Integrating and Scaling Analysis Tools

Dan Blankenberg
Galaxy Team
Overview

• **Types of Tools**

• Simple Tool Configuration

• Datasets and Datatypes

• Advanced Tool Configuration by Example: MAF manipulation

• Additional Tool Configuration

• Getting data into Galaxy

• Sending data out of Galaxy
Types of Tools

• ‘Regular’ tools
• Datasource tools (Getting data into Galaxy)
• Sending your datasets elsewhere
‘Regular’ Tools

- Define input datasets and parameters
- Define output datasets and types
- Provide your own executable (and wrapper when required)
- Tool tests
Datasource Tools

- User configurable inputs are provided by external website
- An executable is provided that will work with the standard Galaxy datasource protocol, but you can also provide your own.
Sending Data out of Galaxy

- ‘Data destination’ tools
- External Display Applications (not actually a Tool)
Overview

- Types of Tools
- **Simple Tool Configuration**
- Datasets and Datatypes
- Advanced Tool Configuration by Example: MAF manipulation
- Additional Tool Configuration
- Getting data into Galaxy
- Sending data out of Galaxy
Defining a tool

- Create an XML file Describing the tool
- Add a reference to this new file to tool_conf.xml
Tool Configuration

- Tool Action - Default tool action should be adequate (Upload tool uses custom tool action)
- Tool Command
- Inputs
  - Action - Used by datasource tools
  - Parameters
- Outputs
- Help
- Tests
A Simple Tool

Here is a representation of a simple tool that computes GC content from a FASTA file:

```
<tool id="fa_gc_content_1" name="Compute GC content">
  <description>for each sequence in a file</description>
  <command interpreter="perl">toolExample.pl $input $output</command>
  <inputs>
    <param format="fasta" name="input" type="data" label="Source file" />
  </inputs>
  <outputs>
    <data format="tabular" name="output" />
  </outputs>

  <tests>
    <test>
      <param name="input" value="fa_gc_content_input.fa"/>
      <output name="out_file1" file="fa_gc_content_output.txt"/>
    </test>
  </tests>

  <help>This tool computes GC content from a FASTA file.</help>
</tool>
```

The tool's interface includes:
- **Source file**: A dropdown selection for the FASTA file.
- **Execute**: Button to run the tool.

This tool computes GC content from a FASTA file.
Overview

- Types of Tools
- Simple Tool Configuration
- Datasets and Datatypes
- Advanced Tool Configuration by Example: MAF manipulation
- Additional Tool Configuration
- Getting data into Galaxy
- Sending data out of Galaxy
Datasets and Datatypes

- All datasets are associated with a Datatype
  - File format
  - Type of Data: genomic intervals, sequence, alignment
    - Hierarchical structure allows accepting more specific datatypes without conversion (tabular <- interval <- bed)
    - Datatype Converters allow use of non-subclassed datatypes as direct input for tools (MAF to FASTA converter allows selection of MAF file as input when FASTA is required)
  - Metadata
    - datatypes_conf.xml and lib/galaxy/datatypes
    - AddingDatatypes on the wiki
Metadata

Information that describes the contents of the Dataset

- Line / Sequence Counts
- Count and type of tabular columns
- Column assignments: start, stop, strand
- Genome build(s): dbkey, species
- Files: indexes

Used by tools to customize interface and results
Overview

• Types of Tools
• Simple Tool Configuration
• Datasets and Datatypes
• Advanced Tool Configuration by Example: MAF manipulation
• Additional Tool Configuration
• Getting data into Galaxy
• Sending data out of Galaxy
An Example Toolset

- A set of tools to manipulate Multiple Alignment Format (MAF) files
- Growing collection of publicly available precomputed alignments
- Extract subregions of interest from cached or user uploaded alignments
- Additional manipulations: format conversions, reverse complement, filters, region coverage, visualization, etc.
The MAF format

# maf version=1
a score=606741.000000
s hg18.chr7   3532968  27 - 158821424 ATGCTGTCCCTCTTCCCCAGCCCAGGG
s panTro2.chr7 3633994  27 - 160261443 ATGCTGTCCCTCTTCCCCAGCCCAGGG
s rheMac2.chr3 3465688  27 - 196418989 ATGCTGCCCCTCTTCCCCAGCCCCGGGG
s canFam2.chr16 40964059  24 -  62570175 ATGCCCCCCC---CCTCCCACCTCAGTG

* Start positions on negative strand are relative to the reverse complement of the source sequence.

