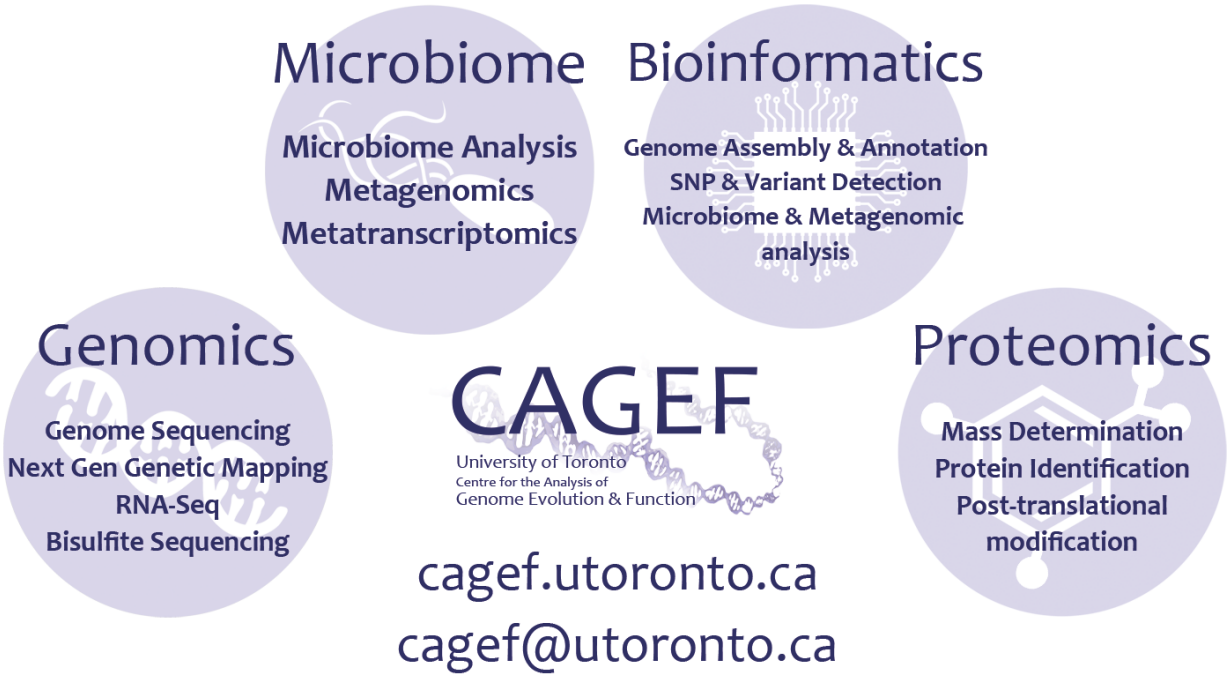


cagef.utoronto.ca
cagef@utoronto.ca

CSB1021HF LEC0131

FUNDAMENTALS OF GENOMIC DATA SCIENCE

0.0.0 Module 2: Assembly and annotation with Galaxy

0.1.0 About Fundamentals of Genomic Data Science

Fundamentals of Genomic Data Science is brought to you by the **Centre for the Analysis of Genome Evolution & Function (CAGEF)** bioinformatics training initiative. This course was developed based on feedback on the needs and interests of the Department of Cell & Systems Biology and the Department of Ecology and Evolutionary Biology.

The structure of this course is a "code-along", hands-on style! A few hours prior to each lecture, materials will be made available for download at QUERCUS (https://q.utoronto.ca/). The teaching materials will consist of a weekly PDF that you can use to follow along with the instructor along with any datasets that you'll need to complete the module. This learning approach will allow you to spend the time coding and not taking notes!

As we go along, there will be some in-class challenge questions for you to solve. Post lecture assessments will also be available for each module, building upon the concepts learned in class (see syllabus for grading scheme and percentages of the final mark).

0.1.1 Where is this course going?

We'll take a blank slate approach here to learning genomic data science and assume you know nothing about programming or working directly with next generation sequencing data. From the beginning of this course to the end we want to guide you from potential scenarios like:

- You don't know what to do with a set of raw sequencing files fresh from a facility like CAGEF.

- You've been handed a legacy pipeline to analyse your data or maintain for the lab, but you don't know what it runs or how.
- You plan on generating high-throughput data but there are no bioinformaticians around to help you out.

and get you to the point where you can:

- Recognize the basic tools in sequence analysis.
- Plan and write your own data analysis pipelines.
- Explain your data analysis methods to labmates, supervisors, and other colleagues.

0.1.2 How do we get there?

In the first half of this course, we'll focus on how to generate analysis pipelines using the Galaxy platform – a user-friendly graphical interface that provides access to common sequence analysis tools. After we are comfortable with these tools, we'll look at life through the lens of a command-line interface. It is here that we will learn the basics of file manipulation and how to program scripts that can carry out multiple tasks for us. From there we'll revisit tools from the first half and learn skills to make your data analysis life easier.

0.2.0 Goals of the module

1. Gain familiarity with the Galaxy interface.
2. Learn how to quality control sequencing reads with Galaxy.
3. Learn how to perform genome assembly and annotation with Galaxy.
4. Learn how to save and extract data and workflows with Galaxy.

0.3.0 Pre-class modules with Coursera

Each week we strongly encourage you to complete the assigned Coursera modules and/or readings ***before*** class. These are meant to provide you with sufficient background material on each week's module so that we can focus on the act of "doing" something with that data rather than spend a lot of time on the origins of it. You'll find a section outlining the next set of Coursera modules and readings ***at the end*** of each module.

0.3.1 Go to www.coursera.org and sign up for an account with your e-mail.

0.3.2 Search the following courses and enroll to audit each course (audit):

- Command Line Tools for Genomic Data Science, Johns Hopkins University.

0.4.0 Setting up your working directory

We suggest that you create a new directory (folder) for this course directly off your root directory called "**FGDS**". Working from your root directory is not necessary, but it will make some of the aspects of the course a little easier to manage. For MacOS users, we suggest you create this as a subfolder in your **user** directory.

0.4.1 Within this directory, create another directory called "**Module2**". This is where we will store the data used in this week's module.

0.4.2 Create a subdirectory called "**downloads**" to store the initial files as we download them before decompressing and working with them in later steps.

1.0.0 Logging into Galaxy

This week we'll being working with the Galaxy platform and becoming familiar with how to navigate around it. There are a few way you can use galaxy between using an account over at https://usegalaxy.org, setting up a server instance using https://genap.ca or trying to set up your own Amazon Web services. Today we'll mainly cover the second option by working with an instance on GenAP.

Why are we using Galaxy? As we mentioned in the class slides, just as with the world of biological science, it's important that our experiments and results are reproducible. While we often encounter detailed methods in manuscripts, we might forget that the same applies to our data analysis.

While we may take this idea for granted, it's important to remember that software tools and packages can undergo revision! Running the same set of data between *different versions* of tools can result in separate outputs. Furthermore, if you don't list your parameter choices in detail, it can make your results harder to interpret and replicate.

While tracking all these details can be quite important, approaching software tools can also be daunting! We use Galaxy as a good way to ease into these software packages, learn to manipulate their parameters, and simplify the process of connecting their output to other tools. The graphical user interface greatly decreases the energy barrier to learning these common bioinformatic tools.

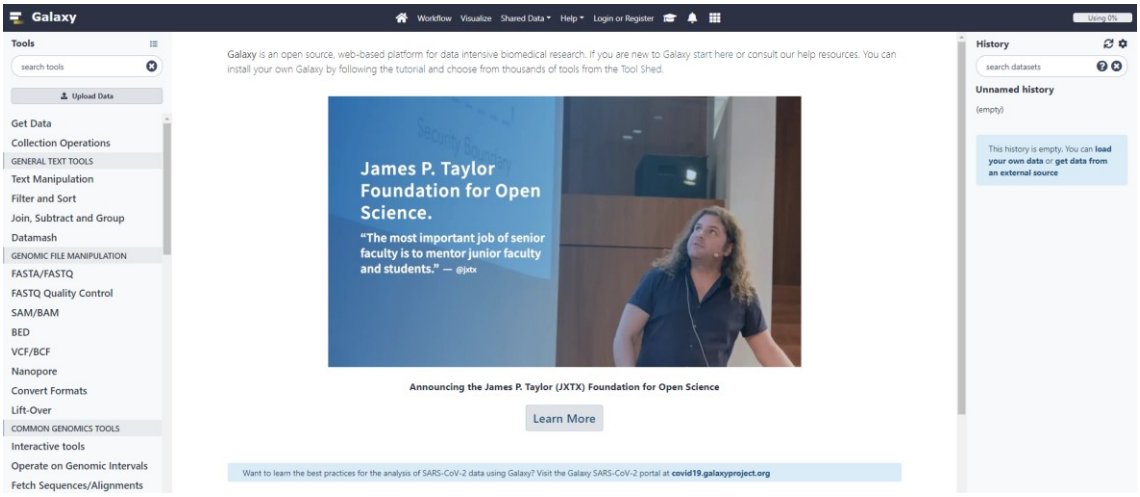
As we'll see later in class, you can also save these steps into workflows (pipelines) which you can use to repeat your analyses on different data sets or to tweak the parameters for a single dataset.

1.1.0 Galaxy Registration

After working with galaxy in this course you may wish to explore it more using the publicly available galaxy server architecture at https://usegalaxy.org. You'll find a number of tutorials, workflows, and datasets available here. After we're done with our work in this course you can use this as an alternate workspace to carry out some of your own pipelines.

1.1.1 Go to https://usegalaxy.org.

1.1.2 Select "Login or Register" located at the top of the page. Fill in all required fields and then submit.



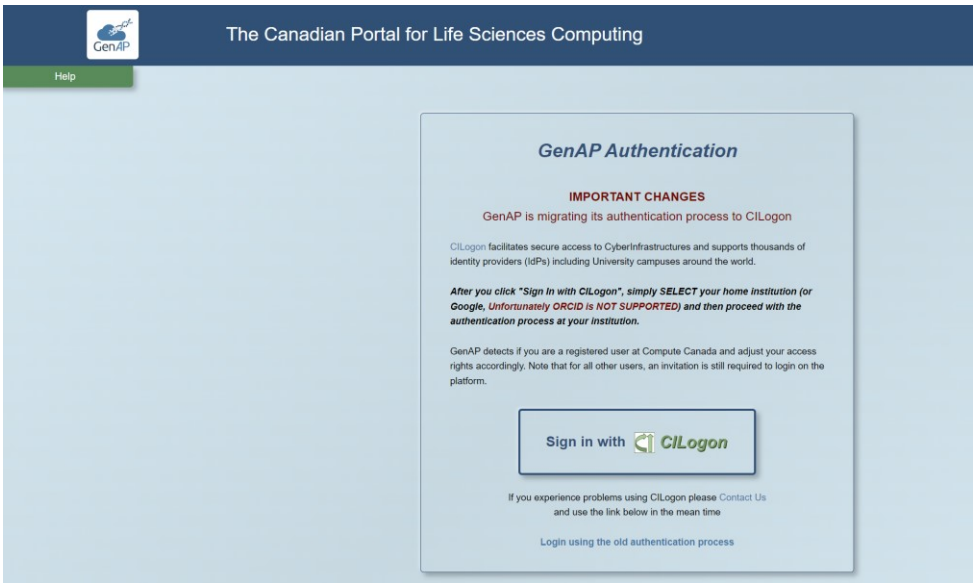
1.1.3 Verify your account by selecting the link in your email after it arrives.

