

---

# **Biohansel Documentation**

**Peter Kruczkiewicz**

**May 28, 2021**



<b>1</b>	<b>Usage</b>	<b>3</b>
1.1	Requirements and Dependencies . . . . .	3
1.2	Quick Installation . . . . .	3
1.3	With Conda . . . . .	3
1.4	With pip from PyPI . . . . .	4
1.5	With pip from Github . . . . .	4
1.6	Install into Galaxy (version >= 17.01) . . . . .	5
1.7	Input Data . . . . .	5
1.8	Output Results . . . . .	5
1.9	Parameters . . . . .	5
1.10	Running biohansel . . . . .	6
<b>2</b>	<b>Tutorial</b>	<b>7</b>
2.1	Testing: . . . . .	7
2.2	NML - Galaxy Access (biohansel) . . . . .	7
2.3	Running biohansel on Terminal (MAC) using Conda . . . . .	10
2.4	Verification Results . . . . .	14
<b>3</b>	<b>Input</b>	<b>17</b>
3.1	Types of Analysis . . . . .	17
3.2	Genotype Metadata Table (Optional) . . . . .	19
<b>4</b>	<b>Genotyping Schemes</b>	<b>21</b>
4.1	Heidelberg, Typhi, Typhimurium, and Enteritidis Genotyping Schemes . . . . .	21
4.2	<i>Mycobacterium tuberculosis</i> (TB) scheme: . . . . .	22
4.3	Genotype Classification System . . . . .	22
4.4	Creating a Genotyping Scheme . . . . .	23
<b>5</b>	<b>Degenerate Base Expansion</b>	<b>31</b>
5.1	Including Degenerate Bases in k-mers: . . . . .	31
5.2	Unchecked Expansion of K-mers: . . . . .	33
5.3	Benchmarking Degenerate Bases . . . . .	34
<b>6</b>	<b>Output</b>	<b>37</b>
6.1	<b>Tech Results.tab</b> . . . . .	37
6.2	<b>Match Results.tab</b> . . . . .	39
6.3	<b>Results.tab</b> . . . . .	41

6.4	<b>Quality_Control</b> . . . . .	44
<b>7</b>	<b>Galaxy Parameters</b>	<b>49</b>
7.1	Sequence Data Type . . . . .	49
7.2	SNP Subtyping Scheme . . . . .	50
7.3	Scheme Subtype Metadata Table (Optional) . . . . .	50
7.4	K-mer Frequency Thresholds . . . . .	50
7.5	Quality Checking Thresholds . . . . .	51
7.6	Developer Options . . . . .	53
<b>8</b>	<b>Command-Line</b>	<b>55</b>
8.1	<b>Required</b> . . . . .	55
8.2	Additional . . . . .	55
8.3	Hansel Help Command . . . . .	56
<b>9</b>	<b>Home</b>	<b>57</b>
<b>10</b>	<b>Command Line</b>	<b>59</b>
10.1	Biohansel on Miniconda Linux Installation Instructions . . . . .	59
10.2	Biohansel installation with pip from PyPI . . . . .	61
10.3	Biohansel installation with pip from Github . . . . .	62
<b>11</b>	<b>Galaxy</b>	<b>63</b>
11.1	Installation Instructions for New Galaxy Users . . . . .	63
11.2	Installation Instructions for Galaxy Admins . . . . .	65
<b>12</b>	<b>Versions</b>	<b>67</b>
12.1	Galaxy versions . . . . .	67
<b>13</b>	<b>Legal</b>	<b>69</b>
<b>14</b>	<b>Contact</b>	<b>71</b>

**Currently debating on changing Subtype to Genotype in all outputs and as such, they are synonymous for biohansel in the current version (2.4.0) and in this manual**

Biohansel genotypes clonal microbial whole-genome sequencing (WGS) data using SNV targeting k-mer genotyping schemes.

This tool works on genome assemblies (FASTA files) or reads (FASTQ files)! Accepts Gzipped FASTA/FASTQ files as input!

Biohansel includes 33 base-pair k-mer SNV genotyping schemes focused on *Salmonella enterica* subsp. *enterica* serovars. K-mers can be any length as long as they are a odd number.

Currently, there are subtyping schemes for the following *Salmonella* serovars:

- Heidelberg
- Enteritidis
- Typhimurium
- Typhi
  - Typhi scheme adapted from Wong et al paper titled: “An extended genotyping framework for *Salmonella enterica* serovar Typhi, the cause of human typhoid”.

These genotyping schemes have been created, maintained, and/or adapted by Genevieve Labbe et al.

There is also an included *Mycobacterium tuberculosis* scheme that was modified from the Francesc Coll et al. paper titled: “A robust SNP barcode for typing *Mycobacterium tuberculosis* complex strains” for biohansel use by Daniel Kein.

Code is available on GitHub under <https://github.com/phac-nml/biohansel>.

- *User Documentation*
- *Installation*
- *Legal*
- *Contact*



Biohansel genotypes clonal microbial whole-genome sequencing (WGS) data using single nucleotide variant (SNV) k-mer genotyping schemes.

SNV k-mer genotyping schemes included in biohansel are currently focused on *Salmonella enterica* enterica serovars and include SNV schemes for serovars Heidelberg, Enteritidis, Typhi, and Typhimurium developed by Genevieve Labbe et al. These schemes are based around 33-mer k-mer pairs using the SNP to distinguish the genotype.

There is also a genotyping scheme for *Mycobacterium tuberculosis* included in the latest version.

Biohansel can be installed with Conda, pip, or within an existing Galaxy infrastructure. View the [install guide](#) of your preference for additional details.

## 1.1 Requirements and Dependencies

This tool has only been tested on Linux (specifically Arch Linux). It may or may not work on OSX.

These are the dependencies required for biohansel

- **Python (>=v3.5)**
  - `numpy` >=1.12.1
  - `pandas` >=0.20.1
  - `pyahocorasick` >=1.1.6
  - `attrs`

## 1.2 Quick Installation

## 1.3 With Conda

Conda is the easiest way to install and run biohansel through the use of the command line.

First, install Conda (Conda installation instructions).

Then, install biohansel through Bioconda (64bit linux and MAC OSX) using the following commands:

```
# OPTIONAL: To create a new Conda environment input this command on terminal:
conda create -n "name of environment" python=3.6
# Then to create/activate conda environment: (*note* name of environment is what user
↳decides to name environment)
source activate "name of environment"

# You can then install biohansel in the new environment
# To deactivate environment, input:
source deactivate

# Setup Conda channels for Bioconda and Conda-Forge (https://bioconda.github.io/#set-
↳up-channels)
conda config --add channels defaults
conda config --add channels bioconda
conda config --add channels conda-forge

#Activate wanted Conda environment (base or user created)
conda activate

# Install biohansel
conda install bio_hansel

#Check installation with the following command; make sure to be in the correct
↳environment
hansel -h
#This will display the usage statement
```

Remember to activate the Conda environment that biohansel is installed into each time you want to run it after opening a new terminal window or you will find that the *hansel* command does not exist.

## 1.4 With pip from PyPI

Install biohansel from PyPI with pip:

```
pip install bio_hansel
```

This will install biohansel along with the required dependencies.

Check that installation is correct with the command:

```
hansel -h
#This will display the usage statement.
```

## 1.5 With pip from Github

Install the latest master branch version directly from Github:

```
pip install git+https://github.com/phac-nml/biohansel.git@master
```

Check that biohansel is working with the command:

```
hansel -h
#This will display the usage statement.
```

## 1.6 Install into Galaxy (version >= 17.01)

Galaxy admins install biohansel from the main Galaxy toolshed ([tutorial](#)):

<https://toolshed.g2.bx.psu.edu/view/nml/biohansel/ba6a0af656a6>

Users can download and set up their own instance of Galaxy following the [get Galaxy tutorial](#) and then install biohansel from the toolshed as an admin using the admin instructions linked above.

## 1.7 Input Data

Biohansel uses genome assemblies (FASTA files) or raw reads (FastQ files) from WGS data as an input. It also accepts these files as their Gzipped FASTA/FASTQ formats. Genomes can be fully assembled or a collection of contigs when analyzed without impacting the output.

SNV genotyping schemes have to be defined for biohansel to run correctly. Four schemes are currently included in biohansel and user created schemes can be developed by creating SNV k-mer pairs in the specified FASTA format used by biohansel. See [Creating schemes](#) for more details.

Genotype metadata schemes can be optionally added to the analysis using the `-M` argument and then specifying a tab delimited file in `.tsv` format. The added metadata is then joined with the genotype/subtype field of the final results. More detailed info on formatting of metadata schemes can be found in the [Input section](#) along with additional information on all of the other input files biohansel can use.

## 1.8 Output Results

Output of the results generated through biohansel will be found in three `.tab` files in the directory that biohansel was run from or in the Galaxy histories window after analysis is complete. The three output files include:

- `tech_results.tab` → Most basic results file giving the genotype and sample coverage (fastq samples)
- `results.tab` → More advanced information on the results generated including how many k-mers were found and what types.
- `match_results.tab` → All k-mer information used to generate the genotype result with the positive kmers First

All outputs contain a quality control (QC) column along with a “qc\_message” column that runs through qc checks to determine if the data is consistent or has any conflicting results that the user should be aware of.

Detailed info about the results outputs and QC can be found in the [output section](#).

## 1.9 Parameters

Parameters can be modified for users of both Galaxy and the command line. These can be changed based on the users need. Modifiable parameters include:

- **K-mer Frequency Thresholds - only apply to raw reads/fastq datasets**
  - Min k-mer frequency/coverage (default 8, cannot lower past 8 in current build)

- Max k-mer frequency/coverage (default 1000)
- **Quality Checking Thresholds - Important parameters for the final results of the QC columns**
  - QC: Frequency below this coverage are considered low coverage (default 20)
  - QC: Min number of k-mers missing for Ambiguous Result (default 3)
  - QC: Decimal Proportion of max allowed missing k-mers (default 0.05)
  - QC: Decimal Proportion of max allowed missing k-mers for an intermediate genotype (default 0.05)
  - QC: Overall k-mer coverage below this value will trigger a low coverage warning (default 20)
- **Command Line Only - Parameters only available on the command line so as to not risk overworking shared Galaxy instances**
  - Max degenerate kmers before program stops to warn you of the dangers of too many kmers (default **WIP**)

Detailed info on biohansels parameters and their functions can be found in the [parameter section](#) or the [command line section](#).

## 1.10 Running biohansel

More detailed information is available under the [Tutorial section](#), the [input section](#), or the [Command Line section](#).

A basic command to run biohansel on an assembled Heidelberg fasta file would be:

```
hansel -s heidelberg -vv -o results.tab -O match_results.tab -S tech_results.tab </  
↪path/to/data_file>
```

This tutorial is for demonstrating how to run biohansel and to confirm that it is installed correctly. For step by step installation instructions view [installation home](#) or for quick instructions view [quick install](#).

### 2.1 Testing:

This page can be used to verify that biohansel is working properly with the different interfaces (command line/Galaxy):

To follow along:

Download either the CP012921.fasta (fasta file) or the SRR2598330(fastq-dump).fastqsanger.gz (raw file) which are the same samples in different formats:

<https://share.corefacility.ca/index.php/s/dRGOuqhDJUNeKmE> (password: biohansel)

CP012921.fasta - is an assembled sequence.

SRR2598330 (fastq-dump).fastqsanger.gz - is the raw reads file that CP012921 was assembled from.

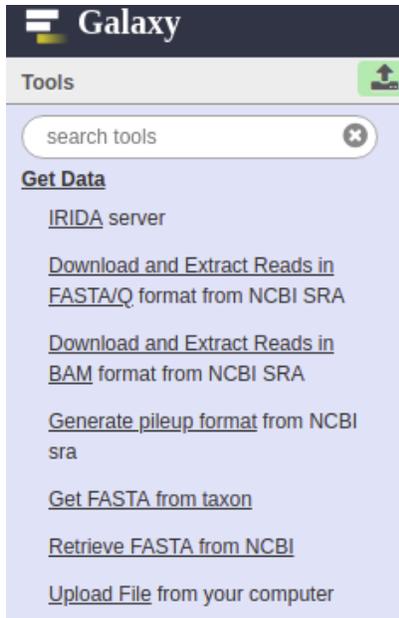
Specific steps regarding the testing/checking that installation was successful using the sample data found above will be found within the (brackets) below each general step involved in the analysis pipeline.

**Both example files are from the same Salmonella Heidelberg sample**

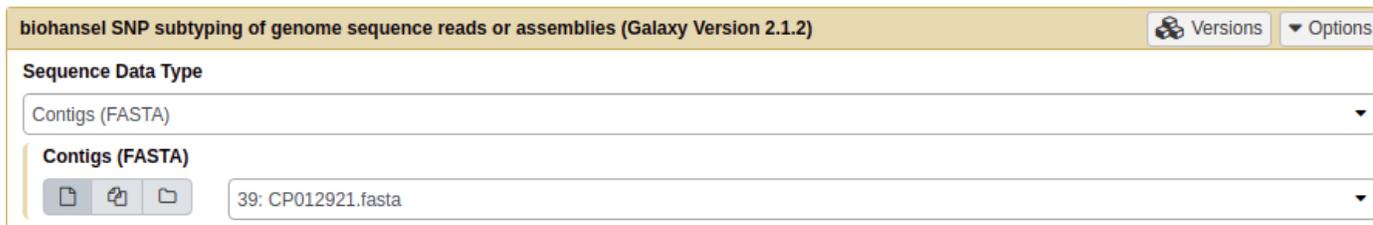
### 2.2 NML - Galaxy Access (biohansel)

1. Create a new history in Galaxy and import either fasta files or raw reads files that are being analyzed into the newly created history using any of the tools from the Get Data “Tools” section.

(For verification: find the sample data in your files, most likely in Downloads, and upload it into Galaxy using the upload file tool.)



