

Analyzing whole genome sequencing data of *Haemophilus* isolates

Author: Margo Diricks ([mdiricks@fz-borstel.de](mailto:mdiricks@fz-borstel.de) or diricks\_margo@hotmail.com)  
🡪 Please let me know if you need any help or if you find any errors! 😊

Contents

[1 Overall workflow 2](#_Toc77082382)

[2 Installation of required packages/tools 2](#_Toc77082383)

[3 Scripts and reference databases 3](#_Toc77082384)

[3.1 How to use 3](#_Toc77082385)

[3.2 Input and output files 4](#_Toc77082386)

[3.3 Parameters 5](#_Toc77082387)

[3.4 Resistance determination 6](#_Toc77082388)

[3.5 MLST analysis 6](#_Toc77082389)

[3.6 Classification and in silico serotyping 7](#_Toc77082390)

[3.7 Analysis of fuc operon 9](#_Toc77082391)

[3.8 De novo assembly 9](#_Toc77082392)

[3.9 Annotation 10](#_Toc77082393)

[3.10 Pan-genome and phylogenetic analysis 10](#_Toc77082394)

[4 References 11](#_Toc77082395)

[5 Supplementary materials 12](#_Toc77082396)

[5.1 Workflow to install and run linux programs on a windows (desktop) PC 12](#_Toc77082397)

[5.2 Some basic linux commands and tips 12](#_Toc77082398)

[5.3 Install miniconda 13](#_Toc77082399)

# Overall workflow

**Note1**: some steps are only relevant for *H. influenzae* (e.g. MLST and in silico serotyping).

**Note2**: so far it is unclear how to properly correct for recombination in *Haemophilus* phylogenies without losing too much resolution (Howell et al. 2014; Pinto et al. 2019).  
**Note3**: Mutational resistance is not included in this scheme because no proper tools exist yet (to the best of my knowledge) to determine this type of resistance.

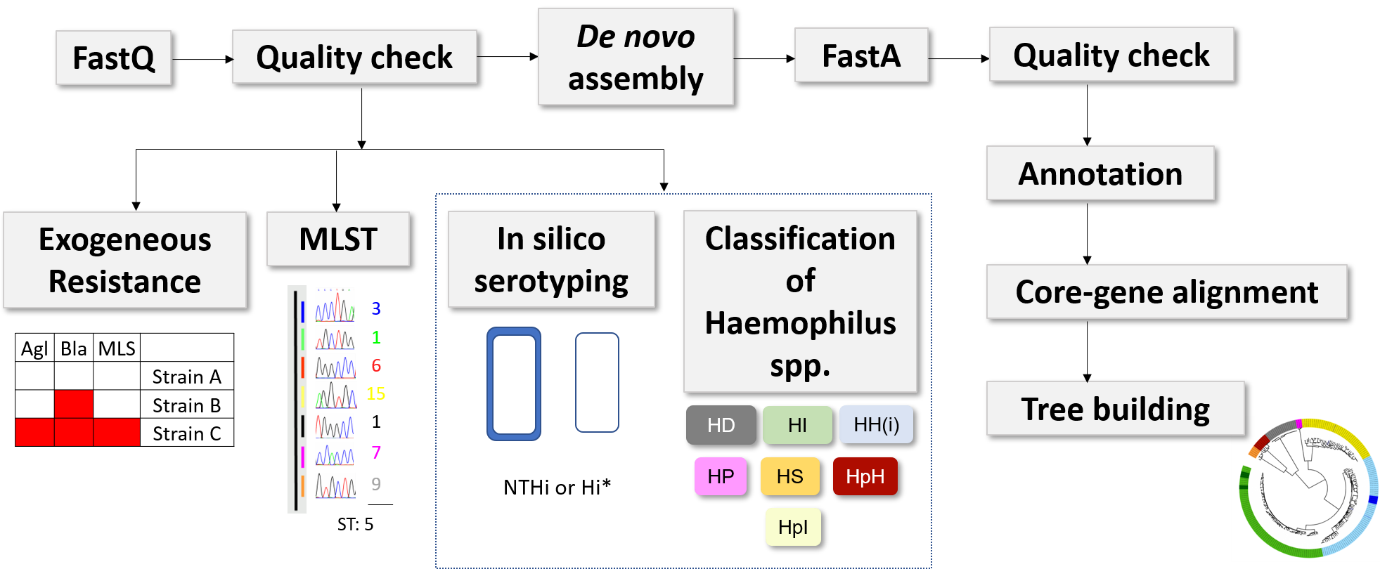


Figure 1 Example workflow for the analysis of whole genome sequencing data from Haemophilus isolates. MLST: multi-locus sequence typing. HI: Haemophilus influenzae; HI\*: serotypeable HI; NTHi: non-typeable HI; HH(i): Haemophilus haemolyticus (subsp. intermedius); HpH: Haemophilus parahaemolyticus; HpI: Haemophilus parainfluenzae; HP: Haemophilus pittmaniae; HS: Haemophilus sputorum; HD: Haemophilus ducreyi.

