Galaxy

Translating workflows into Nextflow with Janis

Galaxy Platform

Online platform for data-analysis

- GUI rather than CLI
- Easy data handling
- Compute provided
- Easy to get started

Functionality is impressive

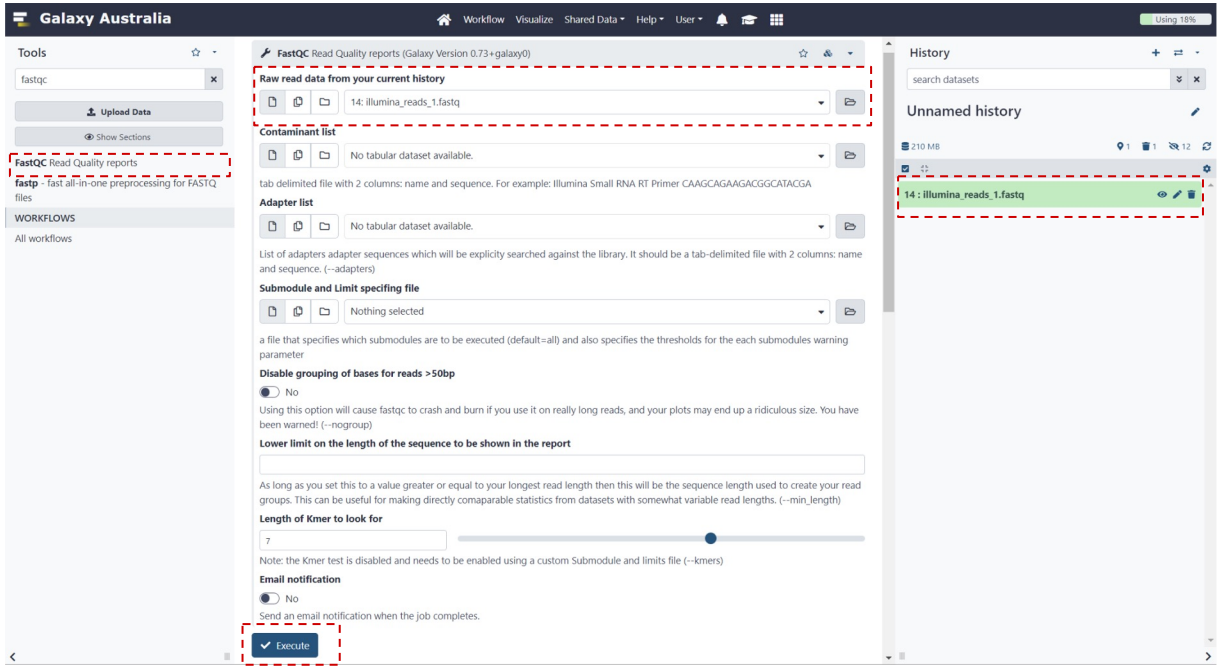
- Each server has collection of Tools
- History documents your analysis
- Can create a workflow for your analysis

Chose to support Galaxy ingest in janis-translate because..

- Many Galaxy tools & workflows publicly available
- Natural progression for users: Galaxy → CLI
- Workflow editor makes prototyping fast

