

---

# **GenPipes**

***Release 3.6.2 ( 0.9 draft )***

**Shaloo Shalini (GSoD 2019) GenPipes**

**Oct 16, 2022**



## GENERAL

<b>1</b>	<b>Welcome to the GenPipes Documentation!</b>	<b>3</b>
----------	---	----------





---

**Note:** GenPipes is sponsored by [Canadian Center for Computational Genomics \(C3G\)](#).

C3G has created a distributed innovation node with broad expertise in bioinformatics. It offers bioinformatics analysis and HPC services for the life sciences research community. These services include customized and case-by-case analysis, along with an extensive suite of software solutions for the genomics community.

---



## WELCOME TO THE GENPIPES DOCUMENTATION!

GenPipes is a flexible Python-based framework that facilitates the development and deployment of multi-step genomic workflows, optimized for High-Performance Computing (HPC) clusters and the cloud. It offers 12 open source, validated and scalable *pipelines* for various *genomics applications*.

---

### Tip: CoVSeq Pipeline - Fighting COVID-19

GenPipes offers CoVSeq pipeline to help researchers sequence and detect mutations quickly to prevent the spread of new strains. See *GenPipes CoVSeq Pipeline User Guide*, for details.

---

GenPipes documentation is organized to address the needs of new users as well as seasoned users and contributors. Refer to the *Documenatation Map* for details on how GenPipes documentation is organized.

Following is the **Table of Contents** of GenPipes documentation.

## 1.1 About

GenPipes is an open source Python-based Workflow Management System (WMS) for Next Generation Sequencing (NGS) genomics pipeline development.

As part of its implementation, GenPipes includes a set of high-quality, standardized analysis pipelines, designed for high-performance computing (HPC) resources and cloud environments.

GenPipes pipelines have been tested, continuously improved through industry standard benchmarks, and used extensively over the past 4 years. By combining both WMS and extensively validated end-to-end genomic analysis workflows, into a simple tool, GenPipes not only enables bioinformatics professionals but also students, researchers in need of bioinformatics tools to perform turnkey analyses for a wide range of bioinformatics applications in the genomics field. It also allows flexible and robust extensions for advanced genomics research.

You can learn more about GenPipes, its target audience, sponsors, how the project has evolved over the years and [license](#) information in the following section.

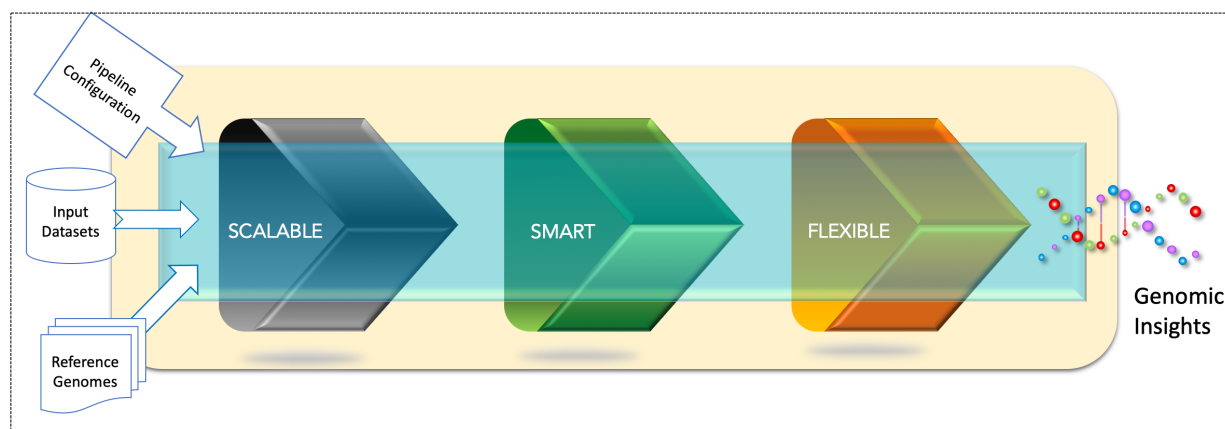
### 1.1.1 Why GenPipes?

Genomic sequencing has become an indispensable tool for modern bioinformatics researchers in their quest to understand biological processes. Next generation sequencing (NGS) is not only complex but is also compute and data intensive. It requires efficient handling of high performance computing infrastructure, managing scalability, flexibility and capability to handle massive amounts of genome reference data while managing intermediate results and inter-dependency between several serial and parallel analysis processes.

GenPipes is a python based bioinformatics tool comprising of several pre-built pipelines that addresses genomic analysis needs for bioinformatics researchers. It is available for public use as open source software. GenPipes was developed at the Canadian Centre for Computational Genomics (C3G). It offers a wide array of genomic sequencing pipelines including RNA-Seq, ChIP-Seq, Whole Genome Sequencing (WGS), Exome sequencing, Bisulfite sequencing, Hi-C, capture Hi-C, metagenomics and SARS-CoV-2 genome sequencing pipeline.



End-to-end WMS for genomic analysis with pre-built sequencing pipelines



### GenPipes Features

- **Multiple Schedulers** - GenPipes is optimized for HPC processing. Currently, it supports 4 schedulers - Slurm, PBS/Torque, Batch, Daemon.
- **Optimal Job Execution Time** - GenPipes minimizes overall job analysis time by job dependency model that leverages job parallelism. This enables jobs to become active and executed as soon as the dependencies are met.
- **Smart Relaunching of Jobs** - Through smart tracking of job progress, GenPipes can determine which jobs failed and at which steps. This helps to relaunch the jobs at the exact point in time, just before the last failure, automatically.
- **Parameter Encapsulation** - GenPipes is a flexible framework that allows user adjustments. It implements a superposed configuration system to reduce the time required to set-up or modify parameters needed during the analysis.
- **Diverse Inputs** - GenPipes is flexible in terms of multiple choice of input files for the analysis. It allows users to skip specific steps in the pipeline if they consider them unnecessary.
- **Customizable Workflows** - GenPipes limits wastage of expensive HPC resources and time as it allows customizable steps in different pipelines enabling users to plug and play with customized pipeline steps.

## Key Differentiators

GenPipes is different from other analysis platforms, workbenches and workflow management systems (WMS) in terms of the following capabilities:

1. **Flexibility:** Easy to modify and configure, multiple type of deployments available – local (containerized), cloud (GCP) and hosted on Compute Canada data centre, support for multiple job schedulers
2. **Scalability:** Optimized for large scale data analysis, simple to scale up or down in terms of processing and data access needs.
3. **Built-in Pipelines:** Pre-built, tested on multiple computing infrastructures, robust industry standard benchmark driven and production quality genomic analysis pipelines for various bioinformatics analyses.

## Comparing GenPipes with other NGS Solutions

GenPipes' strength lies in its robust WMS that comes with one of the most diverse selection of analysis pipelines that have been thoroughly tested. The pipelines in the framework cover a wide range of sequencing applications. The pipelines are end-to-end workflows running complete bioinformatics analyses. While many available pipelines conclude with a bam file or run limited post-bam analysis steps, the pipelines included in GenPipes are extensive, often having as many as 40 different steps that cover a wide range of post-bam processing.

For a tabular comparison of available solutions for NGS [see here](#).

GenPipes is compatible with HPC computing, as well as cloud computing, and includes a workflow manager that can be adapted to new systems. GenPipes also provides job status tracking through JSON files that can then be displayed on a web portal (an official portal for GenPipes will be released soon). GenPipes' available pipelines facilitate bioinformatics processing, while the framework makes it flexible for modifications and new implementations.

GenPipes developers offer continuous support through a Google forum page and a help desk email address ([pipelines@computationalgenomics.ca](mailto:pipelines@computationalgenomics.ca)). Since the release of version 2.0.0 in 2014, a community of users has run GenPipes to conduct approximately 3,000 analyses processing 100,000 samples.

To learn more about what GenPipes is and how it works, refer to the *Getting Started Guide*.

### 1.1.2 Target Audience

GenPipes is primarily geared towards Next Generation Sequencing analysis.

There are two kinds of target audience of GenPipes documentation:

1. End-Users
2. Developers

End-Users

Developers

Researchers and analysts working in bioinformatics and gene-sequencing computational processing. GenPipes strength lies in its simple interface that makes it very easy for students and researchers in need of sophisticated, yet easy to use bioinformatics workflow management tool with built-in pipelines for various genomic analyses.

Software experts and implementers who would like to improve and enhance the GenPipes framework.

Our key focus in GenPipes documentation efforts is to ensure that onboarding GenPipes is intuitive and easy for beginners and also for seasoned users such that they can quickly figure out where to look if they need information on a specific feature or GenPipes capability.

### 1.1.3 Sponsors

GenPipes is developed by Canadian Center for Computational Genomics (C3G). It offers genomic researchers a simple method to analyze different types of data, customizable to their needs and resources, as well as the flexibility to create their own workflows.

C3G provides bioinformatics analysis and HPC services for the life science research community. It has created a distributed innovation node with broad expertise in bioinformatics, providing customized and case-by-case analysis services as well as an extensive suite of software solutions for the genomics community.

#### About C3G

C3G is an academically-affiliated entity that provides bioinformatics analysis and HPC services for the life science research community. Main objectives of C3G are:

1. Writing Open Source software to facilitate bioinformatics analysis. See [here](#) for details.
2. Providing [data analysis services](#) to the genomics community.
3. Training graduate students and researchers through [bioinformatics workshops](#).
4. Providing support for large scale genomics projects like [IHEC](#), [BRIDGET](#).

You can learn more [about C3G](#) and their [partners](#) on the [website](#).

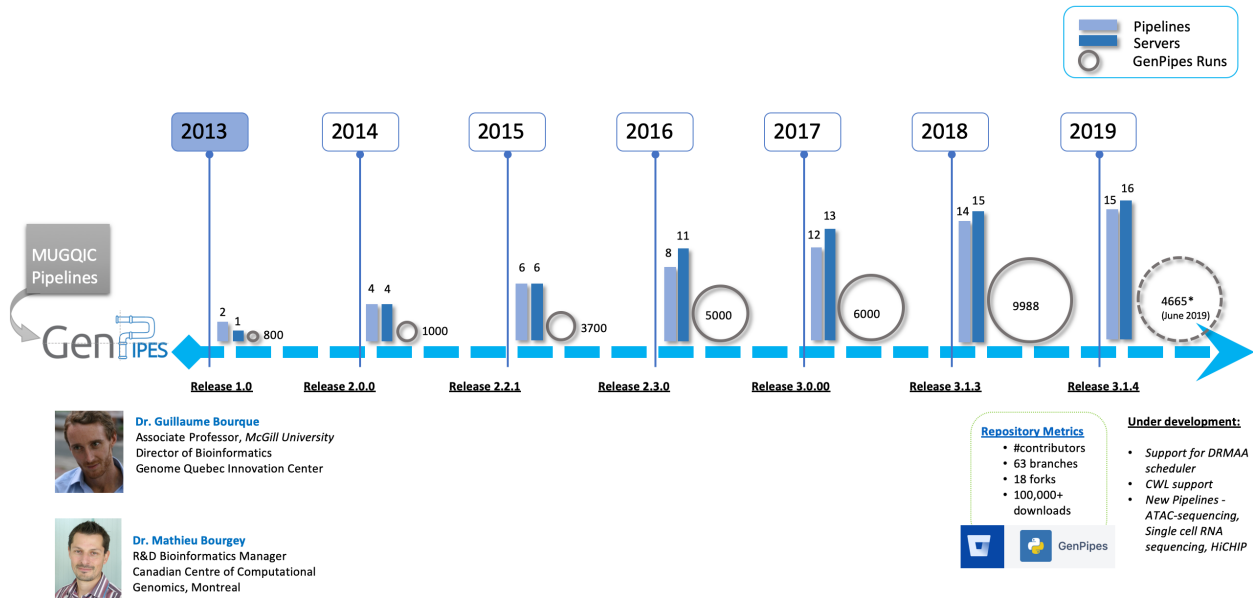
### 1.1.4 History

GenPipes project evolved from the McGill University and Génome Québec Innovation Centre (MUGQIC) genomic analysis project called MUGQIC Pipelines. The centre focuses on genomics applied to populations and its impact on chronic diseases, aging, cancer and genomic responses to environment.

GenPipes is the result of the centre's initiative to develop frameworks that can be used for whole-genome sequencing, as the primary tool to perform genetic analyses in humans and other species.

The following figure shows how GenPipes has evolved over the years, key releases in its evolutionary timeline and the upcoming new features under development. It also shows how the framework itself has grown from supporting only a few pipelines at the start to 14+ genomic analysis pipelines. The number of GenPipes runs utilizing C3G HPC resources have grown significantly as well. Since the release of version 2.0.0 in 2014, a community of users has run GenPipes to conduct approximately 3,000 analyses processing 100,000 samples.

The status of the repository is also indicated in terms of active branches, forks, contributors and total downloads of the open source.



## 1.1.5 License

GenPipes is an open source framework freely distributed and open for external contributions from the developer community. [GenPipes software](#) is available under a [GPLv3](#) open source license and is continuously updated to follow recent advances in genomics and bioinformatics.

For details regarding GenPipes license, refer to [GenPipes License Terms](#).

## 1.1.6 Release Notes

You can refer to the detailed list of changes for past and current releases of GenPipes through the [Release Notes](#).

## 1.1.7 Getting Help

GenPipes is under active development. If you need help, would like to contribute, or simply want to talk about the project with the team members, we have a number of open channels for communication.

To report bugs or file feature requests: use the [GenPipes issue tracker](#). For details, see the [GenPipes Support Page](#).

To talk about the project with people in real time: join the [GenPipes Channel on Google Groups](#). For more details refer to the [Channels](#) page.

To contribute code or documentation changes: submit a pull request on [GenPipes BitBucket](#).

For citations, publications, workshops, test data sets and other resources, visit [Resources](#) page.

## 1.2 GenPipes Real-life Applications

GenPipes offers several *genomic analysis pipelines* that can be used for various bioinformatic analyses.

These can be utilized for performing a wide range of standardized and tailored analysis. Following is a brief summary that highlights potential use case for each of the GenPipes Pipelines.

C3G has extensive experience analyzing data from various sequencing applications with state of the art computation methods. It offers various *services* to next-generation sequencing researchers who wish to use these pipelines for advanced genomic analysis. For customized and case-by-case analysis service, [submit your request today](#).

---

Following is a summary of real-life application use case for each of the GenPipes Pipelines:

### 1.2.1 COVID-19 genome sequencing to detect mutations

GenPipes *CoVSeQ Pipeline* has been used as part of CoVSeQ project. CoVSeQ deals with real-time tracking of Quebec SARS-CoV-2 evolution.

CoVSeQ is a partnership between the [Institut National de Santé Publique du Québec \(INSPQ\)](#) and the [McGill Genome Center](#) to sequence the viral genome of Quebec patients with COVID-19 disease. The viral samples are taken from a Quebec viral biobank, termed the CoVBanQ, which is hosted in the [Laboratoire de Santé Publique du Québec \(LSPQ\)](#).

You can obtain a real-time snapshot of evolving SARS-CoV-2 populations in Québec and access interactive data visualizations. It is meant for virologists, epidemiologists, public health officials and citizen scientists. Through interactive data visualizations, CoVSeQ allows exploration of continually up-to-date datasets, providing a novel surveillance tool to the scientific and public health communities.

For more details, visit [CoVSeQ Website](#) and [Sequencing Workflows](#).

### 1.2.2 RNA Sequencing

The *RNA Sequencing Pipeline* helps to quantify transcripts and genes using a reference genome and test for differential expression across experimental conditions. It is also possible to call single nucleotide variants (SNVs), detect fusion gene events and assess alternative splicing events.

### 1.2.3 De Novo RNA Sequencing

The *De Novo RNA Sequencing Pipeline* assembles reads to transcripts in the absence of a reference genome, annotates transcripts and tests for differential expression. This is well suited to organisms that are not yet characterized.

### 1.2.4 miRNA Sequencing

This pipeline can be used to quantify known miRNAs, discovers novel ones using a reference genome and performs differential expression analysis. Additional analysis may include pathway testing, target candidates analysis or miRNA editing.



### 1.2.5 SARS-CoV-2 Genome Sequencing Pipeline

The *SARS-CoV-2 genome Sequencing Pipeline* is designed for COVID-19 Coronavirus research and surveillance, enabling complete genome sequencing of the new SARS-CoV-2 virus responsible for the COVID-19 pandemic.

### 1.2.6 DNA Methylation Pipeline

The *Methylation Pipeline* helps to analyze data coming from bisulfite converted DNA assayed by various sequencing assays such as RRBS, CpG capture, whole genome sequencing or microarrays. Our analysis computes methylation levels and performs differential analysis between experimental conditions.

### 1.2.7 HiC Pipeline

The *HiC Pipeline* offers data analysis and provides interaction matrices at several resolutions, compartment analysis, topologically associating domain (TAD) predictions, as well as significant chromosomal interactions. Further analyses may include comparisons across samples/conditions and integration of various sample data (expression data, methylation, CTCF/histone binding sites) with Hi-C data.

### 1.2.8 Amplicon Sequencing Pipeline

The *Amplicon Sequencing Pipeline* can process Illumina, PacBio pyrotags amplicons from the 16S, 18S or ITS amplicons. OTUs are picked and diversity is analyzed within and between communities. Further analyses include differential abundance testing or metagenome functional content prediction.

### 1.2.9 DNA Sequencing

The *DNA Sequencing Pipeline* offers state of the art DNA-seq analyses detects and annotates variants in whole exomes, whole genomes or high coverage amplicons. The analysis can also be pushed further by assisting with variant prioritization, or perform advanced cancer related analysis.

### 1.2.10 ChIP Sequencing Pipeline

The *ChIP Sequencing Pipeline* helps in analyzing DNA fragments from immunoprecipitated chromatin by calling alignment peaks on the genome, annotating the said peaks and performing additional analyses such as motif enrichment and discovery. Designed experiments can be analyzed by testing for differential binding between experimental conditions.

## 1.3 Frequently asked questions

Queries related to GenPipes and its usage are categorized broadly as follows:

- General
- New Users
- GenPipes Developers
- C3G Resources

---

Contents

### 1.3.1 General

#### What is GenPipes?

GenPipes is an open source python framework for large scale genomics analysis. Refer to *What is GenPipes* section in GenPipes documentation for details.

#### Where can I find GenPipes Citation reference and text?

Refer to *GenPipes Citation* for citing and example text for GenPipes Citation.

#### Where can I access latest GenPipes source code?

See *Download GenPipes* section of *Deploying GenPipes locally in your server* documentation.

#### Do I need to deploy GenPipes locally on a server or in the cloud to use it?

You do not need to necessarily deploy GenPipes to use it. You can use pre-installed GenPipes on Compute Canada servers. For that you need to have CCDB account. See *Accessing GenPipes on Compute Canada Servers* for details. If you wish to deploy GenPipes locally on your server (bare metal or virtual) or within a container, check out *GenPipes Deployment and Access Guide* for various deployment options and how to setup and access GenPipes.