1. Perform a **quality check** of your WGS data (illumina paired-end fastQ files) e.g. with FastQC (general quality check of coverage, read length,...) and kraken2 (to check for contamination).
2. Determine in silico resistance, MLST, serotype and (sub)species (see below)
3. If you have multiple samples you can build a **phylogenetic** tree (e.g. with FastTree) using an assembly-based core-gene-alignment (e.g. generated by Roary) or mapping-based SNP aligment (e.g. generated by adjusted MTBseq pipeline). Visualize the tree (e.g. with iTOL). Plot presence/absence of marker genes and the final decision (e.g. serotype or species) on your tree using the itol annotation files that are automatically generated by the scripts described below.

# Installation of required packages/tools

If you need to start from scratch you can follow the steps below:

1. Make sure you have access to a linux environment. If you have a laptop/computer with Windows, you could e.g. use a virtual machine or the Windows subsystem for linux (see Supplementary materials section). My scripts were tested using ubuntu 20.04 LTS.
2. Open your ubuntu terminal (In Windows, type ubuntu in the search bar)



1. Install (mini)conda (only needs to be done once) (See Supplementary materials section).
2. Create a conda environment in which you will install your tool (e.g. SRST2) (only needs to be done once) and activate the environment (has to be done each time you want to use the tool/environment):

$conda create --name Analysis\_Haemophilus python=2.7

$ conda activate Analysis\_Haemophilus

1. Install **SRST2** (https://github.com/katholt/srst2) (only needs to be done once). Note that you will need to install it in an environment where python 2.7 is installed. Newer python versions (3 and higher) are not compatible with SRST2.

$ conda install -c bioconda srst2

*This tool detects genes directly from raw sequencing data*

*Note: the scripts were tested with SRST2 v0.2.0.*

1. Install **GNU parallel** (https://www.gnu.org/software/parallel/parallel\_tutorial.html) e.g. using conda (only needs to be done once)

$ conda install -c conda-forge parallel

If you have problems installing GNU parallel with conda (e.g. dependency problems) you can also install it using

$sudo apt install parallel

*This tool allows you to analyze multiple samples in parallel (decreases the computational time)*

*Note: the scripts were tested with GNU parallel v.20200722.*

# Scripts and reference databases

## How to use

Please copy/download the scripts/reference databases from github to a folder on your computer!  
Please install the required tools/packages (see above). If you have used conda to install tools, don´t forget to activate the environment before running my scripts!

Margo`s scripts are typically run using the bash command:

$bash /PathToScripts/script\_name.sh

## Input and output files

Save your input files (typically fastQ files and reference databases) in a folder and specify the paths to these folders in the script you want to use (see below).

The output files depend on the script. However, if the script uses SRST2, following files are automatically generated:

1. Files per sample:

**\*.Log** (Figure 2): log file that contains the user-defined parameters and info regarding the steps of the algorithm.



Figure 2 SRST2 log file for sample H-155 (opened in editor)

**\*\_\_fullgenes\_\_\*.txt** (Figure 3): this file contains all hits with a horizontal coverage > user-defined minimum coverage threshold. Note that this file also contain hits with divergence > user-defined maximum divergence cutoff.

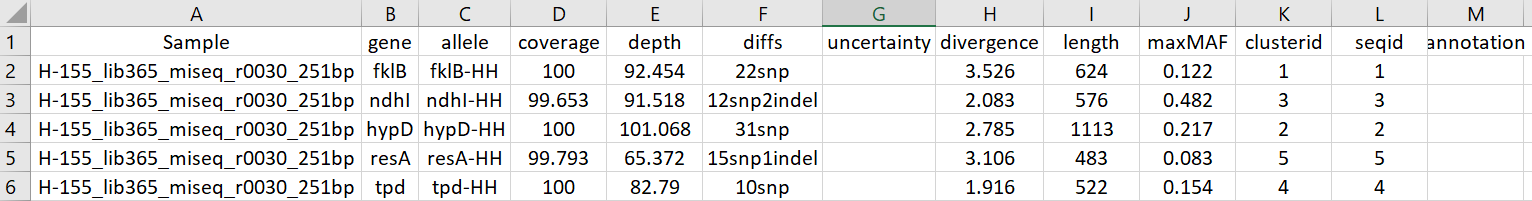


Figure 3 SRST2 fullgenes result file for sample H-155 (opened in excel)

**\*\_\_genes\_\_\*.txt** (Figure 4): this file contains for each loci one hit. All hits meet the criteria set by the user, i.e. horizontal coverage > user-defined minimum coverage threshold and divergence <= user-defined maximum divergence cutoff. Important note: if there are multiple hits meeting the criteria per loci, the last hit as listed in the \_\_fullgenes\_\_ file is used in the \_\_genes\_\_file. This is rather unexpected as one would expect the best hit (highest coverage + lowest divergence towards the reference allele) to be reported. This is highly likely a bug and was reported to the developer(s) in November 2020.

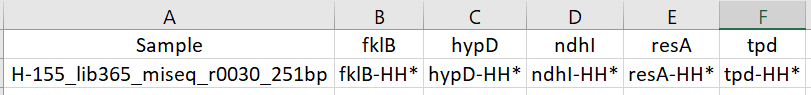


Figure 4 SRST2 genes summary result file for sample H-155 (opened in excel)

**SAM/BAM files and mpilup files:**

My scripts typically contain commands to automatically remove these files as they can take up quite some space

1. Combined file for all samples:

**\*\_\_compiledResults\_\_\*.txt** (Figure 5): this file combines the information from \*\_\_genes\_\_\* files into one summary file.

