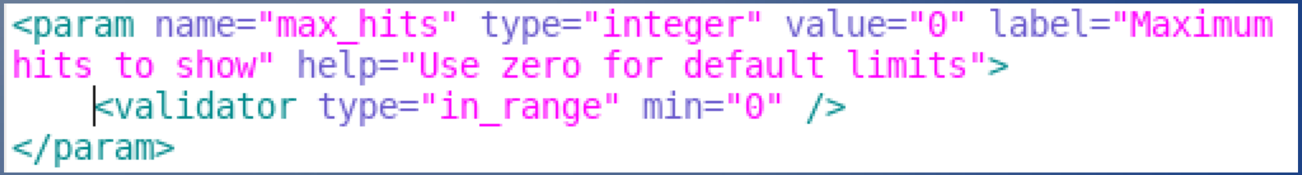
```
<param name="query" type="data" format="fasta"
label="Nucleotide query sequence(s)"/>
```

- Add ' multiple=”true” ' : allows selection of more than one file. These will be passed as a comma-delimited list.

<inputs> <param type=”...”>

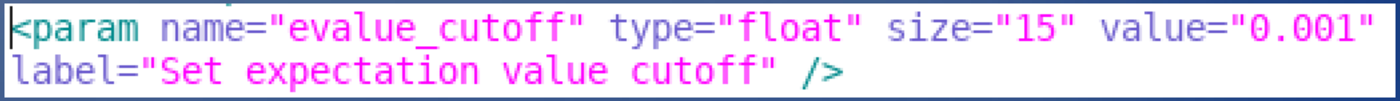
Each form field entry can be passed as a string value to a command line call parameter.

- “integer” : This shows some range validation on the input.



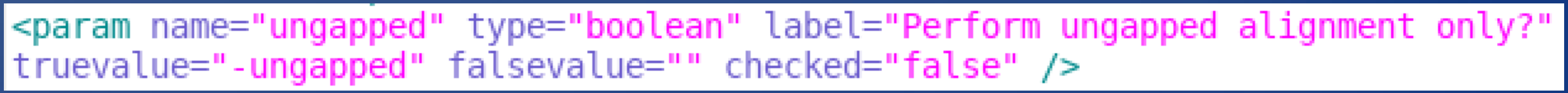
```
<param name="max_hits" type="integer" value="0" label="Maximum
hits to show" help="Use zero for default limits">
    <validator type="in_range" min="0" />
</param>
```

- “float” : Input accepts decimal numbers



```
<param name="evalue_cutoff" type="float" size="15" value="0.001"
label="Set expectation value cutoff" />
```

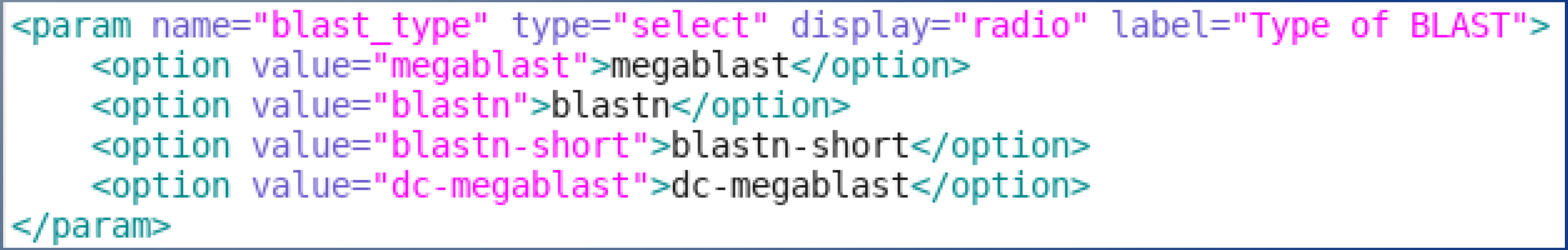
- “boolean” :



```
<param name="ungapped" type="boolean" label="Perform ungapped alignment only?"
truevalue="-ungapped" falsevalue="" checked="false" />
```

<inputs> <param type="...">

- "select" : drop down list or radio/checkboxes



```
<param name="blast_type" type="select" display="radio" label="Type of BLAST">
    <option value="megablast">megablast</option>
    <option value="blastn">blastn</option>
    <option value="blastn-short">blastn-short</option>
    <option value="dc-megablast">dc-megablast</option>
</param>
```

- "text" : provides text input field.
- "data_table" : Filter and display rows / columns from a Galaxy-registered tab-delimited table