The downside to using the online platform is that it can sometimes take several hours or even days for your analysis to get to the front of the queue

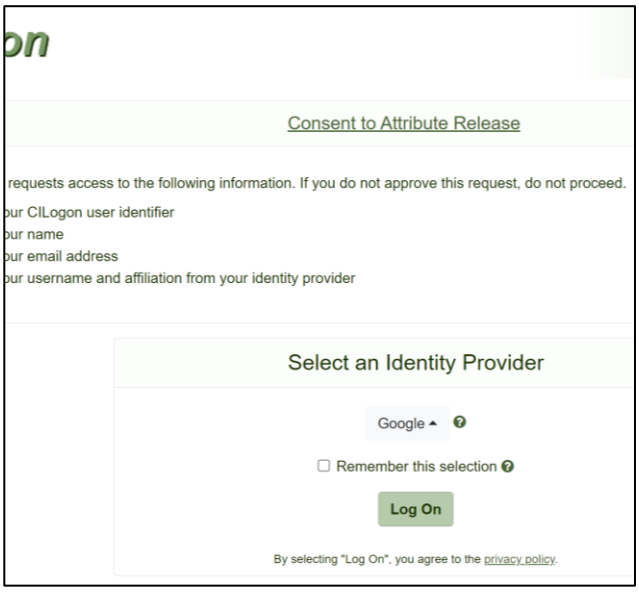
1.2.0 Running on our own galaxy instance through GenAP

GenAP is the Genetics & Genomics Analysis Platform and offers computing infrastructure to Canadian researchers. It is a cloud-based platform service that is built on Compute Canada resources. Therefore, having a Compute Canada Role Identifier (CCRI) can grant you easy access to these resources. For this course we have set up a galaxy instance at https://galaxy-547-p38.p.genap.ca and invited you to access it using your provided CCRIs or a valid gmail address.

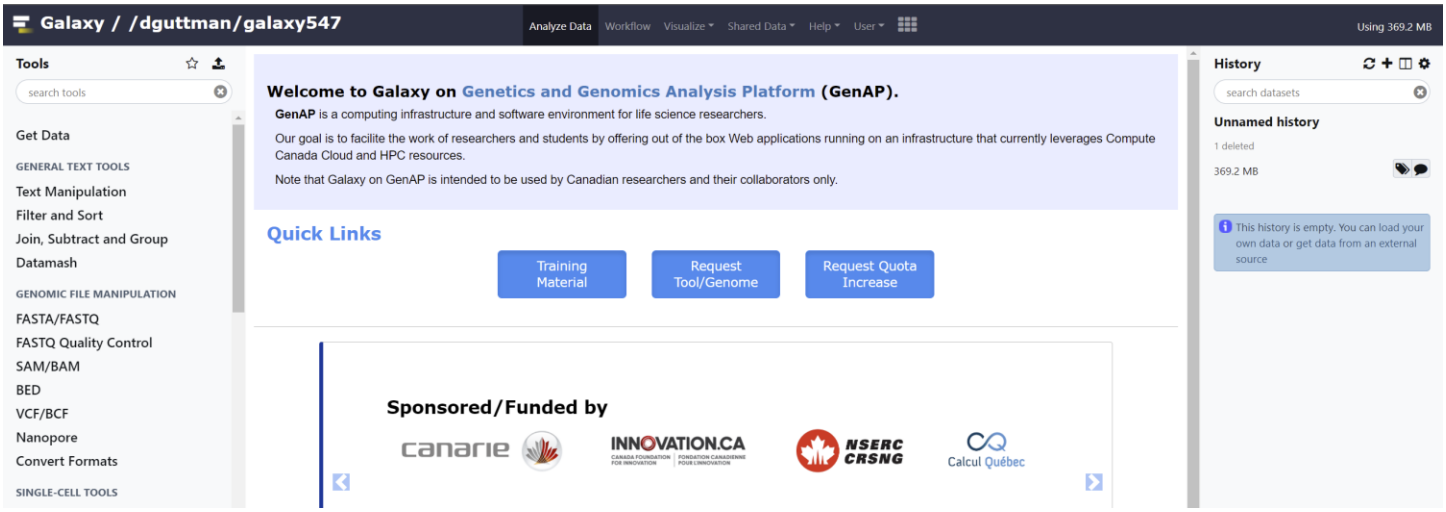
1.2.1 Login using your CILogon credentials – these will either take you to your uToronto page (with 2FA) or a google login page



1.2.1 Login using your University of Toronto or Google credentials



Once you are logged in, you'll be brought to the Galaxy homescreen. It is from here that we'll be doing all of our activities.



1.3.0 The Galaxy interface has four main quadrants

As you look at your galaxy home screen there are four quadrants to identify. Knowing the location and meaning of these will make navigation and following lecture instructions a little easier.

1.3.1 The Galaxy *application screen* is the main portion of your interface. It is centred on the screen and takes up 80% of the application interface. Here you can view results, datasets, and tool options as we navigate through our pipelines.

1.3.2 The Galaxy *menu* is located at the top of the screen. This menu can take you to a few different application screens where you can choose to Analyze Data, choose Workflows, and load Shared Datasets, as well as change user settings etc.

1.3.3 The Galaxy *tools pane* is located on the left side of the screen. This is most useful when working on the **Analyze Data** *application screen*. Scrolling down this tool menu you'll find popular tools. You can also use the *Search* bar to find tools by name.

1.3.4 The Galaxy *History pane* shows us the order of tools and applications that we have used. This can be used to see what you've done but also to put a pipeline together! We'll be referring to this often throughout our instructions.

Now that we've familiarized ourselves with the general interface of Galaxy, we can attempt to use some of the more popular tools and workflows. First, however, we need to import some data.

2.0.0 Importing datafiles in Galaxy

Before we can begin working with any data, we need to import it from somewhere. We can choose to upload data from local sources, from outside servers, or from within the Galaxy server itself. We'll walk through some helpful habits and instructions that will make it easier to import data for your data analysis

2.1.0 Creating a new history

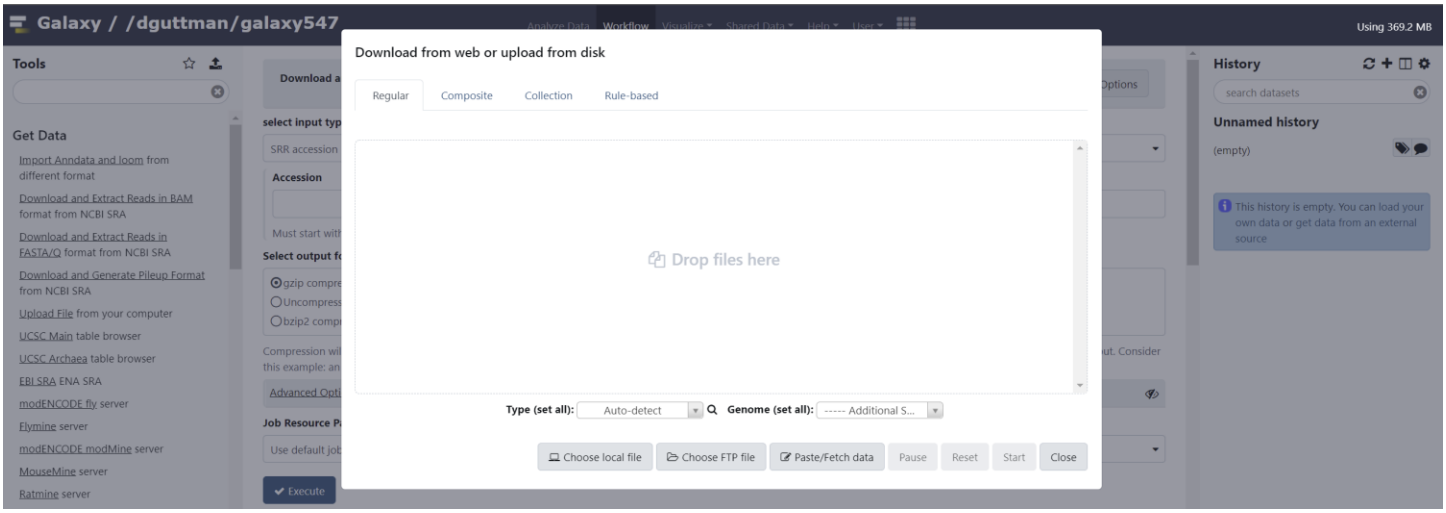
2.1.1 In the History pane, choose the **+** symbol in the top right to create a new history.

2.1.2 Rename the history (directly below the history search bar) from **Unnamed history** to [redacted] and hit the Enter key.

2.2.0 Retrieving local data to analyse on Galaxy

For a typical analysis on Galaxy, you will upload the required data to the server using the **Get Data** tool from the *tool pane*. This option should be the top tool in your *tool pane* and clicking on it you'll see many options! This includes downloading and extracting FASTQ reads directly from the NCBI SRA but we won't do that today.

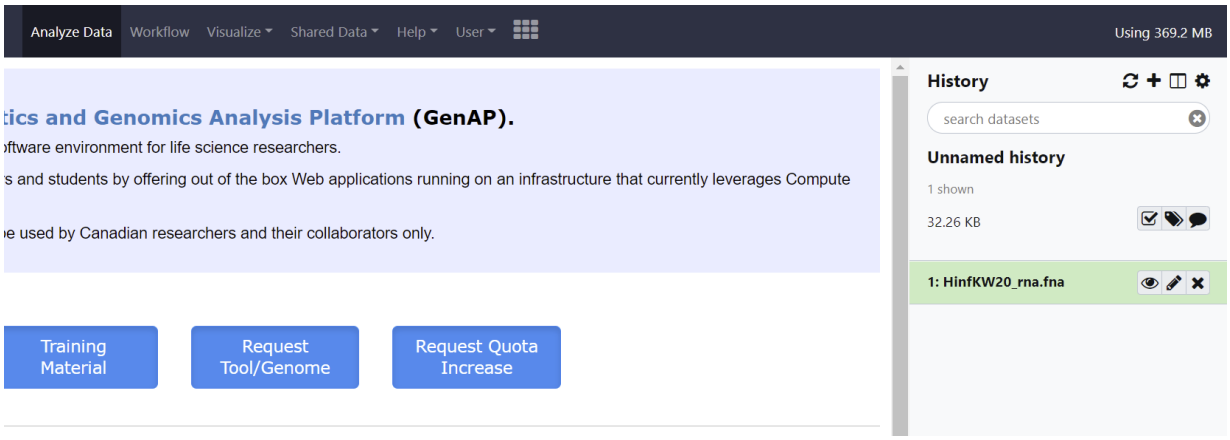
2.2.1 For now, we'll focus on [redacted] Clicking on this option will bring up the following window:



2.2.2 The data you will need for this module is the raw fastq reads for *H. influenzae* str. KW20. Unfortunately, upload and download speeds can be painstakingly slow for large files. Therefore, we will instead do a mock upload with a much smaller file [redacted] to provide an example of how this would work. Use the **Choose local file** button to upload this file data from last week.

2.2.3 Hit the **Start** button to being the upload. Note that you can queue multiple files for upload!

2.2.4 While the file is uploading, you'll see it in the *History pane* first as a grey entry with a small clock icon, then as a yellowish entry with a spinning circle icon. When the operation is complete, the entry will turn a green colour.