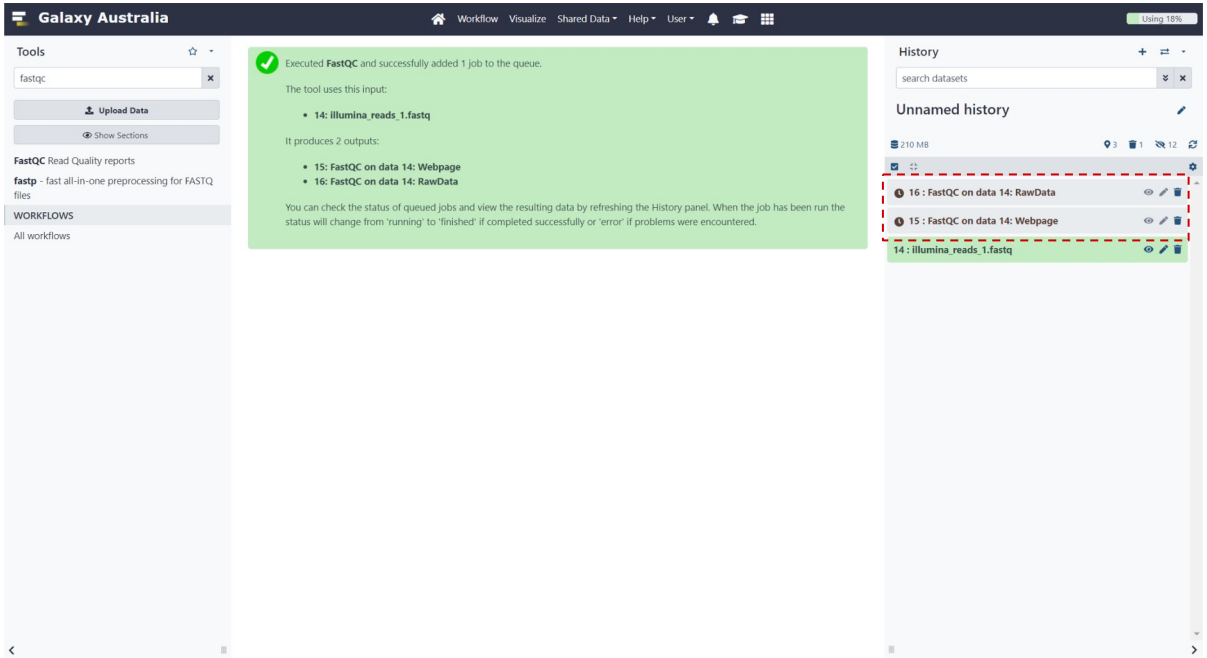
Features

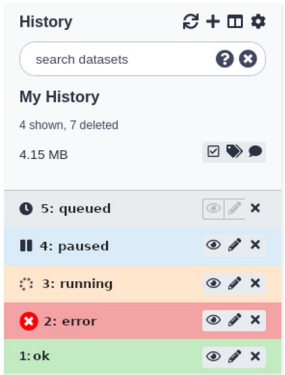
Each server has collection of Tools. Select a tool, supply inputs, execute.