2. Find biohansel on the right-hand side in the “Tools” Section: Under the Experimental Section and click it.
3. For the “Sequence Data Type” parameter, select the proper type of data (FASTA vs. FASTQ (raw)) depending upon the type of data being analyzed  
 (For Verification: If you are using the Fasta file select the fasta option. If you are using the FastQ file, select the paired-end reads data type.)



4. For the “SNP Subtyping Scheme”, select the proper scheme corresponding to the organism in your samples. This can currently be the included *Salmonella* serovars Heidelberg, Enteritidis, Typhimurium, and Typhi schemes, or the *M. tuberculosis* scheme, or a user created FASTA file  
 (For verification: select the “Salmonella Heidelberg subtype scheme”)



5. Optionally, add a Scheme Subtype Metadata Table can be added to the analysis to be included into the end of the results files. This file **must be in the .tsv format** to be added properly or the analysis may fail (.csv may work on Galaxy but if the analysis fails, change it to .tsv).

**Scheme Subtype Metadata Table [Optional]**

47: meta.tsv

CSV or tab-delimited format only. Must contain a 'subtype' column.



6. Click on the eye (  ) to expand or collapse the modifiable parameters to allow adjustments to them to suit your needs. The defaults work well for most analyses but in some situations it may be beneficial to change them. Detailed information on the parameters that Galaxy allows modification to and what they do can be found in the [parameters section](#)

(For verification: leave all of the parameters as their defaults.)

**K-mer Frequency Thresholds**

**Min k-mer frequency/coverage**

  
default = 8 (--min-kmer-freq)

**Max k-mer frequency/coverage**

  
default = 1000 (--max-kmer-freq)

**Quality Checking Thresholds**

**QC: Frequency below this coverage are considered low coverage**

  
default = 20 (--low-cov-depth-freq)

**QC: Min number of tiles missing for Ambiguous Result**

  
default = 3 (--min-ambiguous-tiles)

**QC: Decimal Proportion of max allowed missing tiles**

   
default = 0.05, valid values {0.0 - 1.0} (--max-missing-tiles)

**QC: Decimal Proportion of max allowed missing tiles for an intermediate subtype**

   
default = 0.05, valid values {0.0 - 1.0} (--max-intermediate-tiles)

**QC: Overall tile coverage below this value will trigger a low coverage warning**

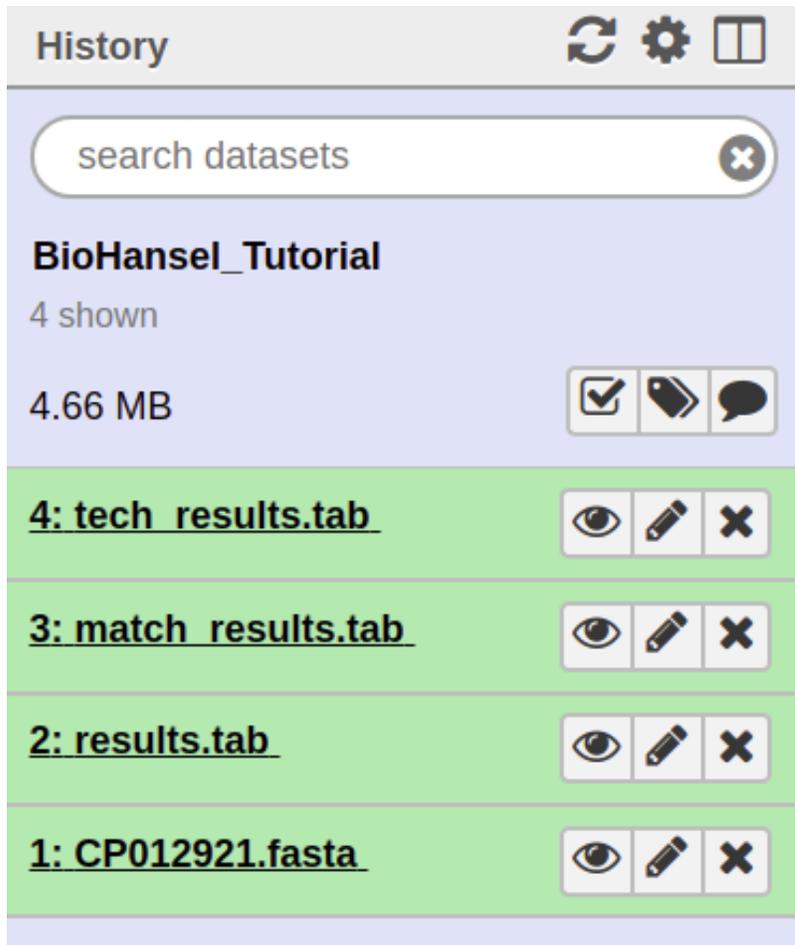
  
default = 20 (--low-cov-warning)

7. Execute your inputs and analyze your data by clicking the large execute button to produce three results files: tech\_results.tab, match\_results.tab and results.tab.

The normal execution time is 0.5 - 60 seconds depending on what the inputs were. Don't be alarmed if it does take longer as it depends on the available computing power and the size of the dataset given (especially on Galaxy).

(For verification: verify that the output was correct by comparing to the [Verification Results](#).)

8. The .tab files can be opened in excel or another spreadsheet program to view the results of the analysis. On Galaxy, the results can be looked at by clicking on the view data eye in the history section.



\*\*For more detailed information on the different types of outputs that are produced by biohansel go to: [Output](#)

## 2.3 Running biohansel on Terminal (MAC) using Conda

### 2.3.1 Steps

1. Go to [Quick Installation instructions](#) or [Full installation instructions](#) (exact same with different details to them) and download Miniconda from the website following the instructions corresponding to your given iOS.

**Skip to step 6 if you have already installed biohansel.**

2. After installing Conda, go on terminal and create a conda environment by inputing this command:

```
conda create -n <name of environment> python=3.6

# For example to create an environment called biohansel the command would be:
conda create -n biohansel python=3.6
```

3. It will ask you to proceed (y/n) afterwards, type in: y

4. Then activate your environment by typing:

```
source activate <name of your environment>

# If you called your environment biohansel, the activate command would be:
source activate biohansel

# Source activate will activate the environment. You know that it is active,
↳if you see
# the environment name beside your name.
```

5. Now install biohansel onto conda environment by inputting:

```
conda install bio_hansel

# Make sure to always activate the environment that biohansel was installed,
↳into
# otherwise it will not run.
```

6. To confirm that biohansel has been installed in the environment, input:

```
hansel -h
#this command shows the numerous types of commands you can use in for,
↳biohansel

# If there is any issue confirm that, if using conda, you are in the correct,
↳environment
# that you installed biohansel to.

# If you installed biohansel with pip and are having issues; confirm that,
↳biohansel
# was installed into the correct python (Should be python 3.6 or higher),
↳with the command:
# which python
```

Additional troubleshooting can be found in the [installation page](#). Go to [command-line](#) to see detailed descriptions of all of the arguments that can be used to run biohansel. Quick descriptions of the *arguments* are found below.

```
(biohansel) dhole@phac5018125:~$ hansel -h
usage: hansel [-h] [-s SCHEME] [--scheme-name SCHEME_NAME]
             [-M SCHEME_METADATA] [-p forward_reads reverse_reads]
             [-i fasta_path genome_name] [-D INPUT_DIRECTORY]
             [-o OUTPUT_SUMMARY] [-O OUTPUT_KMER_RESULTS]
             [-S OUTPUT_SIMPLE_SUMMARY] [--force] [--json]
             [--min-kmer-freq MIN_KMER_FREQ] [--max-kmer-freq MAX_KMER_FREQ]
             [--low-cov-depth-freq LOW_COV_DEPTH_FREQ]
             [--max-missing-kmers MAX_MISSING_KMERS]
             [--min-ambiguous-kmers MIN_AMBIGUOUS_KMERS]
             [--low-cov-warning LOW_COV_WARNING]
             [--max-intermediate-kmers MAX_INTERMEDIATE_KMERS]
             [--max-degenerate-kmers MAX_DEGENERATE_KMERS] [-t THREADS] [-v]
             [-V]
             [F [F ...]]

Subtype microbial genomes using SNV targeting k-mer subtyping schemes.
Includes schemes for Salmonella enterica spp. enterica serovar Heidelberg, Enteritidis, Typhi, and Typhimurium subtyping. Also includes a Mycobacterium Tuberculosis scheme.
Developed by Genevieve Labbe, James Robertson, Peter Kruczkiewicz, Marisa Rankin, Matthew Gopez, Chad R. Laing, Philip Mabon, Kim Ziebell, Aleisha R. Reimer, Lorelee Tschetter, Gary V.
Parnley, David Son, Darian Hole, Philip Mabon, Elissa Giang, Lok Kan Lee, Jonathan Moffat, Marisa Rankin, Joanne MacKinnon, Roger Johnson, John H.E. Nash.

positional arguments:
  F                    Input genome FASTA/FASTQ files (can be Gzipped)

optional arguments:
  -h, --help            show this help message and exit
  -s SCHEME, --scheme SCHEME
                        Scheme to use for subtyping (built-in: "heidelberg",
                        "enteritidis", "typhi", "typhimurium",
                        "tb_speciation"; OR user-specified:
                        /path/to/user/scheme)
  --scheme-name SCHEME_NAME
                        Custom user-specified SNP subtyping scheme name
  -M SCHEME_METADATA, --scheme-metadata SCHEME_METADATA
                        Scheme subtype metadata table (CSV or tab-delimited
                        format; must contain "subtype" column)
  -p forward_reads reverse_reads, --paired-reads forward_reads reverse_reads
                        FASTQ paired-end reads
  -i fasta_path genome_name, --input-fasta-genome-name fasta_path genome_name
                        fasta file path to genome name pair
  -D INPUT_DIRECTORY, --input-directory INPUT_DIRECTORY
                        directory of input fasta files (.fasta|.fa|.fna) or
                        FASTQ files (paired FASTQ should have same basename
                        with "_d.(fastq|fq)" postfix to be automatically
                        paired) (files can be Gzipped)
  -o OUTPUT_SUMMARY, --output-summary OUTPUT_SUMMARY
                        Subtyping summary output path (tab-delimited)
  -O OUTPUT_KMER_RESULTS, --output-kmer-results OUTPUT_KMER_RESULTS
                        Subtyping kmer matching output path (tab-delimited)
  -S OUTPUT_SIMPLE_SUMMARY, --output-simple-summary OUTPUT_SIMPLE_SUMMARY
                        Subtyping simple summary output path
  --force               Force existing output files to be overwritten
  --json                Output JSON representation of output files
  --min-kmer-freq MIN_KMER_FREQ
                        Min k-mer freq/coverage
  --max-kmer-freq MAX_KMER_FREQ
                        Max k-mer freq/coverage
  --low-cov-depth-freq LOW_COV_DEPTH_FREQ
                        Frequencies below this coverage are considered low
                        coverage
  --max-missing-kmers MAX_MISSING_KMERS
                        Decimal proportion of maximum allowable missing kmers
                        before being considered an error. (0.0 - 1.0)
  --min-ambiguous-kmers MIN_AMBIGUOUS_KMERS
                        Minimum number of missing kmers to be considered an
                        ambiguous result
  --low-cov-warning LOW_COV_WARNING
                        Overall kmer coverage below this value will trigger a
                        low coverage warning
  --max-intermediate-kmers MAX_INTERMEDIATE_KMERS
                        Decimal proportion of maximum allowable missing kmers
                        to be considered an intermediate subtype. (0.0 - 1.0)
  --max-degenerate-kmers MAX_DEGENERATE_KMERS
                        Maximum number of scheme k-mers allowed before
                        quitting with a usage warning. Default is 100000
  -t THREADS, --threads THREADS
                        Number of parallel threads to run analysis (default=1)
  -v, --verbose         Logging verbosity level (-v == show warnings; -vv ==
                        show debug info)
  -V, --version         show program's version number and exit
(biohansel) dhole@phac5018125:~$
```

Please click the image to view the command

7. Figure out what directory you are in (which is most likely User/"name of user") using the following command:

```
pwd
# pwd = "print working directory" and will show which directory you are
↳ currently in.
```

8. Using the terminal window, change directories to the directory/folder that contains the data that you want to analyze. This can be done with the following command:

```
cd <path/to/file>
```

Example: if the file was in User/name of user/Downloads you input:

```
cd User/name of user/Downloads
# cd = change directory command
```

9. Once you're in the directory where your data is stored, biohansel can be used to analyze the data in the directory using the following command:

```
hansel -s heidelberg -vv -o results.tab -O match_results.tab -S tech_results.
↳tab <Name of data file>

# If you downloaded the CP012921.fasta, then you would input CP012921.fasta_
↳at the end of the command
# If working with the raw fastq data, you may need to unzip the file to get_
↳the analysis to work.
# To do this use the following g-zip command without the # before it:
# gzip -d <file>
```

## 2.3.2 Arguments

The other arguments needed to run the command are as follows:

-s -> this command is to specify the scheme used by biohansel for the analysis being done (“enteritidis”, “heidelberg”, “typhi”, “typhimurium”, and “tb\_lineage” are the built in schemes right now). You can also use this to specify a custom scheme and then the path to that scheme.

-vv -> this command is used to display more information from the terminal while the command is running. It is not necessary for an analysis but can be extremely useful. Can be added as just -v to show warnings.

-t or --threads <#\_CPUs> -> the number of parallel threads to run analysis

-o -> this command is used to specify the main results file output by biohansel called results.tab (You can change the name to whatever you want. **Remember to add .tab**)

-O -> this command is used to get the more detailed results output known as match\_results.tab (You can change it to whatever name you want. **Remember to add the .tab**)

-S -> this command is used to output the simplest results file generated by biohansel called tech\_results.tab (You can change name to whatever you want. **Remember to add .tab**)

You do not need all of the arguments shown to run the command. You may only choose to look at one or two of the three output files and as such can leave off the file you do not wish to create to save computing power.