#### GenPipes requires familiarity with Unix. As a Windows user, what do I need to setup to use GenPipes?

If you are a Windows user, you need to get a working Unix command line interface to start using GenPipes. Windows users have three main options to get a working Unix command line:

- Ubuntu on Windows (only available on Windows 10)
- Cygwin
- PuTTY

Also, if you plan to run genomics pipelines, you need a genomics data viewer such as *Integrative Genomics Viewer*, a graphical, high-performance visualization tool for genomics data. For details, refer to *instructions to setup your windows laptop*.

#### Where do I find tips on troubleshooting GenPipes runtime and usage issues?

See *Troubleshooting GenPipes Usage issues* section of GenPipes Getting Started Guide for details.

## For new developers, are there any GenPipes development and troubleshooting tips?

See *Troubleshooting GenPipes development issues* section of GenPipes Developers Guide for details.

### 1.3.2 New Users

To create a new CCDB account, what should I fill in the form field: 'position'?

Office phone: [text field]

I prefer correspondence... ☐ in French ☒ in English

**Institution**: [dropdown menu showing 'Other...']

**Institution name**: [text field with 'Indraprastha Institute of Information Technology Delhi']

**Department**: [text field]

Please spell out the correct department name in full to avoid delays in processing your application.

**Position**:

- Principal Investigator**
  - ☒ Adjunct Faculty
  - ☐ Faculty ?
  - ☐ Librarian ?
- Industry / Government PI**
  - ☐ For-profit Principal Investigator ?
  - ☐ Not-for-profit Principal Investigator ?
- Sponsored User**
  - ☐ Undergraduate Student ?
  - ☐ Master's Student
  - ☐ Doctoral Student
  - ☐ Postdoctoral Fellow
  - ☐ External Collaborator (or Visiting Faculty) ?
  - ☐ Researcher ?
  - ☐ Non-research Staff ?
  - ☐ Guest ?
- Compute Canada Staff**
  - ☐ CC Consortium Staff ?
  - ☐ Board Member ?
  - ☐ Member Representative ?
  - ☐ Regional Director

Please describe, in general terms, your main research that requires Compute Canada resources.

#### Response

Use the following option in the form:

external collaborator

For the CCRI field, use the following as input:

bws-221-01

## What email ID should I use if I am an external collaborator?

#### Response

GMail or your work ID should be fine as long as you provide the name of your institution (or college in case of students).

### My account is activated. How do I learn more about Compute Canada Servers and resources available?

#### Response

See [Compute Canada Documentation](#).

### My account is activated but I cannot login into beluga-computecanada or any other node - Cedar, Niagara? What is wrong?

#### Response

Check out the current CCDB server status [here](#). Many a times, not being able to log in might just be due to system unavailability.

Please note that if you try to log in 3 or more times consecutively with a wrong password, your account gets deactivated and your IP address might get blacklisted. You would need to write to [Compute Canada Support](#) to get that reversed.

### What is the best place to report GenPipes bugs?

#### Response

The preferred way to report GenPipes bugs is on our Bitbucket issue page : <https://bitbucket.org/mugqic/genpipes/issues>.

You can always reach us by email at <mailto:pipelines@computationalgenomics.ca> or use [Google Group for GenPipes](#) for any requests.

For more details, visit [GenPipes Support](#) and [GenPipes Channels](#) page in this documentation.

### I was trying to use GenPipes deployed in a Docker Container. What <tag> value should I use?

#### Response

You can use the “latest” tag or one of the tags listed at [GenPipes Docker Hub](#):. If you omit the <tag> Docker will use “latest” by default.

### How do I run GenPipes locally in my infrastructure using a containerized setup? Is a scheduler setup necessary?

GenPipes pipelines use scheduler’s calls (qsub, sbatch) for submitting genomic analysis compute jobs. If you plan to use GenPipes locally using your infrastructure, inside a container, you need to run the GenPipes pipeline python scripts using the “batch mode” option. For local containerized versions of GenPipes, this is the preferred way of running the pipelines, if you don’t have access to a scheduler locally such as SLURM or PBS.

This is how you can run GenPipes pipelines such as [DNA Sequencing Pipeline](#), refer to the command below:

```
dnaseq.py -c dnaseq.base.ini dnaseq.batch.ini -j batch -r your-readsets.tsv -d your-  
→design.tsv -s 1-34 -t mugqic > run-in-container-dnaseq-script.sh  
  
bash run-in-container-dnaseq-script.sh
```

Please note, there is a disadvantage to running GenPipes Pipelines without a scheduler. In the batch mode, which is configured using the “-j batch” option, all the jobs would run as a batch, one after another, on a single node. If your server is powerful enough, this might be your preferable option. Otherwise, if you would like to take advantage of

GenPipes' job scheduling capabilities, you need to install a job scheduler locally in your infrastructure so that GenPipes can work effectively. We recommend SLURM scheduler for GenPipes.

### 1.3.3 GenPipes Developers

#### Pytest command on CCDB server results in command not found error. Pytest install fails.

A new developer trying to setup and run GenPipes tests found the following issues:

I am trying to run some python test cases using "pytest" on beluga cluster.  
I am running into "pytest command not found " error.

```
[harshi@belugal pytest-examples]$ pytest
-bash: pytest: command not found
[harshi@belugal pytest-examples]$ pip install -U pytest
Collecting pytest
  Downloading https://files.pythonhosted.org/packages/98/b8/db268d7c6b517af5b1731fc95d13e89a5032116a1482f2badb423333a420/pytest-4.6.0-py2.py3-none-any.whl (229kB)
    100% |#####| 235kB 4.0MB/s
Collecting funcsigs>=1.0; python_version < "3.0" (from pytest)
Collecting wcwidth (from pytest)
Collecting atomicwrites>=1.0 (from pytest)
Collecting pluggy<1.0,>=0.12 (from pytest)
  Downloading https://files.pythonhosted.org/packages/06/ee/de89e0582276e3551df3110088bf20844de2b0e7df2748406876cc78e021/pluggy-0.12.0-py2.py3-none-any.whl
Collecting pathlib2>=2.2.0; python_version < "3.6" (from pytest)
Collecting py>=1.5.0 (from pytest)
Collecting attrs>=17.4.0 (from pytest)
Collecting packaging (from pytest)
  Downloading https://files.pythonhosted.org/packages/91/32/58bc30e646e55eab8b21abf89e353f59c0cc02c417e42929f4a9546elbid/packaging-19.0-py2.py3-none-any.whl
Collecting six>=1.10.0 (from pytest)
Collecting importlib-metadata>=0.12 (from pytest)
  Downloading https://files.pythonhosted.org/packages/7f/72/5e13a37e989b8b54ac3f52808e122cbd9b878ac980f932dc237ff64f8a00/importlib_metadata-0.17-py2.py3-none-any.whl
Collecting more-itertools<6.0.0,>=4.0.0; python_version <= "2.7" (from pytest)
Collecting scandir; python_version < "3.5" (from pathlib2>=2.2.0; python_version < "3.6"->pytest)
  Downloading https://files.pythonhosted.org/packages/df/f5/9c052db7bd54d0cbf1bc0bb6554362bbal012d003e5888950a4f5c5dadcf4e/scandir-1.10.0.tar.gz
Complete output from command python setup.py egg_info:
Traceback (most recent call last):
  File "<string>", line 1, in <module>
    ImportError: No module named setuptools
-----
Command "python setup.py egg_info" failed with error code 1 in /tmp/pip-build-Gq0otK/scandir/
```

I googled that error and found that it might be due to older version of setup tools.  
I tried to upgrade it and I'm seeing this error now.

```
harshi@belugal:~/pytest-examples
[harshi@belugal pytest-examples]$ pip install --upgrade setuptools
Collecting setuptools
  Downloading https://files.pythonhosted.org/packages/ec/51/f45cea425fd5cb0b0380f5b0f048ebc1da5b417e48d304838c02d6288a1e/setuptools-41.0.1-py2.py3-none-any.whl (575kB)
    100% |#####| 583kB 2.1MB/s
Installing collected packages: setuptools
  Found existing installation: setuptools 26.1.1
    Uninstalling setuptools-26.1.1:
      Exception:
      Traceback (most recent call last):
        File "/cvmfs/soft.computecanada.ca/nix/store/cy2jfc5kxz8yld9lp8b7h2v0g0qjpy2j-python2.7-pip-8.1.2/lib/python2.7/site-packages/pip/basecommand.py", line 215, in main
          status = self.run(options, args)
        File "/cvmfs/soft.computecanada.ca/nix/store/cy2jfc5kxz8yld9lp8b7h2v0g0qjpy2j-python2.7-pip-8.1.2/lib/python2.7/site-packages/pip/commands/install.py", line 317, in run
          prefix=options.prefix_path,
        File "/cvmfs/soft.computecanada.ca/nix/store/cy2jfc5kxz8yld9lp8b7h2v0g0qjpy2j-python2.7-pip-8.1.2/lib/python2.7/site-packages/pip/req/req_set.py", line 736, in install
          requirement.uninstall(auto_confirm=True)
        File "/cvmfs/soft.computecanada.ca/nix/store/cy2jfc5kxz8yld9lp8b7h2v0g0qjpy2j-python2.7-pip-8.1.2/lib/python2.7/site-packages/pip/req/req_install.py", line 742, in uninstall
          paths_to_remove.remove(auto_confirm)
        File "/cvmfs/soft.computecanada.ca/nix/store/cy2jfc5kxz8yld9lp8b7h2v0g0qjpy2j-python2.7-pip-8.1.2/lib/python2.7/site-packages/pip/req/req_uninstall.py", line 115, in remove
          renames(path, new_path)
        File "/cvmfs/soft.computecanada.ca/nix/store/cy2jfc5kxz8yld9lp8b7h2v0g0qjpy2j-python2.7-pip-8.1.2/lib/python2.7/site-packages/pip/utils/_init_.py", line 267, in renames
          shutil.move(old, new)
        File "/cvmfs/soft.computecanada.ca/nix/store/ogcl3xb1w1x8chaskda2qc3oqn5kv00-python-2.7.13/lib/python2.7/shutil.py", line 303, in move
          os.unlink(src)
      OSError: [Errno 30] Read-only file system: '/cvmfs/soft.computecanada.ca/nix/store/ysa1lqlyxiw7p3f4fd3kzvkkqbq5n1-python2.7-setuptools-26.1.1/lib/python2.7/site-packages/setuptools-26.1.1-py2.7.egg'
```

#### Response

Once your account is activated, you can login in CCDB servers such as Beluga, Cedar, Niagra. However, these are National Systems on a shared grid and users don't have permission to install or upgrade the software there.

For more information on what software is installed on Compute Canada infrastructure, refer to [https://docs.computeCanada.ca/wiki/Available\\_software](https://docs.computeCanada.ca/wiki/Available_software).

### What kind of further analysis be done using HiC sequencing Pipeline?

#### Response

GenPipes [Real Life Applications](#) section of the documentation states that the Hi-C pipeline's further analyses may integrate ChIP-seq/RNA-seq data. Refer to the [Bait Intersect](#) and [Capture Intersect](#) steps of the pipeline. These steps help to integrate ChIP-seq / RNA-seq data. You can see which of your regions have ChIP-seq or RNA-seq signal.

### 1.3.4 C3G Resource Usage

#### Where can I find more details on CCDB servers, file system, usage guidelines?

You can visit [Compute Canada Wiki](#) to learn more about the list of Compute Canada Servers, systems and services available, How-to guides and usage policy.

#### Is there a list of software installed on Compute Canada servers?

See list of [available software](#) and globally deployed modules on Compute Canada servers.

#### What are modules and why do we need them for GenPipes?

GenPipes is developed in Python. Modules in Python are a way to load software or tools just in time, when a program needs them.

The modules that come along with GenPipes allow you to use bioinformatics tools like Samtools, Homer, MACS2, without installing them yourself.

For details on why we need modules for GenPipes and which ones are required, deployed and p re-installed on Compute Canada Servers, see [docs\\_gp\\_modules](#).

#### What are GenPipes genomes? Where can I access them from?

GenPipes pipelines are used for genomic analysis and they require reference genomes. C3G, in partnership with Compute Canada, maintains several genomes that are available on several HPC centres including Guillimin, Cedar, Graham and Mammouth. In addition to the fasta sequence, many genomes include aligner indices and annotation files.

To access these genomes, you need to add the following lines to your .bashrc file:

```
## GenPipes/MUGQIC genomes and modules
export MUGQIC_INSTALL_HOME=/cvmfs/soft.mugqic/CentOS6
```

To explore the available genomes, you can type:

```
ls $MUGQIC_INSTALL_HOME/genomes/species/
```

For a list of available genomes, visit [Bioinformatics resources - genomes](#) and see [sources here](#).

## What is meant by test dataset? Where can I find available test datasets?

GenPipes pipelines can be run using your sequencing instruments generated, measured, sampled read datasets in respective formats as required by individual pipelines or test datasets. Refer to [GenPipes Test Datasets](#) for various available test datasets that can be used to run various GenPipes pipelines, in case you don't have your own dataset to be processed.

~

## 1.3.5 GenPipes HPC Jobs

### How do I run GenPipes pipelines when the number of steps or jobs exceeds HPC site queue limit?

Most HPC sites impose resource sharing constraints. One such constraint is limiting the number of jobs submitted to the scheduler wait queue. In such environments user can overcome the queue limits and continue to run GenPipes Pipelines even if the job has more steps than the limit.

#### Response

GenPipes provides utilities such as ``chunk_genpipes.sh`` that can takes pipeline commands as input and chunks them so that each chunk comprises of jobs which are within the specified queue limits for a given HPC environment.

For example, hicseq.py pipeline commands can be chunked as follows:

```
M_FOLDER=path_to_folder

hicseq.py <options> --genpipes_file hicseq_script.sh

$MUGQIC_PIPELINES_HOME/utils/chunk_genpipes.sh hicseq_script.sh $M_FOLDER -n 15
```

Here, ``-n 15`` input specifies that the maximum number of jobs in a chunk is 15. This is an optional parameter. By default, the chunk size is 20.

You can use the ``monitor.sh`` GenPipes utility to monitor the status of these job *chunks*.

For details, refer to [Monitoring GenPipes Pipeline runs](#) and see `genpipes/utils` in the source tree.

**Warning:** ``chunk_genpipes.sh`` script should be invoked **only once** for a pipeline run whereas the monitor script can be run multiple times during a pipeline run for checking the job status.

### In case of an error or job timeout, do I need to re-run the entire GenPipes Pipeline script over again or is there a smarter way to submit only the failed jobs?

When there is an error or timeout with the scheduler, user can avoid canceling all GenPipes jobs and re-submit the entire pipeline script again.

#### Response

GenPipes utilities such as ``chunk_genpipes.sh`` and ``monitor.sh`` can be used to smartly chunk pipeline command script and submit these chunks to the HPC scheduler queue instead of the full pipeline run script at once. The monitoring script can be used to monitor status of job runs. Only the job chunks that timeout or encounter error can be re-submitted to the scheduling queue.

For details, refer to [Monitoring GenPipes Pipeline runs](#) and see `genpipes/utils` in the source tree.

## 1.4 GenPipes: Step by Step

Welcome to the GenPipes **Getting Started Guide!**

This guide shows you how to get started with running genomic analysis pipelines supported by GenPipes, including how to deploy and access the software, reference genomes, tune and customize the pipelines, specify input configurations and setup the environment as per the pipeline pre-requisites.

If you wish to learn more about real life applications of GenPipes and its use cases refer to the [GenPipes Applications](#) section in the left hand side navigation pane.

This getting started guide comprises of the following topics to assist you in understanding and using GenPipes for solving complex bioinformatics research problems.

### 1.4.1 Introducing GenPipes

GenPipes is a workflow management system (WMS) that facilitates building robust genomic workflows. It is a unique solution that combines both a framework for development and end-to-end analysis pipelines, covering a large scope of genomic applications and research domains. It offers researchers a simple method to analyze different types of data, customizable to their needs and resources. In addition, it provides flexibility to create customized workflows besides the ones that are already implemented and validated by GenPipes.

#### What is GenPipes?

GenPipes is an open source, flexible, scalable Python-based framework that facilitates the development and deployment of multi-step computational workflows. These workflows are optimized for High-Performance Computing (HPC) clusters and the cloud.

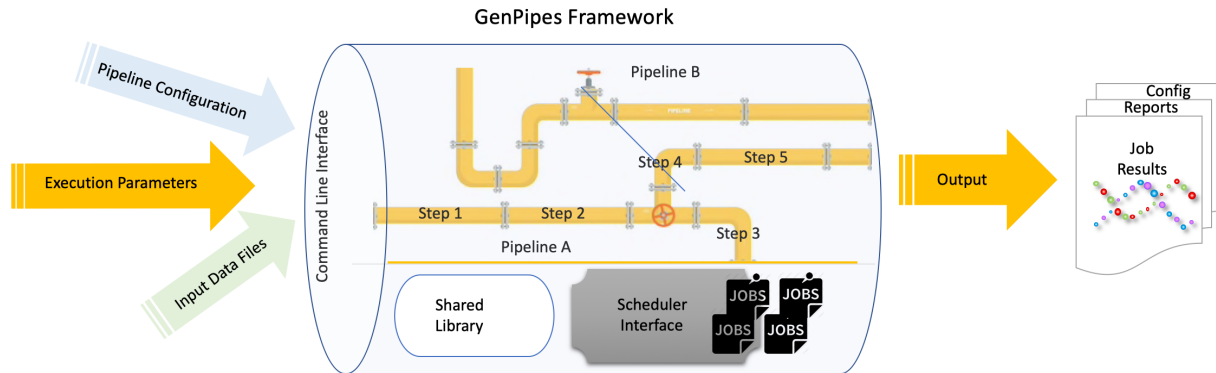
Following genomics application pipelines are already implemented and validated through GenPipes:

- ChIP-Seq
- De-Novo RNA Sequencing
- Deep Whole Genome Sequencing
- Exome Sequencing
- Hi-C / Capture Hi-C
- Illumina raw data processing
- Metagenomics
- SARS-CoV-2 genome sequencing
- RNA Sequencing / Unmapped RNA Quality Control
- Transcriptomics Assembly
- Tumour Analysis
- Whole Exome Sequencing (WES)
- Whole Genome Bisulphate Sequencing (WGBS)/ Reduced Representation Bisulphate Sequencing (RRBS)
- Whole Genome Sequencing (WGS)

GenPipes software is available under a GPLv3 open source [license](#) and is continuously updated to follow recent advances in genomics and bioinformatics. The framework can be accessed through multiple deployment mechanisms. It has already been configured on several servers at C3G HPC computing facility. Its also supports Cloud deployment



through GCP. Besides this, a Docker image is also available to facilitate additional installations on local / individual machines for small dataset analysis.



## Basic Concepts

GenPipes is a Python based object oriented workflow management system that comes pre-built with several genomic analysis pipelines. A typical analysis workflow comprises of several complex actions that are interdependent and need to be managed in terms of input, output, process configuration and pipeline tuning.