Features

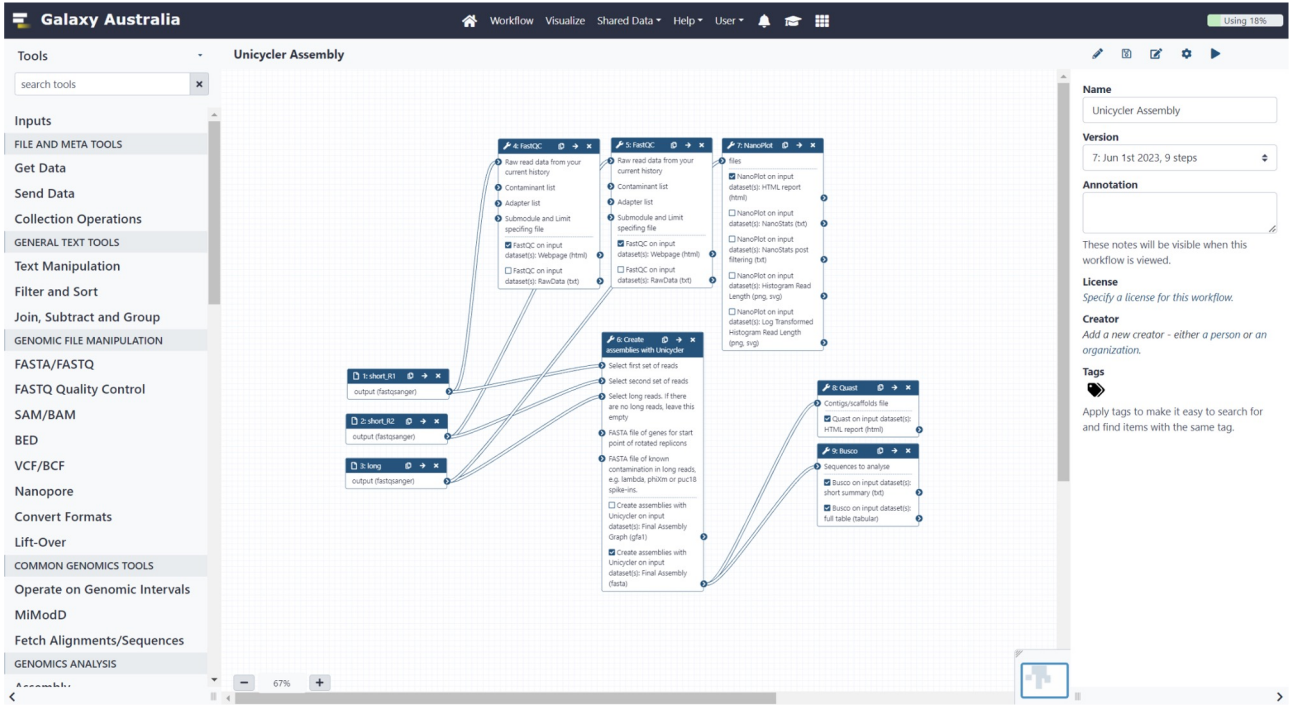
History documents your analysis - queued, running, paused, complete





Features

Create a workflow for your analysis. From analysis history, or from scratch.



Users

Biologists, Career bioinformaticians (due to great accessibility)

A number of prominent research ventures use the Galaxy platform to analyse their data:

- The Encyclopedia of DNA Elements (ENCODE)
- The Vertebrate Genomes Project (VGP)
- Assistance Publique–Hôpitaux de Paris (AP-HP)

Accessing Tools

Tools can be accessed 2 ways:

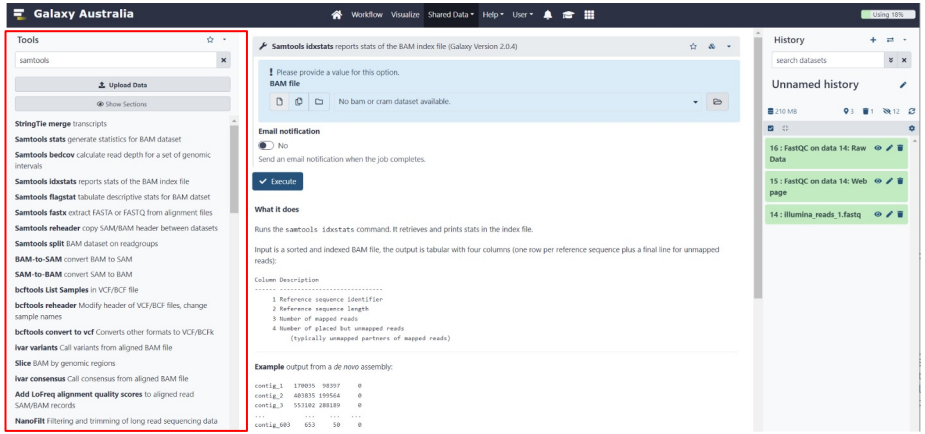
- Via a Galaxy server (https://usegalaxy.org.au/)
- Via the Galaxy Toolshed (https://toolshed.g2.bx.psu.edu/)