After you have defined all of the necessary arguments, input the name of the file at the end of the command and press enter to start the analysis. Normal analysis times will take anywhere from 0.5 - 60 seconds depending on the file size and previous assembly of contigs (if any).

10. The result files that you specified with the correct arguments and their names should be in the directory that the command was run from. For example if you ran the command from a directory called “data”, the results would be located in the “data” directory.

If running the example files, verify that the output was correct by comparing to the [Verification Results](#) tables.

## 2.4 Verification Results

For CP012921.fasta (fasta file):

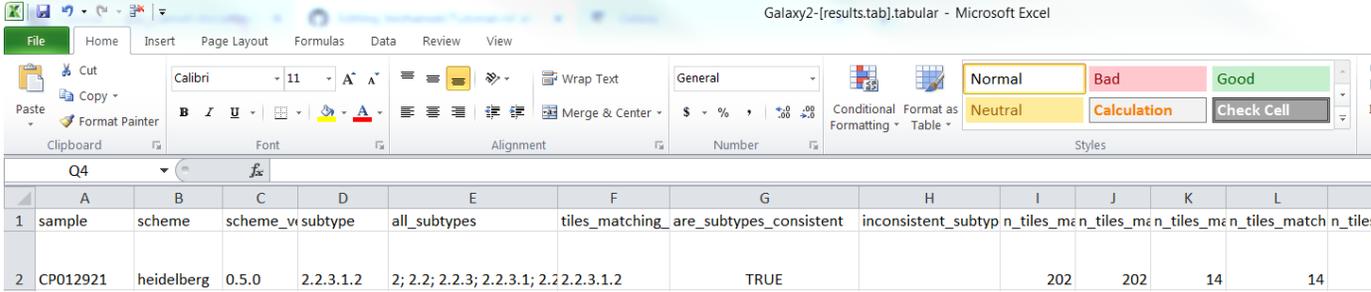
*Fasta match\_result.tab:*

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	tilename	seq	is_revcom	contig_id	match_ind	reposition	subtype	is_pos_tile	sample	file_path	scheme	scheme_ver	qc_status	qc_message
2	negative2	ACTGCCGC	FALSE	CP012921.	2996	2981	2.2.3.1.4	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
3	negative2	GCAAATCC	FALSE	CP012921.	21112	21097	2.2.1.1.1	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
4	negative4	CCTGAAA	FALSE	CP012921.	42247	42232	2.2.2.2.2	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
5	negative6	CGCATGG	FALSE	CP012921.	64724	64709	2.1.3	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
6	negative9	TGCAGTG	FALSE	CP012921.	92681	92666	1.3	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
7	negative1	TAGCGTTT	FALSE	CP012921.	107816	107801	2.2.1.2	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
8	negative1	ACGGTACT	FALSE	CP012921.	157807	157792	2.2.1.1.1	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
9	negative1	CAGTATCT	FALSE	CP012921.	160381	160366	2.2.1.2	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
10	negative1	TACGATCC	FALSE	CP012921.	167035	167020	2.2.2.1.1.1	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
11	negative1	AATAGCGA	FALSE	CP012921.	176723	176708	2.2.2.2.6	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
12	negative1	GTACCGTC	FALSE	CP012921.	198413	198398	2.2.2.1.1.1	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
13	negative2	CTGGAAG	FALSE	CP012921.	202016	202001	1.1	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
14	negative2	CTGTTCCG	FALSE	CP012921.	205853	205838	2.1.1.1.1.1	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
15	negative2	TCCCGGCC	FALSE	CP012921.	227846	227831	2.2.3.2.1	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
16	negative2	TTCCTGTC	FALSE	CP012921.	239167	239152	2.2.2.2.2.1	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
17	negative2	TCGGGTGC	FALSE	CP012921.	271450	271435	2.2.1.1.3	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
18	negative2	GTTGAAAC	FALSE	CP012921.	293743	293728	2.2.1.1	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
19	467662-2.	CTAACTTA	FALSE	CP012921.	467677	467662	2.2.3.1.2	TRUE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
20	negative4	CGGGAATC	FALSE	CP012921.	485008	484993	2.2.2.2.1.2	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
21	negative4	CCTTGTCG	FALSE	CP012921.	489702	489687	2.2.3.2.1	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
22	negative5	CCTGGTGA	FALSE	CP012921.	508774	508759	2.2.2.2.1.3	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
23	negative5	ACCACAAC	FALSE	CP012921.	573274	573259	2.2.2.2.5	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
24	negative6	GTCGGAA	FALSE	CP012921.	600365	600350	2.1.2	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
25	negative6	AGCGGCG	FALSE	CP012921.	600798	600783	1.1	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
26	negative6	TCCAGCGC	FALSE	CP012921.	607453	607438	2.2.1	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	
27	negative6	GGAGCCG	FALSE	CP012921.	650003	649988	2.2.2.1	FALSE	CP012921	/Drives/P/	heidelberg	0.5.0	PASS	

*Fasta tech\_result.tab:*

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	sample	subtype	qc_status	qc_message										
2	CP012921	2.2.3.1.2	PASS											

*Fasta result.tab:*



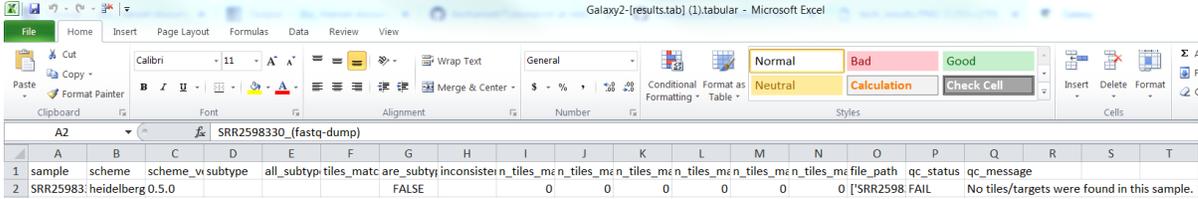
	A	B	C	D	E	F	G	H	I	J	K	L
1	sample	scheme	scheme_v	subtype	all_subtypes	tiles_matching_	are_subtypes_consistent	inconsistent_subtyp	n_tiles_m	n_tiles_m	n_tiles_m	n_tiles_match
2	CP012921	heidelberg	0.5.0	2.2.3.1.2	2; 2.2; 2.2.3; 2.2.3.1; 2.2	2.2.3.1.2	TRUE		202	202	14	14

**For SRR2598330(fastq-dump).fastqsanger.gz (raw file):**

*Raw/FASTQ match\_result.tab:*

*Raw/FASTQ tech\_result.tab:*

*Raw/FASTQ result.tab:*



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	sample	scheme	scheme_v	subtype	all_subtyp	tiles_matc	are_subty	inconsisten	n_tiles_m	file_path	qc_status	qc_message								
2	SRR25983	heidelberg	0.5.0				FALSE		0	0	0	0	0	0	0	[SRR2598	FAIL	No tiles/targets were found in this sample.		



This section describes the main input files that are needed to run biohansel along with the various methods available to analyze datasets using the tool.

The three input files are:

- The WGS data/assembled genome added as a FASTQ or FASTA file
- A chosen genotyping scheme (heidelberg, enteritidis, typhimurium, typhi, or tb\_lineage) or a user-created custom genotyping scheme (FASTA).

More info in the [genotyping schemes section](#).

- A metadata table (**Optional**) in CSV or TSV (tab-delimited) format to add additional information to the results. A **.tsv file is highly recommended** as csv files are currently unstable

More detailed information on the output of what each results file contains can be found in the [Output section](#).

## 3.1 Types of Analysis

### 3.1.1 Analysis of a single FASTA file

Analysis of a single FASTA file would be run on a sample that has already been assembled into contigs through another program/tool. To run the command, the arguments that must be specified include:

- -s “scheme” where the scheme defined can be one of the five built in schemes or a user created one (FASTA format)
- Any combination of the results delimiters and names (file names can be changed but must be included after the argument):
  - -o results.tab
  - -O detailed\_results.tab
  - -S simple\_summary.tab

- The name of the FASTA file at the end of the command

An example command for the analysis of a single FASTA file called SRR1002850.fasta would look like:

```
hansel -s heidelberg -vv -o results.tab -O detailed_results.tab /path/to/  
↪SRR1002850.fasta
```

Or, if you have already changed to the directory containing the dataset, you can use the following command where you do not have to specify the path to the data:

```
hansel -s heidelberg -vv -o results.tab -O detailed_results.tab SRR1002850.  
↪fasta
```

The output of the biohansel tool can be found in the directory that the command was run from.

### 3.1.2 Analysis of a single FASTQ readset

Analysis of a single FASTQ readset would be run on raw sequencing data. To run the command, the arguments that must be specified include:

- -s “scheme” where the scheme defined can be one of the five built in schemes or a user created one (FASTA format)
- Any combination of the results delimiters and names (file names can be changed but must be included after the argument):
  - -o results.tab
  - -O detailed\_results.tab
  - -S simple\_summary.tab
- **The name of the FASTQ file(s) at the end of the command**
  - For single-end reads include the one file
  - For paired-end reads include: -p followed by both files one after the other

An example command for the analysis of a single single-end reads run dataset would look like:

```
hansel -s heidelberg -vv -t 4 -o results.tab -O detailed_results.tab  
↪SRR5646583.fastqsanger
```

An example command for the analysis of a single paired-end reads run dataset would look like:

```
hansel -s heidelberg -vv -t 4 -o results.tab -O detailed_results.tab -p  
↪SRR5646583_forward.fastqsanger SRR5646583_reverse.fastqsanger
```

### 3.1.3 Analysis of all FASTA/FASTQ files in a directory

Analysis on **all** of the FASTA/FASTQ files in the specified directory. This will run on all FASTA/FASTQ files in the directory. Be sure that there are no miscellaneous files that may unnecessarily increase analysis time or lead to unneeded errors.

Biohansel will only attempt to analyze the FASTA/FASTQ files within the specified directory and will not descend into any subdirectories! As such, make sure all of the data to be analyzed is in the same location or organized in a way that suits the project.

Analysis of all of the sequencing files in a directory must include following the arguments to run properly:

- -s “scheme” where the scheme defined can be one of the five built in schemes or a user created one (FASTA format)
- Any combination of the results delimiters and names (file names can be changed but must be included after the argument):
  - -o results.tab
  - -O detailed\_results.tab
  - -S simple\_summary.tab
- -D /path/to/directory\_with\_data

Optionally, you are able to specify the number of threads for an analysis with the `--threads` argument. If you do not specify this, it will default to 1.

- `--threads <#_cpu>` to specify the number of CPUs wanted to run the analysis.

An example of a general command for the analysis of a directory of FASTA/FASTQ files:

```
hansel -s heidelberg -vv --threads <n_cpu> -o results.tab -O detailed_
↳results.tab -D /path/to/fastas_or_fastqs/
```

The chosen output files can be found in the directory that the command was run from or that was specified in the output names and it will contain data from each of the analyzed files run by biohansel.

Ex. If I was running an analysis on samples stored in my “data” directory found in the path `science/user/data`, I could `cd` to my user folder and run the following command:

```
hansel -s heidelberg -vv --threads 1 -o results.tab -O detailed_results.tab -
↳D data/
```

## 3.2 Genotype Metadata Table (Optional)

Optionally you can select a genotype metadata information table to include genotype metadata along with the genotyping results created with biohansel. Metadata tables must be in a tab-delimited format to correctly work. The file extension for your metadata table should be `.tsv` if at all possible or you may end up with an error and no analysis results.

To add a metadata table to the analysis you will add the argument `-M <metadata_scheme.tsv>` to any other analysis command. There are no requirements for the number of columns or the content of each of the columns on the metadata table so long as the **first column** is labeled as “**subtype**”.

A command that incorporates the `-M` command for analysis would be structured following the previously established requirements and looks as follows:

```
hansel -s heidelberg -M <metadata_scheme.tsv> -vv -o results.tab -O detailed_
↳results.tab <data>
```

The biohansel results table will be joined with the genotype metadata table based if a genotype on the metadata table matches one on the results. If a match occurs, the metadata of that genotype will be added to the table at the end of the `results.tab` and `tech_results.tab` results files.

Example metadata table (called `metadata.tsv`):

subtype	Clade	Source	Symptoms
1	I	Geese	Death
1.1	I	Moose	Burns
2.2.1.1.1	II	Mouse	Boils
2.2.2.2.2.1	IIa	Human	Rash

**\*When naming a metadata table make sure there are no spaces or parentheses and that its extension is .tsv or the analysis may fail.\***

The added metadata will appear at the end of the results.tab and the simple\_summary.tab files.

Example: simple\_summary.tab without metadata added:

sample	subtype	qc_status	qc_message
CP012921	2.2.3.1.2	PASS	

Example: simple\_summary.tab with metadata:

sample	subtype	qc_status	qc_message	Clade	Source	Symptoms
CP012921	2.2.3.1.2	PASS		I	Geese	Rash

You can add metadata to the analysis with Galaxy by uploading either a .tsv or a .csv file to your history and specifying that you want it used in the analysis. **A .tsv file is recommended.**

---

## Genotyping Schemes

---

This section will cover the genotyping (**previously called subtyping**) schemes used by biohansel for *Salmonella enterica* subspecies enterica serovar Heidelberg, Typhimurium, Typhi, and Enteritidis. Biohansel also includes a genotyping scheme for *Mycobacterium tuberculosis*. Along with these 4 schemes included in biohansel, this section will provide you with in depth information on how to create a custom genotyping scheme.

The genotyping schemes developed and used by biohansel are specifically designed fasta files that contain many k-mer pairs (split k-mers) of the same length. These k-mer pairs are given a positive (e.g. inclusive) or negative (e.g. exclusive) label for the genotype that they correspond to, allowing analysis to occur through biohansel to determine what the samples genotype is based on the scheme. If you want to see the exact structure, you can click on [K-mer\\_Structure](#) for the exact formatting. k-mers must be formatted this way for biohansel to run correctly. Depending upon which of these k-mers match the target, the final genotype will be obtained.