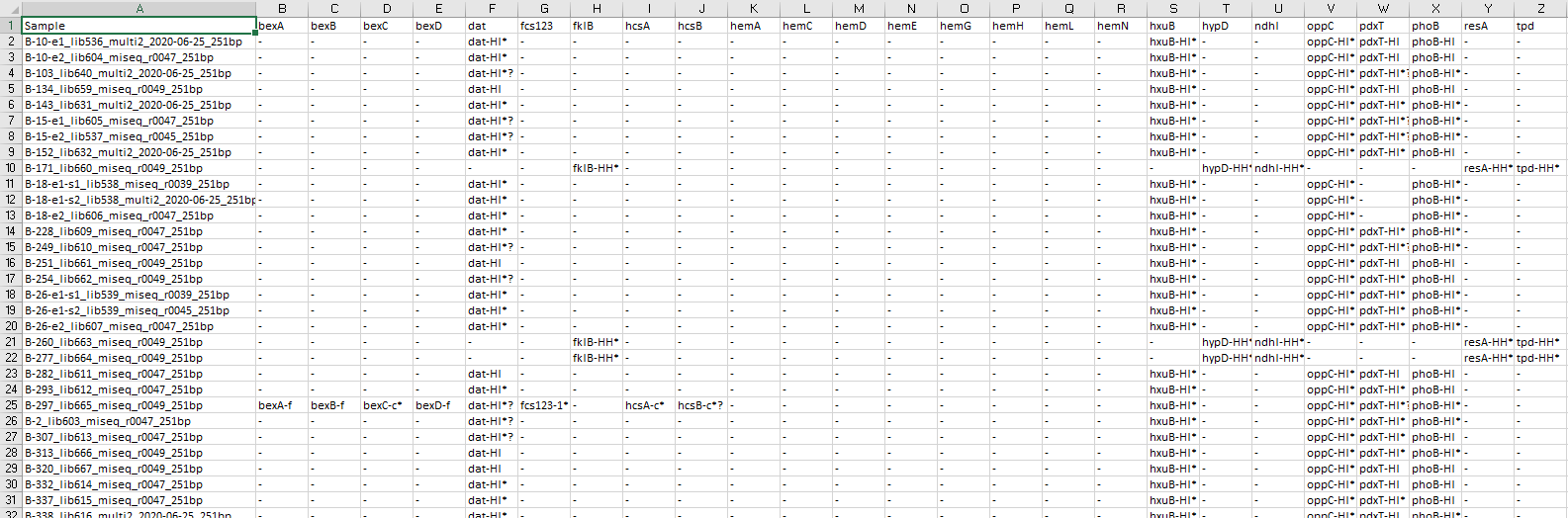


Figure 5 SRST2 compiled results file for all samples (opened in excel)

## Parameters

Always check if ALL parameters of the script are correct (everything before the actual code starts, so before the \*######\*), using e.g. notepad++.  
  
Typical parameters that need to be changed are:

1. **PATH\_input**: Directory where you stored the (raw) paired-end illumina fastQ files e.g. "$HOME/FastQ". Note that when you are working on a windows machine, you need to mount your windows folder e.g. if your FastQ files are on your Desktop then you need to use something like PATH\_input="/mnt/c/Users/diricks/Desktop/Folder\_Where\_FastQ\_Files\_Are\_Stored” in which you change the “diricks” to your own name.
2. **PATH\_output**: Directory where the final files will be stored e.g. "$HOME/Results"
3. **Threads/cpus**: Amount of threads/cpus that can be used to perform calculations. This depends on the performance of your computer/server (e.g. a laptop with a 4-core processor that runs two threads per core, has 8 logical processors 🡪 set threads ≤ 8). Typically, if you put this number on 0, you use all available resources. However, sometimes you get errors or run into problems when Threads/cpus are set at 0, therefore I prefer to set it manually (e.g. to 8).

Typical parameters that need to be changed unless your file name follows this convention „SampleName\_R1.fastq.gz” are:

* **FastqType:** Extension of your fastq file e.g. ".fastq.gz"
* **Fw**: Forward read notation e.g. "\_R1"
* **Rv:** Reverse read notation e.g. "\_R2"

Specific parameters for **SRST2** are:

* **geneDB**: Path to nucleotide reference database e.g. "$HOME/Database/Refdatabase.fasta" (note that you have downloaded this database from my github repository, so use the path were you saved this file)
* **GeneMaxMis** (only if you don´t want to use default values): M
* **MaxDiv** (only if you don´t want to use default values): D
* **MinCov** (only if you don´t want to use default values): C

For *Haemophilus*, I use by default 50M-15D-90C (instead of default SRST2 values 10M-10D-90C). However, if you want to detect very small contaminations or truncated genes, one can e.g. use 200M-80D-40C. Note that decision rules defined in the scripts do not necesserily apply anymore using these relaxed conditions.

* Additional parameters (see bottom of script) can also be added (e.g. minimum depth,…).

Specific parameters for **GNU parallel** are:

1. **Jobs:** Amount of samples to be analyzed in parallel. Similar to threads/cpus. Jobs=0 means you will run as many jobs in parallel as possible (depends on your computational resources, but sometimes leads to problems, so better set manually).