Accessing Tools

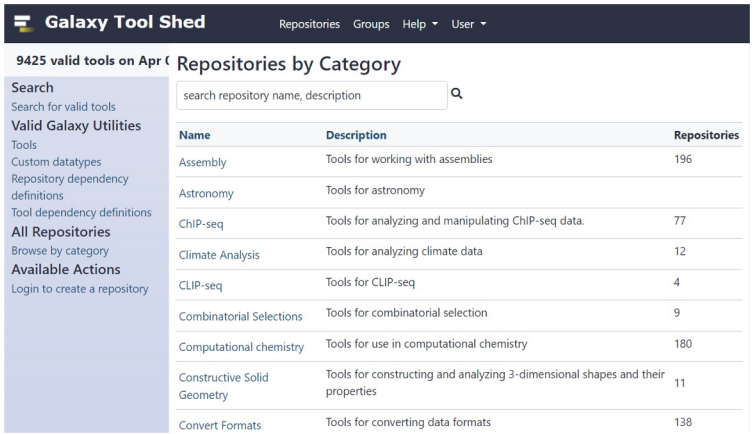
Tools can be accessed 2 ways:

- Via a Galaxy server (https://usegalaxy.org.au/)
- Via the Galaxy Toolshed (https://toolshed.g2.bx.psu.edu/)

Via Galaxy Server



Via Toolshed



Accessing Tools - Tool Wrapper Format

Galaxy Tool Wrappers are written in XML.

They have 2 roles:

- Expose a user interface (UI)
- Execute CLI commands to run software

Accessing Tools - Tool Wrapper Format

Galaxy Tool Wrappers are written in XML.

They have 2 roles:

- Expose a user interface (UI)
- Execute CLI commands to run software

```
<command detect_errors="exit_code"><![CDATA[
    #import re
    #set $sample_name = re.sub('[^\w\-_\.]', '_', $file_input.
    element_identifier)

    ln -sf '${file_input}' ${sample_name} &&

    abricate ${sample_name}
    $adv.no_header
    #if $adv.min_dna_id
        --minid=$adv.min_dna_id
    #end if
    #if $adv.min_cov
        --mincov=$adv.min_cov
    #end if
    2>&1
    --db=$adv.db
    > '$report'
]]></command>
```

abricate.xml command

```
<inputs>
    <param name="file_input" type="data" format="fasta,genbank,embl"
    label="Input file (Fasta, Genbank or EMBL file)" help="To screen for
    antibiotic resistant genes, can be a fasta file, a genbank file or an
    EMBL file." />
    <section name="adv" title="Advanced options" expanded="False">
        <param argument="--db" type="select" label="Database to use -
        default is 'resfinder'" help="Option to switch to other AMR/other
        database">
            <option value="argannot">ARG-ANNOT</option>
            <option value="card">CARD</option>
            <option value="ecoh">EcOH</option>
            <option value="ncbi">NCBI Bacterial Antimicrobial Resistance
            Reference Gene Database</option>
            <option value="resfinder" selected="true">Resfinder</option>
            <option value="plasmidfinder">PlasmidFinder</option>
            <option value="vfdb">VFDB</option>
            <option value="megares">megares</option>
            <option value="ecoli_vf">Ecoli_VF</option>
        </param>
        <param name="no_header" argument="--noheader" type="boolean"
        truevalue="--noheader" falsevalue="" label="Suppress header"
        help="Suppress output file's column headings" />
        <param name="min_dna_id" argument="--minid" type="float"
        value="80" min="0" max="100" optional="true" label="Minimum DNA
        %identity" />
        <param name="min_cov" argument="--mincov" type="float" value="80"
        min="0" max="100" optional="true" label="Minimum DNA %coverage" />
    </section>
</inputs>

<outputs>
    <data name="report" format="tabular" label="${tool.name} on $
    {on_string} report file" />
</outputs>
```

abricate.xml I/O

Accessing Tools - Tool Wrapper Format

Can get super complicated!

Translation becomes very challenging.

- Runtime values can affect the structure of the shell command
- Won't know what the command will look like till it actually runs
- Sometimes impossible to parse correctly.