GenPipes refer to four kinds of objects that are used to manage different components of a typical analysis workflow. These are:

- Pipeline
- Step
- Job
- Scheduler

## Pipeline

It is the main object that controls the overall genomic analysis workflow. For each type of analysis, a specific Pipeline object is defined. Pipeline objects can inherit from one another.

## Step

Each Pipeline object uses Step objects to define the flow of the analysis. It provides flexibility in choosing which steps are called during a pipeline execution. Pipeline instance calls all steps implemented in a pipeline or only a set of steps selected by the user. Each step of a pipeline is a unit of execution block that encapsulates a part of the analysis (e.g., trimming or alignment). The Step object is a central unit object that corresponds to a specific analysis task. The execution of the task is directly managed by the code defined in each Step instance; some steps may execute their task on each sample individually while other steps execute their task using all the samples collectively.

## Job

The key purpose of each Step object is to generate a list of “Job” objects, which correspond to the consecutive execution of single tasks. The Job object defines the commands that will be submitted to the system. It contains all the elements needed to execute the commands, such as input files, modules to be loaded, as well as job dependencies and temporary files. Jobs are submitted in a workflow management system which uses various algorithms to schedule various types of jobs and optimizes time to result and resource usage.

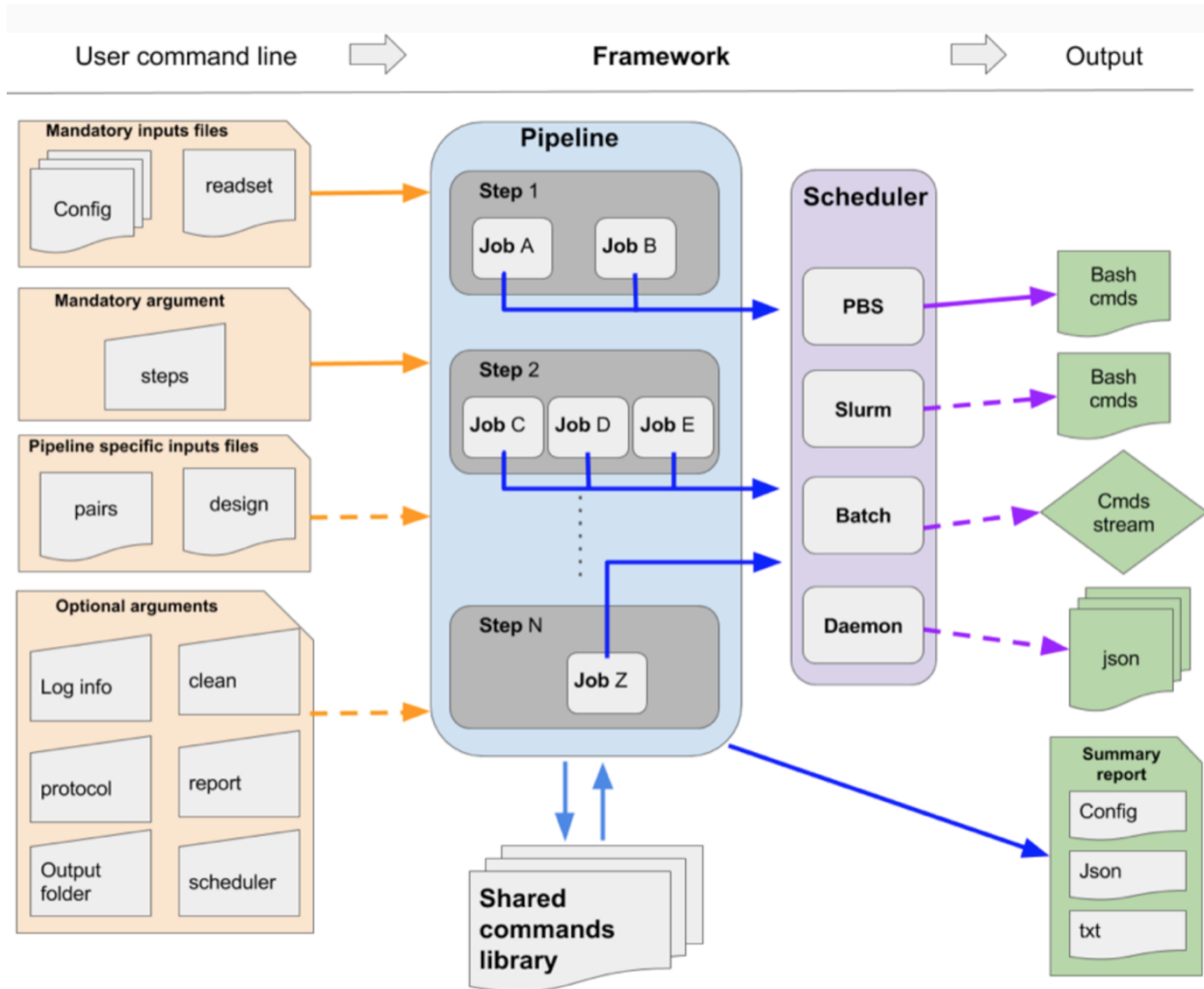
## Scheduler

Each Job object is submitted to the GenPipes workflow management system using a specific “Scheduler” object. The Scheduler object creates execution commands that are compatible with the user’s computing system. Four different Scheduler objects have already been implemented (PBS, SLURM, Batch, and Daemon).

- PBS scheduler creates a batch script that is compatible with a [PBS \(TORQUE\) system](#).
- SLURM scheduler creates a batch script that is compatible with a [SLURM system](#).
- Batch scheduler creates a batch script that contains all the instructions to run all the jobs one after the other.
- Daemon scheduler creates a log of the pipeline command in a [JSON file](#).

## How GenPipes works?

GenPipes is a Python based object-oriented framework that is available to users as a command line tool. Figure below shows the general workflow of GenPipes.



Each GenPipes Pipeline can be launched using a command line instruction. There are three kinds of inputs for each such instruction as shown in the figure above.

- Mandatory command options
- Optional command options
- Input Files

The input files can be of two kinds - mandatory ones, that are needed for every pipeline and pipeline specific input files.

Mandatory input files include Configuration files and Readset files. Configuration files contain details regarding machine environment where the pipeline is executed and parameters that need to be set for each step of the pipeline. Default values are provided and can be changed in case of specific genomic analysis. GenPipes can be deployed locally in your data center or users can access pre-installed GenPipes on Compute Canada servers. For details regarding different kinds of GenPipes deployment, refer to [GenPipes Deployment Guide](#). If you are using GenPipes pre-installed on Compute Canada servers, then the basic configuration files are installed along with GenPipes. These can be supplemented with additional configuration files provided using the '-c' option while running the command line instruction.

Besides the mandatory configuration files, some pipelines have their own specific input file that must be provided. These include Design Files and [Test Dataset files](#). These files are not provided by default and users need to supply them while running the pipelines. For the pipelines that require test dataset files, if you do not have access to any test datasets, you can try out some of the available [Sample Test Dataset Files](#) that are available as additional GenPipes resources for users.

When the GenPipes command is launched, required modules and files will be searched for and validated. If all required modules and files are found, the analysis commands will be produced. GenPipes will create a directed acyclic graph that defines job dependency based on input and output of each step.

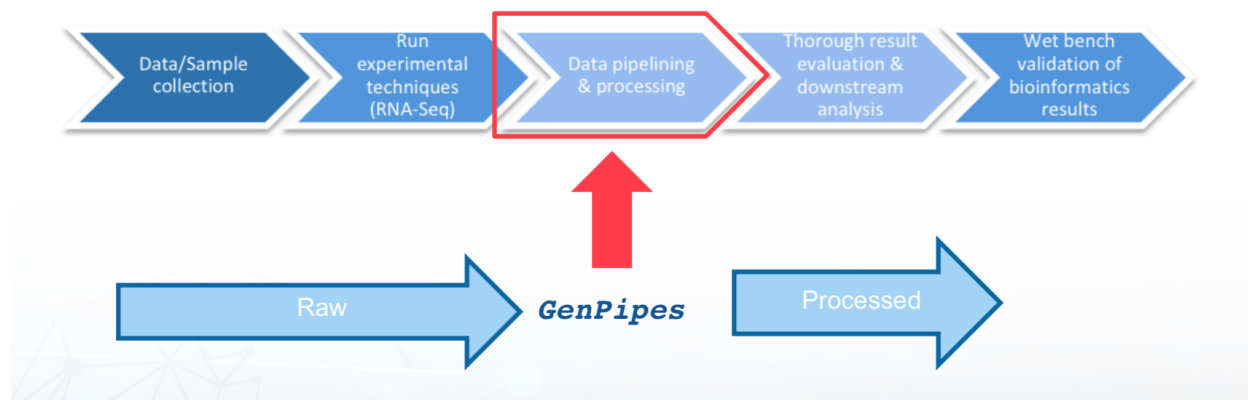
Once launched, the jobs are sent to the scheduler and queued. As jobs complete successfully, their dependent jobs are released by the scheduler to run. If a job fails, all its dependent jobs are terminated and an email notification is sent to the user. When GenPipes is re-run, it will detect which steps have successfully completed, as described in section “Smart relaunch features,” and skip them but will create the command script for the jobs that were not completed successfully. To force the entire command generation, despite successful completion, the “-f” option should be added. The output of the command line instruction are in the form of summary reports and job status. Depending upon the pipeline, there are varied tools that can be used to view and analyze the results. See [Viewing and Analyzing GenPipes Results](#) for further information.

For details on GenPipes usage and various bioinformatics pipelines see [GenPipes User Guide](#).

## Bioinformatics and the role of GenPipes

There has been significant technological evolution in Next Generation Sequencing techniques from improvement in the processes themselves, better infrastructure and software availability as well as in terms of lowering of costs associated with NGS processing. For a good primer on the topic, refer to [Introduction to Next Generation Sequencing](#).

GenPipes plays a key role in data pipelining and processing of next generation sequencing data and cutting edge genomic analysis, as highlighted in the figure below:



### 1.4.2 Using GenPipes for genomic analysis

This document describes the GenPipes execution environment and how to use various genomic analysis pipelines. It assumes you already have access to GenPipes through one of the available [GenPipes Deployment Options](#).

Topics covered in this document includes information regarding what is available in a typical GenPipes deployment and how to access it. Besides that it covers details on required software and environment configurations, various kinds of inputs required to run genomic analysis such as command options, configuration details, readset file and design file. There are example runs highlighting how to issue the pipeline commands with requisite inputs and pipeline specific configurations.

- [GenPipes Executable](#)
- [Running GenPipes](#)

- *Example Run*
- *Example Run with Design File*
- *Monitoring GenPipes Pipeline Runs*
  - *Example: Monitoring hicseq.py*
- *Further Information*

## GenPipes Executable

GenPipes framework can be used to perform various genomic analysis corresponding to the available pipelines. GenPipes is a command line tool. The underlying object-oriented framework is developed in Python. It simplifies the development of new features and its adaptation to new systems; new workflows can be created by implementing a Pipeline object that inherits features and steps from other existing Pipeline objects.

Similarly, deploying GenPipes on a new system may only require the development of the corresponding Scheduler object along with specific configuration files. GenPipes' command execution details have been implemented using a shared library system, which allows the modification of tasks by simply adjusting input parameters. This simplifies code maintenance and makes changes in software versions consistent across all pipelines.

## Running GenPipes

### Pre-Requisites

Before running GenPipes, you may want to visit the checklist of pre-requisites for GenPipes.

### Usage

Here is how you can launch GenPipes. Following is the generic command to run GenPipes:

```
<pipeline-name>.py -c config -r readset-file -s 1-n -g list-of-commands.txt
bash list-of-commands.txt
```

where <pipeline-name> refers to one of the available GenPipes Pipelines and step-range-number-1-n refers to the specific steps in the pipeline that need to be executed.

### Terminology

In the context of GenPipes, you need to be familiar with the following terms. These constitute inputs and configuration required before you can launch the pipelines.

- Readset File
- Configuration files
- Design files
- *Test Datasets*

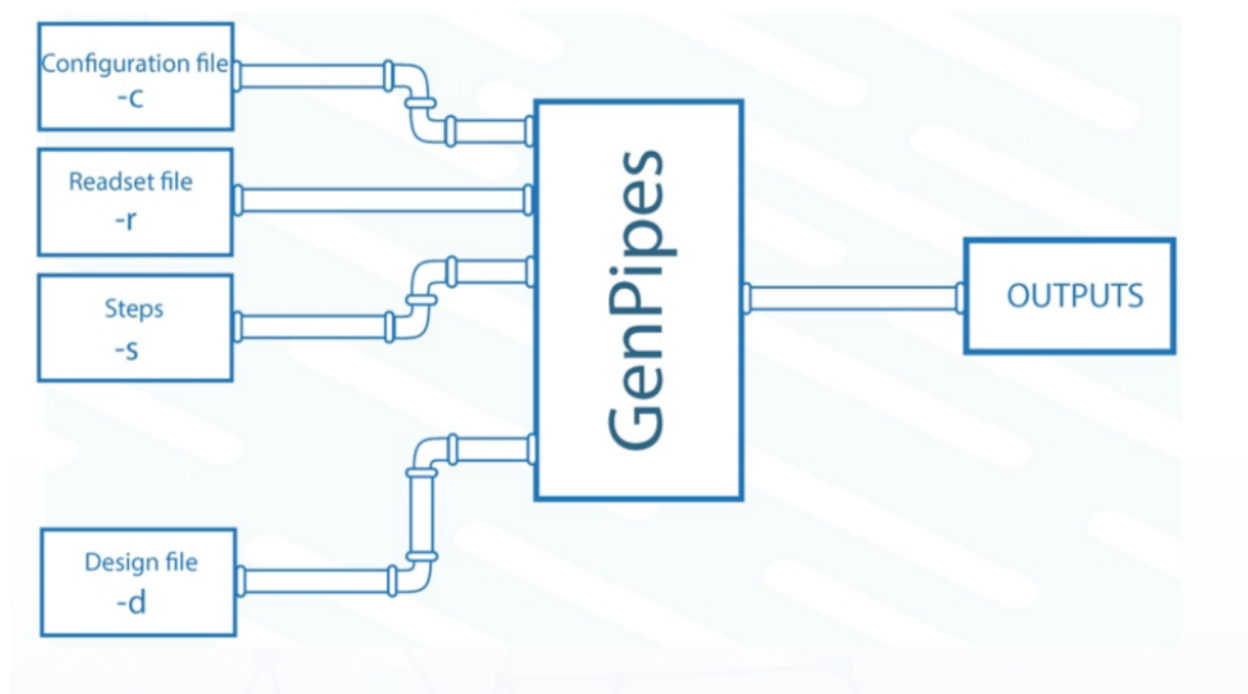
## Launching GenPipes

To launch GenPipes, the following is needed:

1. Name of the pipeline corresponding to one of the available GenPipes Pipelines.
2. A readset file that contains information about the samples, indicated using the flag “-r”. GenPipes can aggregate and merge samples as indicated by the readset file.

3. Configuration/ini files that contain parameters related to the cluster and the third-party tools, indicated using the flag “-c”. Configuration files are customizable, allowing users to adjust different parameters.
4. The specific steps to be executed, indicated by the flag “-s”.

In addition to the configuration files and the input readset file, certain pipelines such as ChIP-Seq and RNA sequencing (RNA-Seq), require a design file that describes each contrast. Custom sample groupings can be defined in the design file. Design files are indicated by the flag “-d”. More information on the design file and the content of each file type can be found in the *GenPipes User Guide*.



## Example Run

The following example shows how you can run Hi-C sequencing pipeline using GenPipes installed on Compute Canada data centres. Please ensure you have login access to GenPipes servers. Refer to checklist of pre-requisites for GenPipes before you run this example.

We will now run the pipeline using a test dataset. We will use the first 2 million reads from HIC010 from Rao et al. 2014 (SRR1658581.sra). This is an in-situ Hi-C experiment of GM12878 using MboI restriction enzyme.

You need to first download the test dataset by visiting this link:

[Hi-C Sequencing Test Dataset](#)

In the downloaded zip file, you will find the two fastq read files in folder “rawData” and will find the readset file (readsets.HIC010.tsv) that describes that dataset.

Please ensure you have access to “guillimin” server in Compute Canada data centre. We will run this analysis on guillimin as follows:

```
hicseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/hicseq/hicseq.base.ini $MUGQIC_PIPELINES_
↪HOME/pipelines/hicseq/hicseq.guillimin.ini -r readsets.HiC010.tsv -s 1-15 -e MboI -g_
↪hicseqScript_SRR1658581.txt
```

To understand what \$MUGQIC\_PIPELINES\_HOME refers to, please see instructions on how to [access GenPipes on Compute Canada servers](#).

In the command above,

-c defines the ini configuration files

-r defines the readset file

-s defines the steps of the pipeline to execute. To check pipeline steps use `hicseq -h`

-e defines the restriction enzyme used in the HiC library

By default, on Compute Canada servers such as “Cedar”, “Graham” or “Mammoth”, SLURM scheduler is used. On guillimin server, you need to use PBS scheduler. For that you need to specify “-j pbs” option as shown below:

```
hicseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/hicseq/hicseq.base.ini $MUGQIC_PIPELINES_
↪HOME/pipelines/hicseq/hicseq.guillimin.ini -r readsets.HiC010.tsv -s 1-15 -e MboI -j_
↪pbs -g hicseqScript_SRR1658581.txt
```

The above command generates a list of instructions that need to be executed to run Hi-C sequencing pipeline. These instructions are stored in the file:

```
hicseqScript_SRR1658581.txt
```

To execute these instructions, use:

```
bash hicseqScript_SRR1658581.txt
```

**Warning:** You will not see anything happen, but the commands will be sent to the server job queue. So do not run this more than once per job.

To confirm that the commands have been submitted, wait a minute or two depending on the server and type:

```
showq -u <userID>
```

where, <userID> is your login id for accessing Compute Canada infrastructure.

In case you ran it several times and launched too many commands you do not want, you can use the following line of code to cancel ALL commands:

```
showq -u <userID> | tr "|" " " | awk '{print $1}' | xargs -n1 canceljob
```

**Note:** Congratulations! You just successfully issued the Hi-C sequencing analysis pipeline commands!!!

After the processing is complete, you can access quality control plots in the `homer_tag_directory/HomerQcPlots`. You can find the compartment data in the `compartments` folder, TADs in the `TADs` folder and significant interactions in the `peaks` folder.

For more information about output formats please consult the webpage of the third party tool used.

**Note:** The Hi-C sequencing pipeline also analyzes capture hic data if the “-t capture” flag is used. For more information on the available steps in that pipeline use:

```
hicseq -h
```

## Example Run with Design File

Certain pipelines that involve comparing and contrasting samples, need a Design File. The design file can contain more than one way to contrast and compare samples. To see how this works with GenPipes pipelines, let's run a ChIP-Sequencing experiment.