**IMPORTANT**: If it is the first time you use the script/database, you need to run this script ones with jobs=1. This is necessary because the first time, fai and bt2 files (required for bowtie2) of the reference database still need to be created. Errors occur when you analyze multiple samples in parallel during this step. Afterwards, you can set it to the value that best matches your computers` performance.

1. **PATH\_temp**: path were temporary files are stored (can be important to change if there is not enough space in your folder)

## Resistance determination

**starter\_Resistome\_SRST2.sh**

Determines the presence/absence of resistance genes (exogeneously acquired resistome) included in a resistance database using SRST2. A resistance database needs to be downloaded/uploaded for this script (see <https://github.com/katholt/srst2>).

INPUT:

* Raw paired-end illumina fastQ files (\*\_R1 and \*\_R2.fastq.gz)
* Resistance database (\*.fasta)

OUTPUT:

* SRST2 output files

## MLST analysis

**starter\_MLST\_SRST2.sh**

Determines the MLST type starting from raw sequencing data with SRST2. A MLST reference database needs to be downloaded/uploaded for this script (see e.g. <https://github.com/katholt/srst2>).

INPUT:

* Raw paired-end illumina fastQ files (\*\_R1 and \*\_R2.fastq.gz)
* MLST database (\*.fasta)
* ST definition file (\*.txt)

OUTPUT:

* SRST2 output files

## Classification and in silico serotyping

**starter\_Classification\_Of\_Haemophilus\_And\_Serotyping\_SRST2.sh**

This script can be used to detect and differentiate between all relevant human-related *Haemophilus* species: *Haemophilus parahaemolyticus* (HpH), *Haemophilus ducreyi* (HD), *Haemophilus parainfluenzae* (HpI), *Haemophilus sputorum* (HS), *Haemophilus pittmaniae*, typeable and non-typeable *Haemophilus influenzae* (HI), putative X-factor dependent and independent *Haemophilus haemolyticus* (HH). It cannot be used to classify “fuzzy” species with aberrant marker patterns. Although it can detect samples with multiple *Haemophilus* species to a certain extent, it cannot detect the presence of multiple strains from same species.

ESTIMATED time:

Using an ASUS ZenBook 14 laptop with an intel core i7 CPU, 8 logical processors and 16 Gb RAM. Tested on simulated fastQ reads (100x coverage depth) of a set of 211 *Haemophilus* isolates downloaded from NCBI

* 44 min for whole set when 8 samples were run in parallel ($jobs=8, parameter of GNU parallel). This is the maximum amount of samples that can be run in parallel as the laptop only has 8 logical processors.
  + Average of 1.6 min per sample (min=47 sec, max=3 min)
* 110 min for whole set when only 1 sample is run at the time ($jobs=1). This would be similar to running SRST2 without GNU parallel.
  + Average of 30 sec per sample (min=20 sec, max=1 min)

MEMORY requirements:

Max. 8 Gb of RAM is required to run this script.

INPUT:

* Raw paired-end illumina fastQ files (\*\_R1 and \*\_R2.fastq.gz)
* Reference database (Class\_Haemophilus\_and\_Serotyping.fasta)

OUTPUT:

**SRST2 output files:**

See section 3.2.

**Classification\_Serotyping\_SRST2\_itol\_Decision\_Species.txt** (Figure 6)**:** this file can be dragged and dropped over an itol tree (https://itol.embl.de/) to visualize the species detected.

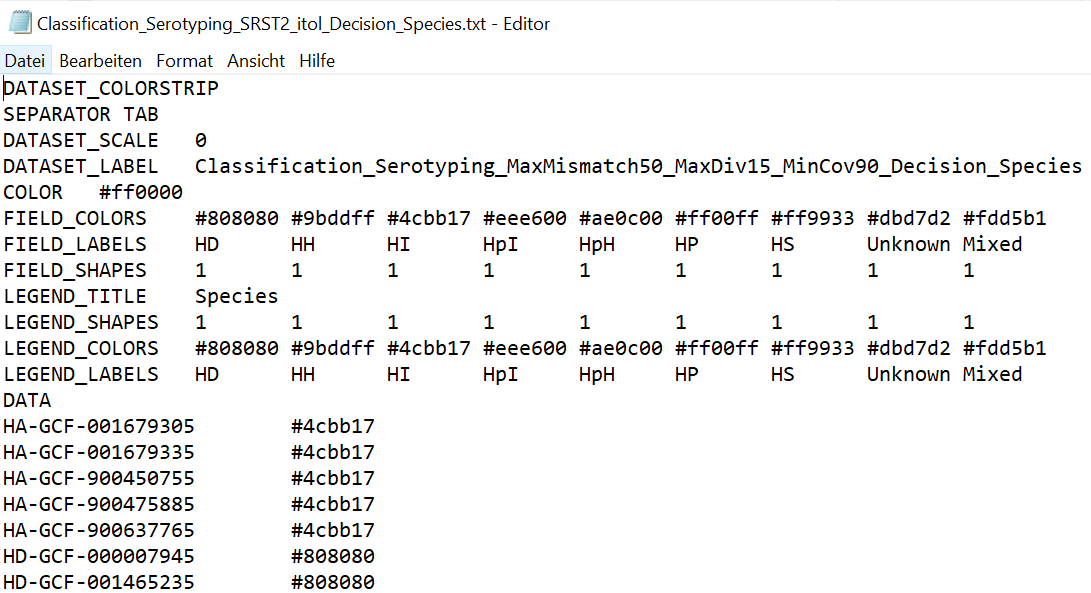


Figure 6 Classification\_Serotyping\_SRST2\_itol\_Decision\_Species example output (opened in editor)