Accessing Tools - Tool Wrapper Format

Can get super complicated!

Translation becomes very challenging.

- Runtime values can affect the structure of the shell command
- Won't know what the command will look like till it actually runs
- Sometimes impossible to parse correctly.

1/3rd of cutadapt.xml command

```
    <command detect_errors="exit_code"><![CDATA[
## Link in the input and output files, so Cutadapt can tell their type

#import re
#set read1 = "input_f"
#set read2 = "input_r"
#set paired = False
#set library_type = str($library.type)
#if $library_type == 'paired':
#set paired = True
#set read1 = re.sub('[^\w\-\s]', '_', str($library.input_1.element_identifier))
#set read2 = re.sub('[^\w\-\s]', '_', str($library.input_2.element_identifier))
#set input_1 = $library.input_1
#set input_2 = $library.input_2
#else if $library_type == 'paired_collection'
#set paired = True
#set input_1 = $library.input_1.forward
#set input_2 = $library.input_1.reverse
#set read1 = re.sub('[^\w\-\s]', '_', str($library.input_1.name)) + "_1"
#set read2 = re.sub('[^\w\-\s]', '_', str($library.input_1.name)) + "_2"
#else
#set input_1 = $library.input_1
#set read1 = re.sub('[^\w\-\s]', '_', str($library.input_1.element_identifier))
#end if

#if $input_1.is_of_type("fastq.gz", "fastqsanger.gz"):
    #set read1 = $read1 + ".fq.gz"
    #set out1 = "out1.gz"
#else if $input_1.is_of_type("fastq.bz2", "fastqsanger.bz2"):
    #set read1 = $read1 + ".fq.bz2"
    #set out1 = "out1.bz2"
#else if $input_1.is_of_type('fasta'):
    #set read1 = $read1 + ".fa"
    #set out1 = "out1.fa"
#else:
    #set read1 = $read1 + ".fq"
    #set out1 = "out1.fq"
#end if
ln -f -s '${input_1}' '$read1' &&

#if $paired:
    #if $input_2.is_of_type("fastq.gz", "fastqsanger.gz"):
        #set read2 = $read2 + ".fq.gz"
        #set out2 = "out2.gz"
    #else if $input_2.is_of_type("fastq.bz2", "fastqsanger.bz2"):
        #set read2 = $read2 + ".fq.bz2"
        #set out2 = "out2.bz2"
    #else if $input_2.is_of_type('fasta'):
        #set read2 = $read2 + ".fa"
        #set out2 = "out2.fa"
    #else:
        #set read2 = $read2 + ".fq"
        #set out2 = "out2.fq"
    #end if
    ln -f -s '${input_2}' '$read2' &&
#end if

## Run Cutadapt

cutadapt

## cutadapt (up to version 1.16) can't be run in multicore mode with these options
#if not any($output_options.info_file, $output_options.rest_file, $output_options.wildcard_file, $output_options.too_short_file, $output_options.too_long_file, $output_options.untrimmed_file))
    -j \${GALAXY_SLOTS:-1}
#end if

#if str( $library.type ) == "single":
    @read1_options@
    --output='$out1'
#else:
    @read1_options@
    @read2_options@
```

Handling Galaxy Tool Wrapper Requirements

Galaxy uses conda to handle tool requirements.

For best-practises pipelines, we need a single container image.