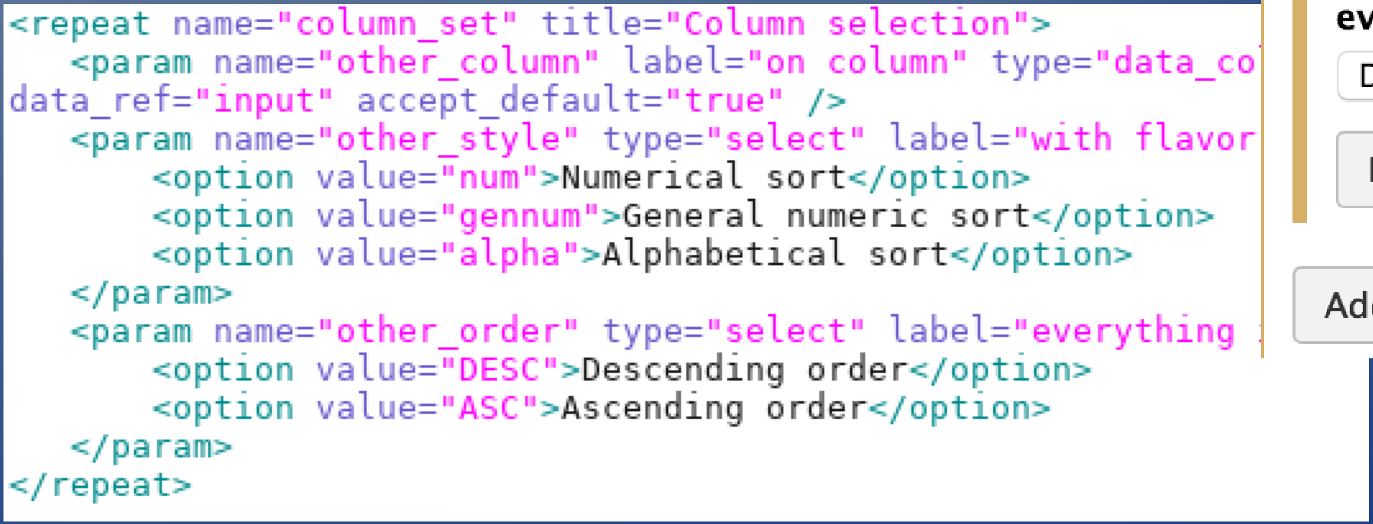
... and others. See:

https://wiki.galaxyproject.org/Admin/Tools/ToolConfigSyntax

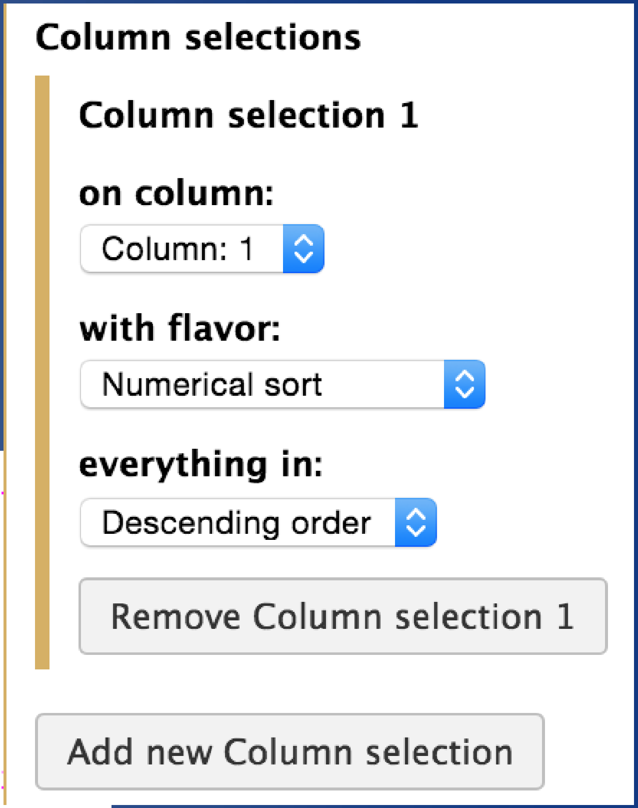
<inputs> <repeat ...>

Use a <repeat> to collect associated parameter inputs.