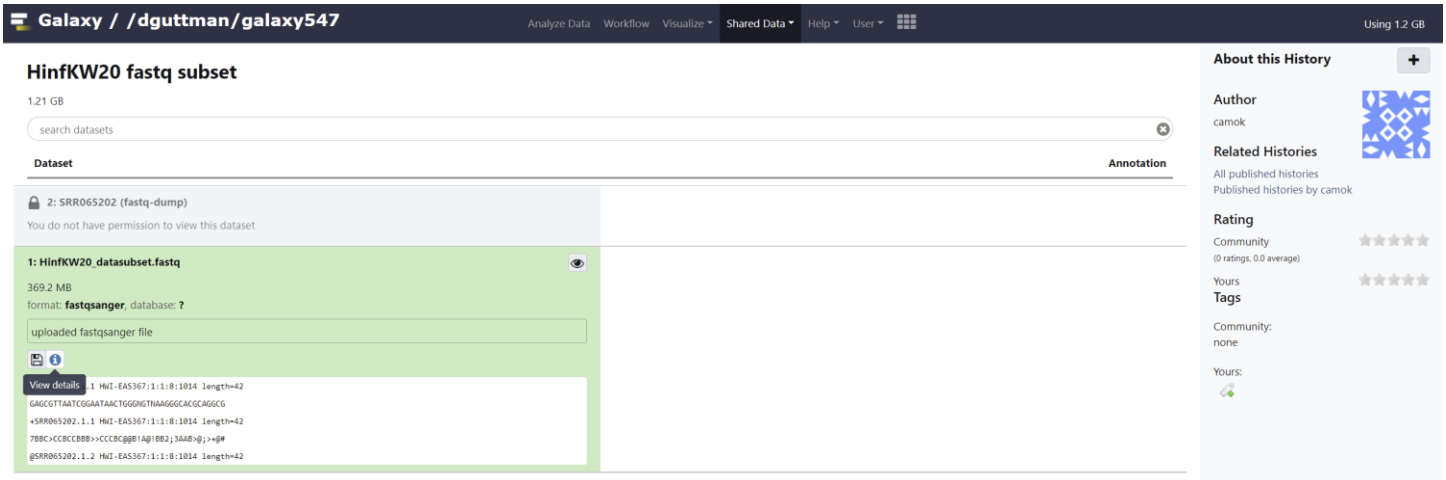
Notice that the entry has a number beside it? Each entry name in your *History pane* will be formatted as **#:history_entry** so that you can keep track of the order for your operations as well. As you create more entries, they will be listed in *descending* order.

2.3.0 Retrieving shared histories to analyse on Galaxy

While the above instructions are a useful example, we will still need the fastq file with our sequencing reads to do assembly and annotation. A subset file of only 2,000,000 reads has been uploaded to our server for you to access and load it to your own history using the “Shared Data” tab:

2.3.1. From the Galaxy *menu* choose [redacted].

2.3.2 From the dataset list, choose **HinfKW20 fastq subset**. This will bring up the following information about the shared history:



2.3.3 In the upper right corner you’ll click on the large + to import the history to your Galaxy instance. This will bring the data into your version of Galaxy without creating multiple permanent copies across the class.

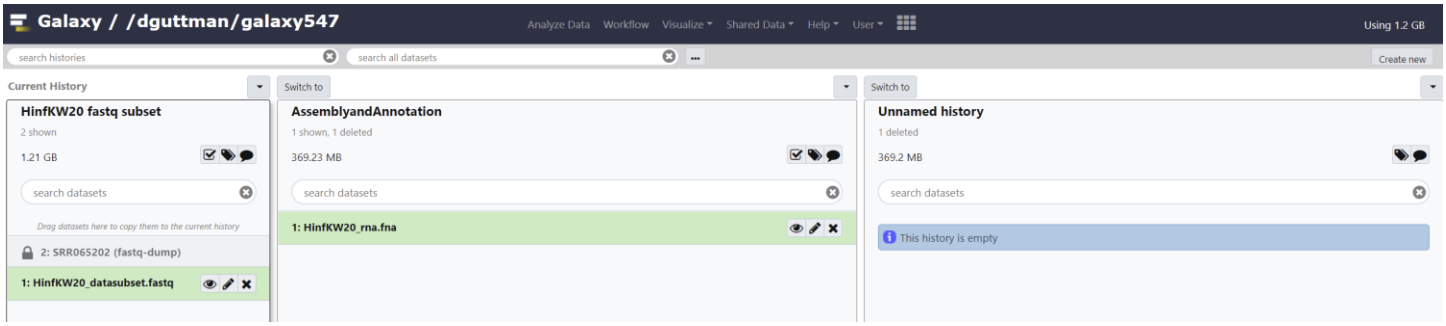
2.3.4 Name this imported history as **HinfKW20 fastq subset**. This action will return you to the main Galaxy homepage.

The process for adding shared datasets to your history will be similar. However, the current setup for the Galaxy instance facilitates shared data through *Shared Data > Histories* instead.

2.4.0 Add an imported history to your own history

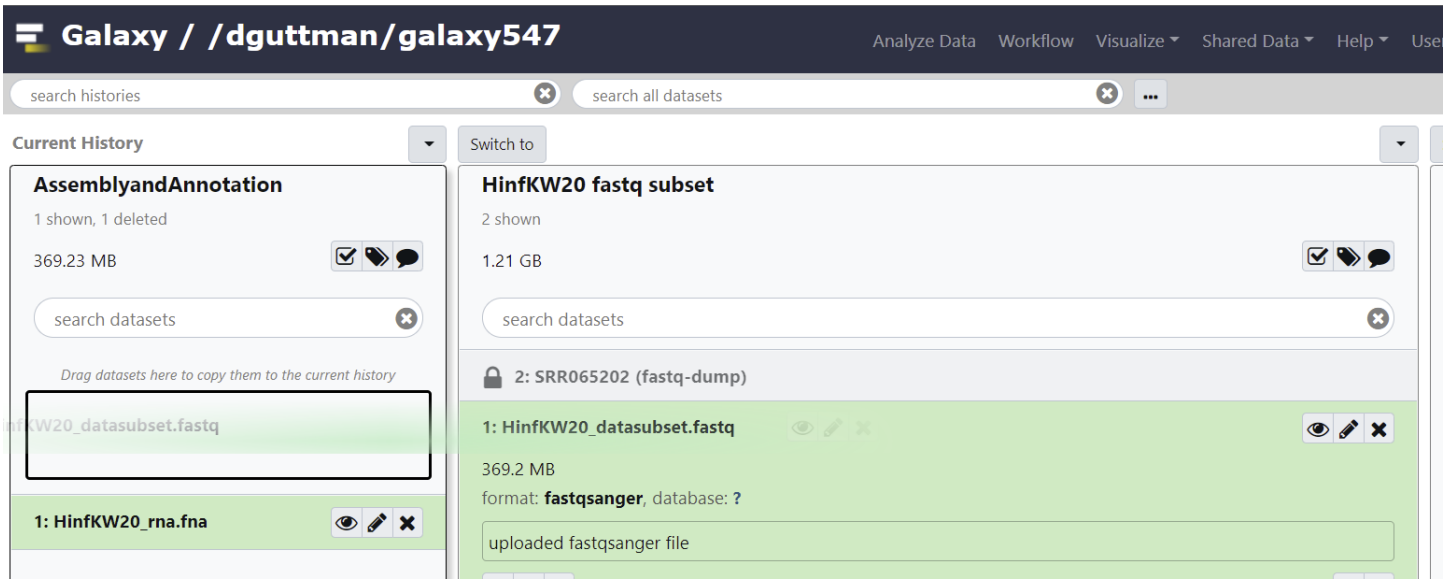
Now that we've imported a history into our instance of Galaxy we want to add that to our current **AssemblyandAnnotation** history.

2.4.1 Click on the **View all histories** icon (double rectangle) at the top of the *History pane*. This will bring you to the history page which provides a fast way to switch between multiple histories. Here we can see the newly imported history as well as our **AssemblyandAnnotation** history.



2.4.2 Above the **AssemblyandAnnotation** history, click on the [redacted] button.

2.4.3 In the **HinfKW20 fastq subset**, select the **HinfKW20_datasubset.fastq** and add it to the **AssemblyandAnnotation** history by *dragging and dropping* it into the supplied box.



2.4.4 Return to the Galaxy homepage by clicking on the top left corner.

2.4.5 You will now see the **HinfKW20_datasubset.fastq** file in your "History" panel. Verify that the datatype is **fastqsanger** using the [redacted]

2.4.6 Change datatype to **fastqsanger** if necessary.

3.0.0 Quality filtering raw sequencing reads

Now that our data subset is available in our history, the first step is to perform a quality inspection with **FastQC** to identify potential issues with our data. After quality inspection, we will perform quality control with **Trimmomatic** (if necessary) to remove any of the identified issues.

3.1.0 Run *FastQC* on raw sequencing reads

The FastQC tool is a popular and helpful way to check the overall read quality of your dataset. While some sequencing platforms like Illumina will perform a set of quality filtering as they run, these are really just eliminating reads with a low passing filter. This means, as a whole the reads show quality issues that make their overall quality questionable. The FastQC tool takes this a step further to look closely at the Q-scores in a number of modules to create a human-readable html report.

3.1.1 From the Galaxy *tools pane* select [redacted]

3.1.2 Scroll down to review the purpose and input/output information for *FastQC*.

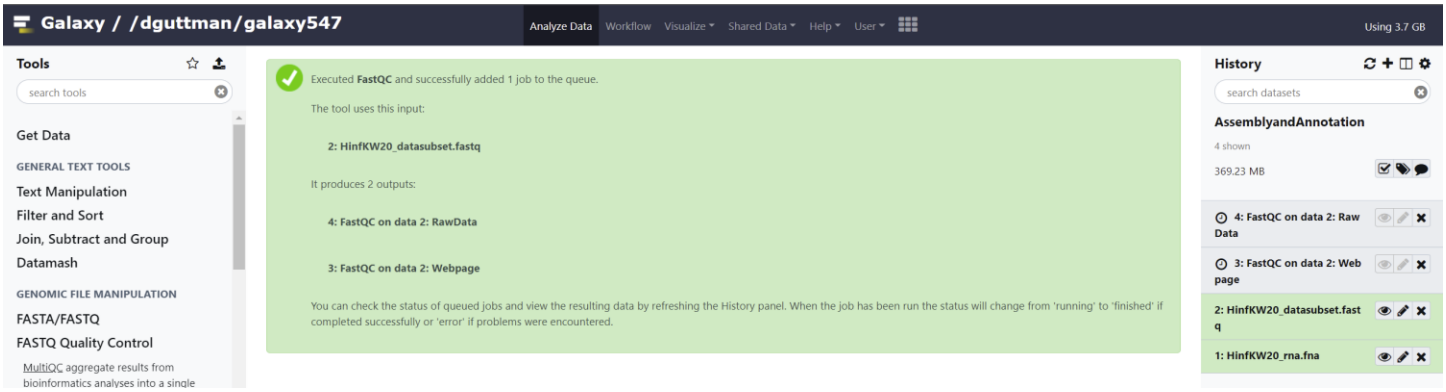
3.1.3 Execute *FastQC* with the following settings:

- Short read data from your current history: **2: HinfKW20_datasubset.fastq**
- Other options: **default**

The run will generate two new history entries:

- **3: FastQC on data 2: Webpage**
- **4: FastQC on data 2: RawData**

Note that in some versions, you may be given the option to title your output. Our GenAP instance uses the history value (***data 2***) as the naming scheme to produce output.