The k-mer genotyping process works due to the clonal (very little genomic change/evolution occurs over time) nature of the serovars found in *Salmonella enterica* or other clonal pathogens. This clonal nature allows SNPs to be mapped to different genotypes that evolved from different lineages over the expanse of a few years (e.g. recent clonal expansions). *Salmonella* serovars are good candidates for this type of genotyping as it is hard to determine a genotype in the lab through conventional means due to all genotypes being genetically similar within the most prevalent *Salmonella* serovars.

This process can be used to genotype other clonal pathogens with biohansel as soon as a genotyping scheme is created and validated for them.

### 4.1 Heidelberg, Typhi, Typhimurium, and Enteritidis Genotyping Schemes

The **Heidelberg** genotyping scheme included with biohansel is in version 0.5.0 and features a set of 202 33-mer pairs with a single nucleotide polymorphism (SNP) distinguishing between the positive and negative condition in of each pair. This distinction between pairs allows for the identification and classification

of different genotypes of Heidelberg serovars based on the number and location of SNPs in a WGS sample that match to the genotyping schemes pairs.

The **Enteritidis** scheme (version 1.0.7) features a similar set of 317 33-mer pairs that follow the same style as the Heidelberg scheme to classify and identify different enteritidis serovar genotypes. It specifies one of 117 genotypes with its scheme.

The **Typhi** genotyping scheme (version 1.3.0) is a new scheme that follows a similar structure to the other two schemes. It features a classification list of 75 k-mer pairs which define 75 possible genotypes. This scheme was adapted by Geneviève Labbé from a publication by Vanessa Wong et al. titled: “[An extended genotyping framework for Salmonella enterica serovar Typhi, the cause of human typhoid](#)”, and more recent publications by Britto et al., Rahman et al., and Klemm et al. When running the Typhi scheme, the previously published hierarchical codes are automatically appended to the results to allow better comparison. In addition to the published genotyping codes, metadata relevant to each genotype is provided to the user based on an analysis of ~8,100 public *S. Typhi* datasets from the NCBI SRA (manuscript in preparation). The genotype metadata includes the number of WGS datasets found to be part of this genotype out of 5,088 *S. Typhi* isolates (under the header “count”) which passed QC and had sufficient source data provided; as well as the earliest year and latest year in which a sample corresponding to this genotype was collected (under the headers “earliest\_year\_collected” and “latest\_year\_collected”), and the most predominant geographic source for this genotype (under the header “top\_geo\_loc\_name”).

The **Typhimurium** genotyping scheme (version 0.5.5) is a scheme that features 430 k-mer pairs and follows the same characteristics of the other schemes above. This scheme is more robust at detecting SNPs and contamination as it has more k-mers for each genotype leading to it being a more specific and streamlined scheme. This scheme specifies for 120 different genotypes.

Changes to schemes occur as new classifications are made and then the schemes are updated on GitHub and Galaxy for the use of others.

The Heidelberg, Enteritidis and Typhimurium schemes were developed by Geneviève Labbé et al.

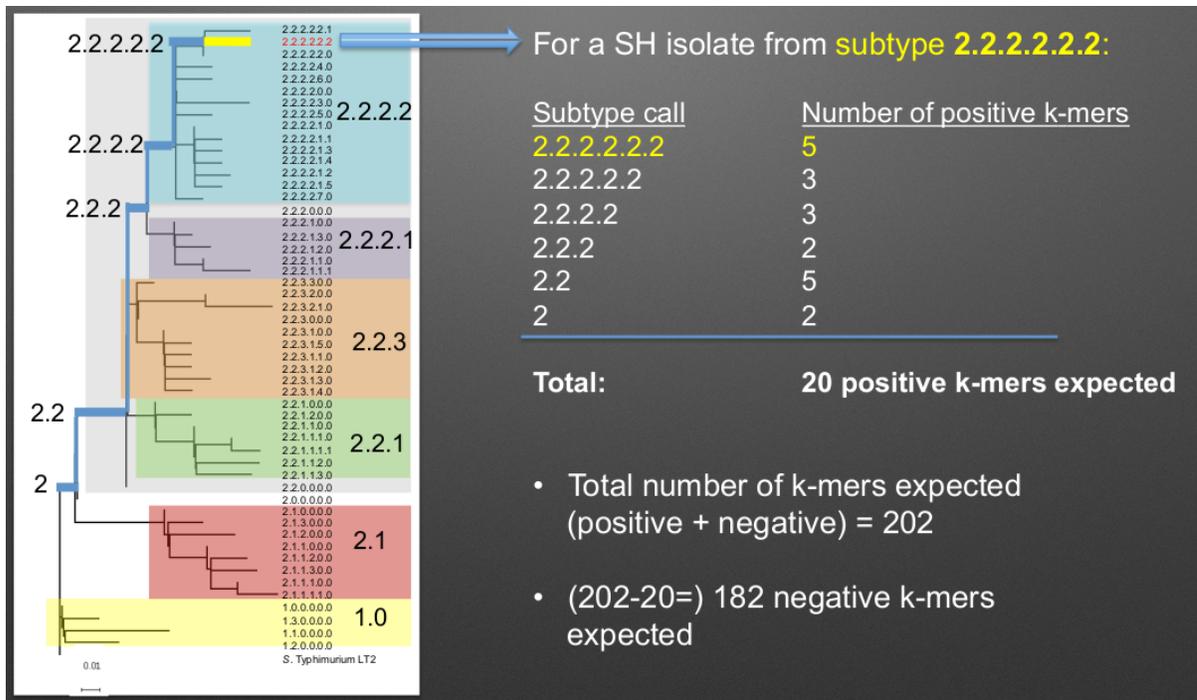
## 4.2 *Mycobacterium tuberculosis* (TB) scheme:

Biohansel now also features a TB scheme developed by Daniel Kein that was adapted from a publication by Francesc Coll et al. titled: “[A robust SNP barcode for typing Mycobacterium tuberculosis complex strains](#)” This scheme currently features a set of 62 33-mers that define 62 different genotypes.

## 4.3 Genotype Classification System

The genotyping classification system created for biohansel follows a nested hierarchical approach to allow relationships between genotypes to be established based on which SNPs they contain. The format designed for the classification system supports modification of the existing genotyping scheme to recognize new branches of new genotypes as they are fit into the existing classification system. The designed system works as a way to easily link outbreak origins and look at places further up the hierarchy where interventions can be done and monitored based on what genotype was found where.

The scheme and process that is used to genotype *Salmonella enterica* subspecies Heidelberg can be seen below:



Following the scheme, each positive k-mer match leads to a more specific classification of the genotype with the first matches determining which lineage the isolate is from (1 or 2 in this case). After the main lineage is determined, the genotype is determined based on all of the matching positive k-mers found in the sample as it follows along the path to the specific genotype. It is important that all/most positive and negative k-mers match a spot in the sample to allow correct genotyping and not generate errors!

It is important to note that for a given genomic SNP position defining a lineage, the “positive” (e.g. inclusive) k-mer means that the SNP base is present “inside” the lineage for all of that lineage and nested ones. The “negative” (e.g. exclusive) k-mers include the SNP bases present “outside” of that lineage such that that specific SNP is not found in any of the hierarchical lineages.

The [Output section](#) contains more details on the errors that can be run into when running a sample.

## 4.4 Creating a Genotyping Scheme

Creating a statistically valid, representative, and well established genotyping/subtyping scheme for biohansel is a large task. Once a scheme is established however, it is easy to modify the scheme to fit the needs of the research and allow for new classifications as they are discovered. When creating a genotyping scheme, keep in mind that the **organism should be clonal**, meaning that the majority of the target pathogen population should belong to a recent clonal expansion, or should represent a pathogen with a very slow genetic evolution (e.g. *M. tuberculosis*). BioHansel functions by finding exact matches to the k-mers defined in the scheme, so populations with a high genetic diversity will not have a sufficient number of k-mer pairs (or split k-mers) that are conserved across the whole population, leading to entire lineages that could fail BioHansel QC. All of the k-mers pairs identified and created for the genotyping scheme should be found in all or almost all the isolates for biohansel to work correctly. If the genetic diversity of the target population is higher (e.g. >3,000 SNPs across the core genome between isolates), a wgMLST or cgMLST scheme would be more appropriate than a SNP-based scheme, as is the case for *Salmonella* serotyping.

To create a well constructed genotyping scheme the steps below should be followed. However, you do not need to follow the steps to create a genotyping scheme and you can create a quick one to identify certain

k-mers instead. As long as the k-mer scheme is followed, the k-mers and their locations can be identified using the match\_results.tab file.

### 4.4.1 Detailed Steps

The detailed steps to create a well structured and accurate genotyping scheme are as follows. These steps were used to create the Genotyping Schemes included in biohansel and have been shown to create accurate results from the test samples run. The steps are:

1. Generate a large dataset that is representative of the organisms population being defined. For best results make sure to:
  - Remove outliers
  - Remove poor quality data
  - de-duplicate the dataset
2. Choose an available reference genome for the organism (ideally a closed/complete genome).
3. Subdivide the population into closely related clonal groups using MASH followed by SNP analysis. This can be done with any Mash clustering tool. An example used to create the included schemes is [Mash version 2](#). The SNP analysis can be done with a number of tools including [SNVPhyl](#), [parsnp](#), [snippy](#), or any tool that you prefer.
  - Aim for groups that are less than 3,000 SNPs between strains over more than 80% of the reference genome

	A	B	C	D	E
1	ref_ID	query_ID	MASH Distance	P-value	Matching-Hashes
2	reference_genome.fastq	reference_genome.fastq	0.0000000	0	1000/1000
3	STR32523.fna	STR6666.fna	0.0003856	0	984/1000
4	STR2131.fna	STR2221.fna	0.0009313	0	962/1000
5	STR4442.fna	STR5122.fna	0.0012112	0	951/1000
6	STR32523.fna	STR7604.fna	0.0017062	0	932/1000
7	STR3213.fna	STR32523.fna	0.0018657	0	926/1000
8	str1411.fna	STR8581.fna	0.0020536	0	919/1000
9	STR6666.fna	STR7604.fna	0.0021077	0	917/1000
10	STR3213.fna	STR6666.fna	0.0022710	0	911/1000
11	STR3213.fna	STR7604.fna	0.0034898	0	868/1000
12	reference_genome.fastq	STR4442.fna	0.0040872	0	848/1000
13	reference_genome.fastq	STR5122.fna	0.0052810	0	810/1000
14	STR32523.fna	STR4124.fna	0.0052810	0	810/1000
15	STR4124.fna	STR6666.fna	0.0057082	0	797/1000

Above is an example of a sorted all against all MASH result based on the matching-hashes column. This result is to see which strains are the most closely related and confirm that all of the samples are similar enough to be able grouped together for a scheme.



Above is an example of a SNP analysis using parsnp and Gingr. These tools can be used to visualize a p phylogenetic tree along with providing a multiple sequence alignment where the SNPs can be easily viewed.

#### 4. Remove rare outliers from the dataset

- these are detected by SNP matrices, number of unaligned bases, number of heterozygous sites, number of bases with low coverage, etc.
- These rare outliers are from suspected poor quality WGS data, mixed culture samples, or large recombinant regions (phage or transposons).

#### 5. De-duplicate the data once again by removing strains that are nearly identical to each other. This can be defined as:

- Strains that are 0-2 SNPs apart over more then 80% of the reference genome
- Strains that MASH cluster with a distance of 0.001

STR32523.fna	STR6666.fna	0.0003856
STR2131.fna	STR2221.fna	0.0009313

According to the MASH clustering result shown above, we have to pick one of STR32523/STR666 and one of STR2131/STR2221 as they are too similar to differentiate properly.

#### 6. Create a Maximum Likelihood (ML) phylogenetic tree from the SNP derived reference assembly of the strains to the reference genome. Here you are looking for:

- Regions that are conserved across the whole population of interest such that the SNPs in the areas are found in 99.5% of all isolates
- SNPs that are at least 20 base pairs from other SNPs or indels
  - **The 20 bases on either side of the SNP should be conserved in at least 99.5% of isolates!**

This can be done with any tool that creates a ML phylogeny. Examples of tools previously used include: SNVPhyl, parsnp, and MEGA.

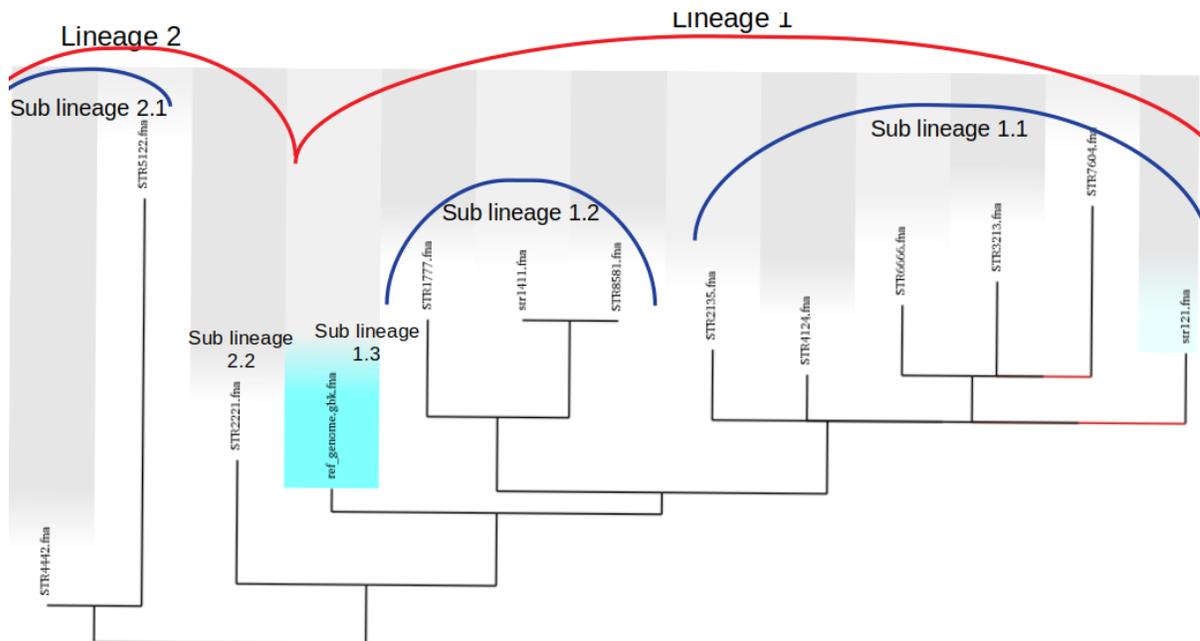
7. Divide the ML tree into main lineages and sub-lineages according to the shape of the tree to allow users to identify

the main clonal expansions. When doing this make sure that:

- Tree branches are at least 2 SNPs long
  - The longer the branch, the better, as there will be more SNP positions to choose from for defining that genotype.