Example from Galaxy tools/sorter.xml file.



```
<repeat name="column_set" title="Column selection">
   <param name="other_column" label="on column" type="data_co
data_ref="input" accept_default="true" />
   <param name="other_style" type="select" label="with flavor
       <option value="num">Numerical sort</option>
       <option value="gennum">General numeric sort</option>
       <option value="alpha">Alphabetical sort</option>
   </param>
   <param name="other_order" type="select" label="everything
       <option value="DESC">Descending order</option>
       <option value="ASC">Ascending order</option>
   </param>
</repeat>
```



Column selections

Column selection 1

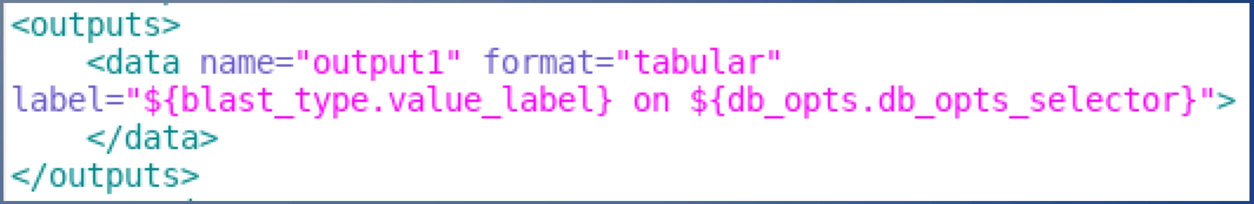
on column: Column: 1

with flavor: Numerical sort

everything in: Descending order

<outputs><data name=”...”>

The <outputs> section has one or more <data ...> tag output file specifications. For each <data> tag Galaxy will set up a new dataset file path to output your app's data to.



```
<outputs>
    <data name="output1" format="tabular"
label="${blast_type.value_label} on ${db_opts.db_opts_selector}">
    </data>
</outputs>
```

format=”...” : Contains one of Galaxy's data types listed in admin tab > Administrator > Server > View data types registry.

The “label” shows up as the dataset name of a history record.

Error handling

Apps generate exit code=0 for normal operation; higher numbers are error or warning codes. How Galaxy should interpret them is up to you; normally 0 means “green” ok; anything else triggers a red “failed” state.

The <stdio> tag holds any codes you want to upgrade or downgrade the error state for.

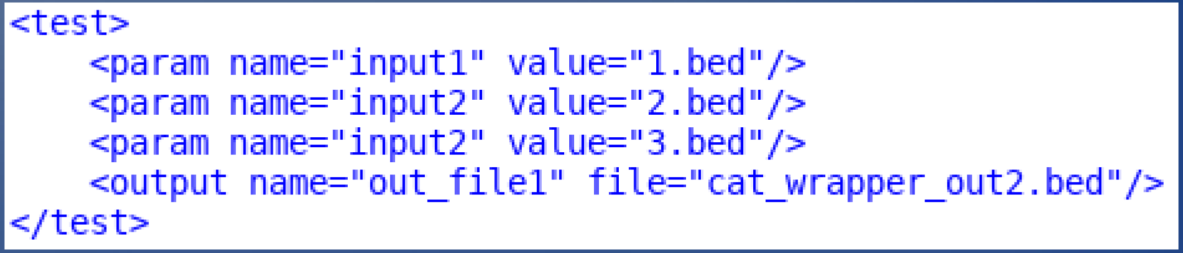
```
<stdio><exit_code range=”1:” level=”fatal”/></stdio>
```

```
<stdio>
      <exit_code range="2"   level="fatal"   description="Out of Memory" />
      <exit_code range="3:5" level="warning" description="Low disk space" />
      <exit_code range="6:"  level="fatal"   description="Bad input dataset" />
      <!-- If the return code has not been set propery check stderr too -->
      <regex match="Error:" />
      <regex match="Exception:" />
</stdio>
```

Robot from Lost in Space (1965)

Tests

- Not required for local use, but for a shared tool, tests are expected.
- Specify input parameters by name and value as they would be entered in form.
- The input and output files are specified in tool's “test-data” folder.



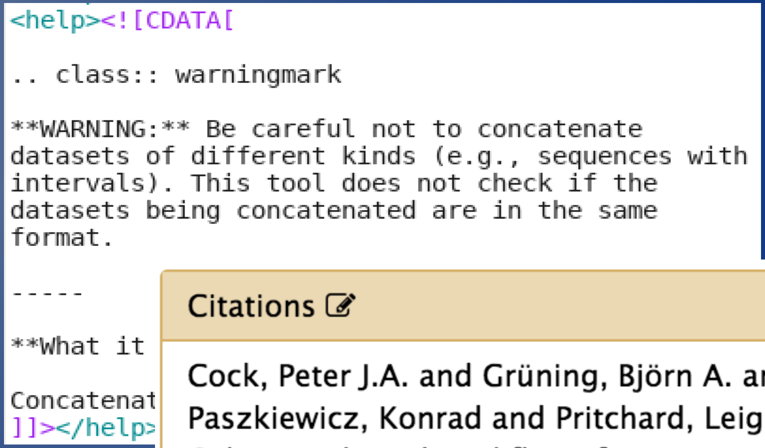
```
<test>
    <param name="input1" value="1.bed"/>
    <param name="input2" value="2.bed"/>
    <param name="input2" value="3.bed"/>
    <output name="out_file1" file="cat_wrapper_out2.bed"/>
</test>
```

