WS1:
Integrating Tools and Data Sources

Daniel Blankenberg

The Galaxy Team
http://UseGalaxy.org

The Galaxy Team

http://wiki.g2.bx.psu.edu/Galaxy%20Team

Overview

- Introduction to Tools
- Adding a basic tool exercise
- About Data Source tools
- Data Source example
- More complex tools and exercises
- The ToolFactory

Galaxy

http://main.g2.bx.psu.edu/

Google

Galaxy | Analyze Data | Workflow | Shared Data | Visualization | Help | User

Tools Options

- Get Data
- Send Data
- ENCODE Tools
- Lift-Over
- Text Manipulation
- Convert Formats
- FASTA manipulation
- Filter and Sort
- Join, Subtract and Group
- Extract Features
- Fetch Sequences
- Fetch Alignments
- Get Genomic Scores
- Operate on Genomic Intervals
- Statistics
- Graph/Display Data
- Regional Variation
- Multiple regression
- Multivariate Analysis
- Evolution
- Metagenomic analyses
- EMBOSS

NGS TOOLBOX BETA

- NGS: QC and manipulation
- NGS: Mapping
- NGS: SAM Tools
- NGS: Indel Analysis
- NGS: Peak Calling

RGENETICS

- SNP/WGA: Data: Filters
- SNP/WGA: QC: LD: Plots
- SNP/WGA: Statistical Models

Workflows

Map with Bowtie for Illumina

Will you select a reference genome from your history or use a built-in index?:

Use a built-in index

Built-ins were indexed using default options

Select a reference genome:

mm9

if your genome of interest is not listed - contact Galaxy team

Is this library mate-paired?:

Paired-end

Forward FASTQ file:

1: E18 PE.1 Reads

Must have Sanger-scaled quality values with ASCII offset 33

Reverse FASTQ file:

1: E18 PE.1 Reads

Must have Sanger-scaled quality values with ASCII offset 33

Maximum insert size for valid paired-end alignments (-X):

1000

The upstream/downstream mate orientation for valid paired-end alignment against the forward reference strand (--fr/--rf/--ff):

FR (for Illumina)

Bowtie settings to use:

Commonly used

For most mapping needs use Commonly used settings. If you want full control use Full parameter list

Suppress the header in the output SAM file:

☑

Bowtie produces SAM with several lines of header information by default

Execute

What it does

Bowtie is a short read aligner designed to be ultrafast and memory-efficient. It is developed by Ben Langmead and Cole Trapnell. Please cite: Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology 10:R25.

History Options

Imported: SNP Pileup Analysis for Sample E18

- 15: Variants from sample E18, consensus different, in RefSeq Genes
- 14: UCSC mm9 RefSeq Genes
- 13: Variants from sample E18 where consensus base different than ref. base
- 10: Variants from sample E18
- 9: Generate pileup on data 8
- 8: SAM-to-BAM on data 7
- 7: Map with Bowtie for Illumina on data 6 and data 5
- 6: E18 PE.2 Reads Groomed, Trimmed
- 5: E18 PE.1 Reads Groomed, Trimmed
- 4: E18 PE.2 Reads Groomed
- 3: E18 PE.1 Reads Groomed
- 2: E18 PE.2 Reads
- 1: E18 PE.1 Reads

Tools

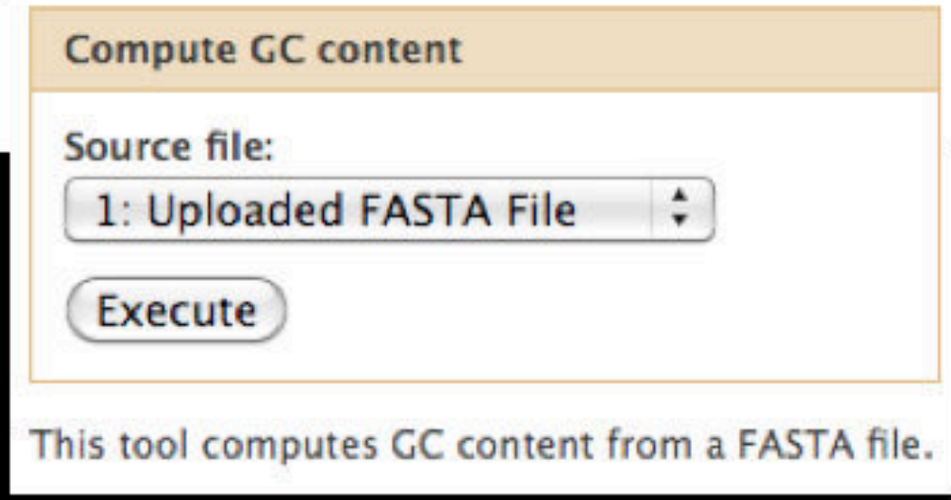
- Any command-line executable
- Accepts zero or more input files and parameters
- Creates one or more output files
- Special types of tools
  - External Data Sources
  - Implicit datatype converters

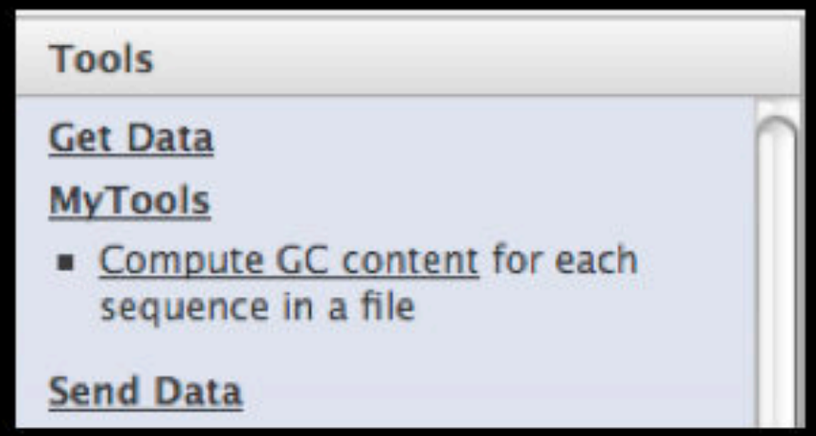
Tool Configuration

- 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 Basic Tool

```
<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>
```






```
<section name="MyTools" id="mTools">
    <tool file="myTools/toolExample.xml" />
</section>
```

tool_conf.xml

Cluster

Cluster intervals of: 6: UCSC Main on Human: knownGene

max distance between intervals: 1 (bp)

min number of intervals per cluster: 2

Return type: Merge clusters into single intervals

Execute

TIP: If your query does not appear in the pulldown menu -> it is not in interval format. Use "edit attributes" to set chromosome, start, end, and strand columns

---

Screencasts!

See Galaxy Interval Operation Screencasts (right click to open this link in another window).

Syntax

- **Maximum distance** is greatest distance in base pairs allowed between intervals that will be considered "clustered". **Negative** values for distance are allowed, and are useful for clustering intervals that overlap
- **Minimum intervals per cluster** allow a threshold to be set on the minimum number of intervals to be considered a cluster. Any area with less than this minimum will not be included in the ouput.
- **Merge clusters into single intervals** outputs intervals that span the entire cluster.
- **Find cluster intervals; preserve comments and order** filters out non-cluster intervals while maintaining the original ordering and comments in the file.
- **Find cluster intervals; output grouped by clusters** filters out non-cluster intervals, but outputs the cluster intervals so that they are grouped together. Comments and original ordering in the file are lost.

Example

cluster.xml

```
<tool id="gops_cluster_1" name="Cluster">
  <description>[[Cluster]] the intervals of a query</description>
  <command interpreter="python2.4">
    gops_cluster.py $input1 $output -1 $input1_chromCol,$input1_startC
                   -d $distance -m $minregions -o $returntype
  </command>
  <inputs>
    <param format="interval" name="input1" type="data">
      <label>Cluster intervals of</label>
    </param>
    <param name="distance" size="5" type="integer" value="1" help="(bp
      <label>max distance between intervals</label>
    </param>
    <param name="minregions" size="5" type="integer" value="2">
      <label>min number of intervals per cluster</label>
    </param>
    <param name="returntype" type="select" label="Return type">
      <option value="1">Merge clusters into single intervals</option>
      <option value="2">Find cluster intervals; preserve comments and
      <option value="3">Find cluster intervals; output grouped by clus
      <option value="4">Find the smallest interval in each cluster</op
      <option value="5">Find the largest interval in each cluster</opt
    </param>
  </inputs>
  <help>

.. class:: infomark

**TIP:** If your query does not appear in the pulldown menu -> it is n

-----

**Screencasts!**

See Galaxy Interval Operation Screencasts_ (right click to open this l

.. _Screencasts: http://www.bx.psu.edu/cgi-bin/trac.cgi/wiki/GopsDesc

-----

**Syntax**

- **Maximum distance** is greatest distance in base pairs allowed betw
- **Minimum intervals per cluster** allow a threshold to be set on the
- **Merge clusters into single intervals** outputs intervals that span
- **Find cluster intervals; preserve comments and order** filters out
- **Find cluster intervals; output grouped by clusters** filters out n
```

Line: 87 Column: 8 XML Soft Tabs: 2

Command-line Executable

- Binaries
- Perl
- Python
- Shell
- R
- ...any others

Input Parameter types

Basic

- Text
- Integer
- Float
- Select
  - Static
  - Dynamic
- Boolean

Genome

- Genome build
- Data column
- Data
- Hidden
- Base URL
- File
- Drill down

Grouping

- Conditional
- Repeat
- Config Files

Input and Output Files

- Datasets
  - History Items
  - Library Items
- File on filesystem
- Metadata in database