<table>
<thead>
<tr>
<th>line indicator</th>
<th>species</th>
<th>chromosome</th>
<th>start position*</th>
<th>number of bases</th>
<th>strand</th>
<th>chromosome length</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>hg18.chr7</td>
<td>3532968</td>
<td>27</td>
<td>-</td>
<td>158821424</td>
<td>ATGCTGTCCCTCTTCCCCAGCCCAGGG</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>panTro2.chr7</td>
<td>3633994</td>
<td>27</td>
<td>-</td>
<td>160261443</td>
<td>ATGCTGTCCCTCTTCCCCAGCCCAGGG</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>rheMac2.chr3</td>
<td>3465688</td>
<td>27</td>
<td>-</td>
<td>196418989</td>
<td>ATGCTGCCCCTCTTCCCCAGCCCCGGGG</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>canFam2.chr16</td>
<td>40964059</td>
<td>24</td>
<td>-</td>
<td>62570175</td>
<td>ATGCCCCCCC---CCTCCCACCTCAGTG</td>
<td></td>
</tr>
</tbody>
</table>
MAF Datatype

- Indexed automatically by Framework
- Metadata: dbkey, species, index file
Example Tool:
Alignment Extractor

Extractors take genomic intervals as the input and return alignments corresponding to these intervals as illustrated below.

tools/maf/interval2maf.xml
Extractor Tool Requirements

- Require an interval dataset of known dbkey (Genome Build)
- Work on MAFs in users History or from locally Cached source
- List of available alignments should be filtered based upon the dbkey of the input interval file
- User can select species to include in extracted MAF
- Different command line is used depending upon MAF source type

tools/maf/interval2maf.xml
Input Parameter types

Basic
- Text
- Integer
- Float
- Select
  - Static
  - Dynamic
- Boolean
- Genome build
- Data column
- Data
- Hidden
- Base URL
- File
- Drill down

- Grouping
- Conditional
- Repeat
- Config Files
Extractor Tool Requirements

- Require an interval dataset of known dbkey (Genome Build)
- Work on MAFs in users History or from locally Cached source
- List of available alignments should be filtered based upon the dbkey of the input interval file
- User can select species to include in extracted MAF
- Different command line is used depending upon MAF source type

tools/maf/interval2maf.xml
Require an interval dataset of known dbkey

- Parameter Validation

tools/maf/interval2maf.xml
Extractor Tool Requirements

- Require an interval dataset of known dbkey (Genome Build)

- **Work on MAFs in users History or from locally Cached source**

- List of available alignments should be filtered based upon the dbkey of the input interval file

- User can select species to include in extracted MAF

- Different command line is used depending upon MAF source type

```
tools/maf/interval2maf.xml
```
Work on MAFs in users History or from locally Cached source

- Conditional Parameters: Allows displaying different sets of parameters based upon the value in a select list

```xml
<conditional name="maf_source_type">
  <param name="maf_source" type="select" label="MAF Source">
    <option value="cached" selected="true">Locally Cached Alignments</option>
    <option value="user">Alignments in Your History</option>
  </param>
  <when value="user">
    <param name="mafFile" label="Choose alignments" type="data">
      <options>
        <filter type="data_meta" ref="input1" key="dbkey" />
      </options>
      <validator type="dataset_ok_validator" />
    </param>
    <param name="species" type="select" display="checkboxes" multiple="true" label="Choose species" help="Select species to be included in the final alignment">
      <options>
        <filter type="data_meta" ref="mafFile" key="species" />
      </options>
    </param>
  </when>
  <when value="cached">
    <param name="mafType" type="select" label="Choose alignments">
      <options from_file="maf_index.loc">
        <column name="name" index="0" />
        <column name="value" index="1" />
        <column name="dbkey" index="2" />
        <column name="species" index="3" />
        <filter type="data_meta" ref="input1" key="dbkey" column="2" multiple="true" separator=" "/>
        <validator type="no_options" message="No alignments are available for the build associated with the selected interval file" />
      </options>
    </param>
    <param name="species" type="select" display="checkboxes" multiple="true" label="Choose species" help="Select species to be included in the final alignment">
      <options from_file="maf_index.loc">
        <column name="uid" index="1" />
        <column name="value" index="3" />
        <filter type="param_value" ref="mafType" name="uid" column="1" />
        <filter type="multiple_splitter" column="3" separator=" "/>
      </options>
    </param>
  </when>
</conditional>
```
Work on MAFs in users History or from locally Cached source

This tool takes genomic coordinates, superimposes them on multiple alignments (in MAF format) stored on the Galaxy site or from your history, and excises alignment blocks corresponding to each set of coordinates. Alignment blocks that extend past START and/or END positions of an interval are trimmed. Note that a genomic interval may correspond to two or more alignment blocks.
Extractor Tool
Requirements

- Require an interval dataset of known dbkey (Genome Build)
- Work on MAFs in users History or from locally Cached source
- **List of available alignments should be filtered based upon the dbkey of the input interval file**
- User can select species to include in extracted MAF
- Different command line is used depending upon MAF source type

tools/maf/interval2maf.xml
List of available alignments should be filtered based upon the dbkey of the input interval file