- Normally time consuming to test in Galaxy directly, so we use “planemo”

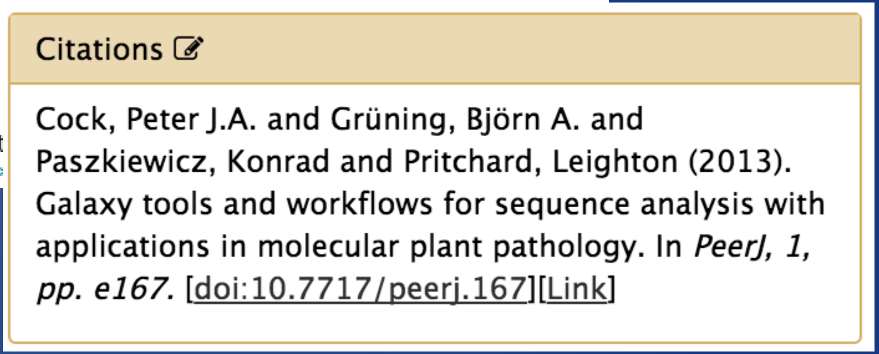
Help Info

"Free" text can only be inserted at form bottom below the Execute button. It uses reStructuredText markdown format: http://docutils.sourceforge.net/docs/user/rst/quickref.html

Add a references section if your app uses 3rd party software or if it is based on your own paper. This can simply be a help section **References** header and text. It can also include an automated lookup:



```
<help><![CDATA[

.. class:: warningmark

**WARNING:** Be careful not to concatenate
datasets of different kinds (e.g., sequences with
intervals). This tool does not check if the
datasets being concatenated are in the same
format.

-----

**What it

Concatenat
]]></help>
```

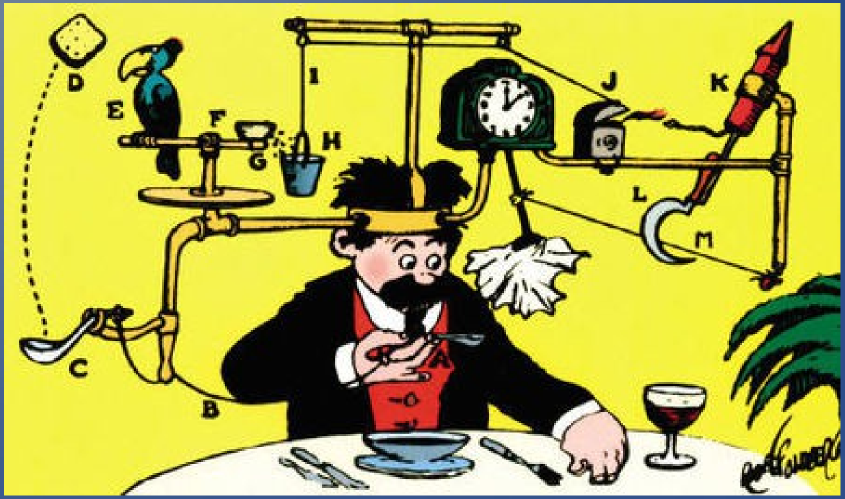


Citations

Cock, Peter J.A. and Grüning, Björn A. and Paszkiewicz, Konrad and Pritchard, Leighton (2013). Galaxy tools and workflows for sequence analysis with applications in molecular plant pathology. In *PeerJ, 1, pp. e167*. [doi:10.7717/peerj.167][Link]

```
<citations>
   <citation type="doi">10.7717/peerj.167</citation>
</citations>
```

Ready?



Self-Operating Napkin
(Copyright © Rube Goldberg™
& © of Rube Goldberg, Inc.)

Get Started on your Galaxy Server

We use the “Planemo” python app to get started with tool development. It can set up a new tool project with a template, validate tool's XML, run Galaxy tests, and launch a tool in a trimmed-down Galaxy.

Use a terminal to “ssh -X ...” to enable X-windows apps. (“ssh -Y ...” for Mac users).