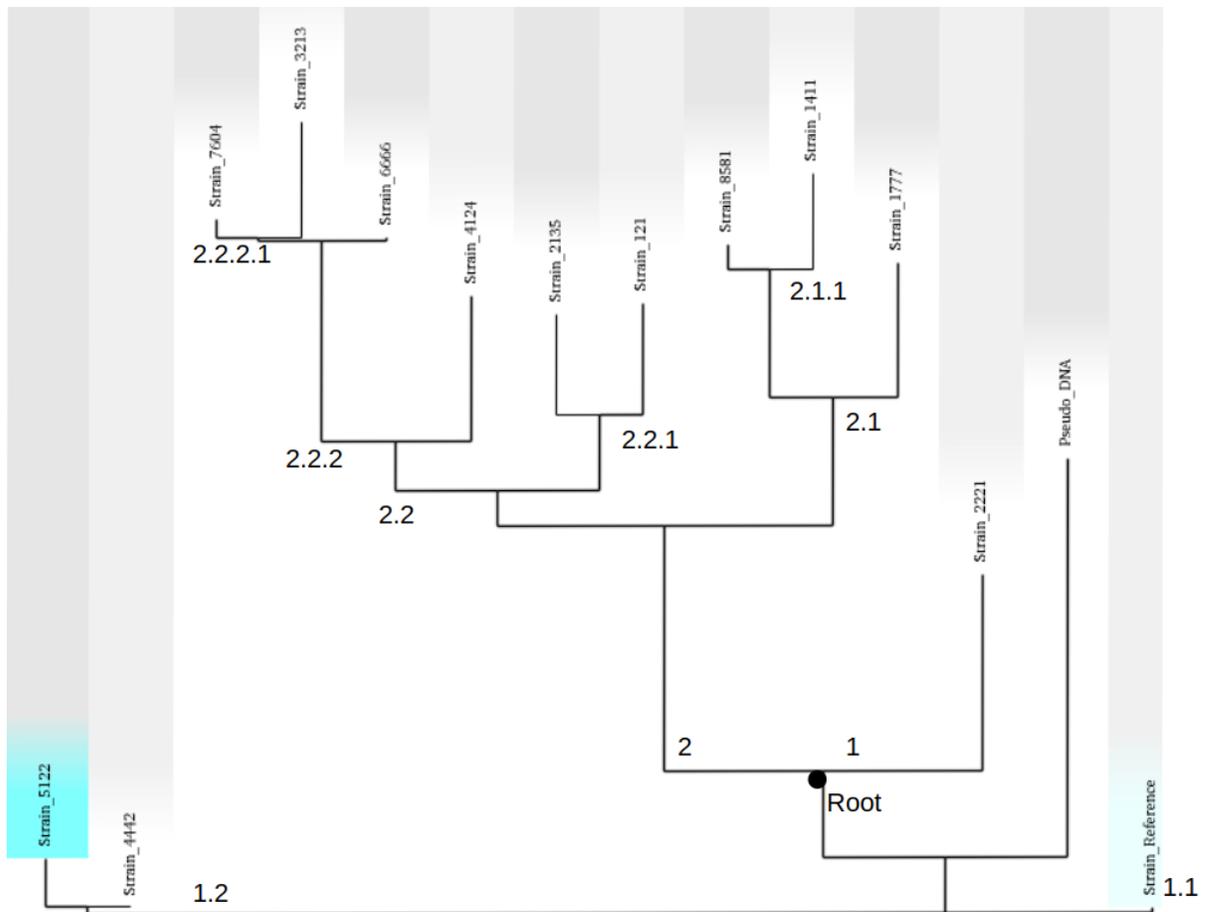
You can look at a SNP file generated previously to look at the SNPs from regions that don't feature any indels and are isolated by at least 15 (preferably 20) nucleotides on each side.

If wanted, you can lower the number of SNP sites to be evaluated into the scheme by removing all of the SNPs that are present in less than 5 isolates and then remaking the tree. The aim is to have at least 5-10 strains per sub-lineage, to keep the scheme focused on clonal expansions.



Above is the ML phylogeny previously generated with lineages and sublineages applied to the strains. These are a preliminary delegation and can change in the next steps. However, it is a good idea to set up lineages now and edit them as better designations are designed.

8. Create a neighbour-joining tree and root it using a distantly related sequence or a pseudo sequence to determine where the root of the tree should be.
9. Give main lineages and sub-lineages determined previously hierarchical codes based on how they cluster in the NJ tree and the SNPs that make up each sequence.



Based on the SNPs seen in the .vcf file and the rooted tree, hierarchical codes are assigned. The root is in an odd spot in this example as it was determined mostly based off of the SNPs seen in the parsnp tree. It is important to verify that the root is correct with an outgroup as the biohansel scheme needs to be strictly hierarchical.

10. Extract from the SNV table or VCF file the canonical SNPs that define the genotype and differentiate it from other strains using **FEHT** which can be installed into bioconda or galaxy.

The installation instructions are found in the link but if you are using bioconda for biohansel, the easiest thing to do is go to the wanted environment and install FEHT there with the following commands:

```
conda activate <name of environment to install feht to>
conda install -c bioconda feht
```

FEHT needs the following specific files to run this process:

- A metadata file with the hierarchical codes
- A SNV table or a VCF file that defines the genotype
- The metadata file will be the info file and the VCF file will be the datafile that is needed for Feht to run.

**Make sure that the isolate names match exactly and both files use a tab delimiter**

The metadata file should look as such and be in a .tsv format:

Strain_name	Level_0	Level_1	Level_2	Level_3	...
SRR1242421444	1	1.1	1.1.2	1.1.2.3	...
SRR1242422313	2	2.2	2.2.2	2.2.2	...

The VCF table should look as such and also be in a **.tsv** format:

	reference	SRR1242421444	SRR1242422313
122123	0	1	0
234142	0	0	1
341251	0	1	1

11. Extract the exact matches to the query using the ratioFilter in FEHT by switching “-F” to “1”.

This is done as the FEHT program performs an all-against-all comparison of all the genotypes, one column (one hierarchy) at a time and we only want the exact matches.

12. From this output, we want to extract the genotype against all else results by searching for the ! sign (ex. search !2.2 instead of 2.2) and compile these results into a new **.tsv** file with the following information:

Genotype	SNP Location	Positive Base	Negative Base
1	395	A	G
1	2998	T	G
1.1	29231	A	G
1.1.1	77889	T	C

The positive base is the base found in the middle of the k-mer and it corresponds to the genotype of the sample. The negative base is the base found in all other samples. Both are equally important for the program to function properly so it is essential that they are properly defined.

13. Create the genotyping scheme with all of the information obtained. The SNP column shows the exact position that the SNP is found in the reference genome. This spot can be made into a 33-mer k-mer used in the scheme by recording 16 bases on each side of the SNP such that the SNP is in position 17 of the 33-mer.

A python script can be written to do this such that it creates 33-mers from the reference genome. Keep in mind that most of them will be of the negative variety and the positive k-mer pair will need to be created in the next step.

14. Finish the genotyping scheme by making sure that each carefully crafted 33-mer has a positive and negative pair attached to the correct genotype. This can be done also using a script (currently being worked on) or the following method:

1. Paste the 33-mers into the correct location in the FEHT filtered output spreadsheet next to the corresponding SNPs.
  
2. The 33 bp sequences are expanded using TextWrangler (replace [A,T,C,G] by the same base+tab), then pasted back into excel, in 33 adjacent columns.



```
>negative2981-2.2.3.1.4  
ACTGCCCGCCGGAGCCGCGTGAAAATATTGTTTA
```

\*\*\*The first distinction between genotypes 1 and 2 (or potentially more genotypes) does not have a negative condition and instead moves samples into one of the two classes established. The setup for the k-mers is similar to the other k-mers shown above and looks like such:

```
>717-1  
ATGCAGAGTCAGTCAGATCAACATGCACCCACA
```

```
>717-2  
ATGCAGAGTCAGTCAGTTCAACATGCACCCACA
```

### Notes

- **The k-mer length can be variable**

The length of the positive and negative k-mers within a pair does not need to be the same.

- **BioHansel can work using a list of positive k-mers exclusively**

In that case however, the user will not benefit from the quality controls that are performed in BioHansel using k-mer pairs.

- **The target canonical SNP can be located anywhere within the k-mer**

The target SNP does not need to be in the center of the k-mer sequence.

- **Since the tool relies on finding exact k-mer matches, the positive k-mer sequence could in theory target an indel sequence**

The target k-mer sequence needs to be conserved in a lineage and absent in the rest of the pathogen population. If there is an indel sequence that is conserved in a lineage and present only in that lineage, the corresponding negative k-mer should be found in the rest of the population and absent in the lineage containing the positive k-mer indel sequence (for example, the negative k-mer sequence could be spanning the target insertion or deletion site in the genome in the rest of the population, if that genome region is unchanged across the rest of the population).

16. Test the created scheme by running biohansel to verify that all of the expected positive target sequences are present in the corresponding strains. Eliminate targeted k-mers from the scheme that do not work well and verify that the targeted k-mers pairs created are present in most of the dataset. Finally test the scheme on a de novo assembly along with raw Illumina sequencing reads to make sure it holds true for both.

---

## Degenerate Base Expansion

---

Biohansel allows users to input custom schemes containing [IUPAC degenerate bases](#) in their kmers. These degenerate bases allow for increased flexibility for kmer matches but can come at the cost of computing power.

In this section, we will look at the how to use degenerate bases and where to be careful when dealing with degenerate bases in your biohansel genotyping schemes.

### 5.1 Including Degenerate Bases in k-mers:

Any of the degenerate bases can be added anywhere in the genotyping schemes k-mers. Once this happens, when the code is run, the k-mers are expanded into all possible combinations of normal DNA bases (A,G,C,T). These degenerate bases can be anywhere in the k-mer so long as there is still a SNP separating the pair, and, if they are not in the SNP location the degenerate base matches in both k-mers of the pair.

#### 5.1.1 Degenerate Base as a SNP:

Example k-mer pair with a single degenerate base as the SNP separating the pair. Here the SNP is bolded to try to make it easier to see. Remember that biohansel scheme k-mers can be any length, but should be long enough to be specific for a particular genome location:

```
>1231-2.2  
TADCT
```

```
>negative1231-2.2  
TACCT
```

The k-mer with the degenerate base would expand into three separate k-mers for the same genotype at the same position. Once the code was ran, these three positive k-mers would only count as 1 k-mer for the

results outputs.

Expanded k-mers where if any of the positive k-mers are found in this set, you have genotype 2.2:

```
>1231-2.2  
TAACT
```

```
>1231-2.2  
TATCT
```

```
>1231-2.2  
TAGCT
```

```
>negative1231-2.2  
TACCT
```

### 5.1.2 Multiple Degenerate Bases Elsewhere in the K-mer:

As stated, you can have any number of degenerate bases in the SNP scheme k-mers. Here is an example of a pair containing two degenerate bases not in the position of the SNP. Here the degenerate bases are bolded along with their expansions:

```
>1231-2.2  
CTTRACTW
```

```
>negative1231-2.2  
CTRCCTW
```

This pair would expand into 4 different k-mers for each pair or **8** new k-mers in total. For the positive ones:

```
>1231-2.2  
CTAACTA
```

```
>1231-2.2  
CTAACTT
```

```
>1231-2.2  
CTGACTA
```

```
>1231-2.2  
CTGACTT
```

And for the negative ones:

>negative1231-2.2  
CTACCTA

>negative1231-2.2  
CTACCTT

>negative1231-2.2  
CTGCCTA

>negative1231-2.2  
CTGCCTT

As you can see, they are still only separated by the A SNP in the positive k-mer and the C SNP in the negative one. Now there are 4 different pairs, with 8 separate sequences making up these 4 pairs. This expanded all from the original pair and all of these new pairs would check for genotype 2.2 at position 1231.

In this example, there would end up being a total of 16 different k-mers due to the reverse compliment also being input into biohansel.

## 5.2 Unchecked Expansion of K-mers:

When calculating the number of k-mers that are being created by a k-mer pair, the IUPAC DNA Nucleotide codes can be thought of as the number of possibilities that they expand to and not what they represent as seen below:

A, G, C, T = 1

R, Y, S, W, K, M = 2

B, D, H, V = 3

N = 4

Using this, we can take any nucleotide k-mer we want and calculate the number k-mers that are being created.

Using the values above, we can substitute the nucleotide with a number and find the number of k-mers created.

Example sequence is "ACGTAGC":

(A)(C)(G)(T)(A)(G)(C)  
(1)(1)(1)(1)(1)(1)(1) = 1

If we had a sequence with degenerate bases, we will start to see how fast the number of k-mers can increase.