New feature added:

- `--build-galaxy-tool-images`
- Builds a new container image on the fly (during translation)
- Contains all software listed in tool <requirements>
- Image will appear as the container requirement in translations

```
<tool id="limma_voom" name="limma" version="@TOOL_VERSION@+galaxy0">

    <description>
        Perform differential expression with limma-voom or limma-trend
    </description>

    <macros>
        <token name="@TOOL_VERSION@">3.50.1</token>
    </macros>

    <requirements>
        <requirement type="package" version="@TOOL_VERSION@">bioconductor-limma</requirement>
        <requirement type="package" version="3.36.0">bioconductor-edger</requirement>
        <requirement type="package" version="1.4.36">r-statmod</requirement>
        <requirement type="package" version="1.1.1">r-scales</requirement>
        <requirement type="package" version="0.2.21">r-rjson</requirement>
        <requirement type="package" version="1.20.3">r-getopt</requirement>
        <requirement type="package" version="3.1.1">r-gplots</requirement>
        <requirement type="package" version="2.4.0">bioconductor-glimma</requirement>
    </requirements>
```

Handling Galaxy Tool Wrapper Requirements

Some notes about this feature:

- Requires docker
- Can be slow (2-30 mins)

In the interest of time, we will not use this feature today.

- Images for relevant tools (limma-voom, hisat2, featurecounts) available on a quay.io repo
- We will replace the container requirement for affected tools

When this feature is turned off...

- janis-translate picks a suitable container based on the main software requirement.

quay.io/grace_hall1/hisat2:2.2.1

```
process HISAT2 {

    container "quay.io/biocontainers/hisat2:2.2.1--h87f3376_5"
    publishDir "${params.outdir}/hisat2"

    input:
    path library_input_1
    path index_path

    output:
    path "${library_input_1.simpleName}.alignment_summary.txt", emit: out_summary_file
    path "${library_input_1.simpleName}.bam", emit: output_alignments

    script:
    """
    hisat2 \
    -U ${library_input_1} \
    -x ${index_path[0].simpleName} \
    --summary-file ${library_input_1.simpleName}.alignment_summary.txt \
    -S out.sam
```

Accessing Workflows

Workflows can be accessed 3 ways

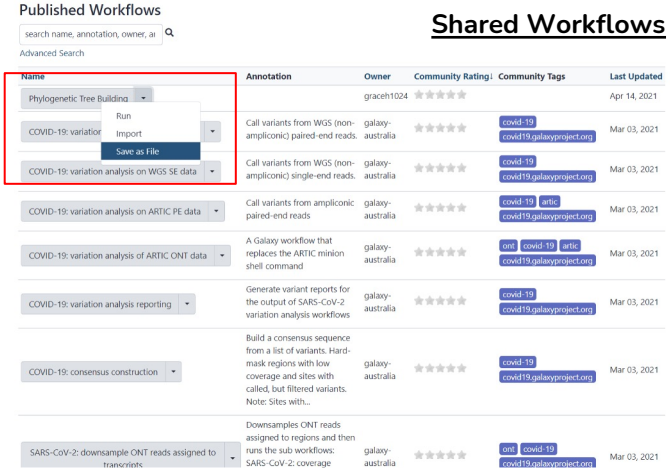
- Via Shared Workflows
- Via User Workflows
- Via Link