Navigate to the Galaxy installation's tools/TESTING/ subfolder.

```
> cd /usr/local/galaxy/galaxytest/galaxy-dist/tools/TESTING/
```

Make a folder for your tool:

```
> mkdir randomish; cd randomish
```

Using Planemo

Try planemo out on a demo project template that has a few tool definition XML files in it (it works with a folder of them at a time):

```
> planemo project_init --template=demo
```

Now edit one of the tool XML files (> 1 tool per folder is ok):

```
> gedit randomlines.xml &
```

OR start from scratch, with some info on your command line app, e.g. for "myapp.py -a file1.fastq > file2.fasta":

```
> planemo tool_init --force --id 'my_example_id' \
    --name 'My example tool name' \
    --example_command 'python myapp.py -a file1.fastq > file2.fasta' \
    --example_input file1.fastq \
    --example_output file2.fasta \
    --test_case \
    --help_from_command 'myapp.py -h'
```

Using Planemo

See planemo validate XML:

> planemo lint

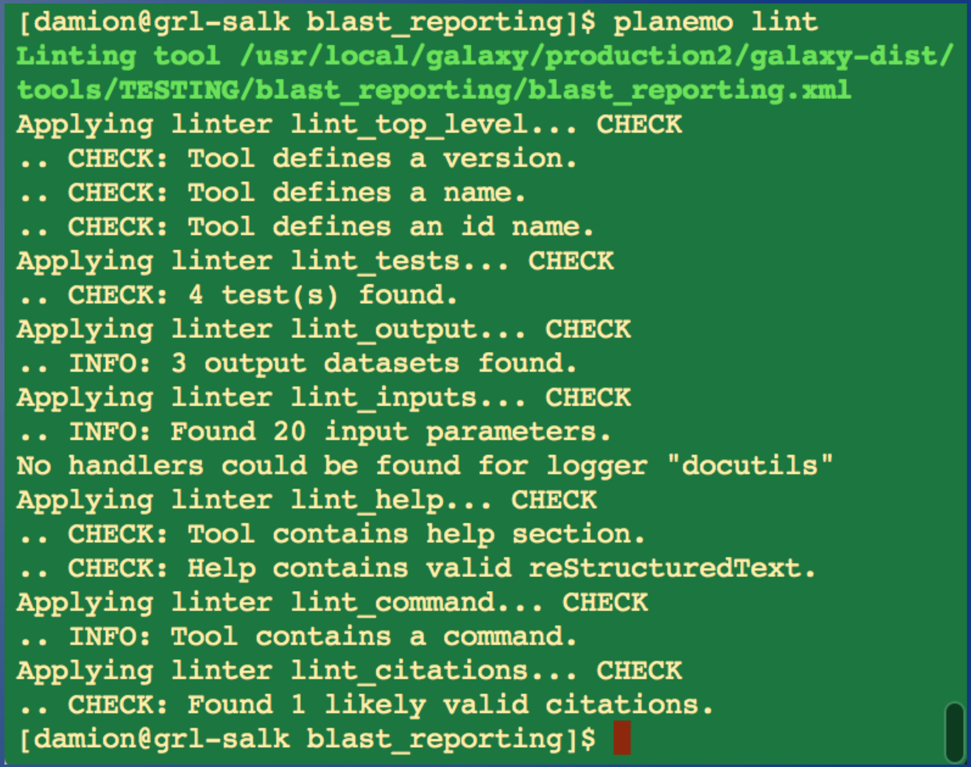
Have planemo generate test report:

> planemo test

... wait a while for the results.

Planemo uses the same tests that Galaxy does – the same input and output files.

Galaxy runs tests on tools that are shared on toolsheds.