### ChIP-Sequencing Test Dataset

We will use a subset of the ENCODE data. Specifically, the reads that map to chr22 of the following samples [ENCFF361CSC](#) and [ENCFF837BCE](#). They represent a ChIP-Seq analysis dataset with the CTCF transcription factor and its control input.

First, you need to download the test dataset from [here](#).

In the downloaded zip file, you will find the two fastq read files in folder rawData and will find the readset file (readsets.chipseqTest.chr22.tsv) that describes that dataset. You will also find the design file (designfile\_chipseq.chr22.txt) that contains the contrast of interest.

Following is the content of the Readset file (readsets.chipseqTest.tsv):

```
Sample Readset Library RunType Run Lane Adapter1 Adapter2 QualityOffset BED FASTQ1_
↳FASTQ2 BAM
ENCFF361CSC_ctrl1 ENCFF361CSC_chr22 SINGLE_END 2965 1 AGATCGGAAGAGCACACGTCTGAACTCCAGTCA_
↳AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT 33 rawData/ENCFF361CSC.chr22.fastq
ENCFF837BCE_ctcf ENCFF837BCE_chr22 SINGLE_END 2962 1 AGATCGGAAGAGCACACGTCTGAACTCCAGTCA_
↳AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT 33 rawData/ENCFF837BCE.chr22.fastq
```

This analysis contains 2 samples with a single readset each. They are both SINGLE\_END runs and have a single fastq file in the “rawData” folder.

Following is the content of the Design file (designfile\_chipseq.txt):

```
Sample CTCF_Input,N
ENCFF361CSC_ctrl1 1
ENCFF837BCE_ctcf 2
```

We see a single analysis CTCF\_Input run as Narrow peaks (coded by “N”; you can use “B” for broad peak analysis). This analysis compares CTCF peaks in ENCFF837BCE\_ctcf to its input control peaks identified from ENCFF361CSC\_ctrl.

Let us now run this ChIP-Sequencing analysis on *guillimin* server at Compute Canada using the following command:

```
chipseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/chipseq/chipseq.base.ini $MUGQIC_
↳PIPELINES_HOME/pipelines/chipseq/chipseq.guillimin.ini -r readsets.chipseqTest.chr22.
↳tsv -d designfile_chipseq.chr22.txt -s 1-15 -g chipseqScript.txt
bash chipseqScript.txt
```

The commands will be sent to the job queue and you will be notified once each step is done. If everything runs smoothly, you should get **MUGQICexitStatus:0** or **Exit\_status=0**. If that is not the case, then an error has occurred after which the pipeline usually aborts. To examine the errors, check the content of the **job\_output** folder.



## Monitoring GenPipes Pipeline Runs

HPC site policies typically limit the number of jobs that a user can submit in a queue. These sites deploy resource schedulers such as Slurm, Moab, PBS and Torque for scheduling and managing sharing of HPC resources.

GenPipes offers a utility script `monitor.sh` to enable better integration with resource schedulers (Slurm, Moab, PBS and Torque) deployed on HPC clusters. It also provides a fail safe mechanism to GenPipes users for re-submitting selected pipeline steps that failed to run due to timeouts. These timeouts are often caused by insufficient resource conditions for jobs running large, complex analysis that need more HPC resources. Another cause of timeouts could be errors in the HPC job queue management system.

The monitoring script lets GenPipes users manage resource constraints in a flexible and robust manner.

### Example: Monitoring hicseq.py

Here is an example of to use the monitoring script with *HiC Sequencing Pipeline*:

```
M_FOLDER=path_to_folder

hicseq.py <options> --genpipes_file hicseq_script.sh

$MUGQIC_PIPELINES_HOME/utils/chunk_genpipes.sh hicseq_script.sh $M_FOLDER

$MUGQIC_PIPELINES_HOME/utils/monitor.sh $M_FOLDER
```

The `chunk_genpipes.sh` script is used to create job chunks of specified size that are submitted at a time. Please note that this script should be executed only once when used in the context of monitoring. The `monitor.sh` script can be invoked multiple times to check on the status of submitted jobs. The `monitor.sh` script runs can be canceled or killed by `Ctrl-C` keystroke or disconnecting the `ssh` shell and restarting monitoring again when required. The `monitor.sh` script has intelligent lock mechanism to prevent accidentally invoking two `monitor.sh` script runs in parallel on the on the same folder or GenPipes pipeline run.

Figure below demonstrates how `monitor.sh` utility works. The pipeline command file output is fed into `chunk_genpipes.sh` script which creates the chunks folder as a one time activity. This chunk folder is monitored by the `monitor.sh` script. The monitoring script can be invoked multiple times during the pipeline run. With this script user can monitor the status of the submitted job chunks, whether they have completed successfully or require to be re-submitted.

For a complete list of available GenPipes utilities, refer to the `genpipes/util` folder in the source tree.

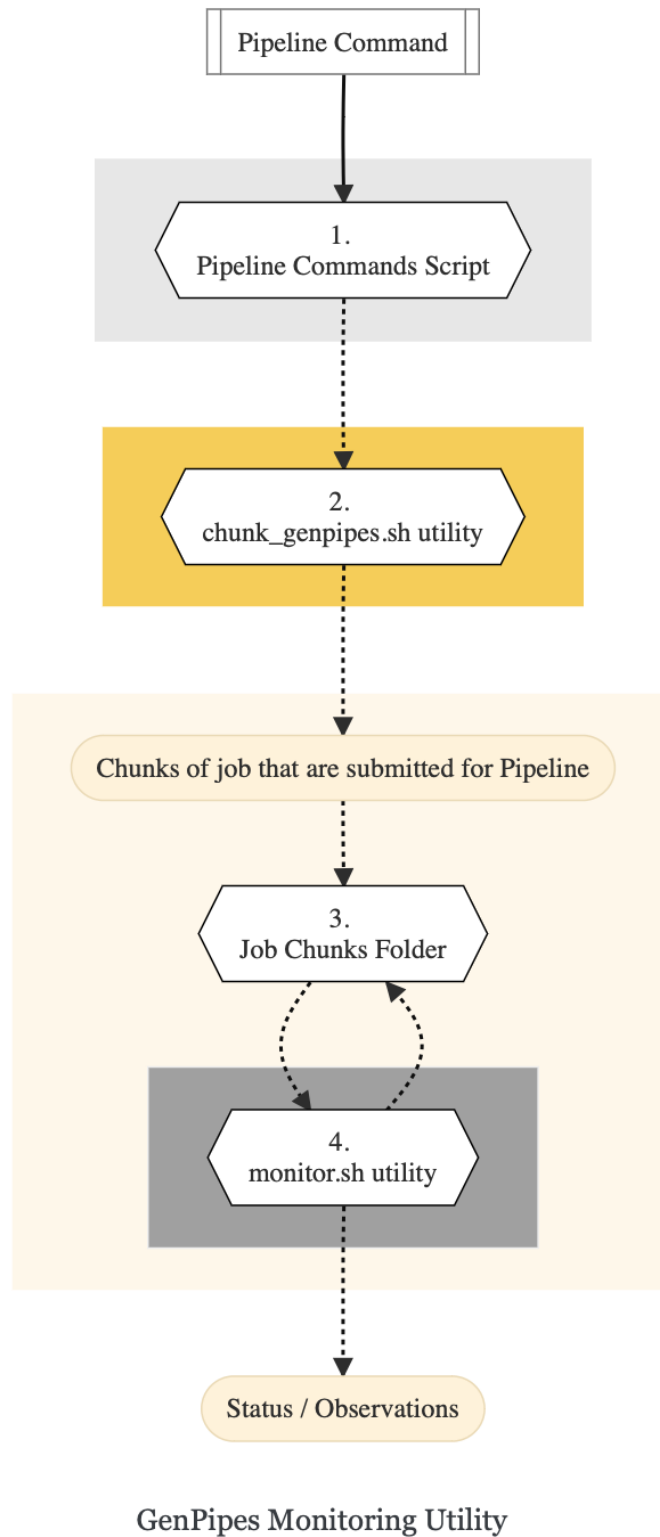
## Further Information

GenPipes pipelines are built around third party tools that the community uses in particular fields. To understand the output of each pipeline, please read the documentation pertaining to the tools that produced the output.

You can see all available GenPipes pipelines for a complete listing of all supported pipelines. To see examples of running other pipelines and also for figuring out how to run pipelines locally or in the cloud on your own GenPipes deployment, refer to *GenPipes Tutorials*.

For further information or help with particular pipelines, you can send us an email to:

[info@computationalgenomics.ca](mailto:info@computationalgenomics.ca)



### 1.4.3 Viewing and Analyzing GenPipes Results

This document contains generic insights related to viewing and analyzing results obtained after running GenPipes Pipelines. It assumes that you are familiar with *GenPipes Basics* and have used one of the pipelines already as demonstrated in *Using GenPipes*.

- *Overview*
- *GenPipes Results*
  - *Exit Codes*
- *GenPipes Errors & Log Files*
- *Visualization and Analysis*
- *GenPipes Relaunch*
- *GenPipes Clean*
- *Tracking GenPipes Environment Parameters for quoting in Publications*

#### Overview

If you are a new user of GenPipes and have successfully run it for the first time, you are likely to wonder, what is next? How can you determine whether all the jobs that were part of one of the *GenPipes Pipelines* were executed successfully? Where are the results located once the pipelines are run and what is the best way to view and analyze GenPipes results in general.

You can find answers to such questions here.

#### GenPipes Results

When a pipeline is run successfully, by default, the output is saved to the same location the pipeline was launched in the first place. This behavior can be changed by modifying the output directory when launching the pipeline using `-o` or `-output-dir`, for example:

```
rnaseq.py -o /PATH/TO/OUTPUT ... (other options)
```

For most pipelines, GenPipes creates an html report with most of the results in the pipeline. To create the report, you need to rerun the same command you ran to create the GenPipes commands, but add `-report` to it.

Refer to the 'job\_output' directory. In that directory, you can find subdirectories that roughly correspond to each step in the pipeline, inside those are the log files. At the top, of the directory, there is a file that is named with the following convention:

```
job_output/PIPELINE_job_list_YEAR-MM-DDTHH.MM.SS
```

where, PIPELINE corresponds to the pipeline name and YEAR-MM-DDTHH.MM.SS to the date and time the pipeline was launched. This job list file in the job\_output folder that can help you determine the status of each job and sub-jobs as well.

**Note:** This job\_list file can be used to check the status of only those jobs that are scheduled using PBS and Slurm schedulers. Also, this feature is not supported when you run *GenPipes in a container*.

You can type in:

```
#$MUGQIC_PIPELINES_HOME/utils/log_report.pl job_output/{TECHNOLOGY}_job_list_{DATE}T  
↪ {TIME}
```

For example:

```
#$MUGQIC_PIPELINES_HOME/utils/log_report.pl job_output/RnaSeq_job_list_2018-06-26T12.54.  
↪ 27
```

This command shown above, returns the status of each job. It outputs a summary file that includes the number of jobs that completed successfully, those that failed, and those that are still active/inactive.

You can save the output as a file and open it in Excel on your laptop. For each job, there is an exit code that indicates job status.

### Exit Codes

Following are some of the common job exit codes:

- 0 - Exit code of 0 means that the pipeline ran without any issues
- 271 - This exit code typically means that there was insufficient RAM allocated and hence the job did not run successfully.
- -11 - Exit code -11 indicates that the job was prematurely killed as it exceeded the allocated walltime - basically insufficient compute resources were assigned for the job.

---

**Note:** For every GenPipes Pipeline run, output is created in the default or specified location. However, please note that what is actually written in the output location varies significantly between each pipeline. Refer to GenPipes User Guide, *Pipelines Reference* section for details regarding the processing performed by different pipelines.

---

### GenPipes Errors & Log Files

When launched, GenPipes creates a job\_output folder where it stores the logs and errors from all the jobs. If errors occur, you need to look into the job\_output folder for the log of the step that failed to see what it last printed before it shut down. This usually helps to understand what potentially happened. GenPipes also sends you an email once each job succeeds or fails. When a job finishes successfully, it will create a file with the extension .done.

GenPipes Logs are stored in the job\_output folder under the appropriate folder for each step. For more details see GenPipes Error Logs.

### Visualization and Analysis

GenPipes output results vary a lot depending upon each specific pipeline and the way it is configured to run. Also, the way results are analyzed is also dependent on the final objective of the analysis. For example, in case of visualizations, the results have to be imported to R or Python or some alternative visualization package.

Tools such as Integrative Genomics Viewer (IGV - Integrative Genomics Viewer), [Genome Browser Gateway](#) and several others are utilized for visualization of results. **These tools vary from pipeline to pipeline.**

Figure below demonstrates one such tool used for RNA Sequencing Analysis.

The best way for new users and beginners is to use the `-report` option while running GenPipes Pipelines. Most pipelines support this and generate an html report that is saved under the report directory.

The screenshot shows the UCSC Genome Browser Gateway interface. The browser address bar displays the URL <https://genome.ucsc.edu/cgi-bin/hgGateway>. The page header includes the UCSC logo and the text "Genome Browser Gateway". Below the header is a navigation bar with links: Genomes, Genome Browser, Tools, Mirrors, Downloads, My Data, Help, and About Us.

The main content area is divided into two sections: "Browse/Select Species" and "Find Position".

**Browse/Select Species**

**POPULAR SPECIES**

Human Mouse Rat Fruitfly Worm Yeast

Enter species or common name

**REPRESENTED SPECIES**

Human Chimp Bonobo Gorilla Orangutan Gibbon Green monkey Crab-eating macaque Rhesus Baboon (anubis) Baboon (hamadryas) Proboscis monkey Golden snub-nosed monkey Marmoset Squirrel monkey Tarsier Mouse lemur Bushbaby

**Find Position**

**Human Assembly**  
Dec. 2013 (GRCh38/hg38)

**Position/Search Term**  
Enter position, gene symbol or search terms  
Current position: chr1:11,102,837-11,267,747

**GO**

**Human Genome Browser - hg38 assembly** [view sequences](#)

UCSC Genome Browser assembly ID: hg38  
Sequencing/Assembly provider ID: Genome Reference Consortium Human GRCh38 (GCA\_000001405.15)  
Assembly date: Dec. 2013  
Assembly accession: [GCA\\_000001405.15](#)  
NCBI Genome ID: 51 (Homo sapiens (human))  
NCBI Assembly ID: 883148 (GRCh38, GCA\_000001405.15)  
BioProject ID: [PRJNA31257](#)

**Search the assembly:**

- **By position or search term:** Use the "position or search term" box to find areas of the genome associated with many different attributes, such as a specific chromosomal coordinate range; mRNA, EST, or STS marker names; or keywords from the GenBank description of an mRNA. [More information](#), including sample queries.
- **By gene name:** Type a gene name into the "search term" box, choose your gene from the drop-down list, then press "submit" to go directly to the assembly location associated with that gene. [More information](#).

Fig. 1: Figure: Genome Browser Gateway

The report generated by using the `–report` flag summarize the most important results in the pipeline, while providing convenient links to the locations of important output files. More advanced users can use those output files to generate their own visualizations or further analyze results using their own methods.

As mentioned earlier, visualization of results varies from pipeline to pipeline. As a reference, you can see RNA Sequencing Analysis Visualization of results.

Figure below shows how the data is displayed once the alignment files are opened on IGV.



Fig. 2: Figure: Data Alignment visualizer using IGV Tool

## GenPipes Relaunch

If GenPipes fails, for any reason, you can recreate the commands and relaunch them. When recreating the commands, GenPipes can detect jobs that have completed successfully and will not rerun them. That being said, unless you understand why a job failed and fix it, relaunched jobs might fail with the same error.

## GenPipes Clean

GenPipes stores some temporary files that are useful to shorten potential reruns. To delete all these files, you can run the GenPipes command with `–clean`. This will delete a lot of files that were marked by GenPipe developers as “removable”. If you are interested in temporary files, avoid the `–clean` command.

## Tracking GenPipes Environment Parameters for quoting in Publications

In order to keep track of all parameters used, GenPipes creates a final .config.trace.ini file each time it is run. It is a good idea to keep a copy of that file in order to keep track of software versions used when publishing your paper or publication.

### 1.4.4 Troubleshooting Runtime Issues

This document lists some commonly faced issues related to GenPipes deployment and usage. These could be related to GenPipes deployment, software dependencies, environment setup or usage options.

The objective here is to list some commonly encountered issues and their fixes so that new users can benefit and focus more on using GenPipes. These are mostly run-time issues that a new user may face. If you are looking at troubleshooting GenPipes as part of feature development or contributing to GenPipes code, please refer to the [Developer Guide](#) and [Troubleshooting Guide](#).

#### Contents

#### Fatal Limit Error: Star RNA Sequencing

When you issue the RNA sequencing GenPipes pipeline with Star option commands, the jobs fails with the fatal limit error:

```
Fatal LIMIT error: the number of junctions to be inserted on the fly =2663181 is larger
↳ than the limitSjdbInsertNsj=10000000
Fatal LIMIT error: the number of junctions to be inserted on the fly =2663181 is larger
↳ than the limitSjdbInsertNsj=10000000
SOLUTION: re-run with at least --limitSjdbInsertNsj 2663181

Nov 29 14:10:58 ..... FATAL ERROR, exiting
MUGQICexitStatus:104
```

It is not clear from the error message where this solution configuration option needs to be specified.

Typically, the Star index options in the `.ini` file supplied for RNA sequencing protocol do not show `--limitSjdbInsertNsj` option.

```
[star_index]

ram = 191000M
io_buffer = 1G
threads = 20
cluster_cpu = -N 1 -c 40
cluster_walltime = --time=15:00:0
cluster_queue = --mem-per-cpu=4775M
star_cycle_number = 99
```

#### Fix

The correct way to specify this option is using `--other-option` flag as shown in the snippet from `.ini` file below:

```
[star_index]
```

(continues on next page)

(continued from previous page)

```

ram = 191000M
io_buffer = 1G
threads = 20
cluster_cpu = -N 1 -c 40
cluster_walltime = --time=15:00:0
cluster_queue = --mem-per-cpu=4775M
star_cycle_number = 99
other_options = --limitSjdbInsertNsj 2500000

```

### Error: qsub command not found

While running a pipeline using specific configuration file, the command text file is generated successfully. However, when the user tries to run the commands using:

```
bash <command-file.txt>
```

the following error shows up:

*qsub: command not found*

```

shalz@ubuntu_srv:~$ hicseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/hicseq/hicseq.ba
se.ini $MUGQIC_INSTALL_HOME/testdata/hicseq/hicseq.GM12878_chr19.ini -r $MUGQIC_IN
STALL_HOME/testdata/hicseq/readset.hicseq.txt -t hic -s 1-16 -e MboI > hicseqScrip
t_tutorial_GM12878_chr19.txt

```

```

shalz@ubuntu_srv:~$ bash hicseqScript_tutorial_GM12878_chr19.txt
hicseqScript_tutorial_GM12878_chr19.txt: line 115: qsub: command not found
shalz@ubuntu_srv:~$

```

### Fix

Check the kind of GenPipes deployment you are working with. If it is a local deployment, on a bare metal server or a virtual server or in a container, you need to make sure you do not specify -j pbs or -j slurm option but -j daemon or -j batch mode.