3.2.0 Reviewing your *FastQC* output

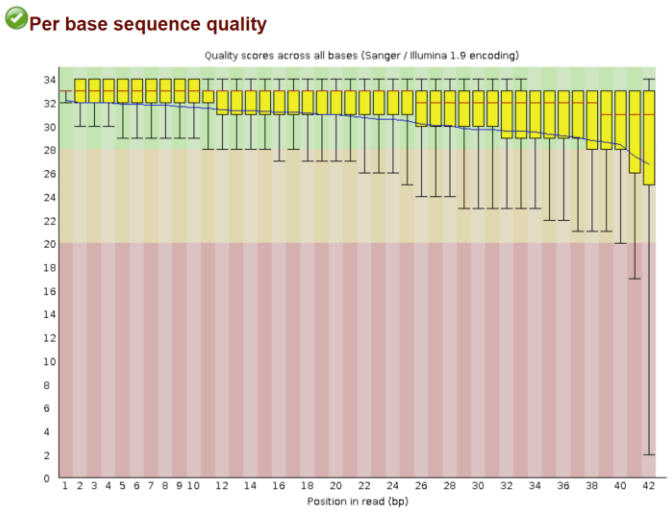
Once the background of the output files is green, the analysis is complete. We can review the results of the FastQC analysis on the *History pane*.

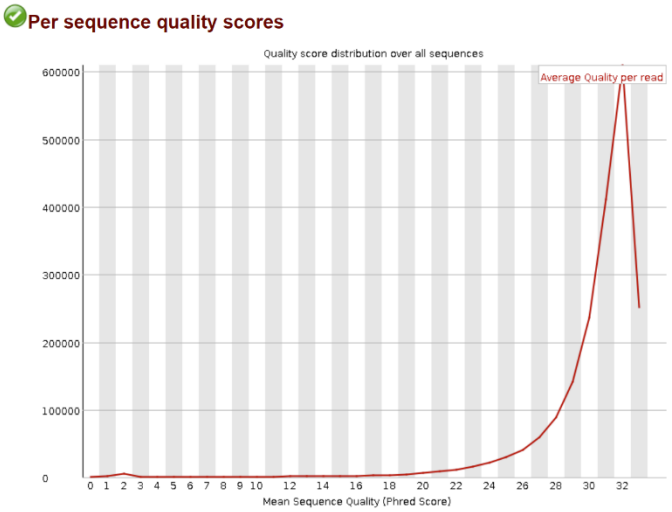
3.2.1 Review the HTML output from [redacted] by selecting the *View data* icon (eye) on the history entry.

3.2.2 You will see the following result sections that you can review:

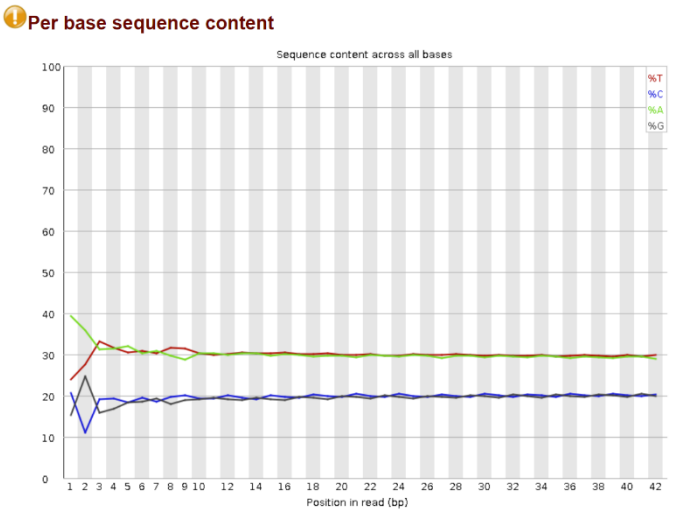
- Basic Statistics (Pass)
- Per base sequence quality (Pass)
- Per tile sequence quality (Pass)
- Per sequence quality scores (Pass)
- Per base sequence content (Warning)
- Per base GC content (Pass)
- Per sequence GC content (Pass)
- Per base N Content (Pass)
- Sequence Length Distribution (Pass)
- Sequence Duplication Levels (Warning)
- Overrepresented Sequences (Pass)
- Adapter Content (Pass)

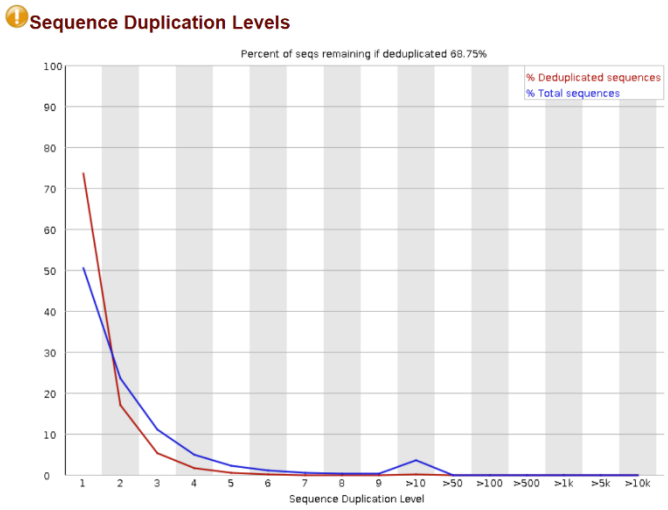
3.2.3 From our analysis we can see that overall, the read quality of along the set of raw reads remains quite high throughout the length of the reads. Even towards the tail end of the reads, the median Q score remains above 28. The average read quality per sequence is ~32.5 and the distribution from 2M reads shows bulk of reads to have a mean Q score above 28.





3.2.4 While this analysis reveals that there are no major problems with our dataset, although sequence content is slightly biased at the start of every read, and there is a partial excess of duplicated sequences (~31%):





An alternative to FastQC: depending on your needs, experimental data types and level of throughput there are other programs that you may use to investigate your read quality. MultiQC is a program that we'll explore in later sessions and it will aggregate results from multiple bioinformatics analyses into a single report. Other options for FASTQ file analysis include quack and fastp.

3.3.0 Use *Trimmomatic* to remove poor-quality reads

Trimmomatic performs a variety of useful tasks to help remove unwanted sequences from your reads. Developed for Illumina sequencing reads, this tool can remove adapter sequences generated during library preparation, while also removing sequences from the ends of your reads based on criteria like quality score.

Using this tool on our dataset may help remove some of the warnings received during our *FastQC* analysis. However, care should always be taken when trimming or removing reads, because strict filters can result in a lot of lost data and create new issues with sequencing reads.

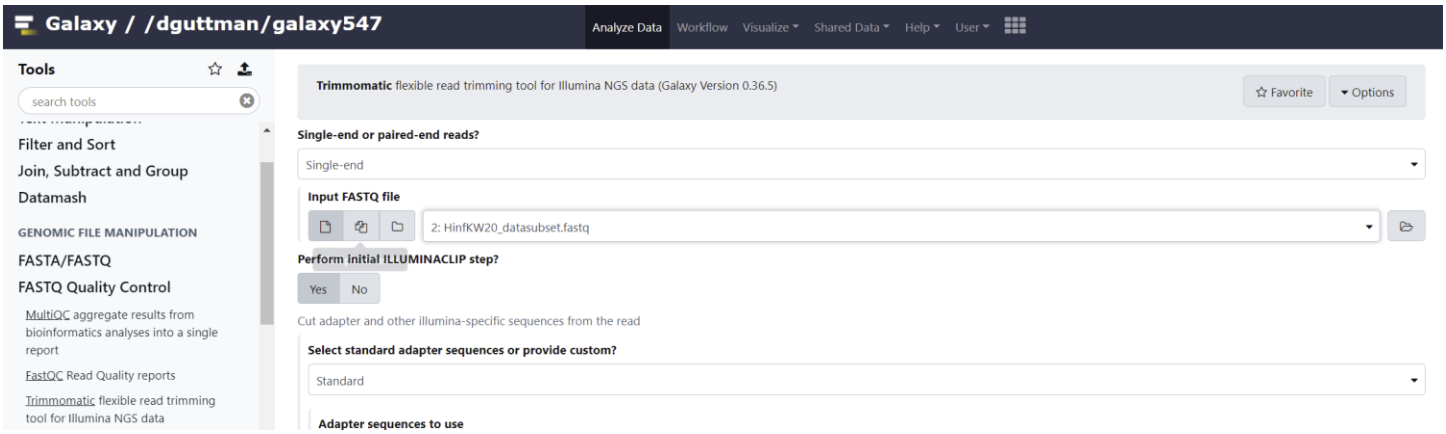
3.3.1

3.3.2 Review the parameters that can be used for *Trimmomatic* and the required inputs. Some commonly employed parameters include:

- **ILLUMINACLIP:** Trim adaptor and other Illumina sequences off the ends of reads.
- **SLIDINGWINDOW:** Trim reads based on average quality in a window of defined size.
- **MINLEN:** Drop all reads below a defined length.
- **LEADING/TRAILING:** Cut bases at the start/end of reads if below defined quality.
- **CROP/HEADCROP:** Cut a defined number of bases at start/end of reads (regardless of quality).
- **AVGQUAL:** Drop all reads below a defined average quality.

3.3.3 Execute Trimmomatic with the following settings:

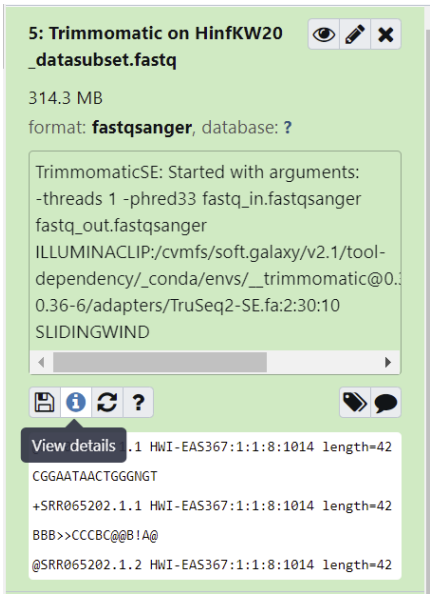
- Single-end or paired-end reads: **Single-end**
- Input FASTQ file: **2: HinfKW20_datasubset.fastq**
- Perform initial ILLUMINACLIP step? **Yes** (generates the following default values)
    - Select standard adapter sequences or provide custom? **Standard**
    - Adapter sequences to use: **TruSeq2 (paired-ended for Illumina GAII)**
    - Maximum mismatch count: **2**
    - Minimum matches between two 'adapter ligated' reads: **30**
    - Minimum matches between adapter sequence and a read: **10**
    - Minimum adapter length that needs to be detected: **8**
- Trimmomatic Operation:
    - **Sliding window trimming (SLIDINGWINDOW)** with default values
        - Number of bases to average across: **4**
        - Average quality required: **20**
- Insert Trimmomatic Operation:
    - [redacted]