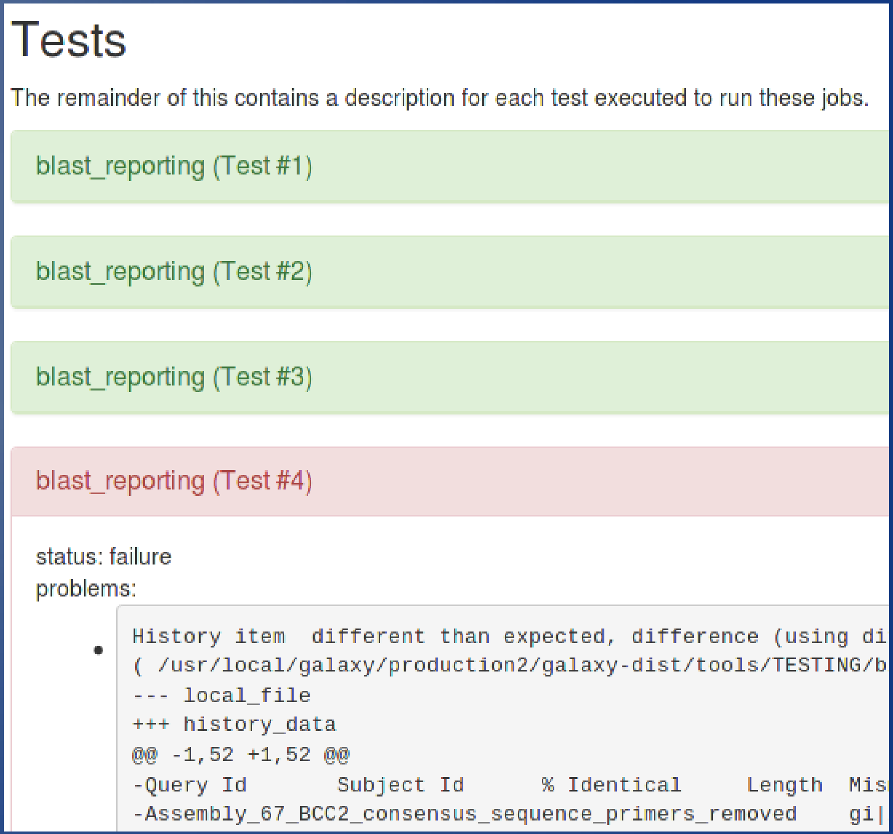


```
[damion@grl-salk blast_reporting]$ planemo lint
Linting tool /usr/local/galaxy/production2/galaxy-dist/
tools/TESTING/blast_reporting/blast_reporting.xml
Applying linter lint_top_level... CHECK
.. CHECK: Tool defines a version.
.. CHECK: Tool defines a name.
.. CHECK: Tool defines an id name.
Applying linter lint_tests... CHECK
.. CHECK: 4 test(s) found.
Applying linter lint_output... CHECK
.. INFO: 3 output datasets found.
Applying linter lint_inputs... CHECK
.. INFO: Found 20 input parameters.
No handlers could be found for logger "docutils"
Applying linter lint_help... CHECK
.. CHECK: Tool contains help section.
.. CHECK: Help contains valid reStructuredText.
Applying linter lint_command... CHECK
.. INFO: Tool contains a command.
Applying linter lint_citations... CHECK
.. CHECK: Found 1 likely valid citations.
[damion@grl-salk blast_reporting]$
```

Using Planemo

Firefox will run via X11 on the server so you can view a planemo test report:

```
> firefox tool_test_output.html &
```



Tests

The remainder of this contains a description for each test executed to run these jobs.

blast_reporting (Test #1)

blast_reporting (Test #2)

blast_reporting (Test #3)

blast_reporting (Test #4)

status: failure
problems:

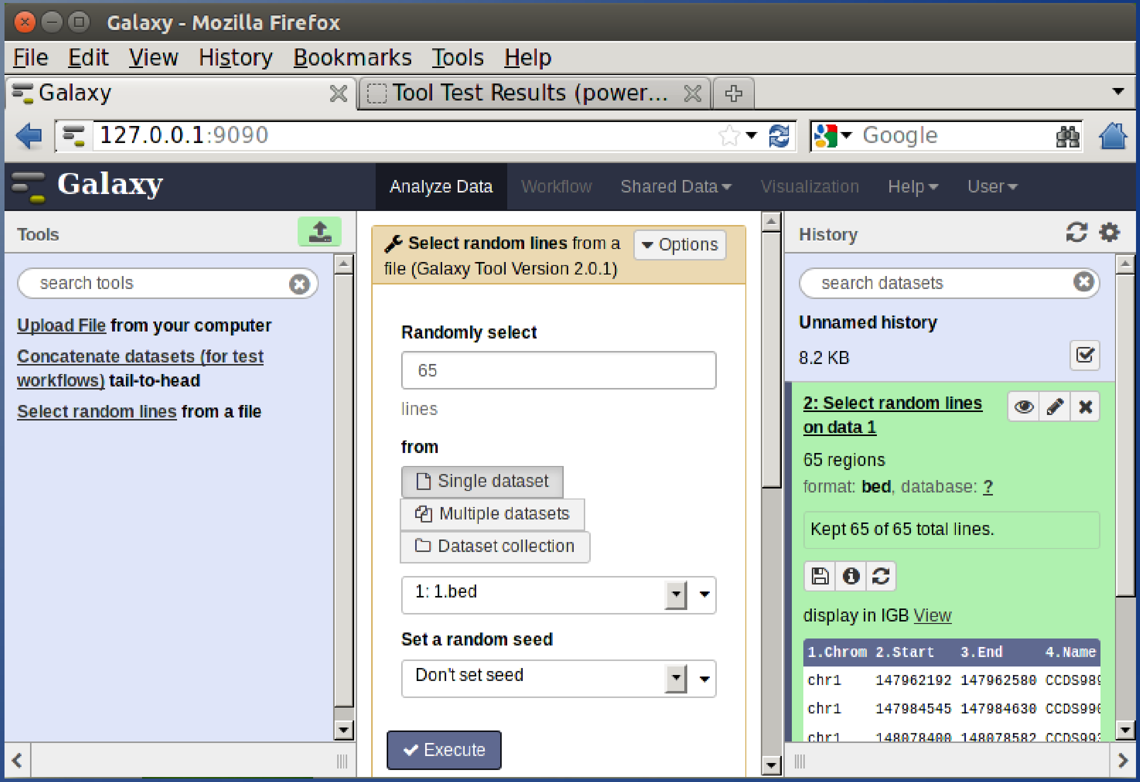
-
  ```
  History item  different than expected, difference (using di
  ( /usr/local/galaxy/production2/galaxy-dist/tools/TESTING/b
  --- local_file
  +++ history_data
  @@ -1,52 +1,52 @@
  -Query Id        Subject Id       % Identical      Length  Mis
  -Assembly_67_BCC2_consensus_sequence_primers_removed    gi|
  ```