In case you are using moab or torque scheduler, ensure that you use -j pbs option in the pipeline command. If you are using mp2b or cedar server, then you need to specify -j slurm as the scheduler option and use the correct configuration file (mp2b.ini or cedar.ini) depending upon which server you are using to run your pipeline jobs.

### Runtime Failure: Job fails on worker nodes

When you issue the pipeline commands, the jobs fail to run on worker nodes.

### Fix

The most common reason for this failure is not setting up the .bashrc with muggic modules. See details on accessing GenPipes on Compute Canada servers - [Step 3: Setting up your user environment for GenPipes access](#). For other GenPipes [Deployment Options](#), make sure you have closely followed the docs\_pre\_req\_chklist before actually issuing GenPipes pipeline run commands.



### Error: RAP\_ID not set

If you try to run GenPipes deployed by C3G on Compute Canada servers, the initial run shows error related to RAP\_ID not set. Sometimes, this same issue manifests in the form of timing error as shown in figure below:

```
[shaloo@ip16-mp2 hicseq]$ bash hicseqScript_tutorial_GM12878_chr19.txt
batch: error:
-----
The specified account bws-221-aa is not in your list of active _cpu allocations...
You cannot use this account to submit this job...
Please use one of the following accounts:
es are loaded (after source bash profile command issued earlier). You may also c
AS default accounts: def-bourqueg,
AC accounts:
compute-Burst accounts:
gic >moduleoutput.txt
output.txt
Use the parameter --account=desired_account when submitting your job
-----
```

Fig. 3: Figure: Error encountered if RAP\_ID not set or set incorrectly

### Fix

Make sure you have updated your .bashrc file as directed in *Step 3: Setting up your user environment for GenPipes access*. Once you set up the correct RAP\_ID when you run the bash commands for your pipeline, they all go through and get scheduled depending on the scheduler (default or as as specified by -j option in pipeline command)

### Error: Missing Genomes and Annotations

Several users have encountered this issue.

### Fix

Most of the GenPipes pipeline commands require you to supply input data in the form of readsets, design files and configuration. If a specific genome that you need to provide to the pipeline is not available in the pre-installed GenPipes setup deployed on Compute Canada servers as listed in test datasets <<https://www.computationalgenomics.ca/test-dataset/>>`\_and available`genomes.

### Error: While using STAR option in RNA Sequencing Pipeline

Users have reported issues while running RNA Sequencing Pipeline. One such issue is as listed by Sophie Ehresmann [here](#).

### Fix

TBD-GenPipes-Dev

### Error: Newbie issue - the pipeline does not execute at all!

First time users may issue the pipeline command and assume it will generate jobs on worker nodes automatically. However, after multiple runs, no execution happens if the pipeline command is executed. For example see Han's issue in [GenPipes Google Group](#).

#### Fix

This is a very common issue. GenPipes pipeline command does NOT issue the commands on its own. When you run the pipeline, it simply generates a bunch of commands to execute but does not execute them. You need to redirect the output of pipeline command into a file and then bash execute that file containing all the commands corresponding to a genomic analysis. See GenPipes Google Group discussions and Mathieu Bourgey's [response for details](#).

### Error: RNA Sequencing Star alignment - Out of Memory

For first time users, it has been observed (see example in [Google GenPipes Forum](#)) that the RNA Sequencing pipeline command execution stops after STAR alignment 1.

#### Fix

try to change the STAR parameters in your ini files to something like in the .ini files of the master branch:

`https://bitbucket.org/mugqic/mugqic\_pipelines/src/master`

The problem should be solved by setting `io_buffer` to a higher value like 1G or 4G. The command you show indicates you are using 8M.

At some point `io_buffer` was decreased in the template .ini but this exposed a bug in STAR where a negative memory allocation is attempted.

## 1.5 How to deploy GenPipes?

This document lists various ways by which GenPipes can be accessed and installed by users. Check out various [available deployment options](#), if you wish to install and setup GenPipes on your infrastructure. If you wish to use pre-installed GenPipes deployed on Compute Canada infrastructure, refer to [Accessing GenPipes on Compute Canada servers](#).

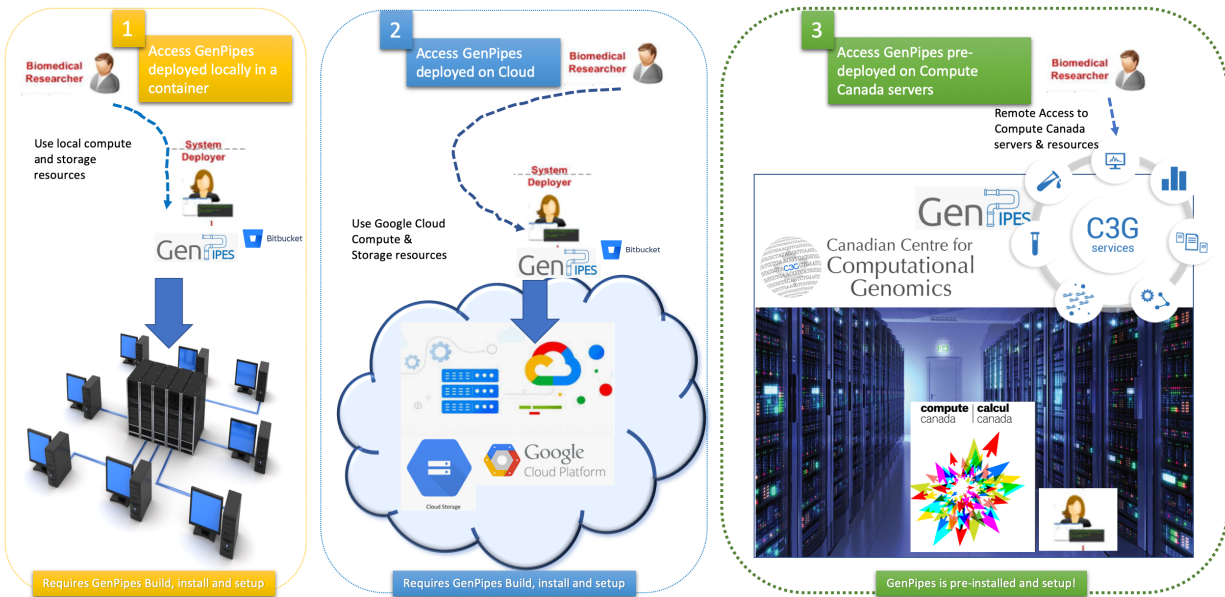
### 1.5.1 Deployment Options

There are multiple ways to access GenPipes and get started with genomic analysis using the pipelines therein. Figure below represents the three options available to bioinformatics researchers to access GenPipes.

1. Remotely access GenPipes deployed at Compute Canada infrastructure
2. GenPipes deployment in the cloud - Google Cloud Platform (GCP)
3. Local deployment
  - Bare metal or virtual server
  - GenPipes in a container

## Accessing GenPipes

The infographic below represents various mechanisms that can be used to access GenPipes today.



## Obtaining GenPipes sources

Refer to the latest [GenPipes sources](#) for instructions on downloading and setting up GenPipes.

## GenPipes on Compute Canada infrastructure

Researchers who have access to Compute Canada resources need not deploy GenPipes for genomic analysis. They can simply login and access Compute Canada servers that have pre-installed stable release of GenPipes. For details refer to *Accessing GenPipes deployment on Compute Canada infrastructure*. External users who do not have access to Compute Canada data centre resources can *apply for the same*.

Through a partnership with the [Compute Canada](#) consortium, the pipelines and third-party tools have also been configured on 6 different Compute Canada HPC centers. This allows any Canadian researcher to use GenPipes along with the needed computing resources by [simply applying to the consortium](#). To ensure consistency of pipeline versions and used dependencies (such as genome references and annotation files) and to avoid discrepancy between compute sites, pipeline set-up has been centralized to 1 location, which is then distributed on a real-time shared file system: the CERN (European Organization for Nuclear Research) Virtual Machine File System [CVM FS](#).

## GenPipes deployment on GCP

If you need to run large scale genomic analysis that requires resource scaling, GenPipes can be deployed and accessed from cloud. At present, Google Compute Platform (GCP) is supported. You may require assistance from System Administrator or your local cloud expert to install and deploy GenPipes in the cloud before you can access and run genomic analysis pipelines provided by GenPipes. For details refer to GenPipes installation guide section titled *“How to deploy GenPipes in the cloud?”*.

## Local deployment

If you wish to deploy GenPipes locally using your own compute and storage infrastructure, you can refer to the BitBucket repository listed earlier. You could either deploy it on your local server / workstation or try GenPipes in a container option.

GenPipes can be installed from scratch on any Linux cluster supporting Python 2.7 by following the instructions in the [README.md file](#). GenPipes can also be deployed via containers approach. A Docker image of GenPipes is available which simplifies the set-up process and can be used on a range of platforms, including cloud platforms. This allows system-wide installations, as well as local user installations via the Docker image without needing special permissions.

Local deployment option can be used for small scale genomic analyses using genome datasets available locally. The GenPipes in a container option is a self-contained image that offers GenPipes software, common reference genomes and all that is needed to run the pre-built analysis pipelines. Bioinformatics researchers who are not familiar with container technology may require assistance from System Administrators in deploying a local copy of GenPipes software. For details refer to GenPipes installation guide section for container deployment *GenPipes in a container*.

## 1.5.2 Accessing GenPipes on Compute Canada Servers

This document will provide information on how to get started with using GenPipes pre-installed on Compute Canada servers.

### Step 1: Get a Compute Canada Data Base (CCDB) account

- a. Go to the website: [https://ccdb.compute canada.ca/account\\_application](https://ccdb.compute canada.ca/account_application)
- b. Agree with the policy and submit your acceptance
- c. Fill the form and submit it.

---

**Note:** If you are a student or a post-doc (or any other kind of Sponsored User), to be eligible to apply to Compute Canada, the Principle Investigator (PI) of your laboratory must also have an account. You will need the Compute Canada Role Identifier (CCRI) of your sponsor/PI. The CCRI has the abc-123-01 form. It is free for Canadian academics to use the Compute Canada servers.

**It will take one or two days before your account request is processed.**

---

### Step 2: Connect to Compute Canada servers

#### Unix / Linux / Mac or Windows (bash)

- a. Open a shell or terminal (bash preferably) and type the following command:

```
ssh myaccount@mp2b.ccs.usherbrooke.ca
```

- b. Enter your Compute Canada account password.

#### Windows (PuTTY)

- a. Open puTTY.
- b. Select “Session” on the left hand side panel
- c. Select “SSH” and fill the “Host Name” entry with the following:

```
mp2b.ccs.usherbrooke.ca
```

d. Click **“Open”**

A terminal will open and ask you to connect using your CC account credentials.

Voila!!! You are all set to use GenPipes deployed on Compute Canada data centre.

**Note:** Canadian Centre for Computational Genomics (C3G), in partnership with Compute Canada, offers and maintains a large set of bioinformatics resources for the community. For a complete list of software currently deployed on several HPC centres, including Guillimin, Cedar, Graham and Mammouth, refer to [Bioinformatics Resources](#). Several reference genomes are also available. You can refer to the [available genomes](#) and the environment setup to access these genomes.

### Step 3: Setting up your user environment for GenPipes access

#### For Abacus, Compute Canada Users only

All of the software and scripts used by GenPipes are already installed on several Compute Canada servers including Guillimin, Mammouth, Cedar and Graham, and will soon be installed on Beluga. To access the tools, you will need to add the tool path to your `bash_profile`. The bash profile is a hidden file in your home directory that sets up your environment every time you log in. You can also use your `bashrc` file.

Genomes and modules used by the pipelines are already installed on a CVMFS partition mounted on all those clusters in `/cvmfs/soft.mugqic/CentOS6`

**Note:** For more information on the differences between the `.bash_profile` and the `.bashrc` profile, consult [this page](#).

```
## open bash_profile
nano $HOME/.bash_profile
```

**Note:** GenPipes have been tested with Python 2.7

Next, you need to load the [software modules](#) in your shell environment that are required to run GenPipes. For a full list of modules available on Compute Canada servers see the [module page](#)

To load the GenPipes modules, paste the following lines of code and save the file, then exit (Ctrl-X):

```
umask 0002

## GenPipes/MUGQIC genomes and modules
export MUGQIC_INSTALL_HOME=/cvmfs/soft.mugqic/CentOS6
module use $MUGQIC_INSTALL_HOME/modulefiles
module load mugqic/python/2.7.14
module load mugqic/genpipes/<latest_version>
export JOB_MAIL=<my.name@my.email.ca>
export RAP_ID=<my-rap-id>
```

You will need to replace the text in “<>” with your account and GenPipes software version specific information.

**JOB\_MAIL** is the environment variable that needs to be set to the email ID on which GenPipes job status notifications are sent corresponding to each job initiated by your account. It is advised that you create a separate email for jobs since

you can receive hundreds of emails per pipeline. You can also de-activate the email sending option by removing the “-M \$JOB\_MAIL” option from the .ini files.

**RAP\_ID** is the Resource Allocation Project ID from Compute Canada. It is usually in the format: rrg-lab-xy OR def-lab.

### Environment settings for MUGQIC analysts

For MUGQIC analysts, add the following lines to your \$HOME/.bash\_profile:

```
umask 0002

## MUGQIC genomes and modules for MUGQIC analysts

HOST=`hostname`;

DNSDOMAIN=`dnsdomainname`;

export MUGQIC_INSTALL_HOME=/cvmfs/soft.mugqic/CentOS6

if [[ $HOST == abacus* || $DNSDOMAIN == ferrier.genome.mcgill.ca ]]; then

    export MUGQIC_INSTALL_HOME_DEV=/lb/project/mugqic/analyste_dev

elif [[ $HOST == ip* || $DNSDOMAIN == m  ]]; then

    export MUGQIC_INSTALL_HOME_DEV=/project/6007512/C3G/analyste_dev

elif [[ $HOST == cedar* || $DNSDOMAIN == cedar.computecanada.ca ]]; then

    export MUGQIC_INSTALL_HOME_DEV=/project/6007512/C3G/analyste_dev

elif [[ $HOST == beluga* || $DNSDOMAIN == beluga.computecanada.ca ]]; then

    export MUGQIC_INSTALL_HOME_DEV=/project/6007512/C3G/analyste_dev

fi

module use $MUGQIC_INSTALL_HOME/modulefiles $MUGQIC_INSTALL_HOME_DEV/modulefiles
module load mugqic/python/2.7.14
module load mugqic/genpipes/<latest_version>

export RAP_ID=<my-rap-id>
```

Also, set JOB\_MAIL in your \$HOME/.bash\_profile to receive PBS job logs:

```
export JOB_MAIL=<my.name@my.email.ca>
```

### How to check the version of GenPipes deployed

To find out the latest GenPipes version available, once you have connected to your CC account, use the following command:

```
module avail 2>&1 | grep mugqic/genpipes
```

---

**Note:** Previous version of GenPipes were named mugqic\_pipelines and are still available for use.

---

### How to ensure bash\_profile changes take effect in the environment variables?

When you make changes to your bash\_profile, you will need to log out and then login again for these changes to take effect. Alternatively, you can run the following command in bash shell:

```
source $HOME/.bash_profile
```

By adding the lines related to module load and environment variable setting via export, you have set up the pipeline environment and are ready to use GenPipes!

This also gives you access to hundreds of bioinformatics tools pre-installed by our team. To explore the available tools, you can type the following command:

```
module avail mugqic/
```

For a full list of all available modules on Compute Canada servers, visit [module page](#).

To load a tool available on Compute Canada servers, for example - samtools, use the following command:

```
# module add mugqic/<tool><version>
module add mugqic/samtools/1.4.1

# Now samtools 1.4.1 is available for use in your account environment. To check, run the
↪ following command:
samtools
```

Several of the GenPipes pipelines may require referencing genomes. To access these pre-installed genomes available in:

```
$MUGQIC_INSTALL_HOME/genomes/species/
```

use the following command to check all available genome species:

```
ls $MUGQIC_INSTALL_HOME/genomes/species
```

All genome-related files, including indices for different aligners and annotation files can be found in:

```
$MUGQIC_INSTALL_HOME/genomes/species/<species_scientific_name>.<assembly>/
## so for Homo Sapiens hg19 assembly, that would be:
ls $MUGQIC_INSTALL_HOME/genomes/species/Homo_sapiens.hg19/
```

For a complete list of all available reference genomes, visit [genome page](#).

## Step 4: Running GenPipes pipelines

Now you are all set to run GenPipes analysis pipelines. Refer to instructions in *Using GenPipes for genomic analysis* for example runs. For specific pipelines supported by GenPipes, their command options refer to GenPipes *User Guide*.

### 1.5.3 Deploying GenPipes locally in your server

This document covers details on how to deploy GenPipes locally on a bare metal or virtual server. If you wish to install GenPipes locally in a container, please refer to *GenPipes in a container* section of deployment guide.

For more details on other available options to deploy and access GenPipes you may refer to *GenPipes Deployment Options Page*.

#### Step 1: Download latest GenPipes sources

First of all, visit GenPipes [Download Page](#) and get a copy of the latest stable release software. To fetch the most recent version of GenPipes, you may use the following command:

```
git clone git@bitbucket.org:mugqic/genpipes.git
```

#### Step 2: Setup environment variables

Add the following line in your your *\$HOME/.bash\_profile*: to set MUGQIC\_PIPELINES\_HOME to your local copy path. For example,

```
export MUGQIC_PIPELINES_HOME=/path/to/your/local/genpipes
```

#### Step 3: Accessing software modules and genomes needed for GenPipe

GenPipes was formerly known as MUGQIC Pipelines. Genomic analysis executed using these pipelines requires [genomes](#) and [software modules](#). You need to load the software modules in your shell environment. To do so, set the environment variable **MUGQIC\_INSTALL\_HOME** to the directory where you want to install those resources in your *\$HOME/.bash\_profile* as follows:

```
## MUGQIC genomes and modules

export MUGQIC_INSTALL_HOME=/path/to/your/local/mugqic_resources
module use $MUGQIC_INSTALL_HOME/modulefiles
```

##### Installing available modules

Software tools and associated modules must be installed in *\$MUGQIC\_INSTALL\_HOME/software/* and *\$MUGQIC\_INSTALL\_HOME/modulefiles/*. Default software/module installation scripts are already available in *\$MUGQIC\_PIPELINES\_HOME/resources/modules/*.