- MAF from user’s history
- Use ‘data_meta’ filter to match dbkeys between genomic intervals and MAF file
List of available alignments should be filtered based upon the dbkey of the input interval file

- **Locally cached MAFs**
- Don’t want to modify the tool each time a new MAF set is added to the cache
- **Dynamic Select**: Use a separate file to describe available cached MAFs

```xml
<param name="mafType" type="select" label="Choose alignments">
  <options from_file="maf_index.loc">
    <column name="name" index="0"/>
    <column name="value" index="1"/>
    <column name="dbkey" index="2"/>
    <column name="species" index="3"/>
  </options>
  <filter type="data_meta" ref="input1" key="dbkey" column="2" multiple="True" separator="","/>
  <validator type="no_options" message="No alignments are available for the build associated with the selected interval.">
  </validator>
</param>
```
Extractor Tool Requirements

• Require an interval dataset of known dbkey (Genome Build)

• Work on MAFs in users History or from locally Cached source

• List of available alignments should be filtered based upon the dbkey of the input interval file

• **User can select species to include in extracted MAF**

• Different command line is used depending upon MAF source type

  `tools/maf/interval2maf.xml`
User can select species to include in extracted MAF

- MAF from a User’s history
- Use *species* metadata value from MAF file to dynamically generate a list of checkboxes
User can select species to include in extracted MAF

- **Locally Cached MAFs**
- Uses same file that was used to generate list of available cached MAFs
- Comma separated list of species is used to dynamically generate a list of checkboxes

```xml
<param name="species" type="select" display="checkboxes" multiple="true" label="Choose species" help="Select species for inclusion in extracted MAF">
  <options from_file="maf_index.loc">
    <column name="uid" index="1"/>
    <column name="name" index="3"/>
    <filter type="param_value" ref="mafType" name="uid" column="1"/>
    <filter type="multiple_splitter" column="3" separator=",">
      <option value="canFam1"/>
      <option value="danRer1"/>
      <option value="fr1"/>
      <option value="galGal2"/>
      <option value="hg17"/>
      <option value="mm5"/>
      <option value="panTro1"/>
      <option value="rn3"/>
    </filter>
  </options>
</param>
```

```bash
$m aaf_index.loc
# Display_name UID
ENCODE_TBA (hg17) ENCODE_TBA_hg17 armadillo,baboon,galGal2,panTro1,Colobus_monkey,cow,canFam1,dusky_titi,elephant,fr1,galago,hedgehog,hg
ENCODE_MAVID (hg17) ENCODE_MAVID_hg17 armadillo,baboon,galGal2,panTro1,Colobus_monkey,cow,canFam1,dusky_titi,elephant,fr1,galago,hedgehog,hg
ENCODE_TBA (hg16) ENCODE_TBA_hg16 armadillo,baboon,galGal2,panTro1,Colobus_monkey,cow,canFam1,dusky_titi,elephant,fr1,galago,hedgehog,hg
8-way multIZ (hg17) 8_WAY_MULTIZ_hg17 canFam1,canRer1,fr1,galGal2,hg17,mm5,panTro1,rn3,canFam1,canRer1,fr1,galGal2,hg17,mm5,p
17-way multIZ (hg18) 17_WAY_MULTIZ_hg18 hg18,panTro1,bosTau2,rheMac2,mm8,rn4,canFam2,echTell,loxAfr1,oryCun1,canRer3,monDom4,dasNov1,g
3-way multIZ (hg18,panTro2,rheMac2) 3_WAY_MULTIZ_hg18 hg18,panTro2,rheMac2,hg18,panTro2,rheMac2 /depot/data2/galaxy/hg18/align.
```
Extractor Tool
Requirements

- Require an interval dataset of known dbkey (Genome Build)
- Work on MAFs in users History or from locally Cached source
- List of available alignments should be filtered based upon the dbkey of the input interval file
- User can select species to include in extracted MAF
- **Different command line is used depending upon MAF source type**

`tools/maf/interval2maf.xml`
Command-line Generation

- Uses Cheetah templating language which allows substitution of user selected input values into command
- Any programming language can be used for a tool
- Control statements can be used: if, else, loops

```python
<command interpreter="python">
    #if $maf_source_type.maf_source == "user"
        interval2maf.py --dbkey=${input1.dbkey} --chromCol=${input1.metadata.chromCol} --startCol=${input1.metadata.startCol} --endCol=${input1.metadata.endCol} --strandCol=${input1.metadata.strandCol} --mafFile=${maf_source_type.mafType} --interval_file=${input1} --output_file=${out_file1} --mafIndexFile=${GALAXY_DATA_INDEX_DIR}/maf_index.loc
    #else
        interval2maf.py --dbkey=${input1.dbkey} --chromCol=${input1.metadata.chromCol} --startCol=${input1.metadata.startCol} --endCol=${input1.metadata.endCol} --strandCol=${input1.metadata.strandCol} --mafFile=${maf_source_type.mafType} --interval_file=${input1} --output_file=${out_file1} --mafIndexFile=${GALAXY_DATA_INDEX_DIR}/maf_index.loc --species=${maf_source_type.species}
    #end if
    #if $split_blocks_by_species_selector.split_blocks_by_species == "split_blocks_by_species"
        remove_all_gap_columns=${split_blocks_by_species_selector.remove_all_gap_columns}
    #end if
</command>
```
Another Example: MAF Visualization