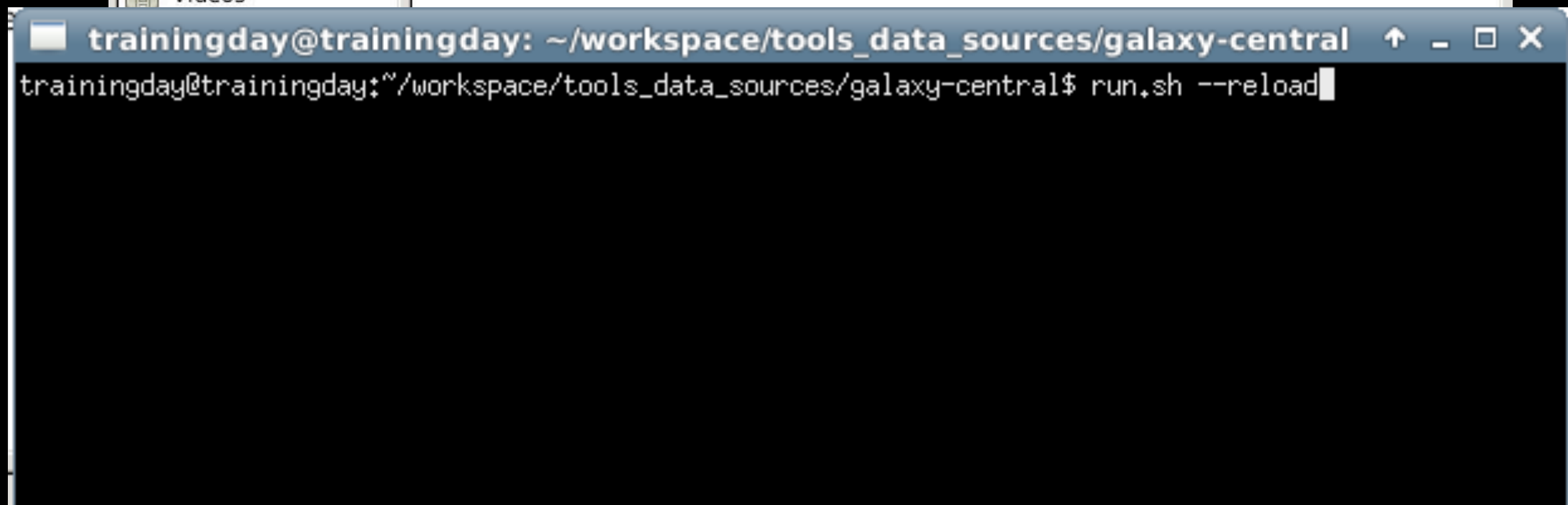
Adding your Own

- Write or download a command-line executable
- Determine number and kind of
  - Input and Output Datasets
  - Input Parameters
- Construct a descriptive tool configuration XML file
  - Write a wrapper script, only if required

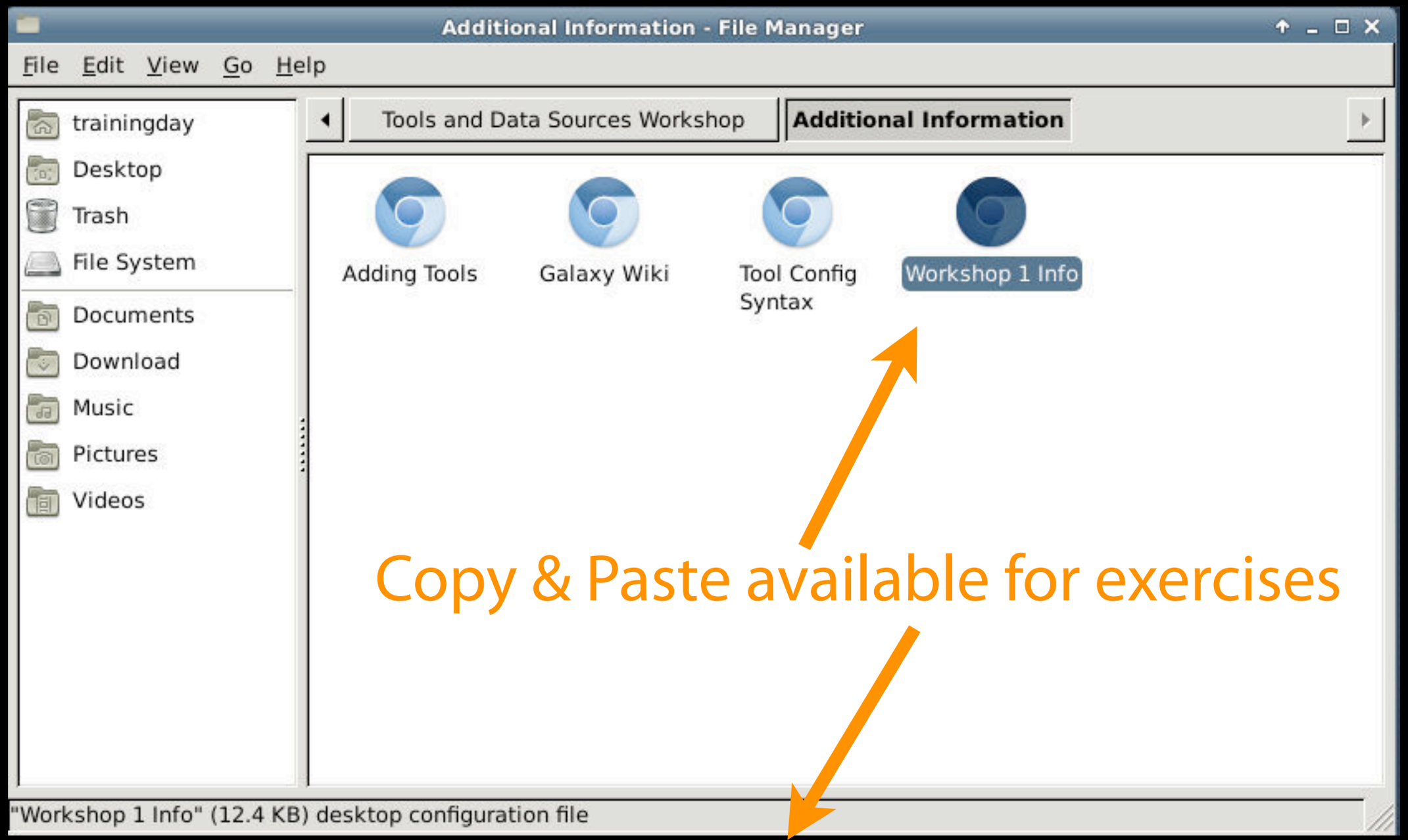
Overview

- Introduction to Tools
- Adding a basic tool exercise
- About Data Source tools
- Data Source example
- More complex tools and exercises
- The ToolFactory

Running Galaxy



Additional Information Folder



Copy & Paste available for exercises

http://wiki.g2.bx.psu.edu/Events/GCC2012/TrainingDay/WS1

Example Command-line Executable

- maf_slice.py
  - Slices a multiple genome alignment by provided genomic regions
    - Inputs
      - Multiple genome alignment file (MAF)
      - Genomic regions file (BED/Interval)

http://wiki.g2.bx.psu.edu/Events/GCC2012/TrainingDay/WS1

Command-line Example

```
python
  maf_slice.py
  --dbkey=hg17
  --mafFile=input.maf
  --interval_file=input.bed
  --output_file=sliced.maf
```

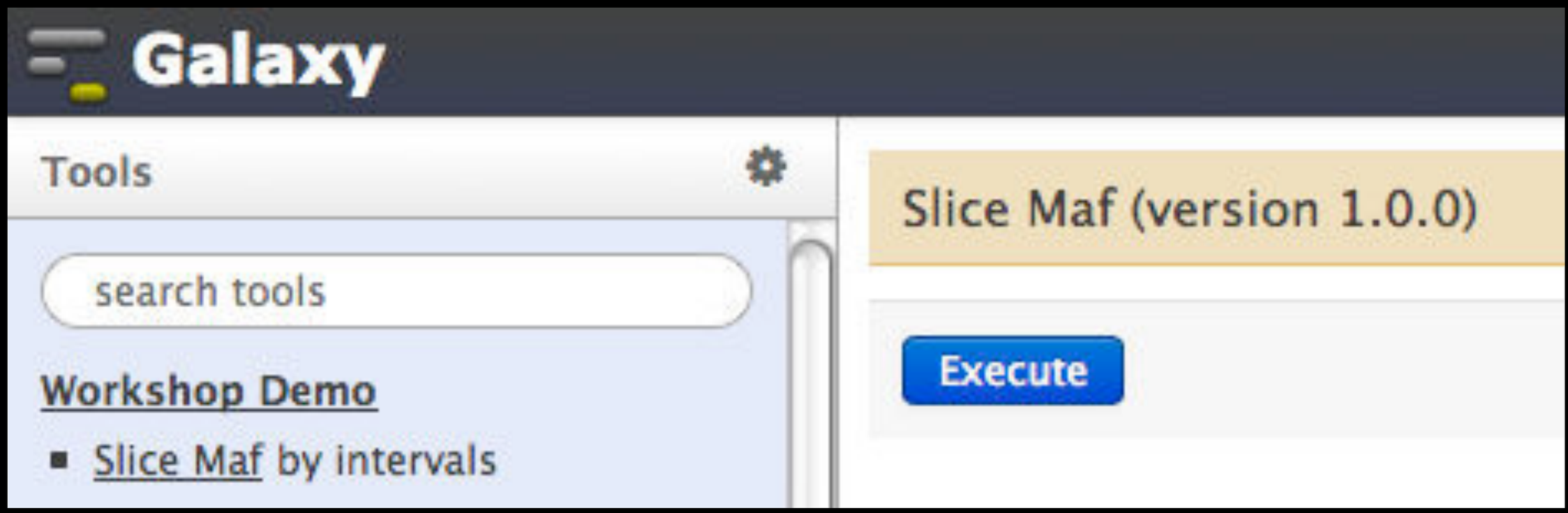
Tool Config Skeleton

```
<tool id="maf_slice" name="Slice MAF" version="1.0.0">
  <description>by intervals</description>
  <command interpreter="python">
maf_slice.py --dbkey=hg17 --mafFile=input.maf --
interval_file=input.bed  --output_file=sliced.maf
   </command>
    <inputs>
    </inputs>
    <outputs>
    </outputs>
    <tests>
    </tests>
    <help>
   </help>
</tool>
```