**Classification\_Serotyping\_SRST2\_itol\_Binary.txt** (Figure 7): this file can be dragged and dropped over an itol tree to visualize the presence/absence of the genes)

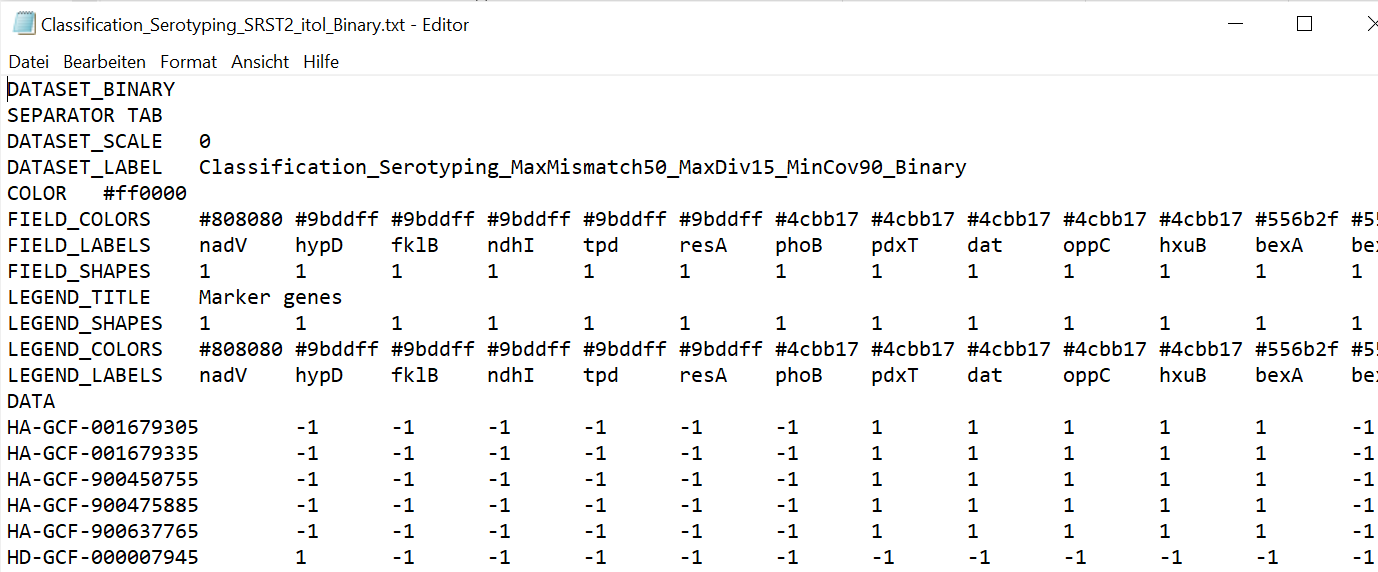


Figure 7 Classification\_Serotyping\_SRST2\_itol\_Binary example output (opened in editor)

**Classification\_Serotyping\_finalDecision.txt** (Figure 8): contains the species and serotype of your strain in a simple text file. Legend is included in the textfile.

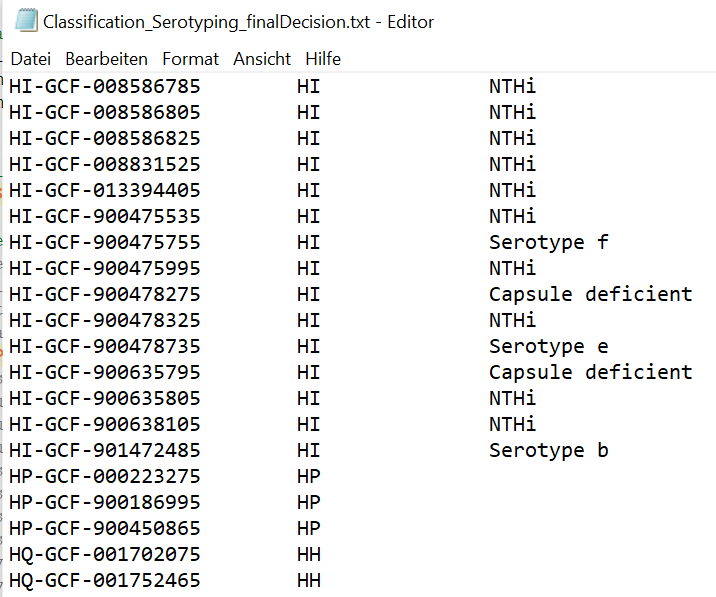


Figure 8 Classification\_Serotyping\_finalDecision example output file (opened in editor)

**Classification\_Serotyping\_SRST2\_itol\_Decision\_Serotype.txt** (Figure 9): this file can be dragged and dropped over an itol tree to visualize the serotype of *H. influenzae* strains.