- We have a Java applet (Gmaj) that is able to view MAF alignments
  - requires a specially constructed Zip file
  - 2 manifest-like Files need to be provided
    - describing how to draw display (title, reference species, etc)
    - describing annotation files
  - User can optionally define Annotations (exons, repeats, highlights, underlays, links) for each species in the alignment
  - A hierarchical list of warnings can be suppressed by the user

  tools/visualization/GMAJ.xml
Gmaj Requirements

- One input MAF file
- Any number of input interval (BED) annotation files
- A hierarchical list of warnings can be suppressed by the user
- 2 Config files
- 1 output dataset of type ‘gmaj.zip’

/tools/visualization/GMAJ.xml
Any number of input interval (BED) annotation files

- Repeat Parameter
- Allows a variable number of a set of parameters
Gmaj Requirements

- One input MAF file
- Any number of input interval (BED) annotation files
- A hierarchical list of warnings can be suppressed by the user
- 2 Config files
- 1 output dataset of type 'gmaj.zip'

tools/visualization/GMAJ.xml
A hierarchical list of warnings can be suppressed by the user.

- Drill Down Parameter

---

Choose Warnings to Suppress:

- MAF File
  - Semantic Assumptions
    - BED Format
    - GFF group is gene name (gff_group)
  - Annotation Files
    - Skipped Items
    - Red Flags
  - Miscellaneous
    - No refseq specified; assuming 'first'
      (default_refseq)
    - One or more bundle entries are not used in parameters
      (file(unused_entry))
    - Skipping blocks for export where reference sequence is hidden or all gaps
      (export_skip)
    - Possible parse error: token ends with an escaped quote
      (escaped_quote)
    - Draggable panel dividers will not be sticky (no_sticky)
    - Selecting a large block may be very slow (big_block)
Gmaj Requirements

- One input MAF file
- Any number of input interval (BED) annotation files
- A hierarchical list of warnings can be suppressed by the user
- 2 Config files
- 1 output dataset of type 'gmaj.zip'

Tools/visualization/GMAJ.xml
Configfile Inputs

• Temporary file is available as input to the tool
• Cheetah templating language is used to define the contents of a temporary file
• Examples
  • Formatted list of input values
  • R (statistics package) scripts
  • Gnuplot Commands
  • Dynamically generated python scripts
Gmaj manifest

```xml
<configfile name="gmaj_file">#gmaj

title = "Galaxy: $maf_input.name"
alignfile = input.maf
refseq = $refseq	bistream = .bed .gff .gtf
#if $nowarn.value:
ownarn = $nowarn
#endif

#set $seq_count = 0
#for $annotation in $enumerate( $annotations ):
#if $annotation.annotation_style.style == "galaxy":
  #set $species_chromosomes = {};
#if $maf_input.dataset.metadata.species_chromosomes:
  #for $line in open( $maf_input.dataset.metadata.species_chromosomes.file_name ):
    #set $fields = $line.split( "\t" )
  #if $fields:
    #set $spec = $fields.pop( 0 )
    #set $species_chromosomes[spec] = $fields
  #end if
#if $species_chromosomes and $annotation.annotation_style['species'].value in $species_chromosomes and $species_chromosomes[$annotation.annotation_style['species'].value, $chrom] for $chrom in $species_chromosomes[$annotation.annotation_style['species']]:
#else:
  #set $seq_names = [$annotation.annotation_style['species']]
#endif
#endif
#end if

#if $seq_count >= 0:
  #for $seq_name in $seq_names:
    seq $(seq_count):
      seqname = $seq_name
      #if $annotation.annotation_style['exons_file'].dataset:
        exons = $(annotation_count).exons.$(annotation.annotation_style['exons_file'].extension)
      #end if
      #if $annotation.annotation_style['repeats_file'].dataset:
        repeats = $(annotation_count).repeats.$(annotation.annotation_style['repeats_file'].extension)
      #end if
      #if $annotation.annotation_style['links_file'].dataset:
        links = $(annotation_count).links.$(annotation.annotation_style['links_file'].extension)
      #end if
      #if $annotation.annotation_style['underlays_file'].dataset:
        underlays = $(annotation_count).underlays.$(annotation.annotation_style['underlays_file'].extension)
      #end if
      #if $annotation.annotation_style['highlights_file'].dataset:
        highlights = $(annotation_count).highlights.$(annotation.annotation_style['highlights_file'].extension)
      #end if
  #end if
#endif
</configfile>
```