Example sequence is “ACTNNANNTTA”

(A) (C) (T) (N) (N) (A) (N) (N) (T) (T) (A)  
(1) (1) (1) (4) (4) (1) (4) (4) (1) (1) (1) = 256

This example is an example how having only 4 ‘N’ nucleotides expands our one sequence into 256 different ones.

These 256 k-mers aren’t including the second part of the pair that this sequence has to belong to and as their is a SNP of ‘A’ in the middle, the other k-mer must also contain those 4 ‘N’s. This means that there are 512 k-mers being used for just this pair alone.

Even then, the 512 k-mers for the positive and negative positions become a total of 1024 different k-mers due to the need to take into account the reverse compliment of all of the sequences.

To put this in perspective, the Heidelberg SNP genotyping scheme contains 202 pairs with 404 sequences and once ran, this is expanded to 808 sequences by biohansel due to the reverse compliment input. The whole genotyping scheme has less k-mers than a single SNP pair in this case.

The goal is the remember that even a small number of degenerate bases can lead to a large number of k-mers and longer run times. ‘N’ is the extreme however and if you were creating a scheme with only the “2” value degenerate bases (ex. ‘R’), then you could have 8 degenerate bases for a single pair and end up with the same 1024 expanded k-mers from the pair.

\* The take away here is to be careful when including degenerate bases in your scheme. The **more degenerate bases included**, the **more kmers** are that are produced by expansion, the **slower the run time**, and the **more RAM is needed to run the sample**.

## 5.3 Benchmarking Degenerate Bases

### 5.3.1 Expand Degenerate Base Module

More degenerate bases = More K-mers = Slower Run Times

In this section we are going to look at the speed of the expand base module and the code itself for different numbers of k-mers.

Here is the speed of running the **expand degenerate bases module** (not biohansel itself) on 1 core using pythons timeit:

K-mer Sequence	Max K-mers Produced	Time
A	1	1.59 microsec
N	4	1.79 microsec
NN	16	2.47 microsec
NNN	64	5.61 microsec
NNNN	256	17.50 microsec
NNNNN	1024	68.30 microsec
NNNNNN	4096	305.0 microsec
NNNNNNN	16384	1.41 msec
NNNNNNNN	65536	6.15 msec
NNNNNNNNN	262144	26.5 msec
NNNNNNNNNN	1048576	112.0 msec
NNNNNNNNNNN	4194394	470.0 msec
NNNNNNNNNNNN	16777216	1.950 sec
NNNNNNNNNNNNN	67108864	8.930 sec
NNNNNNNNNNNNNN	268435456	<b>Died</b>

The higher k-mers are a bit of a stretch but show how much longer the module takes **PER K-MER** if you are not careful.

Remember, the above chart is for a **singular** k-mer and does not take into account the expansion of the whole scheme. If you had a scheme with a lot of these, it would take that amount of time for each k-mer!

### 5.3.2 Whole Biohansel Code

Benchmarking all of the biohansel code using the same input fasta file but increasing the total k-mer count each time.

Remember that the total number of k-mers if there are no degenerate bases is equal to the number of pairs multiplied by 4 to take into account two sequences per pair and the RC of each sequence.

If degenerate bases are present, it is harder to guess the number and running biohansel will tell you if you have over the default 100,000 k-mers and allow you to set the value that you deem acceptable with the “-max-degenerate-kmers” command.

Number of Nucleotides	Number of Scheme K-mers	Execution Time (sec)
4751529	808 -> Base Heidelberg Scheme	0.613
4751529	4,394	0.663
4751529	12,074	0.721
4751529	36,650	0.873
4751529	69,418	0.971
4751529	134,954	1.031
4751529	266,026	1.502
4751529	528,171	2.150
4751529	1,052,459	3.269

This work was done on an assembled fasta file. Note that even with 1,000,000 k-mers, the time it takes to run biohansel is only 3 seconds. BUT, if you’re using fastq files it is going to be much longer and they haven’t been tested for speed with expansion yet! So be careful with large expansions on fastq files.



---

## Output

---

This page describes the three different result files will be produced from running biohansel: *tech results.tab*, *match results.tab* & *results.tab*. The results found in these three files will be the same whether you are using the command line or Galaxy to run an analysis.

## 6.1 Tech Results.tab

### 6.1.1 Structure:

Tech\_results.tab is the simplest output file released by running a biohansel analysis. It contains only the sample name, genotype, and the QC status of the sample allowing this file to be easy to interpret at the cost of not elaborating on any of the specific details of the analysis. Found below are the columns and explanations of the columns for this output file:

Sample	Genotype	Avg_kmer_coverage*	QC_status	QC_message
(Sample Name)	(Corresponding Genotypes Found)	(average k-mer coverage of all the targets)	(PASS/FAIL/WARNING)	(Corresponding QC message)

\*Average\_kmer\_coverage is only found in the output when analyzing fastq files or directories with fastq files.

#### Sample

This column provides the names of samples that were run on biohansel

## Genotype

This column gives the genotype/subtype of the sample determined by the analysis. This column can display a single positive genotype, a list of positive genotypes, or no genotype depending on the results of the analysis. A good analysis will output the following:

1	2	3	4
sample	subtype	qc_status	qc_message
STR2131	2.1	PASS	

If this column does not display a single positive genotype, it will show one of the two following situations:

1. Different genotypes if mixed samples are run or there is an error in a user-created scheme. In this case, biohansel can list multiple different genotypes that were detected.

### Genotype assignment

Genotype assignment uses only the identified positive k-mers and the overall genotype is determined by the most resolved genotyping k-mer(s) identified (e.g. the k-mer(s) associated with the genotype(s) that have the **highest** number of hierarchical levels). In the case of a mixed sample, it can result in two or more genotypes being assigned, if they have the same number of hierarchical levels (e.g.: 1.1.1 and 1.1.2; which both have 3 hierarchical levels).

### Troubleshooting note

The genotype with the highest number of hierarchical levels could correspond to a contaminant, rather than the genotype with the highest genome coverage, in a mixed sample.

1	2	3	4
sample	subtype	qc_status	qc_message
genome1.1.2	1.1.1; 1.1.2	FAIL	"FAIL: 22.22% missing tiles"

2. If no positive target is detected, the column will be blank and the qc\_message will state that no k-mers/targets were found.

1	2	3	4
sample	subtype	qc_status	qc_message
genome1.1.3		FAIL	No tiles/targets were found in this sample.

## Average K-mer Coverage

Displays the average coverage of all of the targets/k-mers that were present in the sample.

## QC Columns

QC Status and QC message are found in full details under their own section as they are a part of all 3 output files. This detailed information is found in the *Quality\_Control* section.

## 6.2 Match Results.tab

### 6.2.1 Fasta File Output Structure:

The following is the scheme for the match\_results.tab file **For a single Fasta file. Running raw reads data has slightly different output columns due to the different nature of the data.** The output columns for the match\_results.tab file are shown below broken into different charts to allow them to fit mostly on one page. In the real generated file, they would all found in the same long row. Below, you will find detailed information for each column.

kmer-name	Se-quence	is_rev	Comp-ting_id	Match	Reflex-posit-ion	Geno-type	is_pos	Sam-ple	File_pa-th	Schem	Schem	QC_Stat	QC_message
(Name of Target/K-mer)	(Cor-responding K-mer Se-quence)	(TRUE/FALSE)	(Name of Contig)	(Match Position)	(Match Position in refer-ence)	(Geno-types in kmer-name)	(TRUE/FALSE)	(Sam-ple Name)	(File Location)	(Schem Name)	(Schem Version)	(PASS/FAIL)	(Cor-responding QC mes-sage)

### 6.2.2 Raw Reads FastQ File Output Structure:

Running raw reads files/FastQ files gives slightly different output columns when compared to the Fasta file match\_results.tab output due to the slight differences in the data that each file contains. The overall output for a match\_results.tab results output from a FastQ file looks as such:

kmer-name	Se-quence	Fre-quency	Ref-posit-ion	Geno-type	is_pos	ikmer	File_pa-th	Sam-ple	Schem	Schem	QC_Stat	QC_message
(Name of Target/K-mer)	(Cor-responding K-mer Se-quence)	(Num-ber of exact matches found)	(Match Position in refer-ence)	(Geno-types in kmer-name)	(TRUE/FALSE)	(ikmer)	(File Location)	(Sam-ple Name)	(Schem Name)	(Schem Version)	(PASS/FAIL)	(Cor-responding QC mes-sage)

### 6.2.3 Detailed Column Information

The detailed information on the meaning of each columns outputs for both files can be found below:

#### Kmername

This column gives the name of the target/k-mer that matched to the sample. It will match to the name of the k-mer in the fasta file following the fasta convention as seen in the [input section](#). The k-mers that match the sample give the genotype of the sample

### Sequence

The column contains the sequence of the k-mer from the kmername column. This sequence is the 33 bp fragment that matched somewhere in the sample.

### is\_revcomp

Is the k-mer found in the forward direction or the reverse direction?

1. FALSE - the target k-mer was found from the 5' to 3' direction
2. TRUE - the target k-mer was found in the 3' to 5' direction in the sample

### Contig\_id

Displays the name of the contig as found in the Fasta file.

### Frequency

Displays the exact number of matches found for the k-mer in the raw reads/FastQ file input.

### Match\_index

Displays the last nucleotide match of a k-mer as its position in the genome.

For example, if the k-mer matched the genome from positions 12312 to 12345, the SNP would be at position 12329 and output of this column would be 12345.

### Refposition

Displays the numerical position of the k-mer/k-mers SNP in the reference genome. This information is also found in the description of the k-mer in the genotyping schemes Fasta file.

### Genotype

Shows the consensus genotype of the sample as determined by the analysis.

This column can display a single positive genotype, a list of positive genotypes, or no genotype depending on the results.

### is\_pos\_kmer

Is the k-mer in question a positive k-mer/target for specific genotype?

1. TRUE - the positive SNP has been found in the sample
2. FALSE - the negative SNP has been found in the sample



### Sample

Provides the names of samples that were run on biohansel

### Scheme

The name of the chosen Scheme used in the analysis.

### Scheme\_Version

The version of the chosen scheme used in the analysis.

### Genotype

Shows the consensus genotype of the sample as determined by the analysis.

This column can display a single positive genotype, a list of positive genotypes, or no genotype depending on the results.

### All\_genotypes

All of the genotypes in all the levels of lineage leading to the final genotype.

4	5
subtype	all_subtypes
2.2.3.1.2	2; 2.2; 2.2.3; 2.2.3.1; 2.2.3.1.2

### kmers\_matching\_genotype

Displays the genotype(s) that the most downstream, specific k-mers have matched to. For good, non-mixed results, it should be the same as the genotype column.

### are\_genotypes\_consistent

1. TRUE - the genotypes are consistent as defined.
  - Consistency -> All positive k-mers within QC parameters have consistent genotypes in downstream sublineages corresponding to parent genotype.

Each k-mer must become more specific to the final genotype while matching all of the previous ones to be considered consistent.

2. FALSE - the genotypes are not consistent.

**inconsistent\_genotypes**

If “are\_genotypes\_consistent” is FALSE, it lists genotypes that are inconsistent to parent.

7	8
are_subtypes_consistent	inconsistent_subtypes
False	['1.1.1', '1.1.2']

**n\_kmers\_matching\_all**

Counting all of the actual k-mer matches (both positive and negative) that make up each genotype lineage as defined by the genotyping scheme used/created.

**n\_kmers\_matching\_all\_expected**

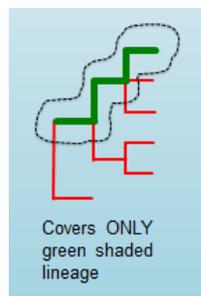
The total number k-mer/target matches expected (both positive and negative) that make up each genotype lineage as defined by the genotyping scheme used/created.

Every/almost every k-mer defined in the scheme should match somewhere in the sample if the sample is of high quality.

9	10
n_tiles_matching_all	n_tiles_matching_all_expected
202	202

**n\_kmers\_matching\_positive**

The number of positive matches in the sample from all of the upstream lineages of the output genotype as defined by the genotyping scheme.



**n\_kmers\_matching\_positive\_expected**

The expected number of positive matches from all of the upstream lineages of the output genotype as defined by the genotyping scheme.

For a good analysis, this value should match the sample.

### **n\_kmers\_matching\_genotype**

The number of positive matches in the sample sublineage only.

### **n\_kmers\_matching\_genotype\_expected**

The expected number of positive matches in the sample sublineage only.

### **File Path**

The file location of the input data.

### **Avg\_kmer\_coverage**

The average frequency of all k-mers, both positive and negative, that were found in the sample. This output column is only found for analysis of raw reads FastQ files and it is an indicator that there was a sufficient amount of overlap in the dataset for the results to be significant.

### **QC Columns**

QC Status and QC message are found in full details under their own section as they are a part of all 3 results files. This detailed information is found in the *Quality\_Control* section.

## **6.4 Quality\_Control**

### **6.4.1 QC Status**

Three possibilities can be shown in this column based on the QC analysis described below: *QC message*

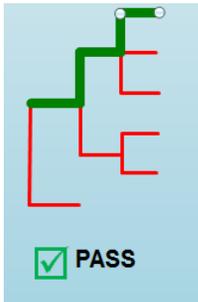
1. PASS
2. FAIL
3. WARNING

### **6.4.2 QC message**

The QC message displayed provides information on what happened in the analysis and where, if there was a warning or fail, the data can be cleaned up/improved to obtain a passing result.

**“Pass”**

A pass occurs when there is no errors in the targeted lineage and its corresponding sublineages:



Once the QC module is declared as a pass, there is no information in the QC message column displayed. The result should be considered a valid analysis.