Accessing Workflows

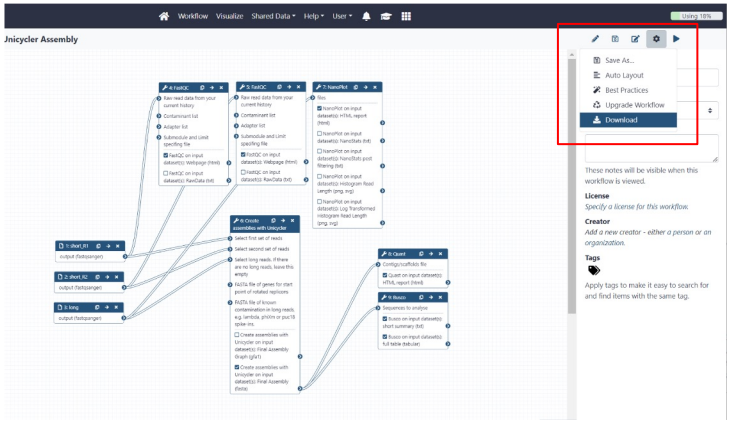
Workflows can be accessed 3 ways

- Via Shared Workflows
- Via User Workflows
- Via Link

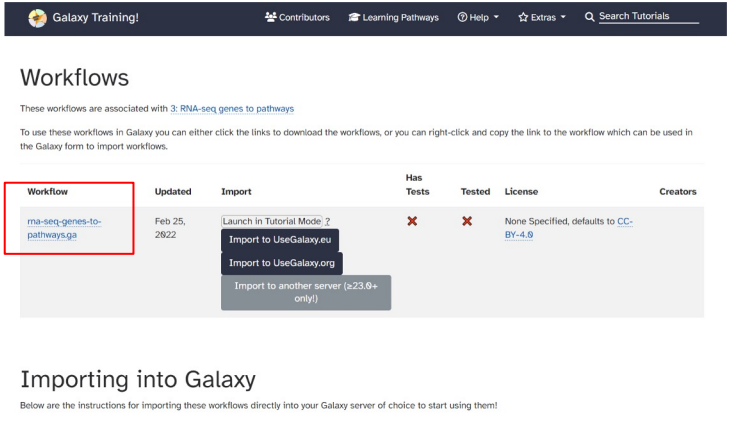
Shared Workflows



User Workflows



Via Link



Accessing Workflows - Workflow Format

Galaxy Workflows can be downloaded as .ga files.

These are json files which describe the metadata a workflow needs to run on a Galaxy server.

- Workflow metadata
- Step metadata
- Step input parameters
- Step tool info (tool name, revision etc)

Accessing Workflows - Workflow Format

Galaxy Workflows can be downloaded as .ga files.

These are json files which describe the metadata a workflow needs to run on a Galaxy server.

- Workflow metadata
- Step metadata
- Step input parameters
- Step tool info (tool name, revision etc)


```
"7": {
    "input_connections": {
        "inputFile": {
            "id": 5,
            "output_name": "output_alignments"
        }
    },
    "name": "MarkDuplicates",
    "outputs": [
        {
            "name": "metrics_file",
            "type": "txt"
        },
        {
            "name": "outFile",
            "type": "bam"
        }
    ],
    "tool_id": "toolshed.g2.bx.psu.edu/repos/devteam/picard/picard_MarkDuplicates/2.18.2.1",
    "tool_shed_repository": {
        "changeset_revision": "f6ced08779c4",
        "name": "picard",
        "owner": "devteam",
        "tool_shed": "toolshed.g2.bx.psu.edu"
    },
    "tool_state": "{\"assume_sorted\": \"true\", \"barcode_tag\": \"\", \"comments\": [],
    \"duplicate_scoring_strategy\": \"SUM_OF_BASE_QUALITIES\", \"inputFile\":
    {\"__class__\": \"ConnectedValue\"}, \"optical_duplicate_pixel_distance\": \"100\",
    \"read_name_regex\": \"\", \"remove_duplicates\": \"false\", \"validation_stringency\":
    \"LENIENT\", \"__page__\": null, \"__rerun_remap_job_id__\": null}",
    "tool_version": "2.18.2.1",
    "type": "tool",
    "uuid": "d68e12ea-a68a-49b0-a4cd-736e4dbd2178",
```

For Today

Will translate 2 Galaxy Tool Wrappers to Nextflow

- https://usegalaxy.org.au/root?tool_id=toolshed.g2.bx.psu.edu/repos/devteam/samtools_flagstat/samtools_flagstat/2.0.4
- https://usegalaxy.org.au/root?tool_id=toolshed.g2.bx.psu.edu/repos/iuc/limma_voom/limma_voom/3.50.1+galaxy0

Will translate 1 Galaxy Workflow to Nextflow

- https://training.galaxyproject.org/training-material/topics/transcriptomics/tutorials/rna-seq-reads-to-counts/workflows/rna-seq-reads-to-counts.ga

Let’s Begin