Using Planemo

You can play with your tool in Galaxy via planemo “serve” which serves the galaxy website at http://127.0.0.1:9090 ... after a while.

> planemo serve

(Ctrl-c cancels)



Upload File from your computer

Concatenate datasets (for test workflows) tail-to-head

Select random lines from a file

Select random lines from a file (Galaxy Tool Version 2.0.1)

Randomly select: 65 lines

from: 1: 1.bed

Set a random seed: Don't set seed

Unnamed history

8.2 KB

2: Select random lines on data 1

65 regions

format: bed, database: ?

Kept 65 of 65 total lines.

display in IGB View

```
1.Chrom 2.Start   3.End     4.Name
chr1    147962192 147962580 CCDS98
chr1    147984545 147984630 CCDS99
chr1    148078400 148078582 CCDS99
```

For a great intro to planemo's other features, see https://planemo.readthedocs.org/en/latest/writing_standalone.html

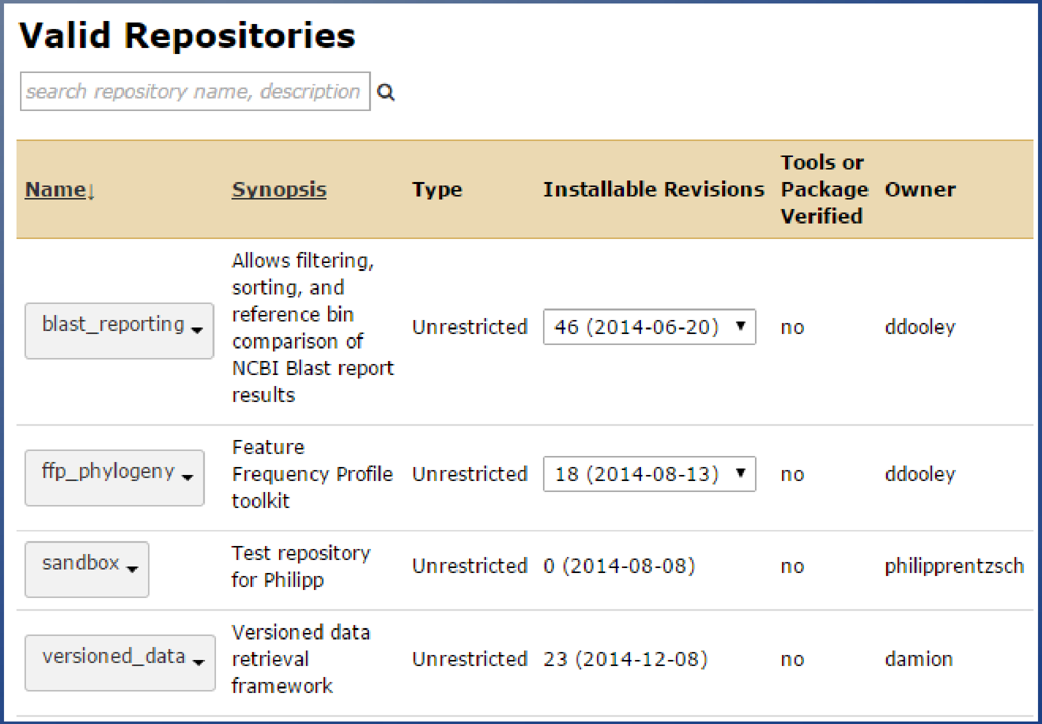
Testing Issues

Small Print: Planemo should report all nasty errors for you to remedy. If however, it fails to do so, or you don't have access to Planemo, and you are trying to install your tool into galaxy directly, then Galaxy will likely not load the tool into its menu system, or even re-start. For this reason tool development can't be easily or safely done on an active Galaxy server. You may find buried in Galaxy's web0.log or paster.log reference to the tool not loading. Startup issues are usually about a tool's malformed XML (incomplete tags, bad nesting, characters that XML sniffs its nose at) or sometimes script compilation issues. For <command> and <help> content this can be avoided by wrapping its text in the <![CDATA[...]]> tag. In the event that this does not remedy the situation, understand that there is no warranty expressed or implied; however possible remedies may be obtained by the very friendly galaxy-dev@lists.galaxyproject.org listserv, or yours truly.

Sharing your Galaxy tool with the world

This involves uploading the tool files into a Galaxy "toolshed".

A Galaxy admin can select them for installation at their own location.



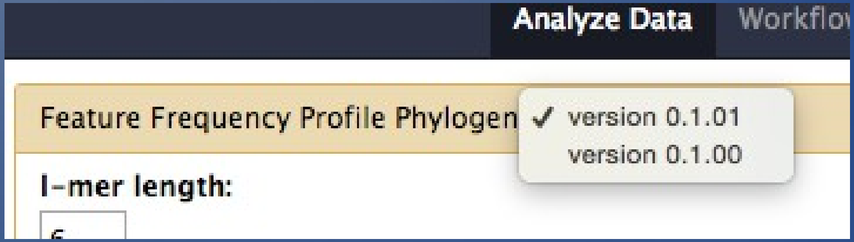
Valid Repositories

search repository name, description

Name	Synopsis	Type	Installable Revisions	Tools or Package Verified	Owner