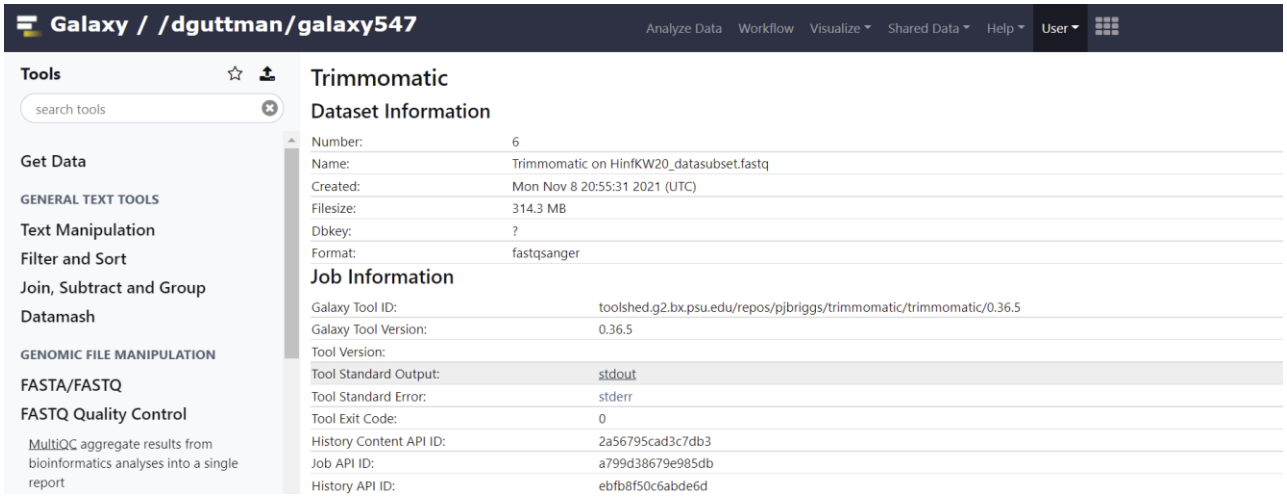
3.3.4 Note that with **Trimmomatic**, the order of operations matters. For example, if you performed the *SLIDINGWINDOW* trimming before using *CROP*, you would be cropping reads that have already been trimmed. This would not usually be what you want, because the *SLIDINGWINDOW* will already have removed many of the low-quality bases at the end of reads.

3.3.5 To view some of the run details, in the *History pane*, click on the entry for **Trimmomatic on HinfKW20_datasubset.fastq**. This will expand the history entry.

3.3.6 Click on the [redacted] which will bring up the running parameters of the *Trimmomatic* run



3.3.7 Under the *Job Information* section, find the *Tool Standard Output* line and select **stdout**.



3.3.8 Note that of 2M reads, 96.94% survived while 3.06% were dropped. The **stdout** (standard out) information also includes the Illumina adapter sequences used in the clipping process.

3.4.0 Rerun your trimmed data in *FastQC*

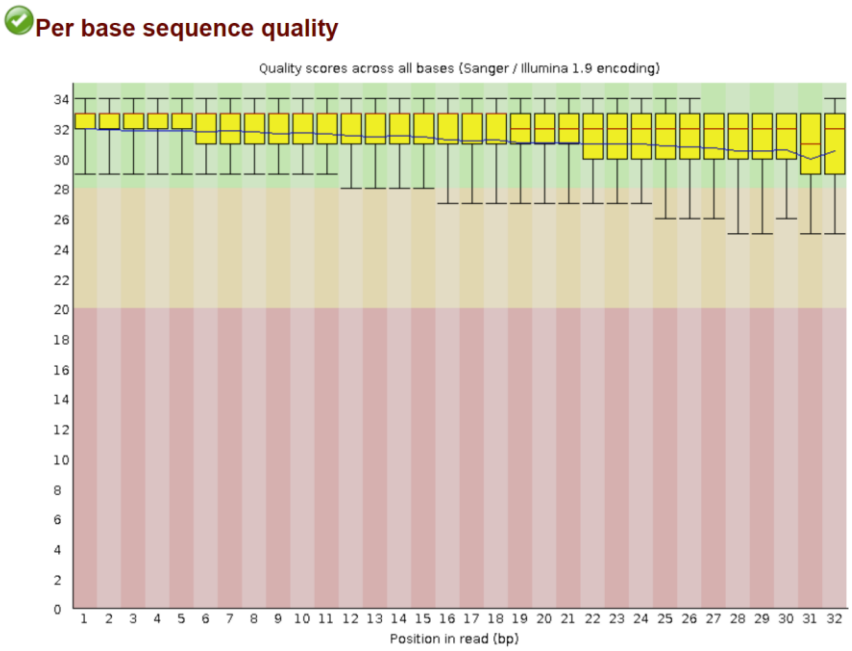
Now that we've trimmed our data of adapters sequences and some poor-quality reads, we can return to examine how this changes our FastQC output.

3.4.1 From the Galaxy *tools pane* select **FASTQ Quality Control > FastQC Read Quality reports**

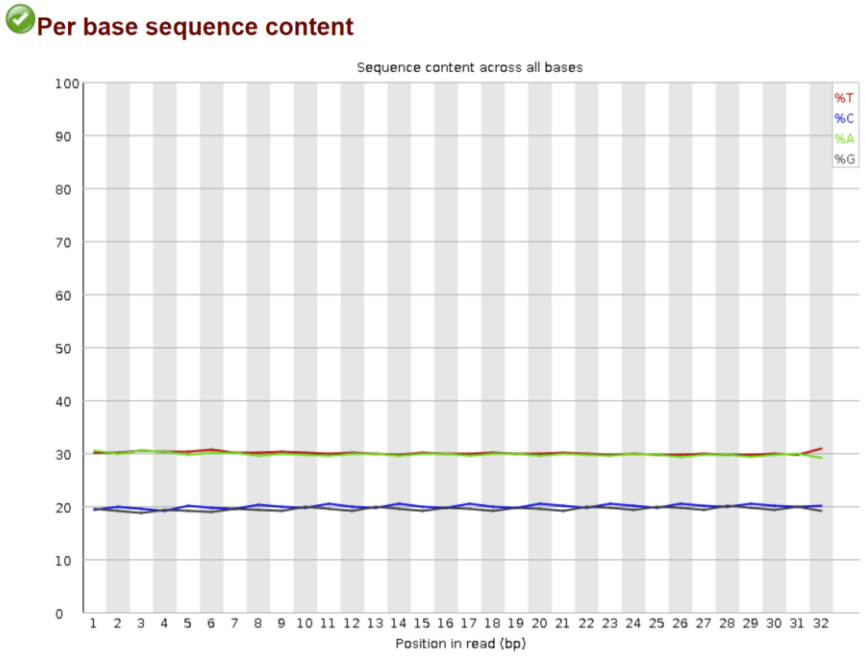
3.4.2 Execute FastQC with the following settings:

- Short read data from your current history: **5: Trimmomatic on HinfKW20_datasubset.fastq**
- Other options: **default**

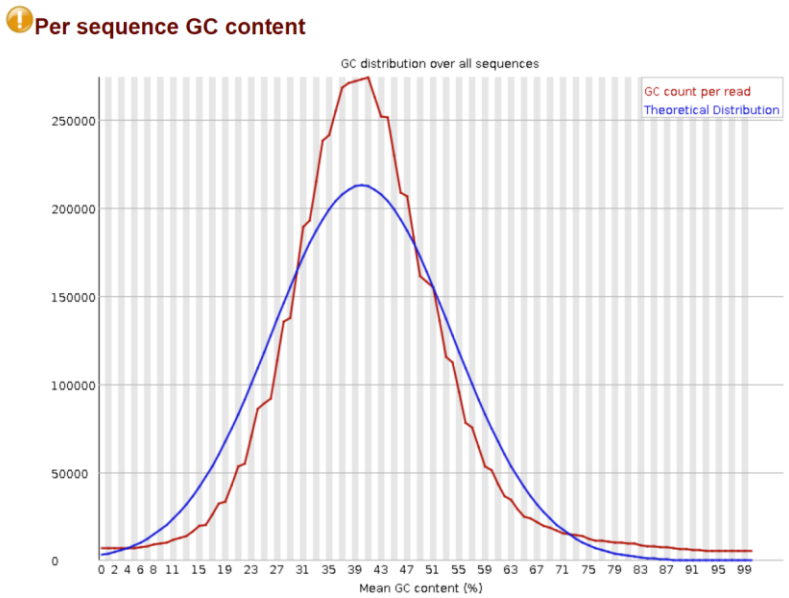
3.4.3 Review the HTML output from **6: FastQC on data 5: Webpage** by selecting the *View data* icon (eye) on the history entry.



- We have solved our initial issues with **Per base sequence content** through the **HEADCROP** command which removed the first 10 bases of each read thus reducing the initial biases at the start of our reads.



- As a consequence, however, our reads are now only 32 bases and we get a warning for the **Per sequence GC content** as the distribution is shifting slightly to a smaller standard deviation.



- We can also see from our **Sequence Length Distribution** that our reads are no longer strictly 42 basepairs but range from 14 to 32 basespairs in length.

4.0.0 Genome Assembly with Velvet

To recap, we have imported raw sequence reads from the *Haemophilus influenza* genome and have trimmed and filtered these reads using a combination of *Trimmomatic* and *FastQC* to analyse the read quality and distribution. We are now ready to assemble these reads into our first draft genome assembly with the Velvet tool.

Given the nature of our reads now ranging in size from 14 to 32 basepairs we can assume that we will end up with several smaller contigs (assembled fragments). Other contributors of small contigs can be low coverage or repetitive regions. Regardless, we'll have an assembly that we can begin to annotate in later steps.

The *Velvet* assembly tool runs in two steps by

1) Use *velveth* to build a ***h***ash table of k-mers in each read ie. it tracks each read by its different subsequences.

2) Use *velvetg* to perform a ***g***enome assembly based on the hash table from *velveth*

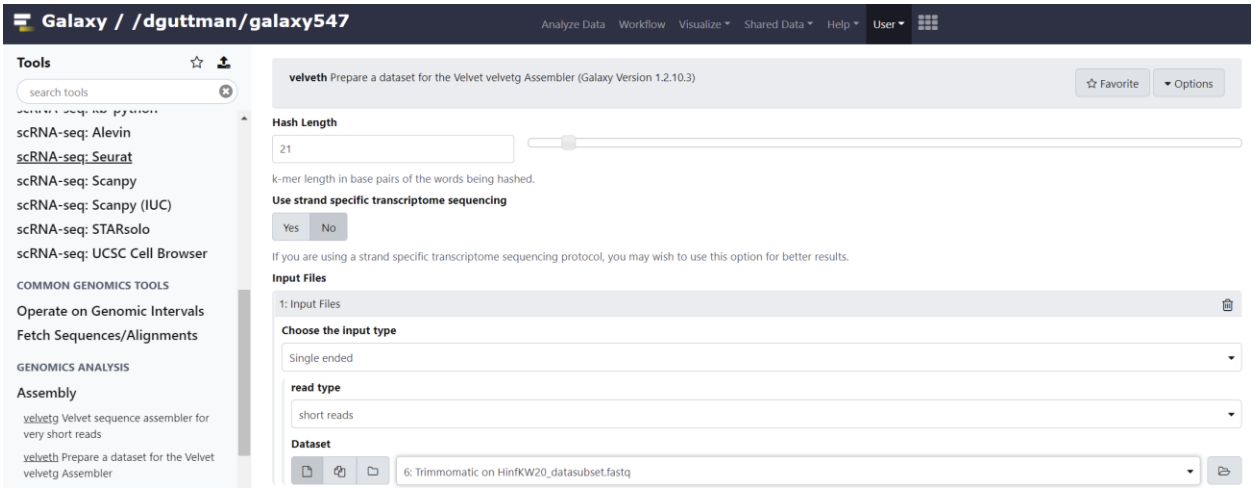
4.1.0 Use *velveth* to produce a hash table of sequences