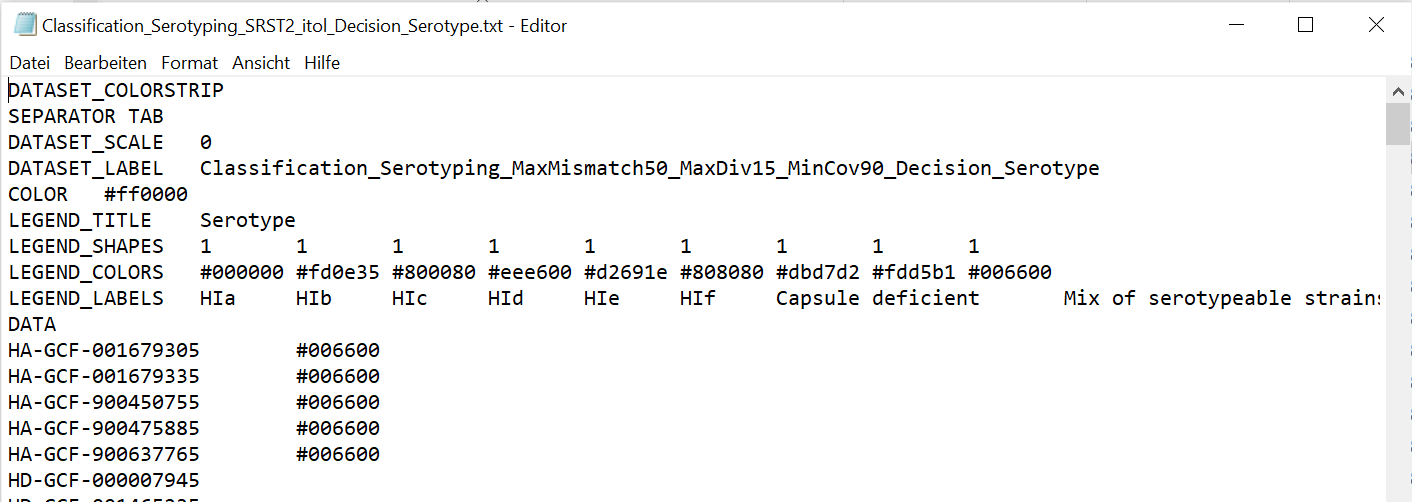


Figure 9 Classification\_Serotyping\_SRST2\_itol\_Decision\_Serotype example output file (opened in editor)

**Classification\_Serotyping\_SRST2\_itol\_Decision\_Subspecies.txt** (Figure 10): this file can be dragged and dropped over an itol tree to visualize the *H. haemolyticus* subsp. *intermedius* strains = putative haemin/X-factor-independent.

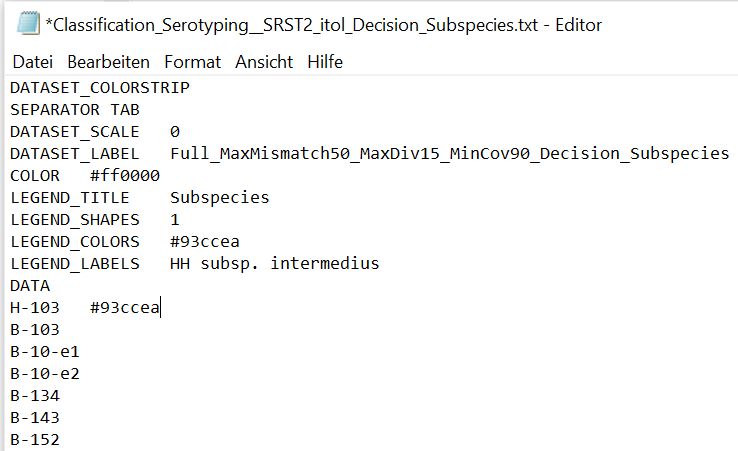


Figure 10 Classification\_Serotyping\_SRST2\_itol\_Decision\_Subspecies example output file (opened in editor)

## Analysis of fuc operon

**Starter\_FucOperon\_SRST2.sh**

This script determines the presence/absence of the six fucose genes (fucK, fucR, fucI, fucA, fucU and fucP) using SRST2.

INPUT:

* Raw paired-end illumina fastQ files (\*\_R1 and \*\_R2.fastq.gz)
* Reference database (fuc\_operon.fasta)

OUTPUT:

* SRST2 output files
* Failed.txt (lists the samples for which an unexpected error occurred)
* OK.txt (lists the samples for which no unexpected error occurred)
* Fuc\_SRST2\_itol\_Binary.txt (this file can be dragged and dropped over an itol tree to visualize the presence/absence of the genes)

## De novo assembly

**starter\_assemble\_Shovill.sh**

This script allows you to make assemblies using the Shovill tool (https://github.com/tseemann/shovill).

Tip: Start with spades as assembler and change it to skesa if this is not working.

Note: If you use the function trimming, you trim with following trimmomatic parameters: LEADING:3 TRAILING:3 MINLEN:30 TOPHRED33. This will remove Illumina adapters, remove leading and trailing low quality or N bases (below quality 3) and will drop reads which are less than 30 bases long after these steps.