tools/demo/maf_slice.xml

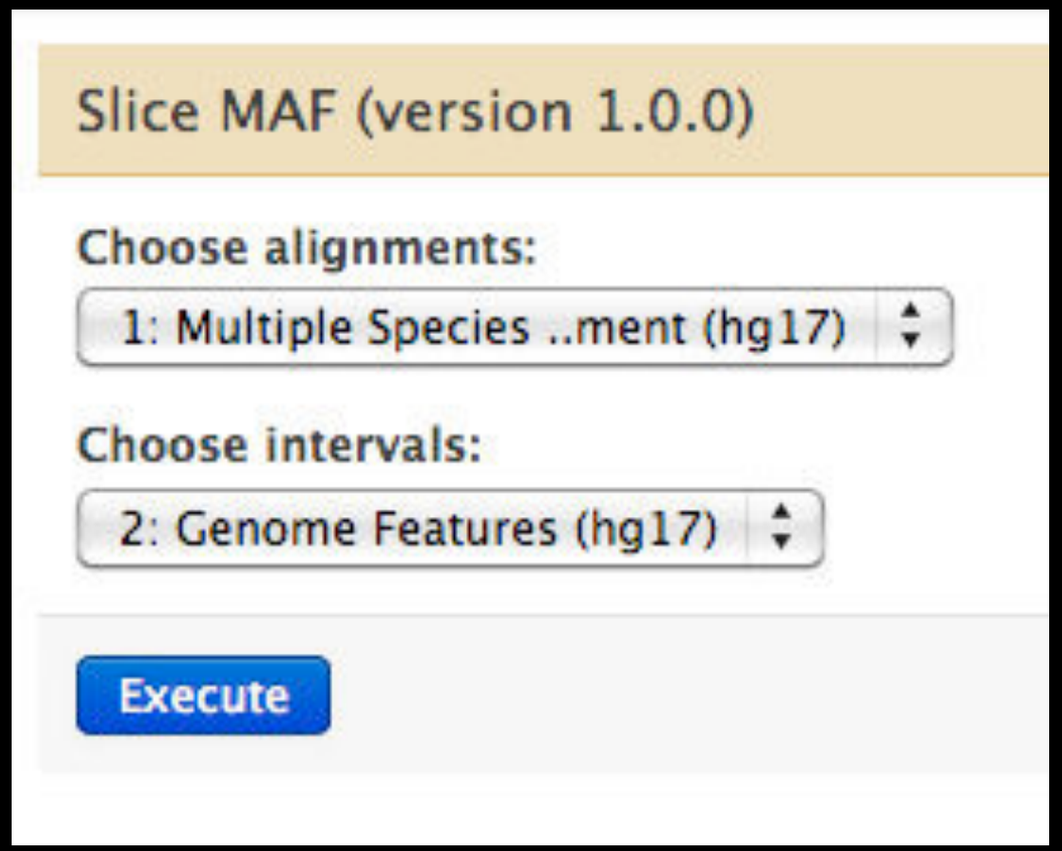
tool_conf.xml

```
<toolbox>
  <section name="Workshop Demo" id="workshop_demo">
    <tool file="demo/maf_slice.xml" />
  </section>
                                        ...
</toolbox>
```


Galaxy
Tools
search tools
Workshop Demo
Slice Maf by intervals
Slice Maf (version 1.0.0)
Execute

Need Some Inputs

```
<inputs>
  <param format="maf" name="maf_input" label="Choose
alignments" type="data"/>
  <param format="bed" name="interval_input" type="data"
label="Choose intervals"/>
</inputs>
```



Get inputs into command-line

```
  <command interpreter="python">
maf_slice.py --dbkey=hg17 --mafFile=${maf_input} --
interval_file=${interval_input} --output_file=sliced.maf
   </command>
```

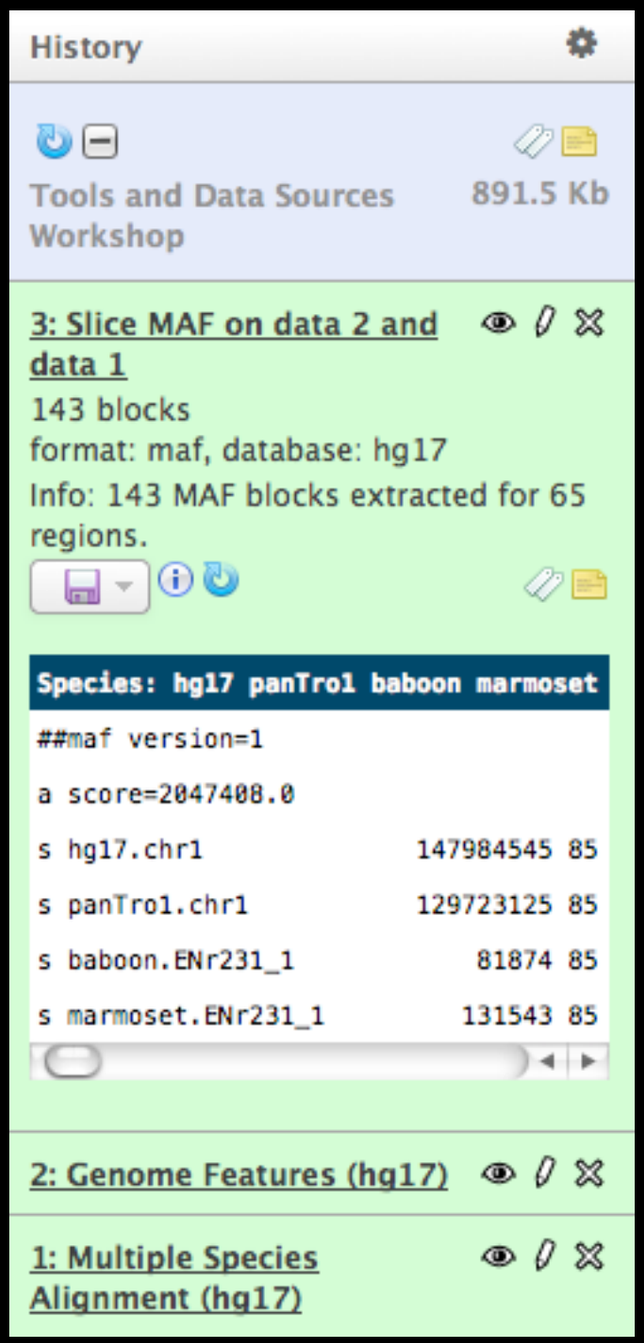
The place-holders (in orange) substitute in the values from the parameters (<param>) having a matching name

The Outputs

```
  <command interpreter="python">
maf_slice.py --dbkey=hg17 --mafFile=${maf_input} --
interval_file=${interval_input} --output_file=${maf_output}
   </command>
```

```
<outputs>
  <data format="maf" name="maf_output"/>
</outputs>
```

Successful Run of a Basic Tool



Overview

- Introduction to Tools
- Adding a basic tool exercise
- About Data Source tools
- Data Source example
- More complex tools and exercises
- The ToolFactory

Data Source Tools

- Special tools
- Get data from external sources into Galaxy
  - Synchronous
    - data ready immediately
  - Asynchronous
    - data needs to be generated by external source first

http://wiki.g2.bx.psu.edu/Admin/Internals/Data Sources

Blankenberg D, Coraor N, Von Kuster G, Taylor J, Nekrutenko A and The Galaxy Team. Integrating diverse databases into an unified analysis framework: a Galaxy approach. Database (Oxford). 2011 Apr 29.

Synchronous Data Source 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

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

Setting Attributes of Newly obtained Datasets

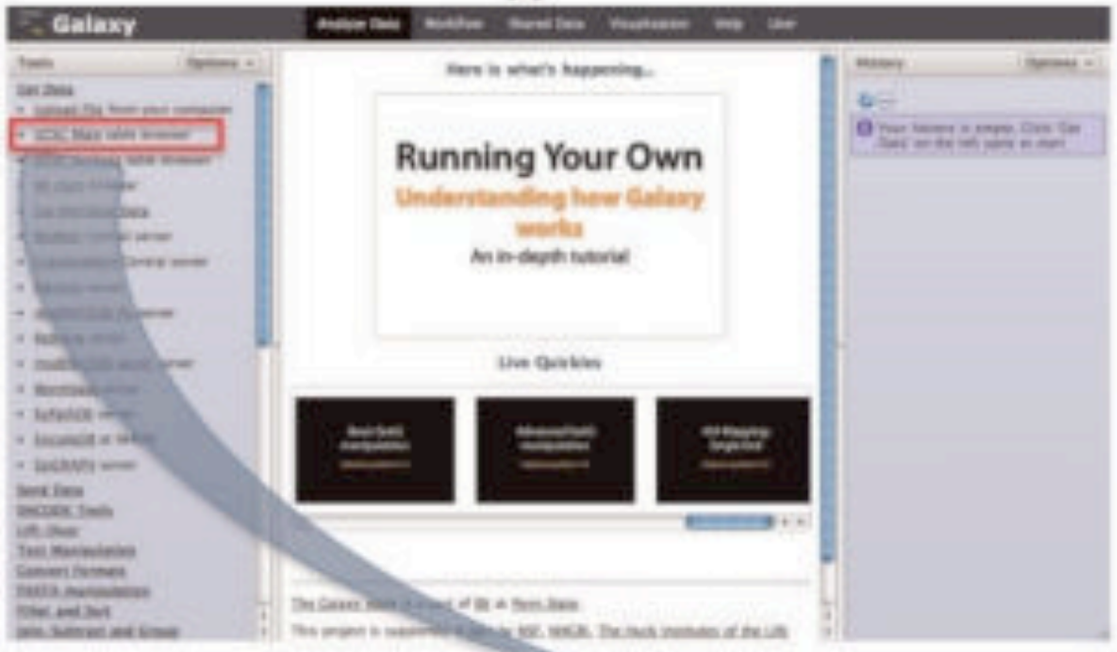
Parameter name	Usage
Name	The external resource can provide a descriptive name for the retrieved data set. If not provided, a default name based upon the name provided in the XML tool configuration is used.
Info	A free-form text string that a resource can use to provide additional information about the data set.
data_type	The type of data returned to Galaxy. Examples include bed, sam, gff and maf.
Dbkey	If the data belongs to a single reference genome, this string is used to store this information. Examples include hg18, mm9 and canFam2.
URL	The user's history will be populated with a new data set containing the results returned by submitting all provided parameters to this URL.

Overview

- Introduction to Tools
- Adding a basic tool exercise
- About Data Source tools
- Data Source example
- More complex tools and exercises
- The ToolFactory

UCSC Table Browser Example

A



B

Galaxy generates a link from the UCSC Table Browser tool configuration.