4.1.1

4.1.2 Scroll down to review the purpose and input/output information about *velveth*.

4.1.3 Execute *velveth* to create your hash table using the following settings:

- Hash Length: **21**
- Use strand specific transcriptome sequencing: **No**
- Input Files:
  - Choose the input type: **Single ended**
  - Read type: **short reads**
  - Dataset: **5: Trimmomatic on HinfKW20_datasubset.fastq**



4.2.0 Use *velvetg* to produce a genome assembly

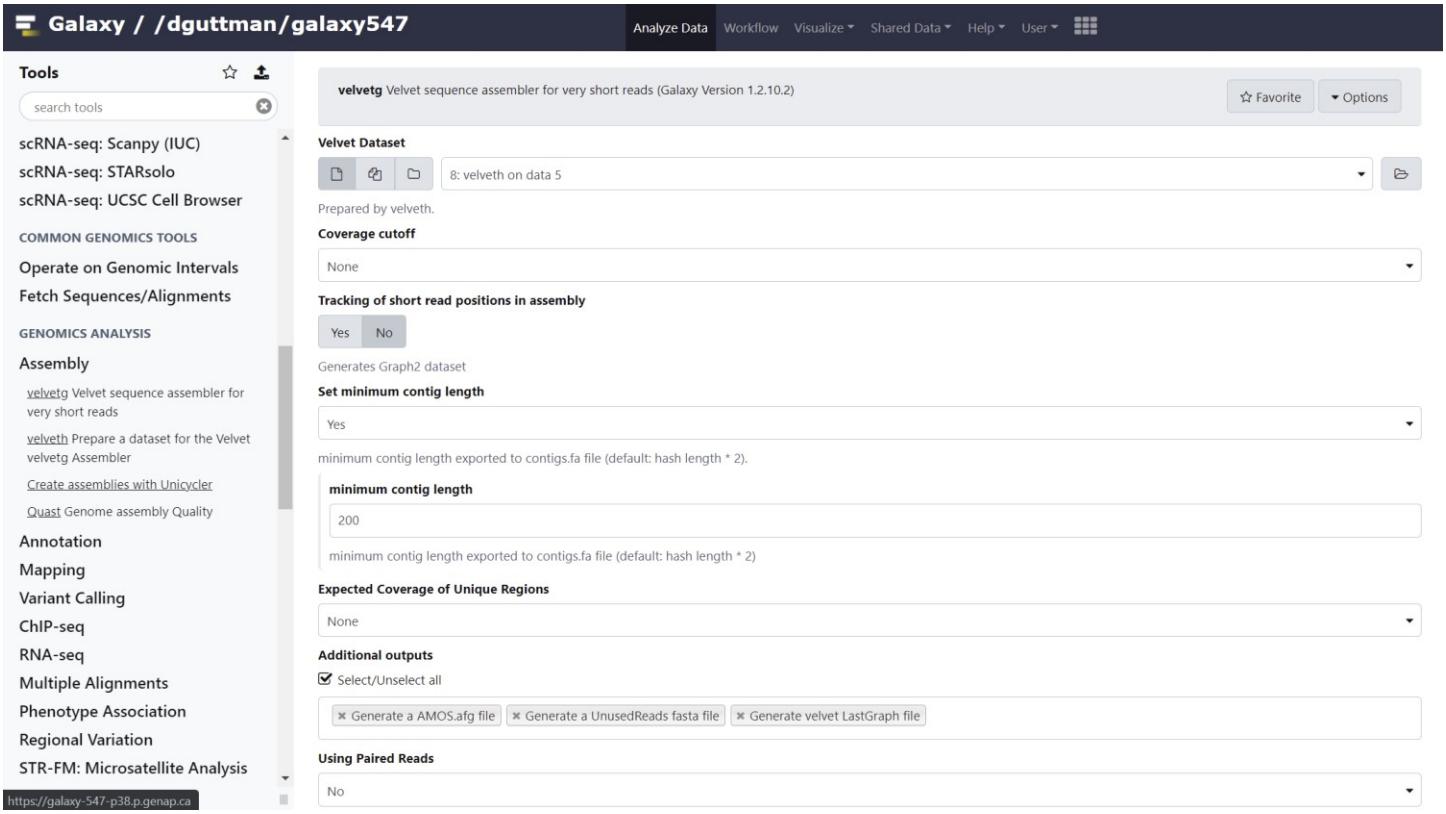
Now that we have our hash table from *velveth* we can send this on the *velvetg* which will use the information to construct a de Bruijn graph by adding the k-mers one-by-one to the graph and then chaining and merging paths and nodes to produce the final contig sequences.

4.2.1

4.2.2 Scroll down to review the purpose and input/output information about *velvetg*.

4.2.3 Execute *velvetg* to produce your assembly using the following settings:

- Velvet Dataset: **velveth on data 5**
- Coverage cutoff: **Automatically determined**
- Tracking of short read positions in assembly: **No**
- Set minimum contig length: **Yes**
    - Minimum contig length: **200**
- Expected Coverage of Unique Regions: **None**
- Additional outputs: **check this box** (if you don't you'll get a Java error)
    - Generate a AMOS.afg file
    - Generate a UnusedReads fasta file
    - Generate a velvet LastGraph file
- Using Paired Reads: **No**



4.3.0 Reviewing your velvetg results

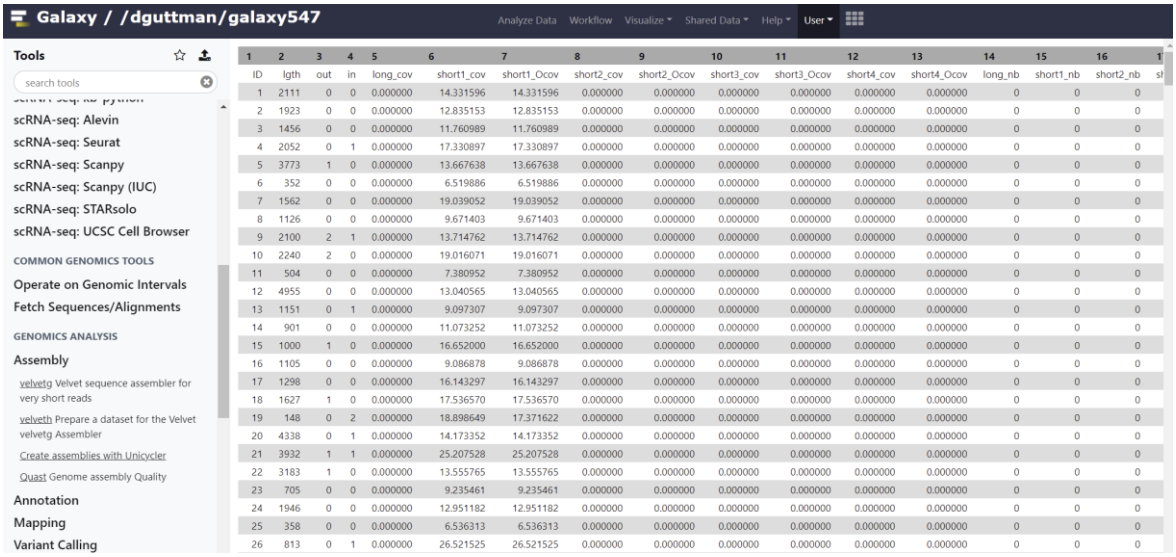
Looking at our *History pane*, we see that *velvetg* has generated five sets of output but we are just interested in two: **velvetg on data 8: Contigs** and **velvetg on data 8: Stats**.

4.3.1 Review the **stdout** file from *velvetg on data 8: Contigs > View Details >* **stdout.**

You will be able to find an N50 and total length for your assembly at the bottom of file, but these values are for your de Bruijn graph so they don't account for contig filtering. N50 is a quality metric that is

defined as the shortest sequence length at 50% of the genome. Higher N50 values mean that you have a more contiguous assembly.

4.3.2 Review the **Stats** output from *velvetg on data 8: Stats > View Data*.



ID	lgth	out	in	long_cov	short1_cov	short1_Ocov	short2_cov	short2_Ocov	short3_cov	short3_Ocov	short4_cov	short4_Ocov	long_nb	short1_nb	short2_nb
1	2111	0	0	0.000000	14.331596	14.331596	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
2	1923	0	0	0.000000	12.835153	12.835153	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
3	1456	0	0	0.000000	11.760989	11.760989	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
4	2052	0	1	0.000000	17.330897	17.330897	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
5	3773	1	0	0.000000	13.667638	13.667638	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
6	352	0	0	0.000000	6.519886	6.519886	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
7	1562	0	0	0.000000	19.039052	19.039052	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
8	1126	0	0	0.000000	9.671403	9.671403	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
9	2100	2	1	0.000000	13.714762	13.714762	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
10	2240	2	0	0.000000	19.016071	19.016071	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
11	504	0	0	0.000000	7.380952	7.380952	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
12	4955	0	0	0.000000	13.040565	13.040565	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
13	1151	0	1	0.000000	9.097307	9.097307	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
14	901	0	0	0.000000	11.073252	11.073252	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
15	1000	1	0	0.000000	16.652000	16.652000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
16	1105	0	0	0.000000	9.086878	9.086878	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
17	1298	0	0	0.000000	16.143297	16.143297	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
18	1627	1	0	0.000000	17.536570	17.536570	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
19	148	0	2	0.000000	18.898649	17.371622	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
20	4338	0	1	0.000000	14.173352	14.173352	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
21	3932	1	1	0.000000	25.207528	25.207528	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
22	3183	1	0	0.000000	13.555765	13.555765	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
23	705	0	0	0.000000	9.235461	9.235461	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
24	1946	0	0	0.000000	12.951182	12.951182	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
25	358	0	0	0.000000	6.536313	6.536313	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0
26	813	0	1	0.000000	26.521525	26.521525	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	0	0

Now that we have our genome assembled, and a list of contigs, what should we do with it? Annotate it of course!

What is a de Bruijn graph? Yes, you have generated a genome using **velvetg** but how does it *actually* work? All the **k-mers** are represented in a graph with connected neighbours being **k-mers** overlapping by (**k-1)** bases. Each node stores information about how often its k-mer is seen within the data. For more information on using the Velvet *de novo* assembler, check out Zerbino 2010 on Pubmed

4.4.0 Determining the quality of your assembly

Now that we've generated an assembly, you may be wondering if it's any good? How well would this compare to the genome assembly for *H. influenza* that we downloaded for Module 1? There are some genome assembly analysis tools that you can use to compare the results from various genome assemblers or variations from the same assembler. Today we'll use a **QU**ality **AS**sessment **T**ool, **QUAST**, to help us compare our *velvet* assembly to a reference genome.

4.4.1 First we'll retrieve a copy of our *H. influenzae* genome with **Get Data > Upload File from your computer.**

4.4.2 Upload the file [redacted] from your computer.

4.4.3 From the Galaxy *tools pane* select **Assembly > Quast Genome assembly Quality**.

4.4.4 Scroll down to review the purpose and input/output information about *QUAST*.

4.4.5 Execute *QUAST* to evaluate your *velvet* assembly using the following settings:

- Contig/scaffolds output file: **velvetg on data 8: Contigs**
- Type of data: **Contig**
- Size of reference genome: [redacted]

- Reference genome: **HinfKW20_genomic.fna**
- Other options: **default**

The QUAST program will produce a series of 8 files:

- **Log**: A quick report of the run log for the package.
- **Misassemblies**: A table outlining potential problems with the assembly including misassembled contigs.
- **Unaligned Contigs**: A report on the unaligned contigs from the assembly or assemblies.
- **Contig view**: A visualization of the contigs and their classes of alignment (correct, misassembled, unaligned).
- **Alignment view**: A visualization of the *where* contigs align to your reference genome. This is also an interactive browser so you can zoom and examine individual contigs and their alignment.
- **Report (Tabular, HTML, PDF)**: Various summaries of your assembly quality, including the alignment size.