**“WARNING: Intermediate Genotype”**

Warnings will be triggered if all four following conditions are met:

**1st condition:** Less than 5% of the k-mers are missing (by default) or more than 95% of the schemes targets are matched (parameters for this is adjustable prior to running biohansel)

**2nd condition:** There should be no clash for “+” and “-” targets for the same genome position (above background noise level)

**3rd condition:** Only a fraction of the k-mers are positive for the final genotype (“# of k-mers matching genotype expected > # of k-mers matching genotype”)

**4th condition:** The targets for the final subtype are a mixture of both “+” and “-” BUT do NOT clash for the same positions.

**“WARNING: Low Coverage”**

If the “Avg k-mer Coverage” is below the parameters given for low coverage (parameters are adjustable) (default min average coverage: 20- fold)

Average coverage calculated from all targets found in the sample (The value is returned to the user)

**Error Type 1: Missing kmers**

\*\*\* The Maximum amount of missing k-mers, either positive or negative, to be allowed before being considered an error/fail. This amount can be edited based on preference and scheme.

Three possible causes:

1. Bacterial scheme does not match target
2. Low genome coverage or low quality data
3. Range of target coverage extends outside of QC limits (k-mer frequency thresholds default = min:8, max:500)

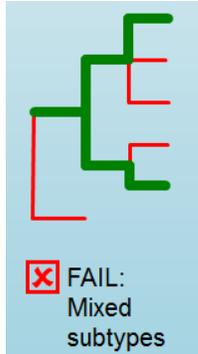
\*\* To determine which cause, the average coverage depth is returned to the user. The value is calculated based on the coverage for all k-mers that were above the minimum coverage threshold (indicated by the QC parameters: default value = 8)

### ***Error Type 2: Mixed Sample***

A mixed sample error is where biohansel is unsure what the final genotype is of the sample due to one of two possible causes:

1. biohansel came out with an “inconsistent result” designation
2. Position conflict: both “+” and “-” targets are found in the same target genome position above background noise level

A possible solution to this error if the average genome coverage is above 100 is to increase the minimum k-mer threshold to at least 10% of the average genome coverage. This will change the background noise tolerated and potentially allow for a positive result to occur.



### ***“Error Type 3: Ambiguous result”***

Caused by both conditions met:

1. Total matching k-mers is within 5% of the expected value
2. 3 or more k-mers are missing for the final genotype call (Error 3a)

***“Error Type 4: Unconfident/Inconclusive result”***

Lineage call is uncertain due to missing targets in downstream sublineage.



---

## Galaxy Parameters

---

This section will explain how to use the Galaxy parameters of biohansel to get the most out of your analyses. The command line is slightly more complicated than Galaxy but also allows more a bit more customization to it. The command line arguments and parameters that can be changed can be found in the [Command-Line section](#)

These are the parameters and their functions using Galaxy. Each of the headers is the name of the section on Galaxy where the underlying parameters can be changed.

### 7.1 Sequence Data Type

The sequence data type settings in Galaxy allow the user to choose what type of file the analysis will be run on. Picking a file type listed in the subsections below will give a few additional options. These options relate to how many files are being analyzed at once and how the data is output. A single file or multiple files can be run with one execute command. The types of jobs that can be run include:

1. A single fasta/fastq file
2. Multiple fasta/fastq files to be analyzed separately
3. A dataset collection of fasta/fastq files that will generate a single output file

Remember that unlike the command line, you cannot run different file type analyses at once with a single run in Galaxy.

#### 7.1.1 Contigs (FASTA)

When the contigs type of sequence data is chosen, Galaxy allows an input of any fasta file currently in the history being worked from.

### 7.1.2 Paired-end reads (FASTQ)

If the paired-end reads data type is selected, two fastq files are needed as input for an analysis to be run. These must have ASCII encoded quality scores. Additionally, the forward and reverse file formats must match.

### 7.1.3 Single-end reads (FASTQ)

If single-end reads is selected, a single fastq file is required as input

### 7.1.4 Paired-end reads collection (FASTQ)

With a paired-end reads collection, you can only select that as the input to run biohansel on.

## 7.2 SNP Subtyping Scheme

Under this heading, select the subtyping scheme that corresponds to the sample type that you are running. If you are running a Heidelberg sample then you would select “Salmonella Heidelberg subtyping scheme” from the drop down menu. A scheme is necessary for a run and if none is selected, there is no run.

If you select the “specify your own custom scheme”, you will need to input the k-mers for the scheme you have created as a fasta file. You can also input multiple developed schemes to run on the input data selected in the Sequence Data Type section. To do this, click on the small k-mers below where it says “Your biohansel SNP Subtyping Scheme”.

### SNP Subtyping Scheme

Select the SNP subtyping scheme you wish to subtype with

#### Your biohansel SNP Subtyping Scheme

### 7.3 Scheme Subtype Metadata Table (Optional)

The Scheme Subtype Metadata Table is an optional input that adds more details to the results files. It can be left blank or a .tsv or .csv file containing a column called subtype can be added in here. Multiple metadata tables can be added and linked to multiple analyses or a collection of them can be added so that they come out in a single output file.

### 7.4 K-mer Frequency Thresholds

The parameters found under this column are only needed when running results on raw read fastq files. They control how much k-mer is needed to be considered acceptable for an analysis.

### 7.4.1 Min k-mer frequency/coverage

The minimum frequency that all of the k-mers of the chosen subtyping scheme should be found in a raw reads Fastq dataset. If a k-mer falls below this coverage, it will not be taken into account in the results.

This parameter should be adjusted based on the average genome coverage of the dataset (or estimated average k-mer coverage output by biohansel). A good min k-mer coverage would be 10% of your estimated genome/k-mer coverage.

### 7.4.2 Max k-mer frequency/coverage

The maximum frequency that all of the k-mers of the chosen subtyping scheme should be found in a raw reads Fastq dataset. If a k-mer is above this chosen frequency, it will not be taken into account. Default is 10,000. Make sure that the Max k-mer frequency is at least 5x more than the average genome coverage of the dataset (or estimated average k-mer coverage output by biohansel).

#### **K-mer Frequency Thresholds**

##### **Min k-mer frequency/coverage**

default = 8 (--min-kmer-freq)

##### **Max k-mer frequency/coverage**

default = 1000 (--max-kmer-freq)

## 7.5 Quality Checking Thresholds

These parameters are used to run the quality control (QC) module that determines if the outcome of biohansel is a PASS or a FAIL. Changing them allows for greater control on what the program will allow a pass, whether this is more strict or less strict.

**Quality Checking Thresholds**

**QC: Frequency below this coverage are considered low coverage**

  
default = 20 (--low-cov-depth-freq)

**QC: Min number of tiles missing for Ambiguous Result**

  
default = 3 (--min-ambiguous-tiles)

**QC: Decimal Proportion of max allowed missing tiles**

   
default = 0.05, valid values {0.0 - 1.0} (--max-missing-tiles)

**QC: Decimal Proportion of max allowed missing tiles for an intermediate subtype**

   
default = 0.05, valid values {0.0 - 1.0} (--max-intermediate-tiles)

**QC: Overall tile coverage below this value will trigger a low coverage warning**

  
default = 20 (--low-cov-warning)

### 7.5.1 QC: Frequency below this coverage are considered low coverage

This QC threshold determines if the coverage of each of the individual k-mers is enough to be considered adequate or low coverage. The lower this value is set, the more lax what is considered low coverage is for QC module warning. The default value is 20. This parameter may be useful for scheme development.

### 7.5.2 QC: Min number of k-mers missing for Ambiguous Result

This QC threshold determines the minimum number of positive k-mers that can be missed to result in an ambiguous output by biohansel. The default value is 3. A higher value means that the QC module is more relaxed resulting in less ambiguous results from datasets that are not as highly covered.

### 7.5.3 QC: Decimal Proportion of max allowed missing k-mers

This QC threshold determines the maximum number of k-mers allowed to be missing in the target dataset as a decimal proportion before the QC module determines the outcome to be a failure. The default decimal proportion is 0.05 or 5%.

### 7.5.4 QC: Decimal Proportion of max allowed missing k-mers for an intermediate subtype

This QC threshold determines the maximum number of k-mers allowed to be missed in the target dataset as a decimal proportion before the QC module determines that the outcome is a failure due to being an intermediate subtype. The default decimal proportion is 0.05 which equals 5%.

### 7.5.5 QC: Overall k-mer coverage below this value will trigger a low coverage warning

This QC threshold is the average k-mer Coverage of an analyzed fastq file which, when the average coverage falls below this number, a warning is triggered by the QC module saying that the sample is of low coverage. The default is 20. A lower number will relax the standard and allow sample of lower quantity to pass which may be needed on metagenomic samples. Doesn't affect the validity of the biohansel results.

**This is meant as a warning that you may want to resequence your sample for more data for other downstream applications.**

## 7.6 Developer Options

There is only one developer option available and it is to output JSON files on top of the normal analysis files. If this option is set to "yes", there will be six output files, three JSON files and three .tab files. The JSON files are used to represent analysis details that are not found in the normal files.



Here, the arguments needed to run biohansel effectively are displayed. The required and additional arguments are shown below to see what must be included in a run.

### 8.1 Required

Make sure to be in the directory containing all of the data needed to run a command or that the path to the input data is put into the command following the argument.

- Genotyping Scheme
  - use -s “scheme”
- Output/Results Files (any combination so long as there is at least one specified. Details in [Output](#))
  - use -S “filename.tab” | for tech\_results.tab
  - use -o “filename.tab” | for results.tab
  - use -O “filename.tab” | for match\_results.tab
    - \* You can also use “.tsv” as the file extension
- Input data
  - use -i <path/to/fasta> | to specify fasta file to analyze
  - use -p <path/to/forward\_reads> <path/to/reverse\_reads> | to analyze paired reads
  - use -D <path/to/directory> | to analyze a full directory of data into 1 file

### 8.2 Additional

**If any of these arguments are left off of the command used to run biohansel, they will be set to default values for the given analysis.**

- M “metadata\_scheme.tsv” -> Used to input a metadata scheme that follows all requirements found in `input`
- force -> Forces the existing outputs to be overwritten
- json -> Output JSON representation of output files
- min-kmer-freq <#> -> Minimum k-mer coverage needed for a raw reads fastq file to be considered acceptable by the quality control module (default is 8)
- max-kmer-freq <#> -> Maximum k-mer coverage for a raw reads fastq file to be considered acceptable (default is 10,000)
- low-cov-depth-freq <#> -> Coverage frequencies of raw read fastq files below this value are considered as low coverage (default is 20)
- max-missing-kmers <#> -> Decimal proportion of maximum allowable missing kmers before being considered an error (0.0 - 1.0) (default is 0.05 or 5%)
- min-ambiguous-kmers <#> -> Minimum number of missing kmers to be considered an ambiguous result (default is 3)
- low-cov-warning <#> -> Overall kmer coverage below this value will trigger a low coverage warning on raw read fastq files. (default is 20)
- max-intermediate-kmers <#> -> Decimal proportion of maximum allowable missing kmers (0.0 - 1.0) to be considered an intermediate genotype (default is 0.05)
- threads <#\_CPUs> -> Number of parallel threads used to run the analysis (default = 1)
- v -> Verbose: Logs verbosity levels where -v == show warnings and -vv == show debug info
- V -> Displays the version of biohansel installed

### 8.3 Hansel Help Command

If you run `hansel -h`, you will be provided with additional information for most of the commands along with following usage statement:

.

## CHAPTER 9

---

Home

---

Detailed installation instructions for anyone who wants to run biohansel. These instructions are tailored to linux or MacOS systems where biohansel has been confirmed to work correctly. If installing to the command line, Bioconda is highly suggested as it shows no issues over the span of many tests.

Pick the way that you want to run biohansel and follow the detailed instructions given on the following pages:

[Command line](#)

[Galaxy](#)

If errors persist in installation, the suggestion is to try another method. If errors occur in running biohansel, then check that all inputs are correctly specified, all files are in the correct formats (fasta/fastq for samples, .tsv for metadata tables), and no other small issues are present (ex. forgetting a column called “subtype” when adding in metadata).



# CHAPTER 10

---

## Command Line

---

There are three ways to install the latest version of biohansel to run analyses through the use of the command line. These instructions have been tested for linux terminal and thus, no comments on the other operating system instructions can be found here at the moment. MAC should be extremely similar once you install Miniconda following the MAC installation steps. biohansel commands have not been tested for the Windows OS terminal.

### 10.1 Biohansel on Miniconda Linux Installation Instructions

Miniconda is a mini version of [Anaconda](#) that includes only conda and its dependencies. If you wish to [install Anaconda](#) then follow the steps found on the Anaconda installation instructions page and join back at step 5.

The steps below will be the fully detailed for Linux operating systems. You can find installation guides for Miniconda for the other operating systems at: <https://conda.io/projects/conda/en/latest/user-guide/install/index.html>

1. Download Miniconda - Python 3.7 from [Conda's Download page](#)
  - Select the correct installer for your operating system (either 64-bit or 32-bit)
2. Path to where the miniconda package was downloaded and run the following command:

```
bash Miniconda-latest-Linux-x86_64.sh
```

You can use the tab key after typing the first few letters of the file to finish the rest.

3. Follow the prompts on the installer screens
4. Close and re-open the terminal window to make the changes take effect
5. Add BioConda to MiniConda with the following commands in the following order:

```
conda config --add channels defaults
conda config --add channels bioconda
conda config --add channels conda-forge
```

6. Create a new environment to run biohansel from. This environment is going to be called “biohansel” but can be whatever you choose. Command:

```
conda create -n biohansel python=3.6
```

7. Activate the newly created conda environment:

```
source activate biohansel
```

8. Check that the channels added earlier are properly included in the new environment by repeating the command in step 5. This can be skipped if step 5 went well:

```
conda config --add channels defaults
conda config --add channels bioconda
conda config --add channels conda-forge
```

The output will either be nothing if the channel is re-added, or it will say that the channel is already present. Either output is good.