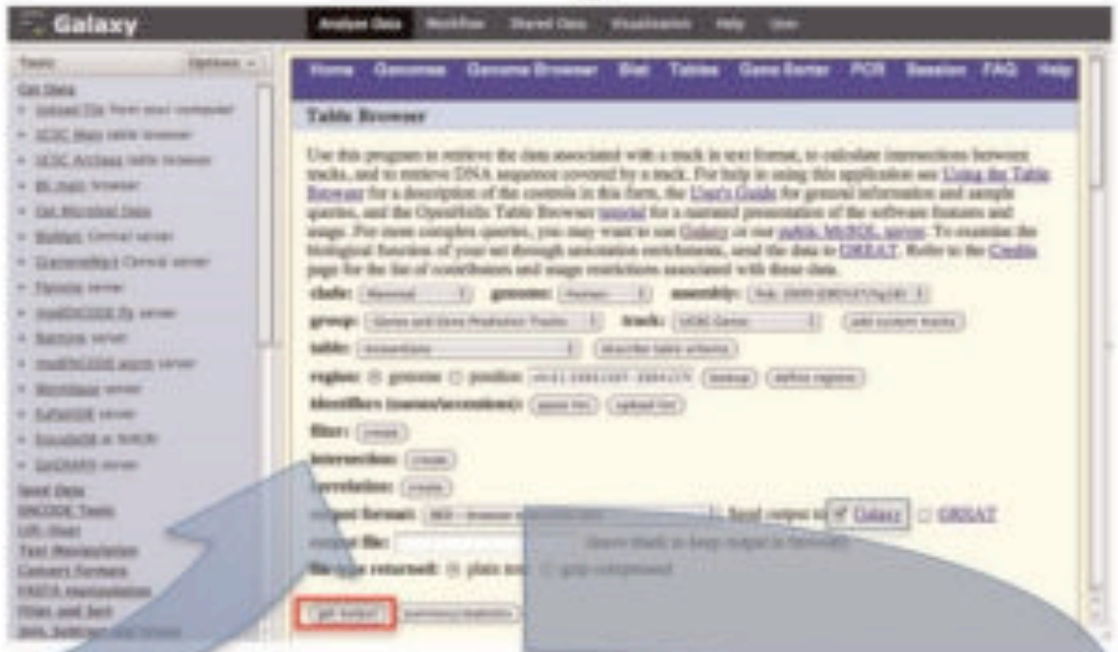
1) http://genome.ucsc.edu/cgi-bin/hgTables

2) ?GALAXY_URL=http%3A//main.g2.bx.psu.edu/tool_runner
3) &tool_id=ucsc_table_direct1

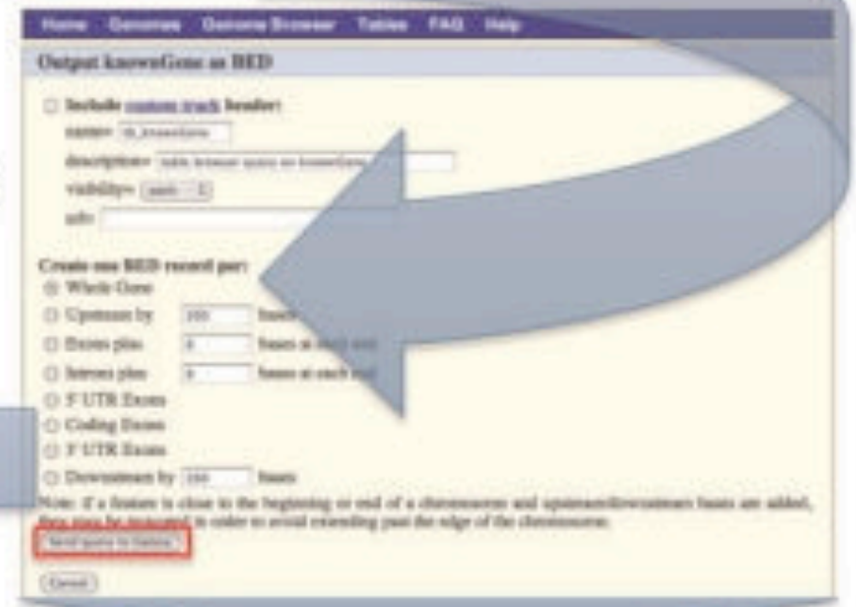
4) &hgta_compressType=none
5) &sendToGalaxy=1
6) &hgta_outputType=bed

1) "inputs action" as specified in datasource tool configuration:
This is the URL of the external resource
2) GALAXY_URL for the originating Galaxy instance
3) Unique tool id assigned to the UCSC table browser tool
Can optionally be encoded as part of GALAXY_URL
4) Resource specific parameter to set default compress type
5) Resource specific parameter to set Galaxy as default target
6) Resource specific parameter to set default return data type

C



D



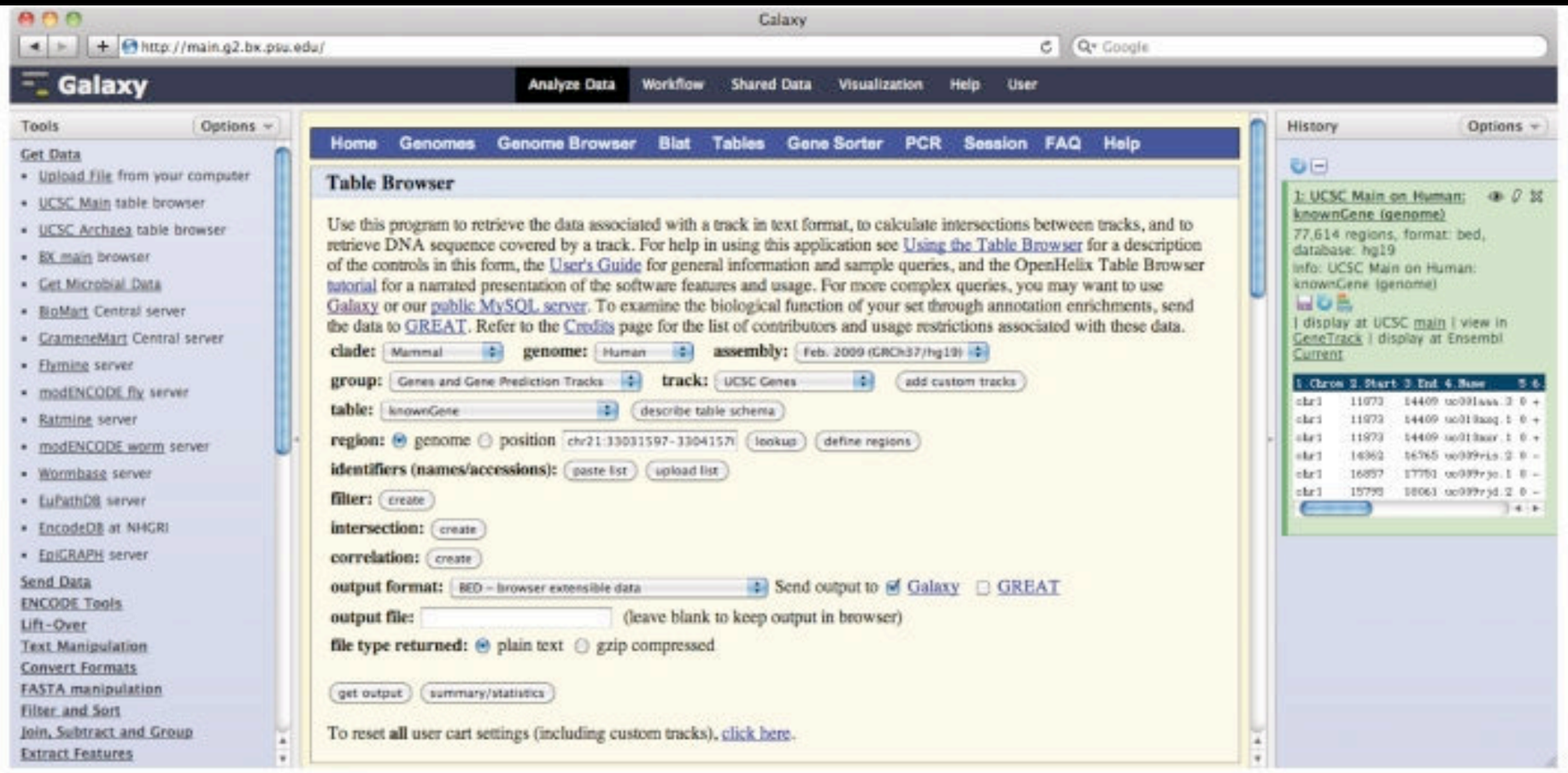
E

```
<FORM ACTION="http://main.g2.bx.psu.edu/tool_runner" METHOD=GET>

<INPUT TYPE=HIDDEN NAME="tool_id" VALUE="ucsc_table_direct1">
<INPUT TYPE=HIDDEN NAME="URL" VALUE="http://genome.ucsc.edu/cgi-bin/hgTables">

<INPUT TYPE=HIDDEN NAME="hgsid" VALUE="169304718">

<INPUT TYPE=HIDDEN NAME="db" VALUE="hg19">
<INPUT TYPE=HIDDEN NAME="org" VALUE="Human">
<INPUT TYPE=HIDDEN NAME="hgta_table" VALUE="knownGene">
<INPUT TYPE=HIDDEN NAME="hgta_track" VALUE="knownGene">
<INPUT TYPE=HIDDEN NAME="hgta_regionType" VALUE="range">
<INPUT TYPE=HIDDEN NAME="hgta_outputType" VALUE="bed">
<INPUT TYPE=HIDDEN NAME="position" VALUE="chr21:33031597-33041570">
<INPUT TYPE=HIDDEN NAME="hgsid" VALUE="171014471">

<INPUT TYPE=HIDDEN NAME="hgta_doSetBed" VALUE="get BED">
<INPUT TYPE=SUBMIT NAME="hgta_doGalaxyQuery" VALUE="Send query to Galaxy" ></FORM>
```

F

UCSC Table Browser Example

A



B

```
<tool name="UCSC Main" id="ucsc_table_direct1" 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">
    <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="dbkey" remote_name="db" missing="?"/>
    <request_param galaxy_name="data_type" remote_name="hgta_outputType" missing="tabular">
      <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 galaxy_value="gtf" remote_value="gff"/>
      </value_translation>
    </request_param>
  </request_param_translation>
  <outputs>
    <data name="output" format="tabular"/>
  </outputs>
  <options sanitize="False" refresh="True"/>
</tool>
```

A name, description and unique tool id are assigned

The default data_source.py script is used to retrieve the data from UCSC

The URL where Galaxy will forward the user when this tool is accessed

Default parameters provided by Galaxy to the data resource, including the automatic selection of the "Send results to Galaxy" option