The Icarus alignment viewer shows the placement of our assembly contigs

Genome assemblies can be complex! Given that we are working with a small genome, the complexity of our assembly is relatively low. You can see that our assembly didn't quite cover the entire reference genome and there are a few possible reasons for why that might be the case. We'll explore some of those in this week's assignment and next week's module as well.

The quality of your genome assembly will depend quite a bit on factors like read length, read depth, single vs. paired-end sequencing format, and is best complemented through additional sequencing methods like long-read sequencing and Hi-C sequencing. These methods can help reduce gaps, and correct contig orientation respectively.

5.0.0 Genome annotation

As is the case with most forms of genome analysis, there are a large number of options available for genome annotation - the process of identifying coding regions in a genome and assigning a function to them.

5.1.0 Genome annotation with Prokka

Prokka is designed for smaller genomes like those of bacteria, archaea, and viruses. Its major advantages are speed and simplicity. **Prokka** can fully annotate a typical 4Mbp genome in less than 10 minutes on most modern personal computers. Furthermore, it integrates coding sequence identification and functional assignment into a single command, which makes it very easy to use.

For eukaryotic genomes, you must consider introns and exons for each gene. It is also important that you mask repetitive sequences in these genomes. Some common programs for the annotation of eukaryotic genomes include **Augustus**, **Maker**, and **Pasa**. Information about these programs is available online.

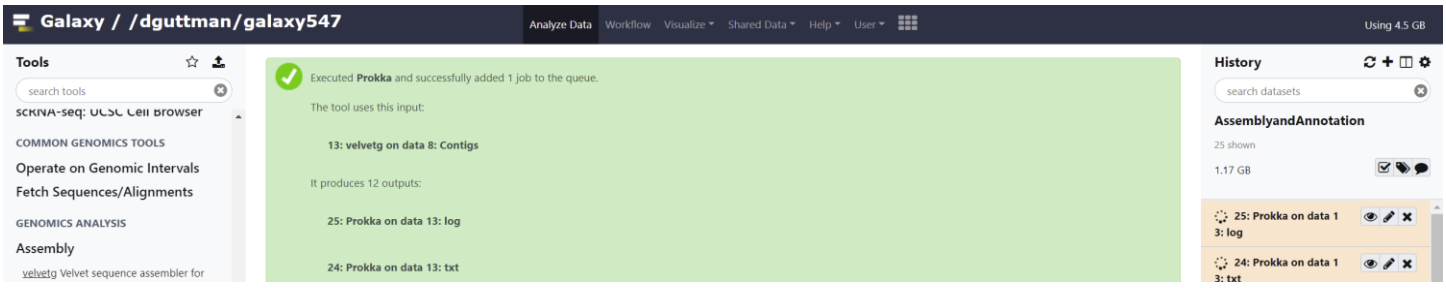
5.1.1 From the Galaxy *tools pane* select **Annotation > Prokka Prokaryotic genome annotation**

5.1.2 Scroll down to review the purpose and input/output information about Prokka.

There are lots of options here that can be useful depending on the goals of the annotation. These include, for example, specifying genus or species-specific databases to prioritize when assigning functions to coding regions. You can also specify which output files you want Prokka to generate. Today, we will simply be running Prokka using default settings and will be assigning functions with the default BLAST database.

5.1.3 Execute Prokka to produce your annotation using the following settings:

5.1.4 Scrolling past the remaining parameters (leave as **default**) we find the *Additional outputs* section where we can choose which files to create from the annotation. Leave these as the default (**Select all**).



5.2.0 Review your *Prokka* output

You'll see that our *Prokka* execution has generated 12 files, many of which should appear familiar to you from module 1: **gff**, **gbk**, **fna**, **faa**, **ffn**, **sqn**, **fsa**, **tbl**. In addition to this it has also produced a coding sequence (**tsv**); a discrepancy report (**err**); a brief summary (**txt**); and a more verbose run report (**log**).

Many of these files are what we would find accompanying our assembly report from NCBI.

6.0.0 Downloading data and saving your pipeline

Downloading output files and applying appropriate naming conventions from Galaxy analyses is not as easy as we might like. This process can be sped up somewhat by gaining familiarity with the command line, but in this section of the course we will stick with **Finder** and **File Explorer**.

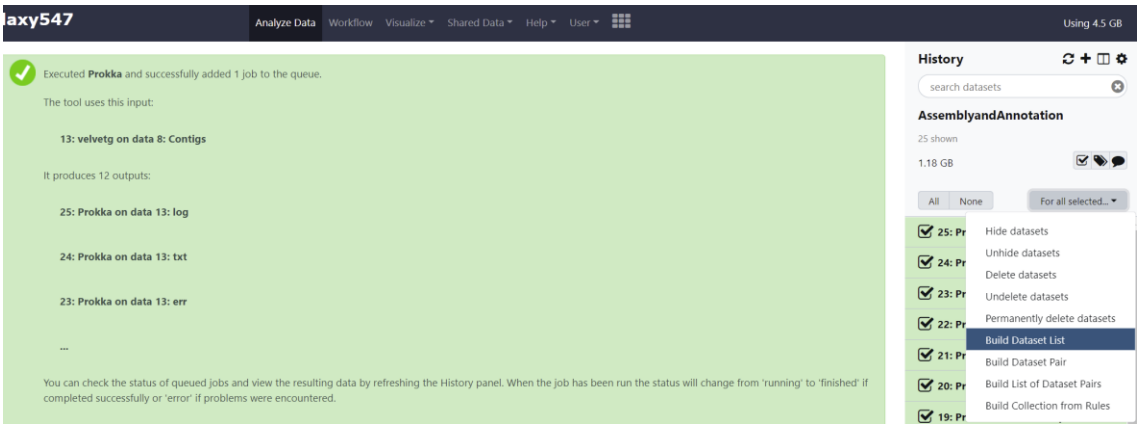
6.1.0 Create a dataset list of your data from the *History pane*

6.1.1 Recall that you have already set up a subfolder for this module under your main FGDS folder.

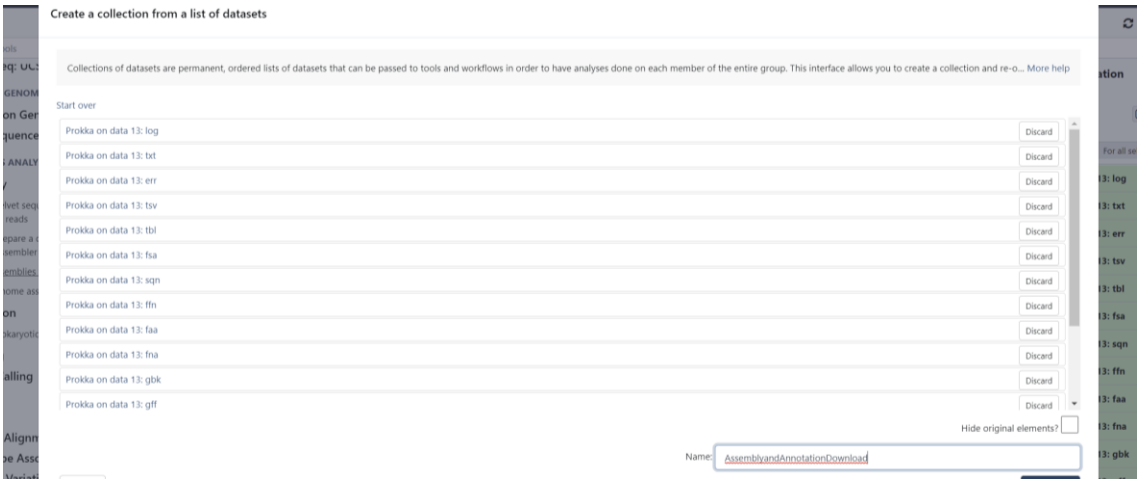
6.1.2 In the Galaxy History pane select the *Operations on multiple datasets* (checkbox) button. You'll see selectable checkboxes are now available for each history entry.

6.1.3 Select the following history entries to download:

6.1.4 From the *History pane* select the *For all selected…* dropdown menu and select **Build Dataset List**



6.1.5 Name the list **AssemblyandAnnotationDownload** before selecting *Create List*. You will now see a new entry in the *History pane* with the name of your newly created list.



6.2.0 Download your dataset list

Now that we've created a list of 16 items we can download it to your own desktop.

6.2.1 Go to the *History pane* and deselect the *Operations on multiple datasets* checkbox icon (if needed) and then select the **AssemblyandAnnotationDownload** entry. This will show the 15 items of your list.

6.2.2 From the *History pane* click on the *Download Collection* button (disk icon) and save the file to your **~/FGDS/Module2/downloads** folder. Notice that a single **.tgz** archive file will download to your directory.

6.2.3 Decompress the **AssemblyandAnnotationDownload.tgz** into **AssemblyandAnnotationDownload.tar** in your favourite program.

6.2.4 Unarchive **AssemblyandAnnotationDownload.tar** and place its contents into **~/FGDS/Module2/** and rename the files (and folders) as follows:

Download name	New name
FastQC on data 2_ Webpage.html	FastQC_RawOutput.html
FastQC on data 2_ Webpage	FastQC_RawOutput
FastQC on data 4_ Webpage.html	FastQC_Trimmed.html
FastQC on data 4_ Webpage	FastQC_Trimmed
Prokka on data x_ err.txt	HinfKW20_prokka.err
Prokka on data x_ faa.fasta	HinfKW20_prokka.faa
Prokka on data x_ ffn.fasta	HinfKW20_prokka.ffn
Prokka on data x_ fna.fasta	HinfKW20_prokka.fna
Prokka on data x_ fsa.fasta	HinfKW20_prokka.fsa
Prokka on data x_ gbk.txt	HinfKW20_prokka.gbk
Prokka on data x_ gff.gff	HinfKW20_prokka.gff
Prokka on data x_ log.txt	HinfKW20_prokka.log
Prokka on data x_ sqn.asn1	HinfKW20_prokka.sqn.asn1
Prokka on data x_ tbl.txt	HinfKW20_prokka.tbl
Prokka on data x_ tsv.tabular	HinfKW20_prokka_cds.tsv
Prokka on data x_ txt.txt	HinfKW20_prokka.txt
Quast: Alignment view.html	HinfKW20_quast_alignView.html
Quast: Contig view.html	HinfKW20_quast_contigView.html
Quast: Log.txt	HinfKW20_quast_log.txt
Quast: Misassemblies.tabular	HinfKW20_quast_misasseblies.tabular
Quast: Report (HTML).html	HinfKW20_quast_report.html

Quast: Report (PDF).pdf	HinfKW20_quast_report.pdf
Quast: Report (tabulal).tabular	HinfKW20_quast_report.tabular
Quast: Unaligned contigs.tabular	HinfKW20_quast_unalignedContigs.tabular
velvetg on data x_ Contigs.fasta	HinfKW20_denovo_assembly.fasta
velvetg on data x_ Stats.tabular	HinfKW20_denovo_assembly_stats.txt

6.3.0 Save your history as a workflow

Now that you've saved your files it's time to convert your history (which is also saved in Galaxy) into a workflow. A workflow is the equivalent of a bioinformatic pipeline – a series of commands that will run on your data to produce a final output.