##### Install new modules

To install a new module or new software tool and associated modules semi-automatically, use the following instructions:

```
cp $MUGQIC_PIPELINES_HOME/resources/modules/MODULE_INSTALL_TEMPLATE.sh $MUGQIC_PIPELINES_
↪HOME/resources/modules/<my_software>.sh
```



Follow the instructions in the file `$MUGQIC_PIPELINES_HOME/resources/modules/<my_software>.sh` and modify it accordingly. Next you need to run the following command with **No arguments**. By default, it will download and extract the remote software archive, build the software and create the associated module, all in `$MUGQIC_INSTALL_HOME_DEV` if it is set.

```
$MUGQIC_PIPELINES_HOME/resources/modules/<my_software>.sh
```

If everything executes OK with no error, you are ready to install the *my\_software* module in production. Use the command:

```
$MUGQIC_PIPELINES_HOME/resources/modules/<my_software>.sh MUGQIC_INSTALL_HOME
```

**Note:** Please note there is no \$ before MUGQIC\_INSTALL\_HOME specified as argument above!.

Next, you need to check if the module is successfully installed and available for use by executing the following command:

```
module avail 2>&1 | grep mugqic/<my_software>/<version>
```

This completes the software module setup for GenPipes execution. Next you need to make sure all required reference genomes are available in your local deployment. Refer to the next section if you wish to install additional genomes.

### Installing genomes

Reference genomes and annotations must be installed in the following directory:

```
$MUGQIC_INSTALL_HOME/genomes/
```

Default genome installation scripts are already available locally in the following directory:

```
$MUGQIC_PIPELINES_HOME/resources/genomes/
```

To install all of the available genomes that are bundled with GenPipes package, use the following script:

```
$MUGQIC_PIPELINES_HOME/resources/genomes/install_all_genomes.sh
```

All species related files are in the following directory:

```
$MUGQIC_INSTALL_HOME/genomes/species/<species_scientific_name>.<assembly>/
```

For example, *Homo Sapiens* assembly *GRCh37* genome directory hierarchy is as follows:

```
$MUGQIC_INSTALL_HOME/genomes/species/Homo_sapiens.GRCh37/
├── annotations/
│   ├── gtf_tophat_index/
│   ├── Homo_sapiens.GRCh37.dbSNP142.vcf.gz
│   ├── Homo_sapiens.GRCh37.dbSNP142.vcf.gz.tbi
│   ├── Homo_sapiens.GRCh37.Ensembl75.geneid2Symbol.tsv
│   ├── Homo_sapiens.GRCh37.Ensembl75.genes.length.tsv
│   ├── Homo_sapiens.GRCh37.Ensembl75.genes.tsv
│   ├── Homo_sapiens.GRCh37.Ensembl75.GO.tsv
│   ├── Homo_sapiens.GRCh37.Ensembl75.gtf
│   ├── Homo_sapiens.GRCh37.Ensembl75.ncrna.fa
│   ├── Homo_sapiens.GRCh37.Ensembl75.rrna.fa
│   └── Homo_sapiens.GRCh37.Ensembl75.transcript_id.gtf
```

(continues on next page)

(continued from previous page)

```

├── Homo_sapiens.GRCh37.Ensembl75.vcf.gz
├── ncrna_bwa_index/
├── rrna_bwa_index/
├── downloads/
│   ├── ftp.1000genomes.ebi.ac.uk/
│   ├── ftp.ensembl.org/
│   └── ftp.ncbi.nih.gov/
├── genome/
│   ├── bowtie2_index/
│   ├── bwa_index/
│   ├── Homo_sapiens.GRCh37.dict
│   ├── Homo_sapiens.GRCh37.fa
│   ├── Homo_sapiens.GRCh37.fa.fai
│   └── star_index/
├── Homo_sapiens.GRCh37.ini
└── log/

```

The assembly name is the one used by the download source. For e.g. “GRCh37” is used for [Ensembl](#).

Each species directory contains a “.ini” file such as:

```
<scientific_name>.<assembly>.ini
```

Among other things, this “.ini” file lists the assembly synonyms. In case of “hg19”, the contents of Homo\_sapiens.GRCh37.ini are as shown below:

```

[DEFAULT]
scientific_name=Homo_sapiens
common_name=Human
assembly=GRCh37
assembly_synonyms=hg19
source=Ensembl
version=75
dbSNP_version=142

```

### Install a new Genome

New genomes and annotations can be installed semi-automatically from [Ensembl](#) (vertebrate species), [Ensemble Genomes](#) (other species) or [UCSC](#) (genome and indexes only; no annotations).

*Example - how to set up genomes for Chimpanzee:*

1. Retrieve the species scientific name on [Ensemble Genomes](#) or [UCSC](#) :

```
Pan troglodytes
```

2. Retrieve the assembly name:

- Ensembl: “CHIMP2.1.4”
- UCSC: “panTro4”

3. Retrieve the source version:

- Ensembl: “78”
- UCSC: unfortunately, UCSC does not have version numbers. Use [panTro4.2bit](#) date formatted as “YYYY-MM-DD”: “2012-01-09”

4. Next, copy the template file to a new file name using the scientific name.

```
cp $MUGQIC_PIPELINES_HOME/resources/genomes/GENOME_INSTALL_TEMPLATE.sh $MUGQIC_PIPELINES_
↪HOME/resources/genomes/<scientific_name>.<assembly>.sh
```

For example, in case of Ensembl, use the following command:

```
cp $MUGQIC_PIPELINES_HOME/resources/genomes/GENOME_INSTALL_TEMPLATE.sh $MUGQIC_PIPELINES_
↪HOME/resources/genomes/Pan_troglodytes.CHIMP2.1.4.sh
```

In case of genomes from UCSC, use the following command to copy the genome install instructions:

```
cp $MUGQIC_PIPELINES_HOME/resources/genomes/GENOME_INSTALL_TEMPLATE.sh $MUGQIC_PIPELINES_
↪HOME/resources/genomes/Pan_troglodytes.panTro4.sh
```

5. Next, you need to modify the following file:

```
$MUGQIC_PIPELINES_HOME/resources/genomes/<scientific_name>.<assembly>.sh
```

Please note that ASSEMBLY\_SYNONYMS can be left empty but if you know that 2 assemblies are identical apart from chr sequence prefixes, document it.

Example below shows the modifications for Ensembl:

```
SPECIES=Pan_troglodytes    # With "-"; no space!
COMMON_NAME=Chimpanzee
ASSEMBLY=CHIMP2.1.4
ASSEMBLY_SYNONYMS=panTro4
SOURCE=Ensembl
VERSION=78
```

Example below shows the modifications for UCSC:

```
SPECIES=Pan_troglodytes    # With "-"; no space!
COMMON_NAME=Chimpanzee
ASSEMBLY=panTro4
ASSEMBLY_SYNONYMS=CHIMP2.1.4
SOURCE=UCSC
VERSION=2012-01-09
```

6. Now you can run the following command to install the genome in \$MUGQIC\_INSTALL\_HOME\_DEV (by default). This will download and install genomes, indexes and, for Ensembl only, annotations (GTF, VCF, etc.).

```
bash $MUGQIC_PIPELINES_HOME/resources/genomes/<scientific_name>.<assembly>.sh
```

**Admin-only** To install it in \$MUGQIC\_INSTALL\_HOME, run the following command:

```
bash $MUGQIC_PIPELINES_HOME/resources/genomes/<scientific_name>.<assembly>.sh MUGQIC_
↪INSTALL_HOME
```

7. **Admin-only** If the new genome has been installed in \$MUGQIC\_INSTALL\_HOME\_DEV, to deploy in \$MUGQIC\_INSTALL\_HOME you can use the following command:

```
rsync -vca --no-o --no-g --no-p --size-only -I -0 --ignore-times $MUGQIC_INSTALL_HOME_
↪DEV/genomes/species/<scientific_name>.<assembly> $MUGQIC_INSTALL_HOME/genomes/species/
```

8. Lastly, add the newly created “.ini” file to the genome configuration files for further use in subsequent genomic analysis pipeline runs by the following command:

```
cp $MUGQIC_INSTALL_HOME/genomes/species/<scientific_name>.<assembly>/<scientific_name>.  
↪<assembly>.ini $MUGQIC_PIPELINES_HOME/resources/genomes/config/
```

#### Step 4: Validating GenPipes local deployment

You are now all set to use GenPipes pipelines. For each pipeline, you can get help about its usage through the help command:

```
$MUGQIC_PIPELINES_HOME/pipelines/<pipeline_name>/<pipeline_name>.py --help
```

Running pipelines requires other inputs such as Configuration File, Readset File and Design File. For details on how to run individual pipelines you can see [Running GenPipes](#) or [GenPipes User Guide](#).

---

**Note:** In case of any issues, you can try GenPipes [Support](#) or check out other [communication channels](#) to view latest discussions around using GenPipes by the community.

---

---

**Note:** You may also want to check the latest GenPipes deployment and setup instructions listed in [GenPipes README.md](#) file.

---

### 1.5.4 Deploying GenPipes in a container

This document covers details on how to deploy GenPipes locally on your infrastructure using container mechanism. For more details on other available options to deploy and access GenPipes you may refer to [GenPipes Deployment Options Page](#).

You can locally deploy GenPipes by creating a container that hosts all necessary software, configuration details to get you started with running GenPipes within the container. You only need \$User privileges to deploy and use GenPipes locally in a container, no root privileges are needed for this option.

GenPipes genomic analysis tools are designed to run on supercomputing infrastructure or HPC data centres such as Compute Canada servers. However, you can generate generate the pipelines scripts and run smaller experiment on a server with container technology. This mechanism is useful if you are a contributor to GenPipes code or wish to add a feature of your own in the code. Containers make it easy for you to debug and develop GenPipes on you local machine, if you do not have access to GenPipes deployed on [Compute Canada servers](#).

#### Step 1: Install a compatible container technology on your local server

GenPipes can be deployed either using [Docker](#) or [Singularity](#) based containers. Refer to the respective container technology tutorial and user manuals to deploy and check if your container setup is working locally.

## Step 2: Setup a GenPipes development environment

Once your container environment and requisite software is all setup and working, proceed to clone GenPipes somewhere locally under \$HOME directory using the following command:

```
git clone https://bitbucket.org/mugqic/genpipes $HOME/some/dir/genpipes
```

Add the following line to your .bashrc file:

```
export GENPIPES_DEV_DIR=$HOME/some/dir/genpipes
```

Next, use instructions below to start your GenPipes container.

## Step 3: Setup GenPipes in the container

For Docker, use the following command:

```
docker run --privileged -v /tmp:/tmp --network host -it -w $PWD -v $HOME:$HOME --user
↳ $UID:$GROUPS -v /etc/group:/etc/group -v /etc/passwd:/etc/passwd [ -v < CACHE_ON_
↳ HOST >:/cvmfs-cache/ ] c3genomics/genpipes:<TAG>
```

For Singularity, use the following command:

```
singularity run [ -B < /HOST/CACHE/ >:/cvmfs-cache/ ] docker://c3genomics/genpipes:<TAG>
```

Please note, <TAG> refers to one of the tagged GenPipes sources as listed [here](#). Click on ‘master’ branch and in the dropdown, choose ‘Tags’ to select the version that you wish to use for GenPipes.

## Step 4: Load GenPipes dependency modules in the container

As shown in previous step, you can initiate the container process on your machine locally. Next, you need to load GenPipes module using the following command:

```
module load dev_genpipes
```

With this command, GenPipes uses whatever commit of branch that has been checked out in \$HOME/some/dir/genpipes directory.

*Voila! Now you can use GenPipes inside the container just like you would use it locally on a server or on Compute Canada servers.*

For each pipeline, you can get help about its usage through the help command:

```
$MUGQIC_PIPELINES_HOME/pipelines/<pipeline_name>/<pipeline_name>.py --help
```

## Step 5: Running GenPipes Pipelines in a container

Running pipelines requires other inputs such as Configuration File, Readset File and Design File. For details on how to run individual pipelines you can see [Running GenPipes](#) or [GenPipes User Guide](#).

You need to make a note of the fact that GenPipes Pipelines use scheduler's calls (qsub, sbatch) for submitting genomic analysis compute jobs. If you plan to use GenPipes locally using your infrastructure, inside a container, you need to run the GenPipes pipeline python scripts using the "batch mode" option. For local containerized versions of GenPipes, this is the preferred way of running the pipelines, if you don't have access to a scheduler locally such as SLURM or PBS.

This is how you can run GenPipes pipelines such as [DNA Sequencing Pipeline](#), refer to the command below:

```
dnaseq.py -c dnaseq.base.ini dnaseq.batch.ini -j batch -r your-readsets.tsv -d your-  
→design.tsv -s 1-34 -t muggic > run-in-container-dnaseq-script.sh  
  
bash run-in-container-dnaseq-script.sh
```

Please note, there is a disadvantage to running GenPipes Pipelines without a scheduler. In the batch mode, which is configured using the "-j batch" option, all the jobs would run as a batch, one after another, on a single node. If your server is powerful enough, this might be your preferable option. Otherwise, if you would like to take advantage of GenPipes' job scheduling capabilities, you need to install a job scheduler locally in your infrastructure so that GenPipes can work effectively. We recommend SLURM scheduler for GenPipes.

---

**Note:** In case of any issues, you can try GenPipes [Support](#) or check out other [communication channels](#) to view latest discussions around using GenPipes by the community.

You may also want to check the latest GenPipes deployment and setup instructions listed in [GenPipes README.md](#) file.

---

## 1.5.5 Deploying GenPipes in the cloud

This document describes how to deploy GenPipes in the cloud. Following are the supported cloud providers where you can deploy and run GenPipes:

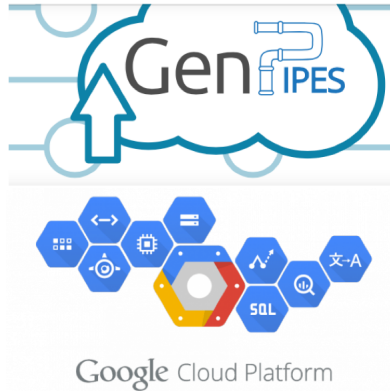
- [GenPipes on Google Cloud Platform \(GCP\)](#)

---

For more details on non-cloud deployment options for GenPipes you may refer to [GenPipes Deployment Options Page](#).

### GenPipes on Google Cloud Platform (GCP)

This section describes how to deploy and run GenPipes using GCP infrastructure.



### Pre-requisites for deploying GenPipes on GCP

If you are a seasoned GCP user and familiar with Google Cloud shell, you can skip this step. For first time GCP users – you may want to try GCP free tier for GenPipes deployment. Follow these steps to create and access GCP account:

- Create an account on GCP. For more instructions, [check out GCP console page](#).
- Get acquainted with Google Cloud Shell. For more instructions, [check out this page](#).
- Create a new project. For detailed instructions [see here](#).

#### Note:

1. Enable Deployment Manager API for your project. For further details, [visit this page](#).
2. You also need to make sure that billing is enabled (even for a GCP free try option).

### Install GenPipes

To install GenPipes on GCP, use Google Cloud Shell Session and download the following install scripts:

```
git clone https://bitbucket.org/mugqic/cloud_deployement.git
```

GenPipes requires [Slurm](#) for scheduling genomic analysis jobs on GCP compute servers. To setup Slurm on your GCP compute infrastructure, run the following commands in your Google Cloud Shell:

```
cd cloud_deployement/gcp/
gcloud deployment-manager deployments create slurm --config slurm-cluster.yaml
```

Once this command is done running, a configuration script is started to install SLURM on the cluster. It will take some time to complete the setup. You will be able to monitor the status of installation once you run the next command. The Cluster configuration is specified in slurm-cluster.yaml file. You can view it to see the controller and worker node setup. By default, only a single node is used for this GCP GenPipes deployment. See node\_count value in slurm-cluster.yaml file.

**Warning:** From here on, your GenPipes cloud is being deployed and your account is getting billed by Google. Remember to shut down the cluster when the analysis is done.

### Access GenPipes Slurm Cluster on GCP

Use the following command to log into the login node of your GCP Slurm cluster:

```
gcloud compute ssh login1 --zone=northamerica-northeast1-a
```

After running the command mentioned above, you are now on GenPipes cloud deployment login node.

### Monitoring GenPipes deployment in GCP

The installation is still running and once you log into the login node of your GCP Slurm cluster, you will see the following welcome message:

```
*** Slurm is currently being installed/configured in the background. ***  
A terminal broadcast will announce when installation and configuration is complete.
```

---

**Note:** Wait for the terminal broadcast. This can take up to 10 minutes. Once you have received it or when you log to the GCP Slurm Cluster login node and there is no warning message displayed as you login, you can go to the next step.

---

### Validate GenPipes on GCP cloud runtime environment

Now that your GCP Slurm Cluster is up and running without any error or warning, you can launch any GenPipes pipeline using the command:

```
<pipeline_name>.py -help
```

For example, to check the help information for GenPipes ChIP Sequencing pipelines, try:

```
chipseq.py -h
```

### GenPipes Test Run in the cloud

To run ChIP Sequencing pipeline using test dataset, use the login node on your GCP Slurm Cluster and issue the following commands in Google Cloud shell corresponding to each step below:

Step 1: Create a new test directory

```
mkdir -p chipseq_test  
cd chipseq_test
```

Step 2: Download test dataset and unzip it as shown below:

```
wget https://www.computationalgenomics.ca/tutorials/chipseq.zip  
unzip chipseq.zip
```

Step 3: GenPipes ChIP Sequencing pipeline needs a configuration file to setup the parameters required by this pipeline. You can download it using the command:

```
wget https://bitbucket.org/mugqic/cloud_deployoment/raw/master/quick_start.ini
```

Step 4: Create ChIP Sequencing pipeline execution command script as shown below:

```
bash # You do not need this line if you did a logout login cycle  
# The next line generates the pipeline script  
chipseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/chipseq/chipseq.base.ini \  
$MUGQIC_PIPELINES_HOME/pipelines/chipseq/chipseq.cedar.ini \  
quick_start.ini \  
-j slurm \  
-r readsets.chipseqTest.chr22.tsv \  

```

(continues on next page)



(continued from previous page)

```
-d designfile_chipseq.chr22.txt \
-s 1-18 > chipseqScript.sh
```

Step 5: Now you can execute ChIP Sequencing pipeline using the following command:

```
bash chipseqScript.sh
```

Step 6: View Progress of your pipeline and jobs by using `squeue` command. For more [Slurm commands](#) and details on monitoring Slurm cluster, you can see [Slurm documentation](#)

There are several ways to check the status of your jobs in the queue. Below are a few SLURM commands to make use of. Use the Linux ‘man’ command to find loads of additional information about these commands as well.

```
squeue <options>
```

where you can use the following options:

```
-u username
-j jobid
-p partition
-q qos
```

For example:

```
[shalz@ubuntu_srv:/$ squeue -u shaloo
JOBID PARTITION  NAME      USER    ST      TIME  NODES NODELIST(REASON)
92311  debug       test      shaloo   R        0:06     2 e06ne9s0e,c17n09
88915  xyz         GPU_test  shaloo   PD        0:00     1 (Priority)
91716  xyz         hell_te   shaloo   R        0:08     2 d19res0e,d16n08
91791  xyz         hello_te  shaloo   PD        0:00     2 (Priority)
91792  xyz         hello_te  shaloo   PD        0:00     2 (Priority)
```

Step 7: Shutdown GCP compute resources (Very Important!!!) You need to make sure that after your jobs are run, you need to shutdown your GenPipes Slurm Cluster on GCP otherwise you will continue to be billed for the same. After all your jobs have run, use the following command to exit out of your login node Google Cloud shell session:

```
exit
```

This command closes the Slurm Login node shell. You are now back on your cloud shell administrative server. You can shut down your GenPipes cloud cluster by running the following script:

```
gcloud deployment-manager deployments delete slurm
```

### Further information

If you run into any issues, please refer to [Troubleshooting runtime issues](#) section of this documentation and visit [GenPipes Support](#) page.