9. Install biohansel and all of its dependencies into this environment:

```
conda install bio_hansel
```

10. Check that the installation was successful. Running the following command to display the usage statement:

```
hansel -h
```

11. When you open a new terminal window to run biohansel, remember to activate the environment you set it to before running a job or it will not work:

```
conda activate biohansel

# Then run the analysis you want to do. Example:
hansel -s heidelberg -o results.tab STR13341
```

If there are problems running/installing biohansel, check to see if any of the following are occurring:

1. Make sure that that all of the system requirements are met for Miniconda on this page: <https://conda.io/projects/conda/en/latest/user-guide/install/index.html#system-requirements>
2. Check that the right python version/installation is being used. It should be found under the /Miniconda3/bin/python3.6/ directory if installed with Miniconda:

```
which python
```

- If the wrong directory or python version is being run by the terminal, then try the following:
  - Restart the terminal window and check again
  - Use the following commands

```
alias python=python3
# This will set python 3 as the working python
```

(continues on next page)

(continued from previous page)

```
# Check that this worked with the command:
python --version
# Should print out the version as 3.x.x depending on which
↳ version is installed.
```

3. If using the Fish shell, make sure that you add the following line to your `fish.config` file if there are problems occurring:

```
source (conda info --root)/etc/fish/conf.d/conda.fish
```

## 10.2 Biohansel installation with pip from PyPI

If you have pip and python3 installed already onto your machine, then the following steps can be used to install biohansel. If not, follow along and install them as prompted:

1. Make sure that python3 is the active python version and that it is installed onto the machine:

```
python -V
# This will print the python version used

# If this doesn't output 3.X and instead outputs 2.X, then type:
alias python=python3

python -V
# Now it should output python version as 3.X.

# If not, then python3 may need to be installed with the following:
apt-install python3.7-minimal
```

Biohansel needs python3 to work correctly. If installed this way, you may need to use the alias command to get the correct version of python active before each run of biohansel.

2. Install biohansel with pip. If pip is not installed on your current machine, then follow the [installing pip tutorial](#):

```
# You can check that pip is installed with the input:
pip

# If pip is installed, then install biohansel with it
pip install bio_hansel

# This will install biohansel along with all of its needed dependencies.
```

3. Check that biohansel has been correctly installed with:

```
hansel -h
```

### Common problems encountered:

1. pip installing biohansel to the wrong python environment. Instead of installing to python 3, it installs to python 2.
  - Set the correct path for pip/python to install files
2. Make sure the correct version of python is being installed to (v3.x)

## 10.3 Biohansel installation with pip from Github

Use the following command:

```
pip install git+https://github.com/phac-nml/biohansel.git@master
```

If that doesn't work, look at the common problems encountered with pip from PyPI or try the PyPI installation instructions. Both installation methods are extremely similar.

## 11.1 Installation Instructions for New Galaxy Users

### 11.1.1 Requirements

- Unix/Linux or Mac OSX
- Python 2.7

### 11.1.2 Steps

If you just want biohansel or other programs to run on your own instance of Galaxy, then follow the steps below to download and install Galaxy. If you wish to work on Galaxy development or use it for another purpose other than installing and running Bioinformatic tools, then read more about [Galaxys uses](#).

For your own instance of Galaxy follow these steps:

1. Go to <https://galaxyproject.org/admin/get-galaxy/> and for our purposes clone the latest release of Galaxy to a Git repository

```
cd <repo directory>
# Change to a directory you want to clone Galaxy to

git clone -b release_18.09 https://github.com/galaxyproject/galaxy.git
# As of writing this, the current release of Galaxy is Version 18.09
# You can always update Galaxy or clone the master branch following the
↪instructions
# of the get-galaxy site
```

2. Galaxy will create config files once the server is run for the first time. Now start Galaxy with:

```
cd galaxy
# Move into the cloned galaxy directory
```

(continues on next page)

(continued from previous page)

```
sh run.sh
# This will run galaxy for the first time should everything be correctly
↳ installed.
# It will automatically create all necessary files
```

Once the start-up is finished, a Galaxy server on the local host will start.

- Galaxy can now be accessed from any web browser at <http://localhost:8080> . After starting, Galaxy output will be printed to the terminal window from the created galaxy server.

If any problems have occurred or you want to customize/modify settings, view the [Galaxy project's get Galaxy website](#) for greater details and instructions.

- Use `Ctrl-C` to shutdown Galaxy and then use the following command to add the sample config file:

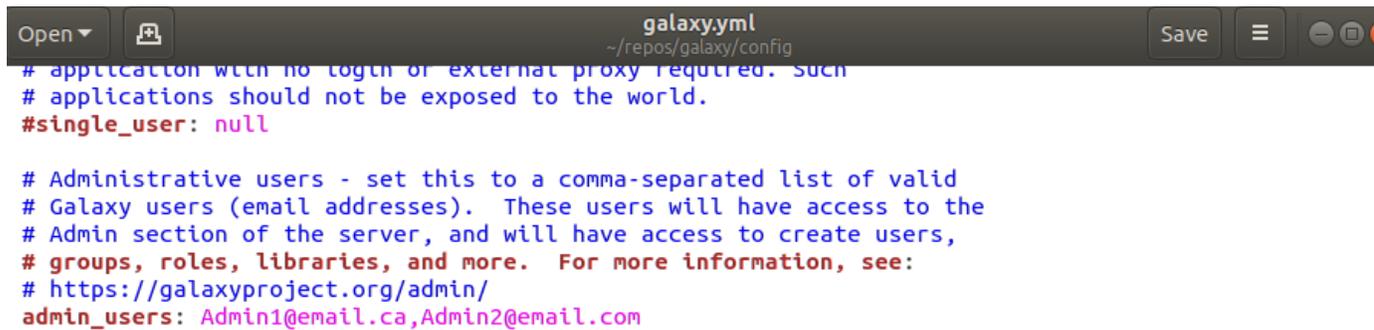
```
cp config/galaxy.yml.sample config/galaxy.yml
```

- Become an Admin so that you may add tools.

- Register with your Galaxy through the register button at the top

- Give Admin-user privileges for your login email to the configuration file `config/galaxy.yml`. Open the file a**

- **Make sure to uncomment (remove the #) the `admin_users` line**



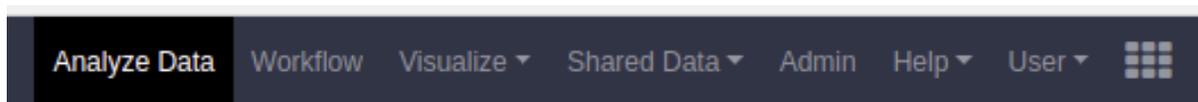
```
galaxy.yml
~/repos/galaxy/config
# application with no login or external proxy required. Such
# applications should not be exposed to the world.
#single_user: null

# Administrative users - set this to a comma-separated list of valid
# Galaxy users (email addresses). These users will have access to the
# Admin section of the server, and will have access to create users,
# groups, roles, libraries, and more. For more information, see:
# https://galaxyproject.org/admin/
admin_users: Admin1@email.ca,Admin2@email.com
```

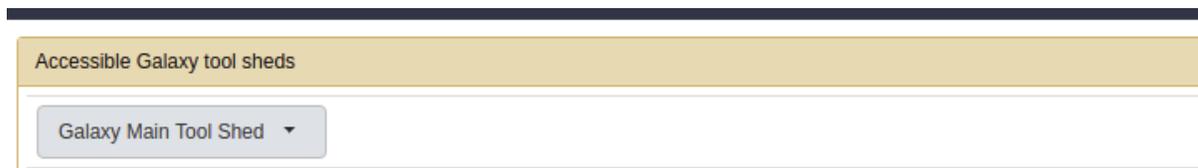
- Restart Galaxy on the terminal again with `Ctrl-C` and the start command:

```
sh run.sh
```

- Now to install biohansel, continue following along here or go to the Galaxy Admin tab at the top bar:



- On the side bar locate install new tools and click on it to show the following page:



- Click on the button and search the Valid Repositories for `biohansel` or `bio_hansel`. Currently, `biohansel` is the more up to date version available but any Galaxy version can be found here.

## Valid Repositories

biohansel ✕

Q

<u>Name:</u>	<u>Synopsis</u>
<div style="border: 1px solid #ccc; border-radius: 3px; padding: 5px; display: inline-block;"> <span style="margin-right: 5px;">biohansel</span> <span style="font-size: 0.8em;">▼</span> </div>	Heidelberg and Enteritidis SNP Elucidation

- Click on `biohansel` and a drop down menu will appear allowing you to preview/install. Click on this to be redirected to a page that will let you browse the repository for `biohansel`, or install it to Galaxy.
- Once you click on *Install to Galaxy*, you will be brought to a page to confirm the installation and the location that the tool can be found under in the toolbar. Set these to your preferences and then click install.
- Additional information on installing tools can be found on the [Galaxy website installing tools tutorial](#).

[link to biohansel repository](#)

## 11.2 Installation Instructions for Galaxy Admins

Follow along steps 7 - 11 of the previous section.

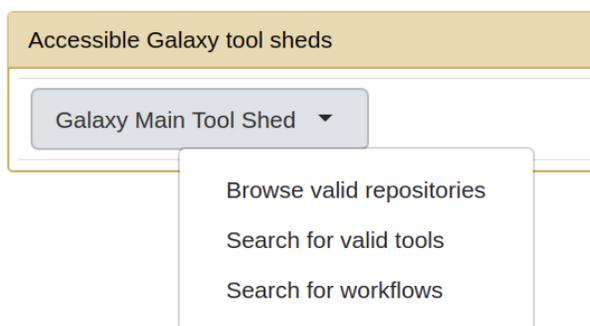
For additional information, see the [Galaxy website installing tools tutorial](#).



### 12.1 Galaxy versions

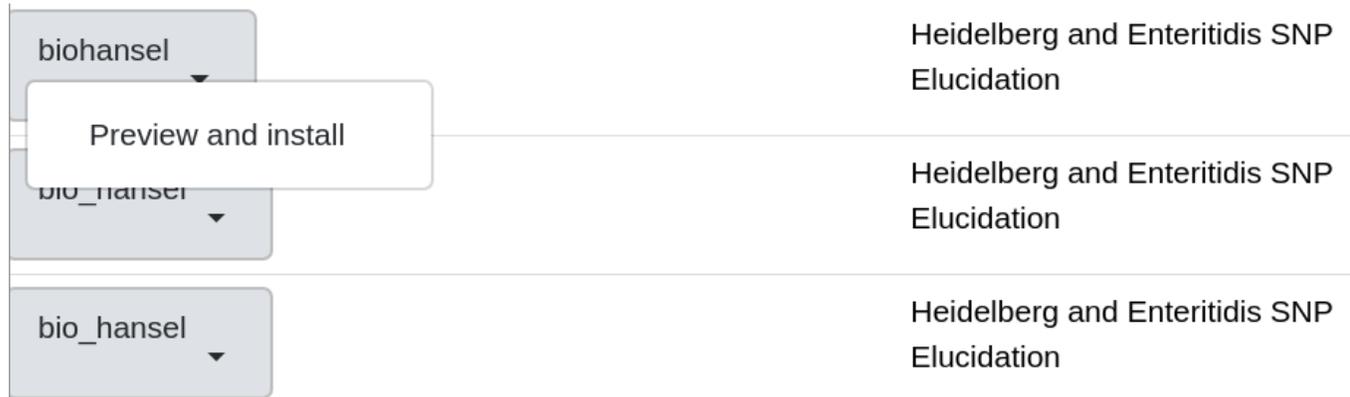
Using Galaxy, you can easily install older versions of biohansel to your own Galaxy workspace. To do this, follow the [Galaxy Installation Page instructions](#) to get Galaxy to run on your local machine. Once Galaxy is installed and you make yourself an admin, you can install any other versions of biohansel released to Galaxy with the following:

1. Click on the Admin button on the top bar to be brought to the Admin page.
2. On the Admin page, on the left side bar under “Tool Management”, click on Install new tools to be brought to the “Accessible Galaxy tool sheds”.
3. Once on this page, click the arrow on the “Galaxy Main Tool Shed” button and then click on the “Browse valid repositories” option.

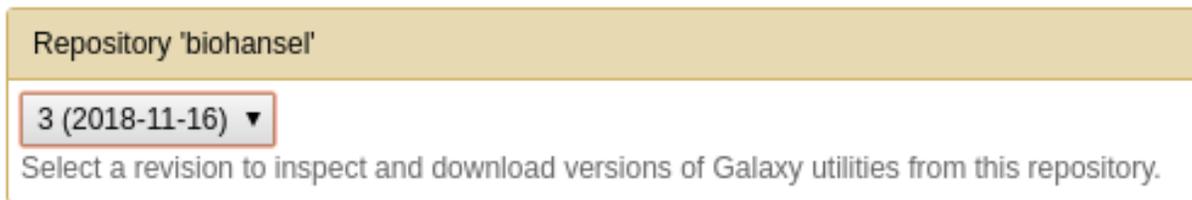


4. Search the valid repos for `biohansel` or `bio_hansel`. Each has different versions that can be installed.
  - `biohansel` can install versions from 2018-11-05 to present
  - `bio_hansel` can install versions from 2018-05-09 to 2017-09-27

Once you know what version you want, click on the “Preview and install” button after clicking the arrow on the biohansel button.



5. Choose the version you wish to install with the drop down menu and then click “install to Galaxy” in the top left corner.



6. Confirm the installation and then the version of biohansel specified will be installed into your local Galaxy

## CHAPTER 13

---

### Legal

---

Copyright Government of Canada 2017

Written by: National Microbiology Laboratory, Public Health Agency of Canada

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this work except in compliance with the License. You may obtain a copy of the License at:

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.



## CHAPTER 14

---

Contact

---

**Gary van Domselaar:** [gary.vandomselaar@canada.ca](mailto:gary.vandomselaar@canada.ca)