INPUT:

* Raw paired-end illumina fastQ files (\*\_R1 and \*\_R2.fastq.gz)

OUTPUT:

* Per sample: Shovill output files (e.g. log files and final assemblies in fasta format)
* For all samples: shovillOutput\_summary.txt (which can be used as a first quality check)

## Annotation

**starter\_Annotate\_Prokka.sh**

This script performs genome annotation using prokka (<https://github.com/tseemann/prokka>).

INPUT:

* Genome (.fasta)

OUTPUT:

* Annotated genome (.gff)

## Pan-genome and phylogenetic analysis

**starter\_Pan-genomeAnalysis-Phylogeny\_Roary-FastTree.sh**

This script calculates a pan and core-genome with Roary and creates a phylogenetic tree file using FastTree.

Note: To prevent over-splitting (Bayliss et al. 2019) of homologues genes into different clusters and thus to increase the size of the core (gene alignment), one can do two things:

* You can turn of paralogue splitting using the -s parameter
* Using a reduced blastP identity of e.g. 60% instead of 95%.

# References

Bayliss SC, Thorpe HA, Coyle NM, Sheppard SK, Feil EJ. 2019. PIRATE: A fast and scalable pangenomics toolbox for clustering diverged orthologues in bacteria. *Gigascience* **8**: 1–9.

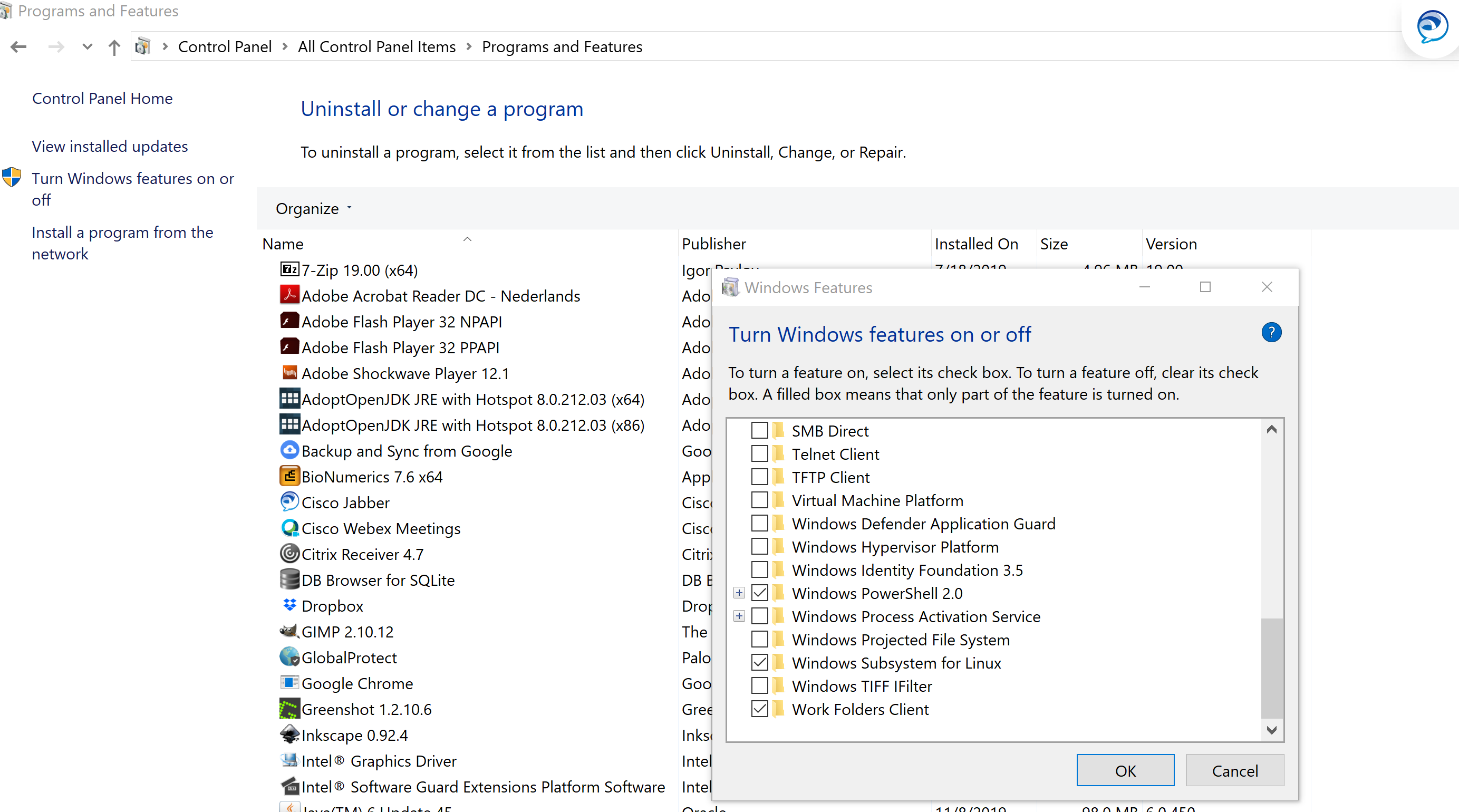
Howell KJ, Weinert LA, Chaudhuri RR, Luan SL, Peters SE, Corander J, Harris D, Angen Ø, Aragon V, Bensaid A, et al. 2014. The use of genome wide association methods to investigate pathogenicity, population structure and serovar in Haemophilus parasuis. *BMC Genomics* **15**: 1179.