A set of parameters provided by the Table Browser are translated into names and values that are Known to Galaxy

A single output dataset is defined

C

```
<toolbox>
  <section name="Get Data" id="getext">
    <tool file="data_source/upload.xml"/>
    <tool file="data_source/ucsc_tablebrowser.xml"/>
    <tool file="data_source/ucsc_tablebrowser_test.xml"/>
    <tool file="data_source/ucsc_tablebrowser_archaea.xml"/>
    <tool file="data_source/bx_browser.xml"/>
    <tool file="data_source/microbial_import.xml"/>
    <tool file="data_source/biomart.xml"/>
    <tool file="data_source/biomart_test.xml"/>
    <tool file="data_source/cbi_rice_mart.xml"/>
    <tool file="data_source/gramene_mart.xml"/>
    <tool file="data_source/fly_modencode.xml"/>
    <tool file="data_source/flymine.xml"/>
    <tool file="data_source/flymine_test.xml"/>
    <tool file="data_source/modmine.xml"/>
    <tool file="data_source/ratmine.xml"/>
    <tool file="data_source/worm_modencode.xml"/>
    <tool file="data_source/wormbase.xml"/>
    <tool file="data_source/wormbase_test.xml"/>
    <tool file="data_source/eupathdb.xml"/>
    <tool file="data_source/encode_db.xml"/>
    <tool file="data_source/epigraph_import.xml"/>
    <tool file="data_source/epigraph_import_test.xml"/>
    <tool file="data_source/hbvar.xml"/>
    <tool file="validation/fix_errors.xml"/>
  </section>
  <section name="Send Data" id="send">
    <tool file="data_destination/epigraph.xml"/>
```

Some Examples

- Basic*
  - tools/data_source/ebi_sra.xml
- Complex*
  - tools/data_source/ucsc_tablebrowser.xml

*from Galaxy's perspective

UCSC Table Browser

tools/data_source/ucsc_tablebrowser.xml

Overview

- Introduction to Tools
- Adding a basic tool exercise
- About Data Source tools
- Data Source example
- More complex tools and exercises
- The ToolFactory

Advanced Tool Topics

- Datasets and Datatypes
- Metadata
- Dynamic Select lists

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

http://wiki.g2.bx.psu.edu/Admin/Datatypes/Adding Datatypes

Datatype 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

Back to the example

- Allow all types of interval files to be used, not just BED
- Allow automatic setting of dbkey based upon provided interval file
- Allow the tool to access precomputed MAF indexes
- Allow the tool to access a local cache of built-in alignment sets
- Allow user to select species to include in output MAF

Generic Interval Format

- 0-based, half-open genomic intervals (just like BED)
- Columns do not have to be arranged in a particular order
- Store this information in the metadata for the dataset

Interval Metadata

```python
class Interval( Tabular ):
    """Tab delimited data containing interval information"""
    file_ext = "interval"
    line_class = "region"

    """Add metadata elements"""
    MetadataElement( name="chromCol", default=1, desc="Chrom column", param=metadata.ColumnParameter )
    MetadataElement( name="startCol", default=2, desc="Start column", param=metadata.ColumnParameter )
    MetadataElement( name="endCol", default=3, desc="End column", param=metadata.ColumnParameter )
    MetadataElement( name="strandCol", desc="Strand column (click box & select)", param=metadata.ColumnParameter, optional=True, no_value=0 )
    MetadataElement( name="nameCol", desc="Name/Identifier column (click box & select)", param=metadata.ColumnParameter, optional=True, no_valu
    MetadataElement( name="columns", default=3, desc="Number of columns", readonly=True, visible=False )
```

lib/galaxy/datatypes/interval.py

```xml
<datatype extension="interval" type="galaxy.datatypes.interval:Interval" display_in_upload="true">
  <converter file="interval_to_bed_converter.xml" target_datatype="bed"/>
  <converter file="interval_to_bedstrict_converter.xml" target_datatype="bedstrict"/>
  <converter file="interval_to_bed6_converter.xml" target_datatype="bed6"/>
  <converter file="interval_to_bed12_converter.xml" target_datatype="bed12"/>
  <converter file="interval_to_bgzip_converter.xml" target_datatype="bgzip"/>
  <converter file="interval_to_tabix_converter.xml" target_datatype="tabix" depends_on="bgzip"/>
  <converter file="interval_to_summary_tree_converter.xml" target_datatype="summary_tree"/>
  <!-- <display file="ucsc/interval_as_bed.xml" inherit="True" /> -->
  <display file="genetrack.xml" inherit="True"/>
  <display file="ensembl/ensembl_interval_as_bed.xml" inherit="True"/>
  <display file="gbrowse/gbrowse_interval_as_bed.xml" inherit="True"/>
  <display file="rviewer/bed.xml" inherit="True"/>
</datatype>
```

datatypes_conf.xml

MAF Metadata

```python
class Maf( Alignment ):
    """Class describing a Maf alignment"""
    file_ext = "maf"

    #Readonly and optional, users can't unset it, but if it is not set, we are generally ok; if required use a metadata validator in the tool de
    MetadataElement( name="blocks", default=0, desc="Number of blocks", readonly=True, optional=True, visible=False, no_value=0 )
    MetadataElement( name="species_chromosomes", desc="Species Chromosomes", param=metadata.FileParameter, readonly=True, no_value=None, visible=
    MetadataElement( name="maf_index", desc="MAF Index File", param=metadata.FileParameter, readonly=True, no_value=None, visible=False, optional

    def init_meta( self, dataset, copy_from=None ):
        Alignment.init_meta( self, dataset, copy_from=copy_from )
    def set_meta( self, dataset, overwrite = True, **kwd ):
        """
        Parses and sets species, chromosomes, index from MAF file.
        """
        #these metadata values are not accessable by users, always overwrite
        indexes, species, species_chromosomes, blocks = COPIED_build_maf_index_species_chromosomes( dataset.file_name )
        if indexes is None:
            return #this is not a MAF file
        dataset.metadata.species = species
        dataset.metadata.blocks = blocks

        #write species chromosomes to a file
        chrom_file = dataset.metadata.species_chromosomes
        if not chrom_file:
            chrom_file = dataset.metadata.spec['species_chromosomes'].param.new_file( dataset = dataset )
        chrom_out = open( chrom_file.file_name, 'wb' )
        for spec, chroms in species_chromosomes.items():
            chrom_out.write( "%s\t%s\n" % ( spec, "\t".join( chroms ) ) )
        chrom_out.close()
        dataset.metadata.species_chromosomes = chrom_file

        index_file = dataset.metadata.maf_index
        if not index_file:
            index_file = dataset.metadata.spec['maf_index'].param.new_file( dataset = dataset )
        indexes.write( open( index_file.file_name, 'wb' ) )
        dataset.metadata.maf_index = index_file
```

lib/galaxy/datatypes/sequences.py

Enhance the tool with Metadata

```
 <command interpreter="python">
maf_slice.py --dbkey=${interval_input.dbkey} --mafFile=${maf_input} --
interval_file=${interval_input}  --output_file=${maf_output}
    --chromCol=${interval_input.metadata.chromCol} --startCol=$
{interval_input.metadata.startCol} --endCol=$
{interval_input.metadata.endCol} --strandCol=$
{interval_input.metadata.strandCol}
    --mafIndex=${maf_input.metadata.maf_index}
  </command>
  <inputs>
   <param format="maf" name="maf_input" label="Choose alignments"
type="data"/>
   <param format="interval" name="interval_input" type="data"
label="Choose intervals"/>
  </inputs>
```

One tool, two different sources for data

- Use a conditional block to allow the user to choose between MAF from the History or a built-in

```
  <inputs>
   <param format="interval" name="interval_input" type="data"
label="Choose intervals"/>
   <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 format="maf" name="maf_input" label="Choose
alignments" type="data"/>
    </when>
    <when value="cached">
     <!-- need some way to access the external data -->
    </when>
   </conditional>
  </inputs>
```

Don’t forget to fix the command-line

```
  <command interpreter="python">
maf_slice.py --dbkey=${interval_input.dbkey} --mafFile=$
{maf_source_type.maf_input} --interval_file=${interval_input}  --
output_file=${maf_output}
    --chromCol=${interval_input.metadata.chromCol} --startCol=$
{interval_input.metadata.startCol} --endCol=$
{interval_input.metadata.endCol} --strandCol=$
{interval_input.metadata.strandCol}
    --mafIndex=${maf_source_type.maf_input.metadata.maf_index}
  </command>
```

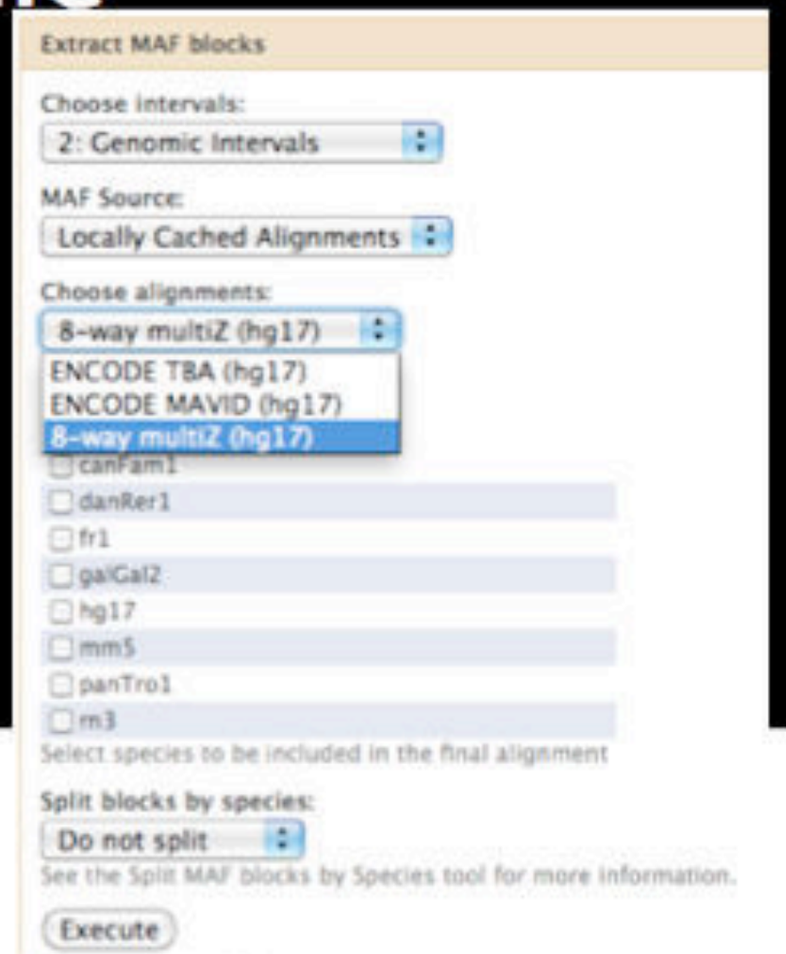
Accessing built-in/cached alignments

- Have an external Tab-delimited file (*location file, *.loc*) that contains information about the available alignments
  - unique ID
  - genomes indexed in the alignment
  - genomes contained in the MAF
  - paths to the alignment files