For advanced GCP cloud setup scenarios and for the latest updates on deploying GenPipes in the cloud, details regarding Slurm stand alone cluster setup, or multi-cluster federation setup or to burst out of on-premise cluster to GCP while running GenPipes, refer to the [README.md](#) file.

## 1.6 GenPipes User Guide

GenPipes is an open source framework developed at [Canadian Centre for Computational Genomics \(C3G\)](#) to facilitate writing and execution of multi-step processing pipelines on HPC clusters. GenPipes is written in Python and is primarily geared towards Next Generation Sequencing analysis. It comes with 12 validated pipelines that include RNA-Seq, ChIP-Seq, DNA-Seq, Methyl-Seq and Hi-C processing.

The new users can get started by referring to the [Quick Start Guide](#). Seasoned users can refer to specifics of any of the pipelines by browsing the [Pipeline Reference section](#).

Users can refer to [latest release notes](#) for GenPipes Version 3.6.2 for more information regarding what is new, which pipelines were updated and what issues are fixed in this release.

If you wish to not only use GenPipes Open Source offering but also contribute to the same or contribute to GenPipes documentation, refer to the section [Getting involved](#) below.

*Happy genomic analysis with GenPipes!*

---

This GenPipes User Guide is organized as follows:

### 1.6.1 GenPipes Quick Start Guide








#### Table of Contents

#### GenPipes Check List

Before you begin using GenPipes for the first time, it would be useful to quickly go over the basic requirements of using GenPipes. This checklist is created to help you get started with GenPipes quickly and get your bearings correct before you embark upon genomic analysis with GenPipes.

## Skill sets and domain know-how

The following table lists some of the skill sets, technologies and know-how that a GenPipes user is expected to have.

	Requirements
	High Performance Computing (HPC) for genomics - environment and software modules used
	Cluster Computing, Cloud environment for genomics
	Compute Canada HPC infrastructure <a href="#">usage account</a> and familiarity
	Use of shared, distributed File Systems such as <a href="#">CVMFS</a>
	HPC Schedulers such as <a href="#">PBS</a> , <a href="#">Slurm</a>
	Bioinformatics Software Dependencies (Local/Pre-installed on CCDB)
	GenPipes <i>Basic Concepts</i> - Pipelines, steps etc.

## Pre-requisites for running GenPipes

Besides the skills and technologies listed above, you need powerful hardware and some pre-installed software modules before you can run GenPipes. For a complete list, please refer to [docs\\_pre\\_req\\_chklist](#) for details.

## Choose GenPipes Deployment Option

Before you can quickly get started with GenPipes, you need some clarity regarding your GenPipes deployment preference.

This choice governs where you will execute or run GenPipe pipelines for conducting genomic analysis. As you may be aware, genomic analysis is time consuming and resource intensive processing that requires significant amount of raw compute power, memory and reference dataset storage space.

GenPipes is available as pre-deployed software that can be run using compute and storage resources available for C3G users through the partnership with [Compute Canada](#) datacenter. If you already have a [Compute Canada account](#), this may be your fastest option to use GenPipes.

Otherwise, there are other options available as well but they may need additional time and expertise to download and deploy one of the [GenPipes Release Builds](#) that come in the form of a compressed file with some basic modules, test datasets and reference genomes that can be used for basic genomic analysis. This kind of deployment works for simple analysis unless you have access to high end compute power or local HPC resources.

Local deployment can be either on bare metal server, on a VM or inside a container.

For details on available GenPipes deployment options and how to deploy, please visit [How to deploy GenPipes?](#) Guide.

## Get GenPipes

Once you have made your *choice* regarding GenPipes deployment option, if you chose to use C3G pre-built and deployed GenPipes setup on Compute Canada servers, you don't need to get and setup any build or code. It is already taken care of. You simply need to start using your Compute Canada account and access the latest copy of GenPipes deployed therein.

However, if you choose to deploy GenPipes on your own instead of using the pre-installed setup managed by C3G using Compute Canada servers, you can use either of the options listed below. You can either obtain a downloadable GenPipes Release Build or obtain a copy of GenPipes sources and build, deploy them yourself.

- Pre-built, packaged GenPipes via the [GenPipes Download Page](#)
  - For detailed instructions on how to setup and configure GenPipes locally on your server refer to [Deploying GenPipes locally](#).
  - If you wish to deploy and setup GenPipes inside a container, [see here for instructions](#).
  - You can also choose to deploy GenPipes on Google Compute Cloud. [See here](#) for details.
- Build your own version of GenPipes by downloading [GenPipes Source Code](#)
  - For details on GenPipes build process, see [GenPipes Developer Guide](#). Once you generate the build, you can follow the same instructions as for using a downloadable release to setup and configure GenPipes.

## Run GenPipes

Once you have *chosen* your GenPipes use option and have access to the GenPipes executable through Compute Canada server access or on your local server, you need to setup and configure your GenPipes environment as per your deployment choice. After you have configured it, you are ready to execute GenPipes.

First of all, test that it works, without actually running a pipeline, by using the `-help` or `-h` option. For details, please refer to the Getting Started Guide [Run GenPipes section](#).

Each of the available GenPipes Pipeline has its own unique execution command. You can find details for each in the [Pipeline Reference Guide](#). It also has sections demonstrating *example runs* for each pipeline.

New users may benefit from the [GenPipes Tutorial](#) section of this documentation that provides step by step instructions on how to execute a few sample GenPipes pipelines. There is also a tutorial available for [running GenPipes in the cloud using Google Compute Platform](#).

In case you run into any runtime issues or errors, do refer to [Troubleshooting GenPipes Runtime Issues](#) or browse the GenPipes [Support](#) sections.

Happy analysis with GenPipes! We would love to hear your feedback on GenPipes or the documentation.

[Contributions](#) to GenPipes or its documentation are most welcome!

## Getting Help on GenPipes

If you need help on GenPipes, you can refer to [Support](#) section in the left hand navigation pane.

Alternatively, it is recommended that you check out various [Channels](#). It is quite possible that the issue you are facing is already being discussed and resolved there with some workaround.

If you are locally situated near C3G centre in Montreal or Toronto, you can avail the [Open Door](#) sessions at C3G.

Besides these, if you need deeper consultation regarding usage of GenPipes Pipelines for a real life research scenario, you can save time and money by requesting for [C3G Bioinformatics Services](#) tailored to your specific needs.

For any support issues, you can also drop an email to the GenPipes Help Desk on the address:

`pipelines%40computationalgenomics.ca`

## Get Involved with GenPipes

GenPipes bioinformatics pipelines were originally developed at the Canadian Centre for Computational Genomics (C3G), as part of the GenAP project. These, are now available for public use. GenPipes includes a wide array of *pipelines*, including RNA-Seq, ChIP-Seq, WGS, exome sequencing, Bisulfite sequencing, Hi-C, capture Hi-C, metagenomics and the latest SARS-CoV-2 genome sequencing pipeline.

## Contributing to GenPipes

### We love your input!

The objective of this document is to make contributing to GenPipes as easy and transparent as possible, for any of the following types of contributions from the wider community:

1. Reporting a bug or issue
2. Submitting a fix
3. Proposing new features
4. Discussing code optimizations
5. Becoming a maintainer

## Tracking GenPipes Issues and Feature Requests

GenPipes uses [Bitbucket](#) to host its source code, track issues and feature requests, as well as pull requests.

Pull requests are the best way to propose changes to the code base. Please ensure that you follow the following process as GenPipes is complex and requires careful validation of any updates. Also, you may need computational resources as GenPipes is demanding in terms of compute resources.

- Fork the repo and create your branch from master.
- If you have added code that should be tested, add tests.
- If applicable, update the documentation.
- Ensure the test suite passes.
- Make sure your code lints.
- Issue that pull request!

## License

GenPipes is available under GPL v3 [license](#). Any contributions you make will be under the GPL (v3) license.

## Reporting a bug or issue

You can report a bug or an issue related to GenPipes using the following template:

- Single line summary briefly describing the issue
- **Steps to reproduce**
  - Specific instructions
  - If required, provided commands and code that caused the issue
  - Make sure you specify GenPipes version number, deployment type
- What was expected and what actually happened
- Other insights or logs that can help debug the issue.

This guide is adapted from [FaceBook Draft.js Guide](#).

---

This guide aims at helping you get started with GenPipes quickly (under 15 minutes!) and locate the information you require to meet any of the objectives listed below:

- **Objective:** *GenPipes Check List*  
Figure out what is needed to run GenPipes in terms of software, hardware, login accounts and other resources. **See** [GenPipes Check List](#) section in Quick Start Guide Table of Contents above.
- **Objective:** *Familiarise with GenPipes basics*  
Understand GenPipes basics, concepts and figure out how it works in order to apply it for your specific genomic analysis use case. **See** [Introduction to GenPipes`](#) and [Why GenPipes?](#) section of this documentation.
- **Objective:** *Choose GenPipes deployment*  
Should I deploy GenPipes on my resources or use a pre-installed copy of the same deployed on Compute Canada servers? **Refer** to [Choose GenPipes deployment option](#) section in Quick Start Guide Table of Contents above.
- **Objective:** *Get Latest GenPipes*  
How to obtain the latest copy of GenPipes (sources, pre-packaged build, documentation etc.)? **Refer** to [Get GenPipes](#) section in the Table of Contents above.
- **Objective:** *Run GenPipes for the first time*  
How do I run / execute GenPipes in 3 simple steps? **See** [Run GenPipes](#) section of this guide as shown in Table of Contents above.
- **Objective:** *Get Help!*  
How can I achieve this with GenPipes or I ran into issues - where do I find help? **Check** out [Getting Help on GenPipes](#) above.
- **Objective:** *Solve commonly known issues in GenPipes Usage*  
I ran into issues - is there a workaround or has it been resolved already? For this, you may want to **check** out [GenPipes FAQ](#) or browse [GenPipes Channels](#) for more insights.

## 1.6.2 Pipelines Reference Guide

GenPipes implements standardized genomics workflows, including DNA-Seq, tumour analysis, RNA-Seq, de novo RNA-Seq, ChIP-Seq, SARS-CoV-2 genome sequencing, methylation sequencing, Hi-C, capture Hi-C, and metagenomics. All pipelines have been implemented following a robust design and development routine by following established best practices standard operating protocols. The pipelines accept a binary sequence alignment map ([BAM](#)) or a [FASTQ](#) file as input.

This guide contains usage manual and reference information for all available GenPipes genomic analysis pipelines. Visit [GenPipes Real-life Applications and use cases](#) to see how these pipelines can be deployed for complex next-generation genomic analysis in real life scenarios.

### Amplicon Sequencing Pipeline

Amplicon sequencing (ribosomal RNA gene amplification analysis) is a metagenomic pipeline. It is based on the established [Quantitative Insights into Microbial Ecology \(QIIME\)](#) procedure for amplicon-based metagenomics. It assembles read pairs using [Fast Length Adjustment of Short Reads \(FLASH\)](#), detects chimeras with [UCHIME](#), and picks operational taxonomic units using [VSEARCH](#). Operational taxonomic units are then aligned using [PyNAST](#) and clustered with [FastTree](#). Standard diversity indices, taxonomical assignments, and ordinations are then calculated and reported graphically.

- [Introduction](#)
- [Version](#)
- [Usage](#)
- [Example Run](#)
- [Pipeline Schema](#)
- [Pipeline Steps](#)
- [Step Details](#)
- [More information](#)

### Introduction

Amplicon sequencing is a highly targeted gene sequencing approach used to analyze genetic variation in specific genomic regions. Amplicons are Polymerase Chain Reaction (PCR) products and the ultra-deep sequencing allows for efficient variant identification and characterization. Amplicon sequencing uses oligonucleotide probes that target and capture genomic regions of interest and then uses next-generation sequencing techniques.

#### Uses of Amplicon sequencing

1. Diagnostic microbiology utilizes amplicon-based profiling that allows to sequence selected amplicons such as regions encoding 16S rRNA that are used for species identification.
2. Discovery of rare somatic mutations in complex samples such as tumors mixed with germline DNA.

See [More Information](#) section below for details.

## Version

3.6.2

For the latest implementation and usage details refer to Amplicon Sequencing implementation [README](#) file.

---

## Usage

```
ampliconseq.py [-h] [--help] [-c CONFIG [CONFIG ...]] [-s STEPS]
                [-o OUTPUT_DIR] [-j {pbs,batch,daemon,slurm}] [-f]
                [--no-json] [--report] [--clean]
                [-l {debug,info,warning,error,critical}]
                [--sanity-check]
                [--container {wrapper, singularity} <IMAGE PATH>]
                [--genpipes_file GENPIPES_FILE]
                [-t {qiime,dada2}] [-d DESIGN] [-r READSETS] [-v]
```

## Optional Arguments

`-t {qiime,dada2}, --type {qiime,dada2}`

AmpliconSeq analysis **type**

`-d DESIGN, --design DESIGN`

design file

`-r READSETS, --readsets READSETS`

readset file

`-h` show this help message **and** exit

`--help` show detailed description of pipeline **and** steps

`-c CONFIG [CONFIG ...], --config CONFIG [CONFIG ...]`

config INI-style **list** of files; config parameters  
are overwritten based on files order

`-s STEPS, --steps STEPS` step **range** e.g. '1-5', '3,6,7', '2,4-8'

`-o OUTPUT_DIR, --output-dir OUTPUT_DIR`

output directory (default: current)

`-j {pbs,batch,daemon,slurm}, --job-scheduler {pbs,batch,daemon,slurm}`

job scheduler **type** (default: slurm)



<code>-f, --force</code>	force creation of jobs even <b>if</b> up to date (default: false)
<code>--no-json</code>	do <b>not</b> create JSON file per analysed sample to track the analysis status (default: false i.e. JSON file will be created)
<code>--report</code>	create ' <b>pandoc</b> ' command to merge <b>all</b> job markdown report files <b>in</b> the given step <b>range</b> into HTML, <b>if</b> they exist; <b>if</b> <code>--report</code> <b>is</b> set, <code>--job-scheduler</code> , <code>--force</code> , <code>--clean</code> options <b>and</b> job up-to-date status are ignored (default: false)
<code>--clean</code>	create ' <b>rm</b> ' commands <b>for</b> <b>all</b> job removable files <b>in</b> the given step <b>range</b> , <b>if</b> they exist; <b>if</b> <code>--clean</code> <b>is</b> set, <code>--job-scheduler</code> , <code>--force</code> options <b>and</b> job up-to-date status are ignored (default: false)
<code>-l {debug,info,warning,error,critical}, --log {debug,info,warning,error,critical}</code>	log level (default: info)
<code>--sanity-check</code>	run the pipeline in 'sanity check mode' to verify all the input files needed for the pipeline to run are available on the system (default: false)
<code>--container {wrapper, singularity} &lt;IMAGE PATH&gt;</code>	run pipeline inside a container providing a container image path <b>or</b> accessible singularity hub path
<code>-v, --version</code>	show the version information <b>and</b> exit
<code>-g GENPIPES_FILE, --genpipes_file GENPIPES_FILE</code>	Commands <b>for</b> running the pipeline are output to this file pathname. The data specified to pipeline command line <b>is</b> processed <b>and</b> pipeline run commands are stored <b>in</b> GENPIPES_FILE, <b>if</b> this option <b>is</b> specified . Otherwise, the output will be redirected to stdout . This file can be used to actually " <b>run the GenPipes Pipeline</b> ".



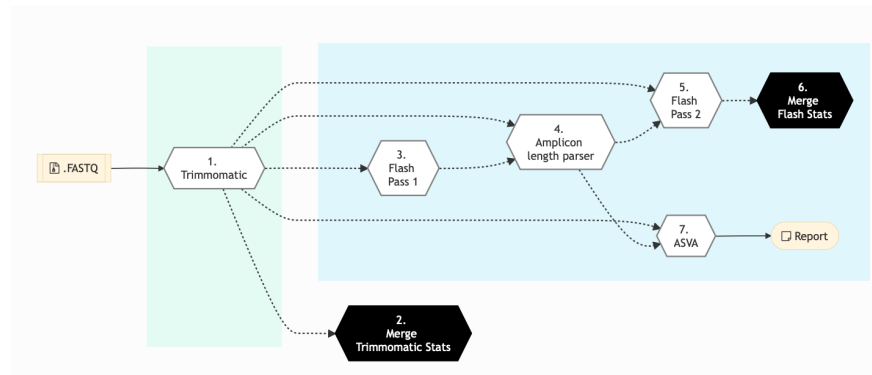
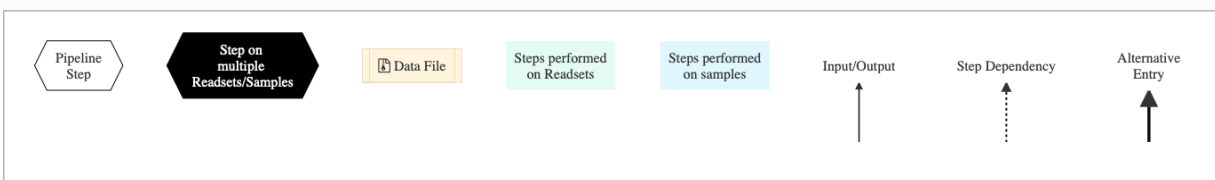


Fig. 5: Figure: Schema of DADA2 Amplicon Sequencing protocol



The following figure shows the pipeline schema for **DADA2 Pipeline** type of amplicon sequencing protocol.

## Pipeline Steps

The table below shows various steps that constitute the Amplicon Sequencing QIIME and DADA2 type genomic analysis pipelines.