`tools/visualization/GMAJ.xml`
Gmaj Requirements

• One input MAF file
• Any number of input interval (BED) annotation files
• A hierarchical list of warnings can be suppressed by the user
• 2 Configfiles
• 1 output dataset of type ‘gmaj.zip’

tools/visualization/GMAJ.xml
Gmaj Result
Overview

• Types of Tools
• Simple Tool Configuration
• Datasets and Datatypes
• Advanced Tool Configuration by Example: MAF manipulation
• Additional Tool Configuration
• Getting data into Galaxy
• Sending data out of Galaxy
Additional Tool Configuration

- Many input parameter types available
- More dynamic select sources and filters
- Additional parameter validators
- A Note on Tool Outputs
- Onscreen Help
- Functional Tool Tests
Input Parameter types

- Basic
  - Text
  - Integer
  - Float
  - Select
    - Static
    - Dynamic
  - Boolean
  - Genome build
  - Data column
  - Data
  - Hidden
  - Base URL
  - File
  - Drill down

- Grouping
- Conditional
- Repeat
- Config Files
Dynamic Select Parameter Sources

- Options from a File on Disk (*.loc)
- Options from the contents of a Dataset
- Options from Metadata of a Dataset
- Options can be filtered based upon the value of other tool parameters as well as the above
Options from the contents of a Dataset

<param name="feature" type="select" multiple="true" label="Extract features">
    <options from_dataset="input1">
        <column name="name" index="0"/>
        <column name="value" index="0"/>
        <filter type="unique_value" name="unique" column="0"/>
    </options>
</param>

Extract features:

| chr  | bed2gff CDS1001.1_cds 0 0 chr1 148325019_r | 148325018 148325075 . + . score '0'; |
| chr2 | bed2gff CDS1201.1_cds 0 0 chr2 134288584_r | 134288583 134288638 + . . score '0'; |
| chr3 | bed2gff CDS3922.1_cds 0 0 chr3 50241230_r | 50241230 50241237 + . . score '0'; |
| chr4 | bed2gff CDS4855.1_cds 0 0 chr4 41411593_r | 41411593 41411644 + . . score '0'; |
| chr5 | bed2gff CDS5491.1_cds 0 0 chr5 26907299_r | 26907299 26907364 + . . score '0'; |
| chr6 | bed2gff CDS6141.1_cds 0 0 chr6 128764157_r | 128764157 128764189 + . . score '0'; |
| chr7 | bed2gff CDS7248.1_cds 0 0 chr7 55251624_r | 55251624 55252124 + . . score '0'; |
| chr8 | bed2gff CDS7725.1_cds 0 0 chr8 1731397 1731476 r . . score '0'; |
| chr9 | bed2gff CDS8736.1_cds 0 0 chr9 3844095_r | 3844095 3844339 + . . score '0'; |
| chr10| bed2gff CDS9526.1_cds 0 0 chr10 112361895_r | 112361895 112361953 + . . score '0'; |
| chr11| bed2gff CDS9949.1_cds 0 0 chr11 98710241_r | 98710241 98712285 + . . score '0'; |
| chr12| bed2gff CDS10016.1_cds 0 0 chr12 41489673_r | 41489673 41489705 + . . score '0'; |
| chr13| bed2gff CDS10395.1_cds 0 0 chr13 37430 37557 r . . score '0'; |
| chr14| bed2gff CDS13891.1_cds 0 0 chr14 23780115_r | 23780115 23780321 + . . score '0'; |
| chr15| bed2gff CDS12866.1_cds 0 0 chr15 50668506_r | 50668506 50668564 + . . score '0'; |
| chr16| bed2gff CDS13249.1_cds 0 0 chr16 33308234_r | 33308234 33308423 + . . score '0'; |
| chr17| bed2gff CDS13014.1_cds 0 0 chr17 32707033_r | 32707033 32707192 + . . score '0'; |
| chr18| bed2gff CDS13807.1_cds 0 0 chr18 30120224_r | 30120224 30120265 + . . score '0'; |
| chrX | bed2gff CDS14065.1_cds 0 0 chrX 122745048_r | 122745048 122745924 + . . score '0'; |

tools/extract/extract_GFF_Features.xml
# Dynamic Select Filters

<table>
<thead>
<tr>
<th>Filter</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>data_meta</td>
<td>Options by a dataset metadata value</td>
</tr>
<tr>
<td>param_value</td>
<td>Filter options by the value of another parameter</td>
</tr>
<tr>
<td>static_value</td>
<td>Filter options by a static value</td>
</tr>
<tr>
<td>unique_value</td>
<td>Filter options to all unique values</td>
</tr>
<tr>
<td>multiple_splitter</td>
<td>Split an option into multiple options</td>
</tr>
<tr>
<td>add_value</td>
<td>Add an option by value</td>
</tr>
<tr>
<td>remove_value</td>
<td>Remove an option by value</td>
</tr>
<tr>
<td>sort_by</td>
<td>Order options</td>
</tr>
</tbody>
</table>
# Parameter Validation