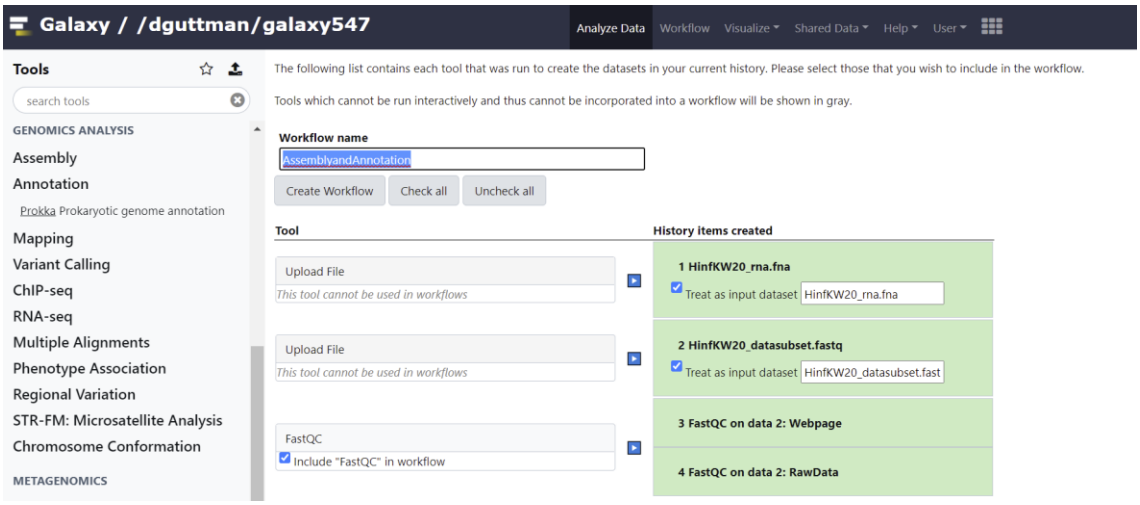
Saving workflows will allow you to perform the same analyses that we just performed on a completely different dataset. For example, if there was a second *H. influenzae* strain that you had sequenced, you could run the same analysis we just ran on the *H. influenzae* KW20 strain with the same parameters in a single step. This is important for both efficiency and reproducibility.

6.3.1 Go to the *History pane* and select the *History options* button.

6.3.2 Under the *CURRENT HISTORY* section select I

Notice that the Upload File tools include a description "This tool cannot be used in workflows". These will inevitably be removed from the workflow when we create it.

6.3.3 Simplify the *Workflow name* to **AssemblyandAnnotation**

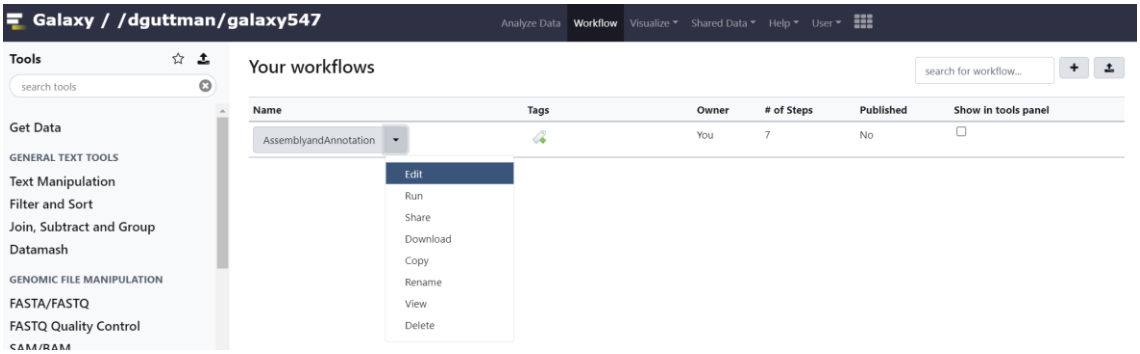


6.3.4 Select the *Create Workflow* button.

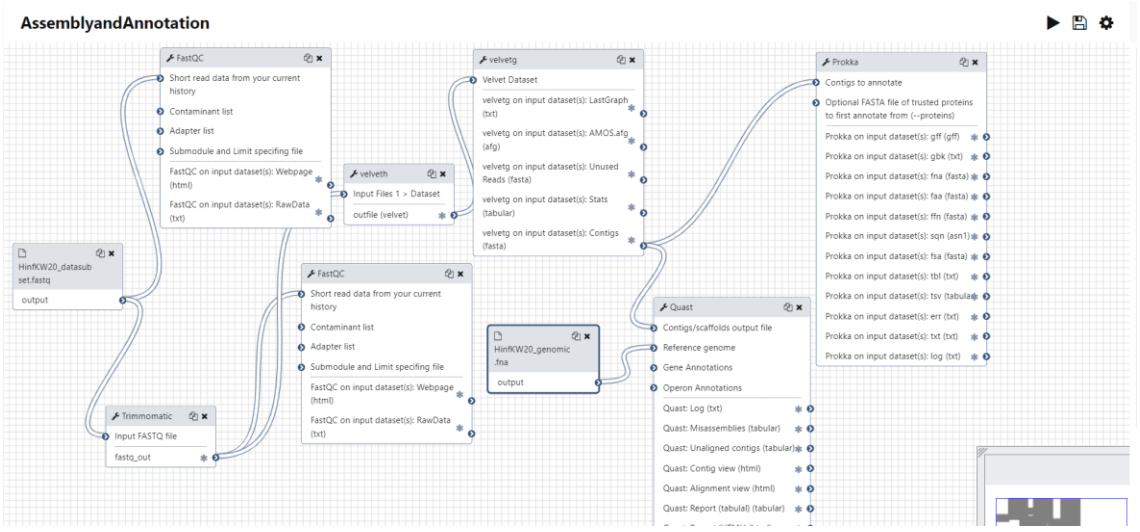
6.4.0 Editing your workflow

Now that a workflow has been created you can edit and rearrange portions as you see fit. You can even alter some of the parameters, add/remove steps, specify the files you want to include as output and resave the workflow or save it as a completely new one.

6.4.1 On the Galaxy *menu* select *Workflows*



6.4.2 On the Workflows page select the pulldown menu for *AssemblyandAnnotation* and select the [redacted]. This will load the workflow editor as follows.



6.4.3 Use the visual editor to re-arrange the steps into a clear set of analyses. This will make it easier to see how you might want to modify the input/output files. Notice how the tools are connecting the output of one with the input of another.

6.4.4 In the right-hand *Details pane* you can see the specific parameters set for each tool when you click on the individual tool information boxes.

Notice that in this pane you can also label and annotate the individual steps. This can be a useful way to comment on what you are doing and why.

6.4.5 Click on the

In the *Details pane* you can now see additional tags and annotations you can add to the overall workflow. These are helpful attributes to add if you are sharing your workflow with others as it will make it searchable and descriptive. You can also see previous saved versions of this workflow in the *Version* dropdown menu.

6.4.6 Use the *Save* icon (floppy disk) to save your workflow.

6.5.0 Run your saved workflow

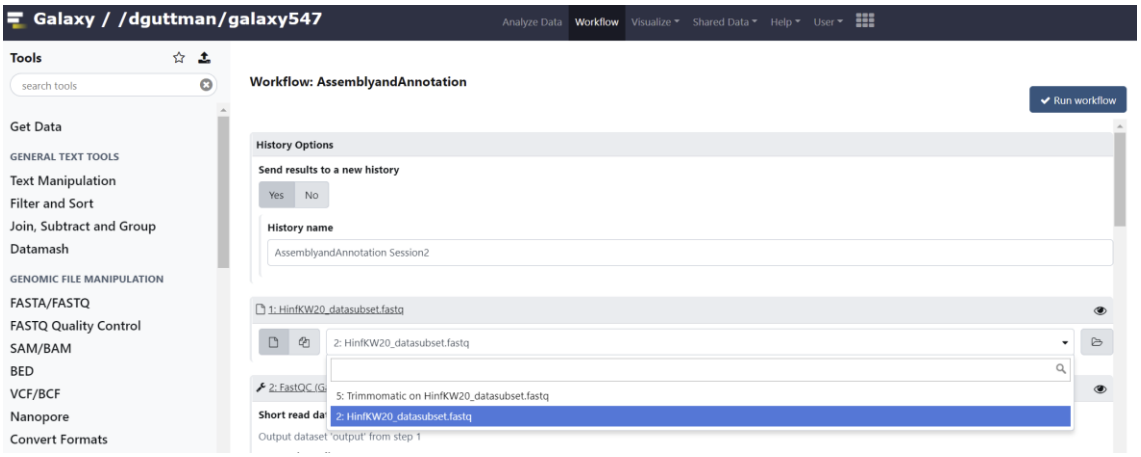
Now that we've saved our updated workflow, how do we apply this to another dataset? Recall that our saved workflow appears to already include a dataset.

6.5.1 On the Galaxy *menu* select *Workflows*

6.5.2 On the Workflows page select the pulldown menu for *AssemblyandAnnotation* and select the *Run* option. This will load the workflow in it's run order as follows.

6.5.3 Under the *History Options* section, decide whether you'd like the results of this run to go to your current history or a new one. If you choose **Yes**, then be sure to give your history a name.

6.5.4 *Step 1* will ask you to enter an initial file for analysis. The dropdown menu will include any available fastq files that are present in your current *History pane*. It is here that you will choose a set of fastq reads that are available in your history.



6.5.5 Double check the settings/parameters across your tool. You'll see that they can be edited individually but if you are saving this as a workflow, you'll want to keep them constant for consistency. If you're experimenting with your settings, you should note they will not be saved and you should experiment with these in the **Workflow Edit** page instead.

7.0.0 Class summary

That concludes our second lecture and introduction to working with the Galaxy platform. Next week we'll dig into some other tools on the platform to explore RNAseq. Altogether we've explored the following:

- Importing data files into Galaxy.
- Quality control analysis of raw and filtered reads with *FastQC*.
- Trimming and filter reads with *Trimmomatic*.
- Small genome assembly using *Velvet*.
- Quality analysis of genome assemblies using *QUAST*
- Prokaryotic genome annotation with *Prokka*.
- Downloading your datasets.
- Saving, editing, and running workflows based on your previous history.

7.1.0 Post-lecture assessment (10% of final grade)

Soon after this lecture, a homework assignment will be made available on Quercus in the assignment section. It will build on the ideas and/or data generated within this lecture. Each homework assignment will be worth 10% of your final mark. If you have assignment-related questions, please try the following steps in the order presented:

- Check the internet for a solution – read forums and learn to navigate for answers.
- Generate a discussion on Quercus outlining what you've tried so far and see if other students can contribute to a solution.
- Contact course teaching assistants or the instructor.

7.2.0 Suggested class preparation for Module 3

Next week we will begin exploring the Galaxy interface and learning how to perform quality control and genome assembly on data from sources like the SRA. To prepare for this, we suggest the following reading which will be available from the Module 2 section of Quercus:

- *Altmann et al., 2012, Human Genetics, 131:1541-1554.*

CAGEF University of Toronto Centre for the Analysis of Genome Evolution & Function