tool-data/maf_index.loc

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




```
<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 inter
  </options>
</param>
```

```
$ cat maf_index.loc
#Display_name UID       indexed_for:build1,build2,build3       exists_in_maf:build1,build2,build3      Comma_Separated_Full_Paths_To_Files
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,hed
ENCODE TBA (hg16)       ENCODE_TBA_hg16 armadillo,baboon,galGal2,panTro1,colobus_monkey,cow,canFam1,dusky_titi,elephant=elephant,fr1,galago,he
8-way multiZ (hg17)     8_WAY_MULTIZ_hg17       canFam1,danRer1,fr1,galGal2,hg17,mm5,panTro1,rn3        canFam1,danRer1,fr1,galGal2,hg17,mm5,p
17-way multiZ (hg18)    17_WAY_MULTIZ_hg18      hg18,panTro1,bosTau2,rheMac2,mm8,rn4,canFam2,echTel1,loxAfr1,oryCun1,danRer3,monDom4,dasNov1,g
3-way multiZ (hg18,panTro2,rheMac2)     3_WAY_MULTIZ_hg18       hg18,panTro2,rheMac2    hg18,panTro2,rheMac2    /depot/data2/galaxy/hg18/align
```

tool_data_table_conf.xml

- .loc files can be accessed directly by path
  - Move .loc file, need to modify maf_slice.xml
- Tool data tables provides an abstraction layer between the filesytem location of the data and the name of the data
- Move .loc file, no need to modify the tool

```
<!-- Locations of MAF files that have been indexed with bx-python -->
<table name="indexed_maf_files">
    <columns>name, value, dbkey, species</columns>
    <file path="tool-data/maf_index.loc" />
</table>
```

Setting Parameters

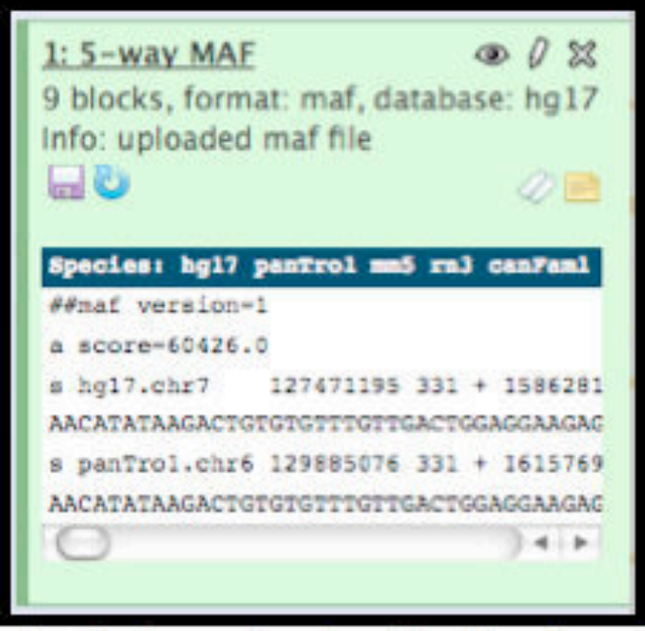
```
    <when value="cached">
      <param name="mafType" type="select" label="Choose
alignments">
        <options from_data_table="indexed_maf_files">
          <filter type="data_meta" ref="interval_input" key="dbkey"
column="dbkey" multiple="True" separator=","/>
          <validator type="no_options" message="No alignments are
available for the build associated with the selected interval file"/>
        </options>
      </param>
    </when>
```

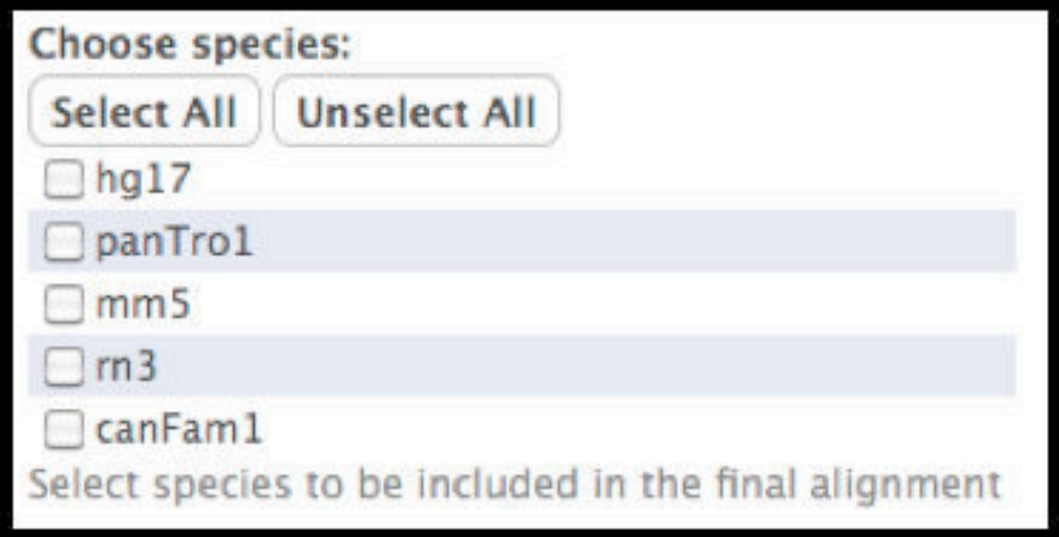
Enhancing command-line

```
 <command interpreter="python">
   maf_slice.py --dbkey=${interval_input.dbkey} --interval_file=$
{interval_input}  --output_file=${maf_output}
   --chromCol=${interval_input.metadata.chromCol} --startCol=$
{interval_input.metadata.startCol} --endCol=$
{interval_input.metadata.endCol} --strandCol=$
{interval_input.metadata.strandCol}
   #if $maf_source_type.maf_source == "user":
      --mafFile=${maf_source_type.maf_input}
      --mafIndex=$
{maf_source_type.maf_input.metadata.maf_index}
   #else:
      --mafType=$maf_source_type.mafType
      --mafIndexFile=${GALAXY_DATA_INDEX_DIR}/maf_index.loc
   #end if
 </command>
```

Selecting Species in output

- Use species metadata value from MAF file to dynamically generate a list of checkboxes






```
<param name="species" type="select" display="checkboxes" multiple="true" label="Choose species" help="Select sp
  <options>
    <filter type="data_meta" ref="mafFile" key="species" />
  </options>
</param>
```

```
    <when value="user">
      <param format="maf" name="maf_input" label="Choose
alignments" type="data"/>
      <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="maf_input" key="species" />
        </options>
      </param>
    </when>
```

Add to command-line

```
<command interpreter="python">
    maf_slice.py --dbkey=${interval_input.dbkey} --interval_file=$
{interval_input}  --output_file=${maf_output}
    --chromCol=${interval_input.metadata.chromCol} --startCol=$
{interval_input.metadata.startCol} --endCol=$
{interval_input.metadata.endCol} --strandCol=$
{interval_input.metadata.strandCol}
    #if $maf_source_type.maf_source == "user":
        --mafFile=${maf_source_type.maf_input}
        --mafIndex=${maf_source_type.maf_input.metadata.maf_index}
        --species=${maf_source_type.species}
    #else:
        --mafType=$maf_source_type.mafType
        --mafIndexFile=${GALAXY_DATA_INDEX_DIR}/maf_index.loc
    #end if
  </command>
```

Species from .loc

```
    <when value="cached">
      <param name="mafType" type="select" label="Choose alignments">
        <options from_data_table="indexed_maf_files">
          <filter type="data_meta" ref="interval_input" key="dbkey" column="dbkey" 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_data_table="indexed_maf_files">
          <column name="uid" index="1"/>
          <column name="value" index="3"/>
          <column name="name" index="3"/>
          <filter type="param_value" ref="mafType" column="uid"/>
          <filter type="multiple_splitter" column="name" separator=","/>
        </options>
      </param>
    </when>
```

Add to command-line

```
  <command interpreter="python">
    maf_slice.py --dbkey=${interval_input.dbkey} --interval_file=$
{interval_input}  --output_file=${maf_output}
    --chromCol=${interval_input.metadata.chromCol} --startCol=$
{interval_input.metadata.startCol} --endCol=$
{interval_input.metadata.endCol} --strandCol=$
{interval_input.metadata.strandCol}
    --species=${maf_source_type.species}
    #if $maf_source_type.maf_source == "user":
      --mafFile=${maf_source_type.maf_input}
      --mafIndex=${maf_source_type.maf_input.metadata.maf_index}
    #else:
      --mafType=$maf_source_type.mafType
      --mafIndexFile=${GALAXY_DATA_INDEX_DIR}/maf_index.loc
    #end if
  </command>
```

Slice MAF (version 1.0.5)

Choose intervals:

2: Genome Features (hg17)

MAF Source:

Alignments in Your History

Choose alignments:

1: Multiple Species ..ment (hg17)

Choose species:

Select All | Unselect All

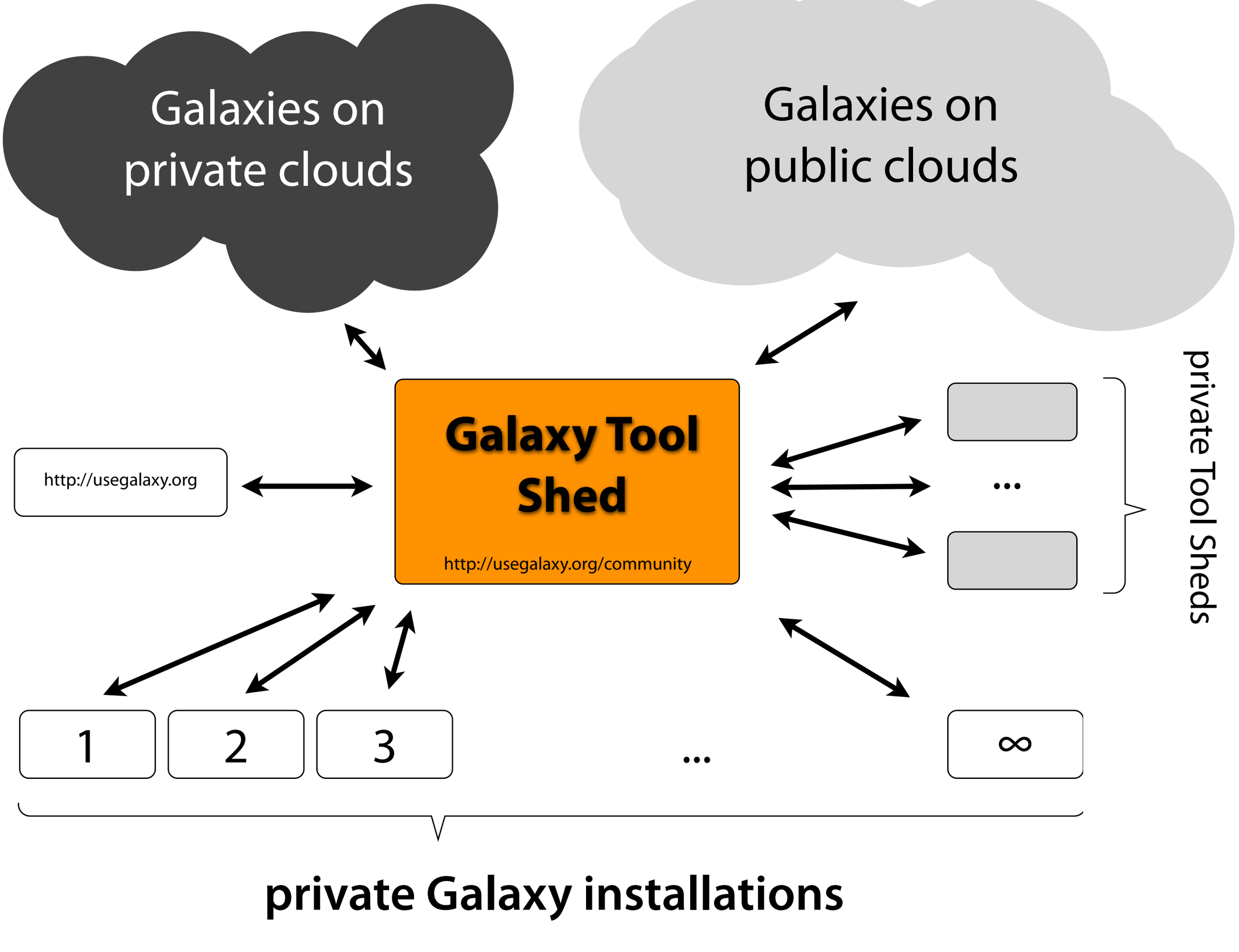
- [ ] hg17
- [ ] panTro1
- [ ] baboon
- [ ] marmoset
- [ ] galago
- [ ] rn3
- [ ] mm6
- [ ] rabbit
- [ ] cow
- [ ] canFam1
- [ ] rfbat
- [ ] shrew
- [ ] armadillo
- [ ] tenrec
- [ ] monDom1
- [ ] tetNig1
- [ ] fr1
- [ ] rheMac1
- [ ] galGal2
- [ ] xenTro1
- [ ] danRer2
- [ ] elephant
- [ ] platypus
- [ ] hedgehog
- [ ] colobus_monkey
- [ ] dusky_titi
- [ ] owl_monkey
- [ ] mouse_lemur

Select species to be included in the final alignment

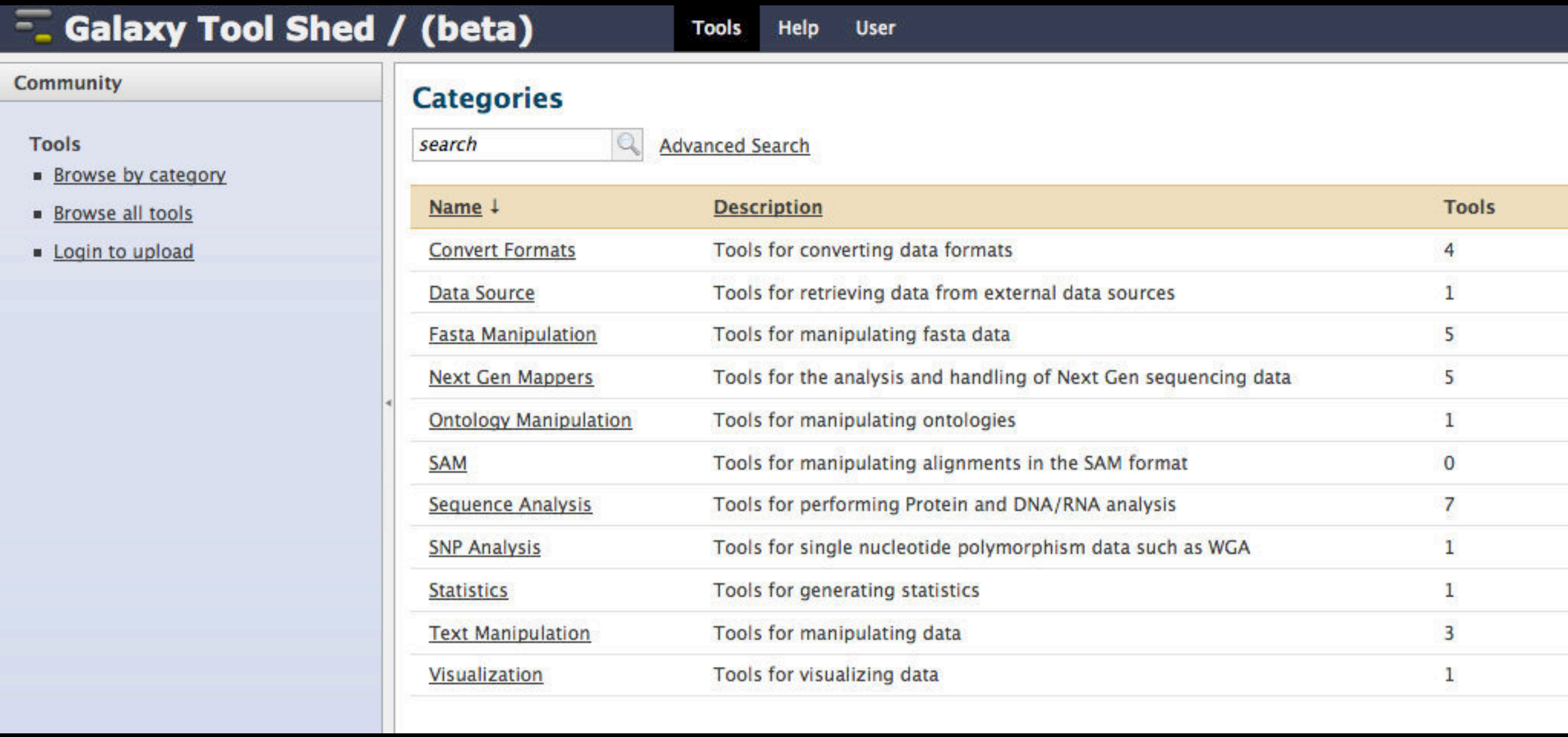
Execute

I created a tool...now what?

- **Share it!**
- Galaxy Tool Shed
- Workshop by Greg Von Kuster
  - 2PM, Room C
- Upload and Download contributed Tools
- Rate and provide feedback


Galaxies on private clouds
Galaxies on public clouds
Galaxy Tool Shed
http://usegalaxy.org/community
http://usegalaxy.org
...
private Tool Sheds
1
2
3
...
∞
private Galaxy installations

Get and Contribute Tools



http://usegalaxy.org/community

Additional Advanced Tool Topics

- Tool Output actions
  - changing formats, setting metadata based upon user inputs
- Filtering outputs
- Parameter Validation

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

Tool Output Actions

- Allow run-time attributes (parameter settings, input metadata, datatype) to change attributes of a created output file
- Examples:
  - Cut tool
    - Default output is tabular
    - What if input was BED and all required interval columns are kept?
  - NGS mappers (e.g. bowtie)
    - Set dbkey of output when using pre-built indices