<table>
<thead>
<tr>
<th>Validator</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>expression</td>
<td>Evaluates a python expression using the parameter value</td>
</tr>
<tr>
<td>regex</td>
<td>Checks if parameter value matches the specified regular expression</td>
</tr>
<tr>
<td>in_range</td>
<td>Ensures a numerical parameter value falls between a min and max</td>
</tr>
<tr>
<td>length</td>
<td>Ensures the length of a text parameter value falls between a min and max</td>
</tr>
<tr>
<td>metadata</td>
<td>Checks if specified metadata is missing for a dataset</td>
</tr>
<tr>
<td>unspecified_build</td>
<td>Checks if the dbkey for a dataset has been set</td>
</tr>
<tr>
<td>no_options</td>
<td>Ensures at least one option has been set for select parameters</td>
</tr>
<tr>
<td>empty_field</td>
<td>Ensures a text field has not been left empty</td>
</tr>
<tr>
<td>dataset_metadata_in_file</td>
<td>Checks if a particular metadata value for a dataset exists in a File</td>
</tr>
<tr>
<td>dataset_ok</td>
<td>Ensures an input dataset is in the OK state</td>
</tr>
</tbody>
</table>
Tool Outputs

• Each tool should have at least One output

• Outputs can be made optional based upon user input

```xml
<outputs>
  <!-- Variable number of outputs. Either one (for single-end) or two (for paired-end) -->
  <data name="output1" format="fastqsanger"/>
  <data name="output2" format="fastqsanger">
    <filter>paired[ 'pairedSingle' ] == 'paired'</filter>
  </data>
</outputs>
```

• When number of outputs is indeterminate, additional outputs can be created by placing specially named files in a certain directory

• Composite Datatypes: a single history item composed of several files
Onscreen Tool Help

- Written using the reStructuredText Format (RST)
- Human Readable plaintext markup syntax
- RST parser converts plaintext into HTML for display with each Tool
This tool allows you to plot values contained in columns of a dataset against each other and also allows you to have different series corresponding to the same or different datasets in one plot.

This tool throws an error if the columns selected for plotting are absent or are not numeric and also if the lengths of these columns differ.

**Example**

Input file:

1 68 4.1
2 71 4.6
3 62 3.8
4 75 4.4
5 58 3.2
6 60 3.1
7 67 3.8
8 68 4.1
9 71 4.3
10 69 3.7

Create a two series XY plot on the above data:
- Series 1: Red Dashed-Line plot between columns 1 and 2
- Series 2: Blue Circular-Point plot between columns 3 and 2

Create a two series XY plot on the above:
- Series 1: Red Dashed-Line plot between
- Series 2: Blue Circular-Point plot between

.. image:: ../static/images/xy_example.jpg

</help>
Tool Tests

• Specify User Inputs
• Provide Files (test-data/)
• Input Files
• File to compare to Output

<tests>
  <test>
    <param name="input1" value="3.maf" ftype="maf"/>
    <param name="species" value="canFam1"/>
    <param name="fasta_type" value="concatenated"/>
    <output name="out_file1" file="cf_maf2fasta_concat.dat" ftype="fasta"/>
  </test>
  <test>
    <param name="input1" value="4.maf" ftype="maf"/>
    <param name="species" value="hg17,panTrol,rheMac2,rn3,mm7,canFam2,bosTau2,dasNov1"/>
    <param name="complete_blocks" value="partial_allowed"/>
    <param name="fasta_type" value="multiple"/>
    <output name="out_file1" file="cf_maf2fasta_new.dat" ftype="fasta"/>
  </test>
</tests>
## A failed test

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Status</th>
<th>Output</th>
<th>Exception</th>
</tr>
</thead>
<tbody>
<tr>
<td>functional.test_toolbox.TestForTool_substitutions1.test_tool</td>
<td>Fetch substitutions (substitutions1) &gt; Test-1</td>
<td>success</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>functional.test_toolbox.TestForTool_hyphy_nj_tree_wrapper1.test_tool</td>
<td>Neighbor Joining Tree (hyphy_nj_tree_wrapper1) &gt; Test-1</td>
<td>success</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>functional.test_toolbox.TestForTool_cshl_fastx_artifacts_filter.test_tool</td>
<td>Remove sequencing artifacts (cshl_fastx_artifacts_filter) &gt; Test-1</td>
<td>success</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>functional.test_toolbox.TestForTool_cshl_fastx_artifacts_filter.test_tool</td>
<td>Remove sequencing artifacts (cshl_fastx_artifacts_filter) &gt; Test-2</td>
<td>failure</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>functional.test_toolbox.TestForTool_subtract_query1.test_tool</td>
<td>Subtract Whole Query (subtract_query1) &gt; Test-1</td>
<td>success</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>functional.test_toolbox.TestForTool_subtract_query1.test_tool</td>
<td>Subtract Whole Query (subtract_query1) &gt; Test-2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overview