blast_reporting	Allows filtering, sorting, and reference bin comparison of NCBI Blast report results	Unrestricted	46 (2014-06-20)	no	ddooley
ffp_phylogeny	Feature Frequency Profile toolkit	Unrestricted	18 (2014-08-13)	no	ddooley
sandbox	Test repository for Philipp	Unrestricted	0 (2014-08-08)	no	philipprentzsch
versioned_data	Versioned data retrieval framework	Unrestricted	23 (2014-12-08)	no	damion

Galaxy Tool – Versioning

With a few tool versions installed on a Galaxy server, a user will have a choice of which to run.



Preparing a Toolshed Update

Every tool should have a version number at the top of its XML file:

```
<tool id=”blast_reporting” name=”BLAST Reporting” version=”1.0.5”>
```

Uploads to main Galaxy toolshed should include a version number increment.

However, try to limit uploads to significant changes.

Mask of Yul Brynner, Westworld (1973)

If version number stays the same: the upload becomes an update to the existing toolshed version. If a Galaxy admin downloads your changes they will be applied to their currently loaded tool (if same version). This may be a problem for experimental reproducibility

Preparing a Toolshed Update

Make a tar archive of the tool folder contents.

Select which files or subfolders to include or exclude in the archive. E.g. include all files **except** text editor backups, compiled python templates, and planemo output reports:

```
> tar -zcvf [tool name].tar.gz * --exclude "*~" --exclude "*.pyc"
 --exclude "tool_test_output*" --exclude "*.gz"
```

Download to your local computer so that it can be uploaded to a Galaxy toolshed. The last “./” places the file in your current folder.

```
> scp [your user name]@[your server name]:
/usr/local/galaxy/galaxytest/galaxy-dist/tools/TESTING/[folder]/
[tool name].tar.gz ./
```

Links

The Galaxy developer's email list is a great place to get help
http://dev.list.galaxyproject.org

Galaxy Documentation
https://wiki.galaxyproject.org/Admin/Tools/WritingTests
https://wiki.galaxyproject.org/Admin/Tools/AddToolTutorial
https://wiki.galaxyproject.org/Tools/BestPractices
https://wiki.galaxyproject.org/ToolShed/PublishTool

The Galaxy tool parameter syntax (not complete)
https://wiki.galaxyproject.org/Admin/Tools/ToolConfigSyntax

Planemo Tool tester and Tutorial
http://planemo.readthedocs.org

Cheeta expression language
http://cheetahtemplate.org/learn.html