Cut Tool

```
<outputs>
  <data format="tabular" name="out_file1" >
    <actions>
      <conditional name="delimiter">
        <when value="T">
          <conditional name="input">
            <when datatype_isinstance="interval">
              <action type="format" default="tabular">
                <option type="from_param" name="columnList" column="0" offset="0"> <!-- chromCol is 1-->

                  <filter type="insert_column" column="0" value="interval"/>

                  <filter type="insert_column" ref="columnList" /> <!-- startCol -->

                  <filter type="insert_column" ref="columnList" /> <!-- endCol -->

                  <filter type="multiple_splitter" column="1" separator=","/>
                  <filter type="column_strip" column="1"/> <!-- get rid of all external whitespace -->
                  <filter type="string_function" column="1" name="lower" />
                  <filter type="param_value" column="1" value="^c\d{1,}$" compare="re_search" keep="True"/>
                  <filter type="column_strip" column="1" strip="c"/> <!-- get rid of c's  -->
                  <filter type="boolean" column="1" cast="int" />

                  <filter type="multiple_splitter" column="2" separator=","/>
                  <filter type="column_strip" column="2"/> <!-- get rid of all external whitespace -->
                  <filter type="string_function" column="2" name="lower" />
                  <filter type="param_value" column="2" value="^c\d{1,}$" compare="re_search" keep="True"/>
                  <filter type="column_strip" column="2" strip="c"/> <!-- get rid of c's  -->
                  <filter type="boolean" column="2" cast="int" />

                  <filter type="multiple_splitter" column="3" separator=","/>
                  <filter type="column_strip" column="3"/> <!-- get rid of all external whitespace -->
                  <filter type="string_function" column="3" name="lower" />
                  <filter type="param_value" column="3" value="^c\d{1,}$" compare="re_search" keep="True"/>
                  <filter type="column_strip" column="3" strip="c"/> <!-- get rid of c's  -->
                  <filter type="boolean" column="3" cast="int" />

                  <filter type="metadata_value" ref="input" name="chromCol" column="1" />
                  <filter type="metadata_value" ref="input" name="startCol" column="2" />
                  <filter type="metadata_value" ref="input" name="endCol" column="3" />

                </option>
              </action>
```

```
<conditional name="out_file1">
  <when datatype_isinstance="interval">
    <action type="metadata" name="chromCol">
      <option type="from_param" name="columnList" column="0" offset="0"> <!-- chromCol is 0-->
        <filter type="multiple_splitter" column="0" separator=","/>
        <filter type="column_strip" column="0"/> <!-- get rid of all external whitespace -->
        <filter type="string_function" column="0" name="lower" />
        <filter type="param_value" column="0" value="^c\d{1,}$" compare="re_search" keep="True"/>
        <filter type="column_strip" column="0" strip="c"/> <!-- get rid of c's  -->
        <filter type="insert_column" value="1" iterate="True" column="0"/>
        <filter type="boolean" column="1" cast="int" />
        <filter type="metadata_value" ref="input" name="chromCol" column="1" />
      </option>
    </action>

    <action type="metadata" name="startCol">
      <option type="from_param" name="columnList" column="0" offset="0"> <!-- startCol is 0-->
        <filter type="multiple_splitter" column="0" separator=","/>
        <filter type="column_strip" column="0"/> <!-- get rid of all external whitespace -->
        <filter type="string_function" column="0" name="lower" />
        <filter type="param_value" column="0" value="^c\d{1,}$" compare="re_search" keep="True"/>
        <filter type="column_strip" column="0" strip="c"/> <!-- get rid of c's  -->
        <filter type="insert_column" value="1" iterate="True" column="0"/>
        <filter type="boolean" column="1" cast="int" />
        <filter type="metadata_value" ref="input" name="startCol" column="1" />
      </option>
    </action>

    <action type="metadata" name="endCol">
      <option type="from_param" name="columnList" column="0" offset="0"> <!-- endCol is 0-->
        <filter type="multiple_splitter" column="0" separator=","/>
        <filter type="column_strip" column="0"/> <!-- get rid of all external whitespace -->
        <filter type="string_function" column="0" name="lower" />
        <filter type="param_value" column="0" value="^c\d{1,}$" compare="re_search" keep="True"/>
        <filter type="column_strip" column="0" strip="c"/> <!-- get rid of c's  -->
        <filter type="insert_column" value="1" iterate="True" column="0"/>
        <filter type="boolean" column="1" cast="int" />
        <filter type="metadata_value" ref="input" name="endCol" column="1" />
      </option>
    </action>

    <action type="metadata" name="nameCol" default="0">
      <option type="from_param" name="columnList" column="0" offset="0"> <!-- nameCol is 0-->
        <filter type="multiple_splitter" column="0" separator=","/>
        <filter type="column_strip" column="0"/> <!-- get rid of all external whitespace -->
        <filter type="string_function" column="0" name="lower" />
        <filter type="param_value" column="0" value="^c\d{1,}$" compare="re_search" keep="True"/>
        <filter type="column_strip" column="0" strip="c"/> <!-- get rid of c's  -->
        <filter type="insert_column" value="1" iterate="True" column="0"/>
        <filter type="boolean" column="1" cast="int" />
        <filter type="metadata_value" ref="input" name="nameCol" column="1" />
      </option>
    </action>
```

Cut Tool

```
<action type="metadata" name="strandCol" default="0">
  <option type="from_param" name="columnList" column="0" offset="0"> <!-- strandCol is 0-->
    <filter type="multiple_splitter" column="0" separator=","/>
    <filter type="column_strip" column="0"/> <!-- get rid of all external whitespace -->
    <filter type="string_function" column="0" name="lower" />
    <filter type="param_value" column="0" value="^c\d{1,}$" compare="re_search" keep="True"/>
    <filter type="column_strip" column="0" strip="c"/> <!-- get rid of c's  -->
    <filter type="insert_column" value="1" iterate="True" column="0"/>
    <filter type="boolean" column="1" cast="int" />
    <filter type="metadata_value" ref="input" name="strandCol" column="1" />
  </option>
 </action>
</when>
</conditional>

</when>
</conditional>
</when>
</conditional>
</actions>
</data>
```

tools/filters/cutWrapper.xml

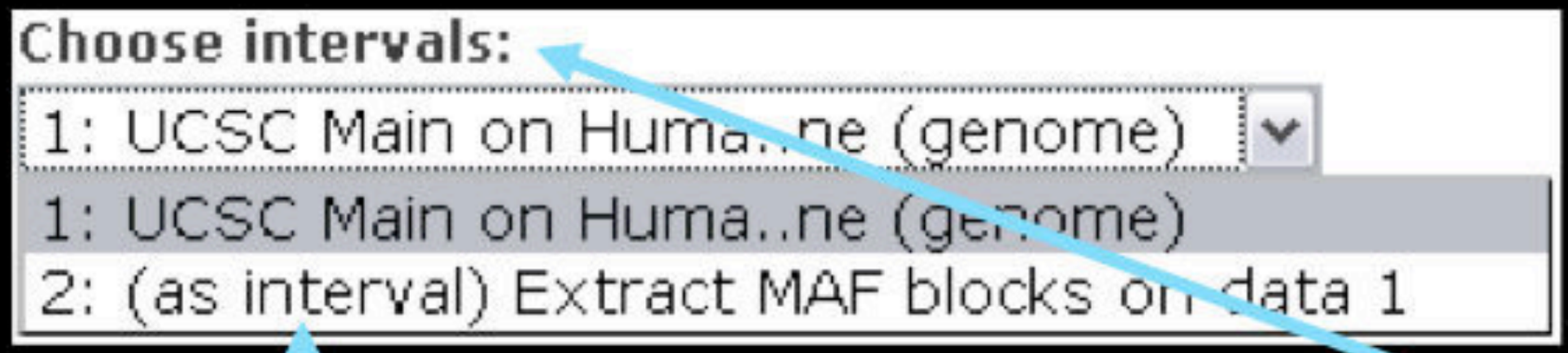
Filtering Outputs

- Parameter settings can be used to change the outputs that are created

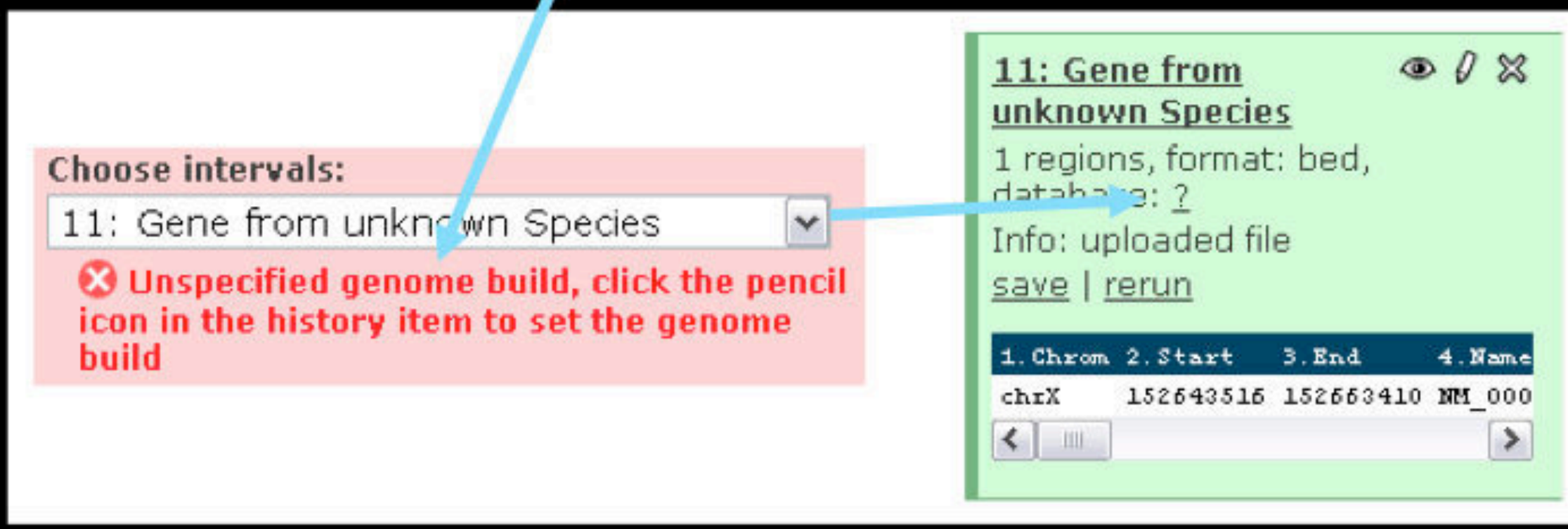
```
  <outputs>
    <data name="output_bed_file" format="bed" label="${tool.name} on ${on_string} (peaks: bed)"/>
    <data name="output_xls_to_interval_peaks_file" format="interval" label="${tool.name} on ${on_string}
(peaks: interval)">
      <filter>xls_to_interval is True</filter>
    </data>
    <data name="output_xls_to_interval_negative_peaks_file" format="interval" label="${tool.name} on $
{on_string} (negative peaks: interval)">
      <filter>xls_to_interval is True</filter>
      <filter>input_type['input_control_file1'] is not None</filter>
    </data>
    <data name="output_treatment_wig_file" format="wig" label="${tool.name} on ${on_string} (treatment:
wig)">
      <filter>wig_type['wig_type_selector']=='wig'</filter>
    </data>
    <data name="output_control_wig_file" format="wig" label="${tool.name} on ${on_string} (control: wig)">
      <filter>wig_type['wig_type_selector'] == 'wig'</filter>
      <filter>input_type['input_control_file1'] is not None</filter>
    </data>
    <data name="output_extra_files" format="html" label="${tool.name} on ${on_string} (html report)"/>
  </outputs>
```

tools/peak_calling/macs_wrapper.xml

Parameter Validation




```
<param format="interval" name="input1" type="data" label="Choose intervals">↓
  <validator type="unspecified_build" />↓
</param>↓
```



tools/maf/interval2maf.xml

Available Validators

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

Overview

- Introduction to Tools
- Adding a basic tool exercise
- About Data Source tools
- Data Source example
- More complex tools and exercises
- The ToolFactory