- Types of Tools
- Simple Tool Configuration
- Datasets and Datatypes
- Advanced Tool Configuration by Example: MAF manipulation
- Additional Tool Configuration
  - Getting data into Galaxy
- Sending data out of Galaxy
Datasource Tools

- **Synchronous**
  - data ready immediately

- **Asynchronous**
  - data needs to be generated by external source first
Synchronous Datasource Tools

- When the user has finished at the external site, the external site directs the user back to Galaxy
- Parameters and a retrieval URL are provided
- In the background, Galaxy creates a new dataset and then uses provided URL and parameters to retrieve the new dataset content
UCSC Datasource Tool

<tool name="UCSC Main" id="ucsc_table_directasdf1" tool_type="data_source">
  <description>table browser</description>
  <command interpreter="python">data_source.py $output $__app__.config.output_size_limit</command>
  <inputs action="http://genome.ucsc.edu/cgi-bin/hgTables" check_values="false" method="get">
    <display>go to UCSC Table Browser $GALAXY_URL</display>
    <param name="sendToGalaxy" type="hidden" value="1" />
    <param name="hgta_compressType" type="hidden" value="none" />
    <param name="hgta_outputType" type="hidden" value="bed" />
  </inputs>
  <request_param_translation>
    <request_param galaxy_name="URL_method" remote_name="URL_method" missing="post" />
    <request_param galaxy_name="URL" remote_name="URL" missing="" />
    <request_param galaxy_name="dbkey" remote_name="db" missing="?" />
    <request_param galaxy_name="organism" remote_name="org" missing="unknown species" />
    <request_param galaxy_name="table" remote_name="hgta_table" missing="unknown table" />
    <request_param galaxy_name="description" remote_name="hgta_regionType" missing="no description" />
    <value_translation>
      <value galaxy_value="tabular" remote_value="primaryTable" />
      <value galaxy_value="tabular" remote_value="selectedFields" />
      <value galaxy_value="wig" remote_value="wigData" />
      <value galaxy_value="interval" remote_value="tab" />
      <value galaxy_value="html" remote_value="hyperlinks" />
      <value galaxy_value="fasta" remote_value="sequence" />
    </value_translation>
  </request_param>
  <uihints minwidth="800" />
  <outputs>
    <data name="output" format="tabular" />
  </outputs>
  <options sanitize="False" refresh="True" />
</tool>

tools/data_source/ucsc_tablebrowser.xml
Table Browser

Use this program to retrieve the data associated with a track in text format, to calculate intersections between tracks, and to retrieve DNA sequence covered by a track. For help in using this application see [Using the Table Browser](#) for a description of the controls in this form, the [User's Guide](#) for general information and sample queries, and the OpenHelix Table Browser [tutorial](#) for a narrated presentation of the software features and usage. For more complex queries, you may want to use [Galaxy](#) or our [public MySQL server](#). To examine the biological function of your set through annotation enrichments, send the data to [GREAT](#). Refer to the [Credits](#) page for the list of contributors and usage restrictions associated with these data.

**clade:** Mammal  
**genome:** Human  
**assembly:** Mar. 2006 (NCBI36/hg18)  
**group:** Genes and Gene Prediction Tracks  
**track:** UCSC Genes  
**table:** knownGene  
**region:**  
- genome  
- ENCODE Pilot regions  
- position  
**identifiers (names/accessions):** paste list  
**filter:** create  
**intersection:** create  
**correlation:** create  
**output format:** BED – browser extensible data  
**GREAT**  
**output file:**  
- plain text  
- gzip compressed  
**file type returned:**
Asynchronous Datasource Tools

- When the user has finished at the external site, the external site directs the user back to Galaxy.
- Parameters and an URL is provided.
- In the background, Galaxy creates a new dataset and then creates a new GALAXY_URL (uniquely identifying this new dataset) and sends this and provided parameters to the data source.
- External data source then runs processes to generate dataset contents and sends a URL back to GALAXY_URL.
Overview

• Types of Tools
• Simple Tool Configuration
• Datasets and Datatypes
• Advanced Tool Configuration by Example: MAF manipulation
• Additional Tool Configuration
• Getting data into Galaxy
• Sending data out of Galaxy
Sending Data out of Galaxy