Pinto M, González-Díaz A, Machado MP, Duarte S, Vieira L, Carriço JA, Marti S, Bajanca-Lavado MP, Gomes JP. 2019. Insights into the population structure and pan-genome of Haemophilus influenzae. *Infect Genet Evol* **67**: 126–135.

# Supplementary materials

## Workflow to install and run linux programs on a windows (desktop) PC

1. Activate Windows subsystem for linux in control panel > Programs and features > Turn Windows features on or off (only once)



1. Install latest version ubuntu (linux environment) on your PC from microsoft store (sometimes only works when VPN is disabled): e.g. Ubuntu 20.04 LTS
2. Open Ubuntu terminal (you can search for it in start or make a desktop icon)



1. Install whatever you want (e.g. see below).

## Some basic linux commands and tips

* In this document, code is preceded by “$” and visualized as grey blocks. In your ubuntu terminal, you will also see a $ sign, so no need to type this character before your code.
* By default you will be in your home folder e.g. mdiricks@my-molmyc65 (username ubuntu@computername) when starting the terminal. This corresponds to windows explorer directory C:\Users\diric\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu20.04onWindows\_79rhkp1fndgsc\LocalState\rootfs\home\mdiricks
* Avoid using folders with spaces or - in their names; always use \_
* When you are typing folders names in the ubuntu terminal (e.g. for the cd command see below), you can use tab to auto-fill (so you don´t need to type the folder name manually)
* Show path to working directory:

$pwd

* Change directory:

$cd /path/to/directory

* Returning to your home folder:

$cd or $cd $HOME or $cd ~

* To check free space:

$df -h

* Making new directories:

$ mkdir

* Listing the folders in your current folder:

$ls /home/folder

* Commands you repeatedly use can be combined in a bash script: open a new text document in a folder, give it a name and extension .sh eg ExampleFile.sh. Edit the file with notepad++ but make sure you first put it in linux mode by pressing ´edit > EOL conversion > Unix. Open ubuntu, cd to folder where shell script is located and type

$bash ExampleFile.sh

or alternatively from within any folder type

$bash /path/to/file/ExampleFile.sh

* Usage of the $ in a command like $(echo foo) means run whatever is inside the parentheses in a subshell and return that as the value.
* |: pipeline. This character indicates that the part left of it is used as input for part right of it. This allowys for communication between processes
* ; : Permits putting two or more commands on the same line.
* You can edit files (e.g.) bashrc file using a linux texteditor e.g vim (or nano):

$vim ~/.bashrc

* If you want to install/download programs or databases, put off your VPN otherwise it will be too slow!
* Downloading of scripts or databases from github: cd to location where you want to put file; Copy link from github; $wget and paste link 🡪 DOES NOT WORK if you copy the link on your PC and past it on remote linux server! In that case you download the html file rather then the true file and you will get an error <!DOCTYPE html> when you try to execute the file. Instead, download the full zip folder of the tool on the start site
* Wget is only downloading a single URL, git clone is a whole version control system that downloads or updates a directory

## Install miniconda

Packages/tools can be installed in several ways. I like to use e.g. miniconda (<https://docs.conda.io/en/latest/miniconda.html>)

For the scripts/tools I use, a miniconda installation (instead of anaconda) is more then enough.

For some basic conda commands, check the cheatsheet: https://docs.conda.io/projects/conda/en/4.6.0/\_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf

1. Open ubuntu terminal
2. Get the installation file

Go to <https://repo.anaconda.com/miniconda/> and check which version you want. Typically you take last version, but if you would get any errors, try installing a previous version. A version that worked for me last time was [Miniconda3-4.7.12-Linux-x86\_64.sh](https://repo.anaconda.com/miniconda/Miniconda3-4.7.12-Linux-x86_64.sh).

$wget https://repo.anaconda.com/miniconda/Miniconda3-4.7.12-Linux-x86\_64.sh

1. Run the installation script:

$bash Miniconda3-4.7.12-Linux-x86\_64.sh

1. Read the license agreement and follow the prompts to accept. Install in directory you want  
   When it asks you if you'd like the installer to prepend it to the path, say yes. Optionally install VS Code when prompted.
2. Close the terminal and reopen it (to reload .bash configs).
3. To test that it worked, run

$which python

It should print a path that has miniconda in it. Mine is /home/mdiricks/miniconda3/bin/python.

#If it doesn't have miniconda in the path, do the next step. Manually add the miniconda bin folder to your PATH. To do this, I added "export PATH=/home/diric/miniconda3/bin:$PATH" to the bottom of my ~/.bashrc file. (change „/home/diric/“ by the folder where you installed miniconda3).

1. Update conda to latest version

$conda update conda

1. Create new environments, activate them and install programs. I prefer to create a new environment every time I install a new tool or set of tools. E.g.

$conda create --name Analysis\_Haemophilus python=2.7

$ conda activate Analysis\_Haemophilus

$ conda install -c bioconda srst2

Or

$conda create --name shovill

$conda install -c conda-forge -c bioconda -c defaults shovill

$shovill –check