- Data Destination Tools
  - Appear on the left-hand side tool menu
- External Display Applications
  - Appear as links in right-hand side, for each applicable History item preview
- These are not tools
Data Destination Tools

After clicking the **Execute** button, you will be redirected to the EpiGRAPH test website. Please be patient while the dataset is being imported. Inside EpiGRAPH:

**What it does:**
This tool sends the selected dataset to EpiGRAPH for analysis.

---

**EpiGRAPH outline**
The EpiGRAPH web service sends the dataset and runs:

- **17: MACS on data 12**

---

- **Send Data**
  - Perform genome analysis and prediction with EpiGRAPH

**ENCODE Tools**
- Lift-Over
- Text Manipulation
External Display Applications

- Define an XML configuration which describes how and where to present the data to the External Web Application.
  - Static
  - Dynamic - display options can be loaded from a file.
- Inform Galaxy about the new display by adding to the appropriate datatype in `datatypes_conf.xml`.

wiki:ExternalDisplayApplications/Tutorial
Displaying BAM files at UCSC

- The data to be displayed is provided by giving a public URL to the UCSC genome browser of the form: http://genome.ucsc.edu/cgi-bin/hgTracks?db=UCSC_GENOME_BUILD&hgt.customText=URL_OF_CUSTOM_TRACK

- 3 data files are needed to be provided by URL:
  - a custom track definition
  - the BAM file
  - the BAM index

- With the following requirement: the index must have the same name as the BAM file, but have the additional suffix of '.bai'
Static External Display Application
BAM at UCSC
What if...

- We have multiple mirrors and want a link for each?
- Want to only provide links for certain Genome Builds (filtering by metadata: dbkeys)?
Dynamic External Display Application

<display id="ucsc_bam" version="1.0.0" name="display at UCSC">
  <!-- Load links from file: one line to one link -->
  <dynamic_links from_file="tool-data/shared/ucsc/ucsc_build_sites.txt" skip_startwith="#" id="0" name="0">

    <!-- Define parameters by column from file, allow splitting on builds -->
    <dynamic_param name="site_id" value="0"/>
    <dynamic_param name="ucsc_link" value="1"/>
    <dynamic_param name="builds" value="2" split="True" separator=","/>

    <!-- Filter out some of the links based upon matching site_id to a Galaxy application configuration parameter and builds from the category parameter. -->
    <filter>${site_id in $APP.config.ucsc_display_sites}</filter>
    <filter>${dataset.dbkey in $builds}</filter>

    <!-- We define url and params as normal, but values defined in dynamic_param are available by specified name -->
    <url>${ucsc_link}db=${qp($bam_file.dbkey)}&amp;hgt.customText=${qp($track.url)}"></url>
    <param type="data" name="bam_file" url="galaxy_${DATASET_HASH}.bam" strip_https="True"/>
    <param type="data" name="bai_file" url="galaxy_${DATASET_HASH}.bam.bai" metadata="bam_index" strip_https="True"/>
    <param type="template" name="track" viewable="True" strip_https="True">
      track type=bam name="${bam_file.name}" bigDataUrl="${bam_file.url} db=${bam_file.dbkey}
    </param>

  </dynamic_links>
</display>
Specifics available at the Galaxy wiki

For tool developers and labs

(if you want to run your own Galaxy or add tools)

Galaxy is an easy-to-use, open-source, scalable framework for tool and data integration. Tool developers, stop wasting time writing interfaces and get your tools used by biologists! Labs, installing your own local instance of Galaxy is easy, and allows you to use your own dedicated computational resources and custom tools.

New features:
- Read the details of newly introduced features.
- Check out our preliminary site map

The basics
- Installing your own local Galaxy instance and handling data integration
- Adding a tool
- Add an external display application
- Demo scripts

The details
- Configuring Galaxy for a production environment (including cluster support)
- More on tool integration:
  - A complete description of tags used for tool integration.
  - A tutorial on writing of functional tests
- Plugging external data sources into Galaxy
- Adding multiple alignments
- Adding new datatypes
- Data security in Galaxy
- A word about Galaxy's Python eggs

Questions or problems?

If you think you've seen a bug – please, report it!

Please visit our FAQ page to check out responses to some frequently asked questions. If your question is still not answered, here is how to get help:

- To report a bug, an issue with the public instance (i.e., http://usegalaxy.org), or to suggest improvements, use galaxy-bugs.
- To ask a question about how to use Galaxy for data analysis, send an e-mail to galaxy-user.
- For installation, configuration, and tool integration issues, send an e-mail to galaxy-dev.

You can also subscribe to galaxy-user and galaxy-dev to become a member of our rapidly growing community.
But Wait, I have a tool I want to integrate how can I...?

• Ask at the Bar