	<i>QIIME sequencing Steps</i>	<i>DADA2 sequencing Steps</i>
1.	<i>Trimmomatic16S Step</i>	<i>Trimmomatic16S Step</i>
2.	<i>Merge Trimmomatic Stats</i>	<i>Merge Trimmomatic Stats</i>
3.	<i>Flash Pass 1</i>	<i>Flash Pass 1</i>
4.	<i>Amplicon Length Parser</i>	<i>Amplicon Length Parser</i>
5.	<i>Flash Pass 2</i>	<i>Flash Pass 2</i>
6.	<i>Merge Flash Stats</i>	<i>Merge Flash Stats</i>

continues on next page

Table 1 – continued from previous page

	<i>QIIME sequencing Steps</i>	<i>DADA2 sequencing Steps</i>
7.	<i>Catenate</i>	<i>ASVA</i>
8.	<i>UCHIME Step</i>	
9.	<i>Merge UCHIME Stats</i>	
10.	<i>OTU Picking</i>	
11.	<i>OTU Rep Picking</i>	
12.	<i>OTU Assigning</i>	
13.	<i>OTU Table</i>	
14.	<i>OTU Alignment</i>	
15.	<i>Filter Alignment</i>	
16.	<i>Phylogeny</i>	
17.	<i>QIIME Report</i>	
18.	<i>Multiple Rarefaction</i>	
19.	<i>Alpha Diversity</i>	
20.	<i>Collate Alpha</i>	
21.	<i>Sample Rarefaction Plot</i>	
22.	<i>QIIME Report 2</i>	
23.	<i>Single Rarefaction</i>	

continues on next page

Table 1 – continued from previous page

	<i>QIIME sequencing Steps</i>	<i>DADA2 sequencing Steps</i>
24.	<i>CSS Normalization</i>	
25.	<i>Rarefaction Plot</i>	
26.	<i>Summarize Taxonomy</i>	
27.	<i>Plot Taxonomy</i>	
28.	<i>Plot Heatmap</i>	
29.	<i>Krona</i>	
30.	<i>Plot to Alpha</i>	
31.	<i>Beta Diversity</i>	
32.	<i>Principal Coordinate Analysis</i>	
33.	<i>PCoA Plot</i>	
34.	<i>Plot to Beta</i>	

## Step Details

Following are the various steps that are part of GenPipes Amplicon Sequencing genomic analysis pipeline:

### Trimmomatic16S

MiSeq raw reads adapter & primers trimming and basic QC is performed using [Trimmomatic](#). If an adapter FASTA file is specified in the config file (section ‘trimmomatic’, param ‘adapter\_fasta’), it is used first. Else, Adapter1, Adapter2, Primer1 and Primer2 columns from the readset file are used to create an adapter FASTA file, given then to Trimmomatic. Sequences are reversed-complemented and swapped.

This step takes as input files:

1. MiSeq paired-End FASTQ files from the readset file.

### Merge Trimmomatic Stats

The trim statistics per readset are merged in this step.

### **Flash Pass 1**

TBD-GenPipes-Dev

### **Amplicon Length Parser**

In this step, we look at [FLASH](#) output to set amplicon lengths input for [DADA2](#). As minimum eligible length, a given length needs to have at least 1% of the total number of amplicons.

### **Flash Pass 2**

TBD-GenPipes-Dev

### **Merge Flash Stats**

The paired end merge statistics per readset are merged in this step.

### **Catenate**

This step catenates all the reads in one file for further analysis. As input, it takes the merged FASTQ files from the previous FLASH step.

### **UCHIME**

This step takes catenated FASTA file from the previous Catenate step and uses it for reference based chimera detection using [VSearch tool](#).

### **Merge UCHIME Stats**

The chimeric sequences filtered out statistics per readset are merged at this step.

### **Operational Taxonomic Unit (OTU) Picking**

This step takes catenated FASTA file from the previous Catenate step as input. The OTU picking step (de novo & close\_ref) assigns similar sequences to operational taxonomic units (OTUs) by clustering sequences based on a user-defined similarity threshold. Method per default uses the following tools:

- [VSearch tool](#)
- [QIIME tool](#)

### **OTU Representative Picking**

After picking OTUs, this step pick a representative sequence for each OTU. This step takes as input files:

- OTU file from previous step
- Catenated and filtered FASTA file from filter\_chimeras step.

### **OTU Assigning**

Given a set of OTUs, this step attempts to assign the taxonomy of each OTU using [Uclust algorithm](#). As input, it takes OTU representative sequence file from the previous step.

### **OTU Table**

This step make a consensus OTU table in biom format. It tabulates the number of times an OTU is found in each sample, and adds the taxonomic predictions for each OTU.

This step takes as input files:

- OTU picking file.
- Taxonomy assignment for each OTU from the previous step.

### OTU Alignment

OTU Alignment uses OTU representative sequence file as input and aligns the OTU representative sequence using [PyNAST tool](#).

### Filter Alignment

This step takes the alignment sequence file as input and filters it by removing positions that are gaps in every sequence.

### Phylogeny

This step uses filtered alignment sequence file from previous step and builds a phylogenetic tree from a multiple sequence alignment using [FastTree algorithm](#).

### QIIME Report

In this step, the first part of the report for taxonomic affiliation is created.

### Multiple Rarefaction

This is the first of the four steps (1/4) for rarefaction plot. It takes OTU non-rarefied table in biom format as input. It rarefies OTU table by random sampling (without replacement) at different depth in order to perform rarefaction analysis. You need to provide the minimum/maximum number of sequences per samples and the size of each steps between the min/max of seqs/sample.

### Alpha Diversity

This is the second of the four steps (2/4) for rarefaction plot. It takes as input, multiple OUT rarefied table in biom format from the previous step and calculates alpha diversity on each sample using a variety of alpha diversity metrics (chao1, shannon, observed OTUs).

### Collate Alpha

This is the third of the four steps (3/4) for rarefaction plot. It merges all the alpha diversity computed in the previous step.

### Sample Rarefaction Plot

This is the fourth and last of the four steps (4/4) for rarefaction plot. Here, rarefaction curve for each sample is plotted.

### QIIME Report 2

In this step, the second part of the report for taxonomic affiliation is created.

### Single Rarefaction

Single rarefaction takes OTU table in biom format as input. This step is recommended. It subsamples (rarefy) all the samples to an equal number of sequences for further comparison. You have to provide the number of sequences to subsample per sample in the configuration file (single\_rarefaction\_depth).

### CSS Normalization

CSS Normalization takes OTU table in biom format as input. This is the alternative method for normalization to rarefaction. The CSS Matrix normalization step is recommended.

### Rarefaction Plot

In this step, rarefaction curve for each sample is plotted on the same plot.

### Summarize Taxonomy

This is the first of three (1/3) steps for taxonomic affiliation plot. If available, it takes OTU rarefied table in biom format as input. Otherwise, it takes OTU non-rarefied table in biom format. It summarizes information of taxonomic groups within each sample at different taxonomic level.

### Plot Taxonomy

This is the second of three (2/3) steps for taxonomic affiliation plot. It takes summarization information from the previous step as input and makes taxonomy summary bar plot based on taxonomy assignment.

### **Plot Heatmap**

This is the third and last of three (3/3) steps for taxonomic affiliation plot. It takes summarized information from previous step and makes heatmap at phylum level.

### **Krona**

This step plots the [Krona chart](#) for taxonomic affiliation.

### **Plot to Alpha**

This step generates the first part of the final report for Amplicon sequencing pipeline. The report displays results related to taxonomy, heatmap and alpha diversity.

### **Beta Diversity**

This is the first of three (1/3) steps for 2D [PCoA analysis](#) plot.

It takes the following data as input files:

- OTU rarefied table in biom format
- Tree file

This step calculates beta diversity (pairwise sample dissimilarity) on OTU table. The OTU table has to be normalized. Only works with  $\geq 4$  samples

Principal Coordinate Analysis ([PCoA analysis](#)) is the second step of three (2/3) for PCoA plot. It takes the matrix produced in the previous step and computes coordinates for PCoA.

### **PCoA Plot**

This is the last of three (3/3) steps for 2D [PCoA analysis](#) plot.

### **Plot to Beta**

Here the last part of the final report for Amplicon sequencing pipeline is generated. The report displays results related to beta diversity PCoA plots.

### **Asva**

This step checks for the design file (required for [PCoA Plot](#))

---

## **More information**

For the latest implementation and usage details refer to Amplicon Sequencing implementation [README.md](#) file.

- [Amplicon sequencing techniques](#)
  - [Amplicon Sequencing Primer](#)
  - [High-throughput amplicon sequencing.](#)
  - [Trimmomatic - flexible trimming.](#)
-



## ChIP Sequencing Pipeline

Chromatin Immunoprecipitation (ChIP) sequencing technique is used for mapping DNA-protein interactions. It is a powerful method for identifying genome-wide DNA binding sites for transcription factors and other proteins. The ChIP-Seq workflow is based on the [ENCODE Project](#) workflow. It aligns reads using the [Burrows-Wheeler Aligner](#). It creates tag directories using [Homer routines](#). Peaks are called using [Model based Analysis for Chip Sequencing \(MACS2\)](#) and annotated using Homer. Binding motifs are also identified using Homer. Metrics are calculated based on [IHEC requirements](#). The ChIP-Seq pipeline can also be used for assay for transposase-accessible chromatin using sequencing (ATAC-Seq) samples. At GenPipes, we are developing a pipeline that is specific to [ATAC-Seq](#).

- [Introduction](#)
- [Version](#)
- [Usage](#)
- [Example Run](#)
- [Pipeline Schema](#)
- [Pipeline Steps](#)
- [Step Details](#)
- [More Information](#)

### Introduction

ChIP-Seq experiments allows the isolation and sequencing of genomic DNA bound by a specific transcription factor, covalently modified histone, or other nuclear protein. The pipeline starts by trimming adapters and low quality bases and mapping the reads (single end or paired end ) to a reference genome using [Burrows-Wheeler Aligner \(BWA\)](#). Reads are filtered by mapping quality and duplicate reads are marked. Then, Homer quality control routines are used to provide information and feedback about the quality of the experiment. Peak calls is executed by MACS and annotation and motif discovery for narrow peaks are executed using Homer. Statistics of annotated peaks are produced for narrow peaks and a standard report is generated.

For more details, see [ChIP-Seq Guidelines](#), and [MUGQIC\\_Bioinfo\\_Chip-Seq.pptx](#).

### Version

3.6.2

For the latest implementation and usage details refer to Hi-C Sequencing implementation [ChIP-Seq Pipeline README](#) file.

## Usage

```
chipseq.py [-h] [--help] [-c CONFIG [CONFIG ...]] [-s STEPS]
           [-o OUTPUT_DIR] [-j {pbs,batch,daemon,slurm}] [-f]
           [--no-json] [--report] [--clean]
           [-l {debug,info,warning,error,critical}] [--sanity-check]
           [--container {wrapper, singularity} <IMAGE PATH>]
           [--genpipes_file GENPIPES_FILE]
           [-d DESIGN] [-t {chipseq, atacseq}] [-r READSETS] [-v]
```

## Optional Arguments

`-t {chipseq, atacseq}, --type {chipseq, atacseq}`

Type of pipeline (default chipseq)

`-d DESIGN, --design DESIGN`

design file

`-r READSETS, --readsets READSETS`

readset file

`-h` show this help message **and** exit

`--help` show detailed description of pipeline **and** steps

`-c CONFIG [CONFIG ...], --config CONFIG [CONFIG ...]`

config INI-style **list** of files; config parameters are overwritten based on files order

`-s STEPS, --steps STEPS` step **range** e.g. '1-5', '3,6,7', '2,4-8'

`-o OUTPUT_DIR, --output-dir OUTPUT_DIR`

output directory (default: current)

`-j {pbs,batch,daemon,slurm}, --job-scheduler {pbs,batch,daemon,slurm}`

job scheduler **type** (default: slurm)

`-f, --force` force creation of jobs even **if** up to date (default: false)

`--no-json` do **not** create JSON file per analysed sample to track the analysis status (default: false i.e. JSON file will be created)

<code>--report</code>	create 'pandoc' command to merge all job markdown report files in the given step range into HTML, if they exist; if --report is set, --job-scheduler, --force, --clean options and job up-to-date status are ignored (default: false)
<code>--clean</code>	create 'rm' commands for all job removable files in the given step range, if they exist; if --clean is set, --job-scheduler, --force options and job up-to-date status are ignored (default: false)
<code>-l {debug,info,warning,error,critical}, --log {debug,info,warning,error,critical}</code>	log level (default: info)
<code>--sanity-check</code>	run the pipeline in `sanity check mode` to verify all the input files needed for the pipeline to run are available on the system (default: false)
<code>--container {wrapper, singularity} &lt;IMAGE PATH&gt;</code>	run pipeline inside a container providing a container image path or accessible singularity hub path
<code>-v, --version</code>	show the version information and exit
<code>-g GENPIPES_FILE, --genpipes_file GENPIPES_FILE</code>	Commands for running the pipeline are output to this file pathname. The data specified to pipeline command line is processed and pipeline run commands are stored in GENPIPES_FILE, if this option is specified. Otherwise, the output will be redirected to stdout. This file can be used to actually "run the GenPipes Pipeline".

**Note:** ChIPSeq Pipeline Design File Format

Sample	MarkName	EW22_EW3_vs_EW7_TC71
EW22	H3K27ac	1
EW3	H3K27ac	1
EW7	H3K27ac	2
TC71	H3K27ac	2

where, the numbers specify the sample group membership for this contrast:

'0' or '': the sample does not belong to any group;  
 '1': the sample belongs to the control group;  
 '2': the sample belongs to the treatment test case group.

The design file only accepts 1 for control, 2 for treatment and 0 for other samples that do not need to compare.

---

**Warning:** Incorrect Design File Format

Please note that the first and second column in the design file must be sample name and histone mark/binding protein respectively.

**If the user specifies any value other than the permitted ones above in the design file, the pipeline will fail.**

### Example Run

You can download [ChIP-Seq test dataset](#) and use the following command to execute the ChIP-Seq genomics pipeline:

```
chipseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/chipseq/chipseq.base.ini $MUGQIC_
↳ PIPELINES_HOME/pipelines/chipseq/chipseq.beluga.ini -r readset.chipseq.txt -d design.
↳ chipseq.txt -s 1-20 -g chipseqScript.txt

bash chipseqScript.txt
```

**Warning:** While issuing the pipeline run command, use ``-g GENPIPES_FILE`` option (see example above) instead of using the ``> GENPIPES_FILE`` option supported by GenPipes so far, as shown below:

```
[genpipes_seq_pipeline].py -t mugqic -c $MUGQIC_PIPELINES_HOME/pipelines/[genpipes_
↳ seq_pipeline]/[genpipes_seq_pipeline].base.ini $MUGQIC_PIPELINES_HOME/pipelines/
↳ [genpipes_seq_pipeline]/[genpipes_seq_pipeline].guillimin.ini -r readset.[genpipes_
↳ seq_pipeline].txt -s 1-6 > [genpipes_seq_pipeline]_commands_mugqic.sh

bash [genpipes_seq_pipeline]_commands_mugqic.sh
```

``> scriptfile`` should be considered deprecated and ``-g scriptfile`` option is recommended instead.

Please note that redirecting commands to a script ``> genpipe_script.sh`` is still supported for now. **But going forward, this mechanism might be dropped in a future GenPipes release.**

The commands will be sent to the job queue and you will be notified once each step is done. If everything runs smoothly, you should get MUGQICexitStatus:0 or Exit\_status=0. If that is not the case, then an error has occurred after which the pipeline usually aborts. To examine the errors, check the content of the job\_output folder.

---

### Pipeline Schema

Figure below shows the schema of ChIP-Seq protocol.

Following is the schema for the ChIP-Seq pipeline using the -t atacseq option:

---

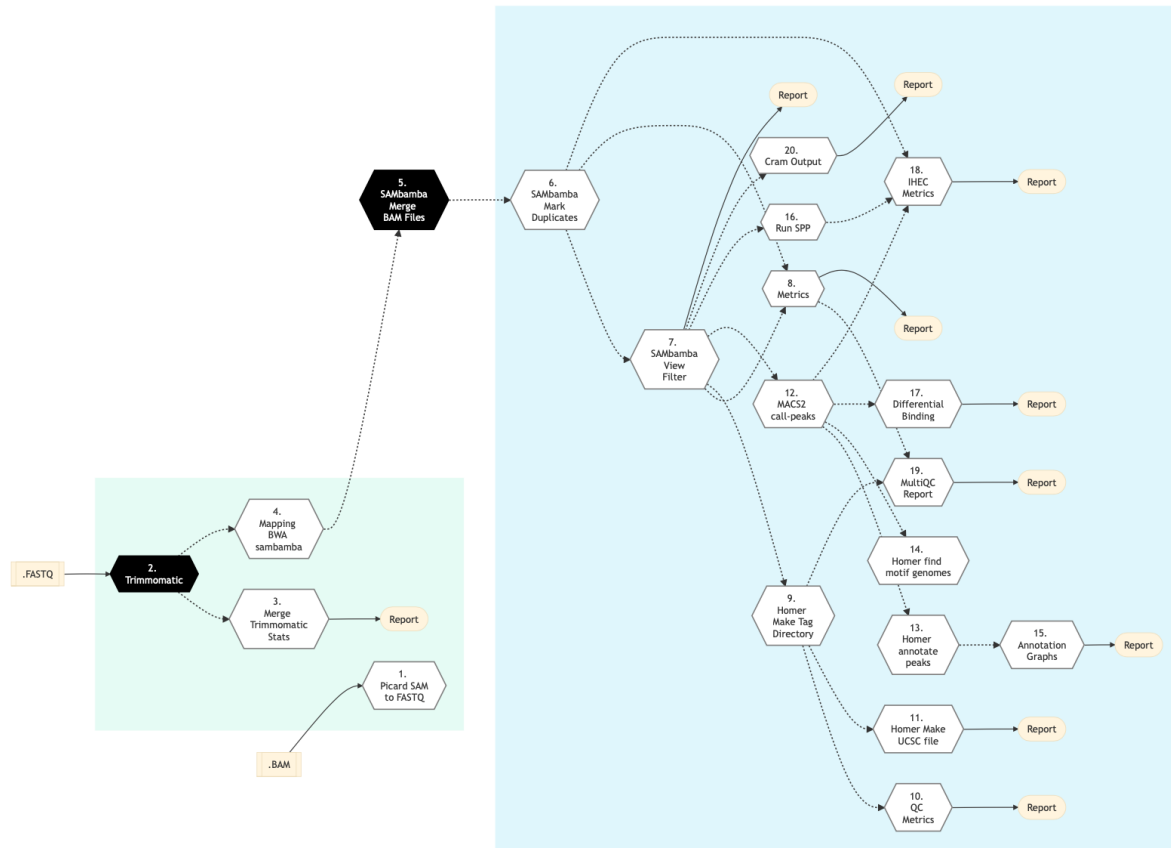
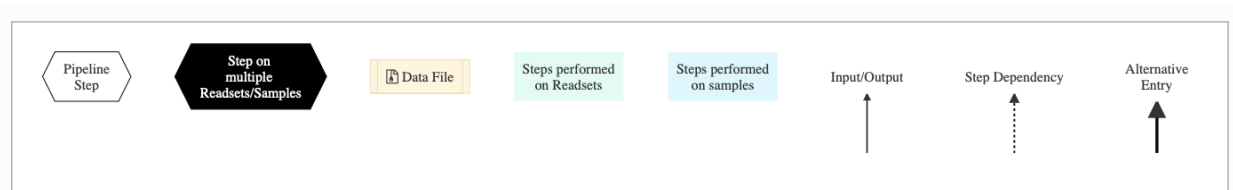


Fig. 6: Figure: Schema of ChIP Sequencing protocol



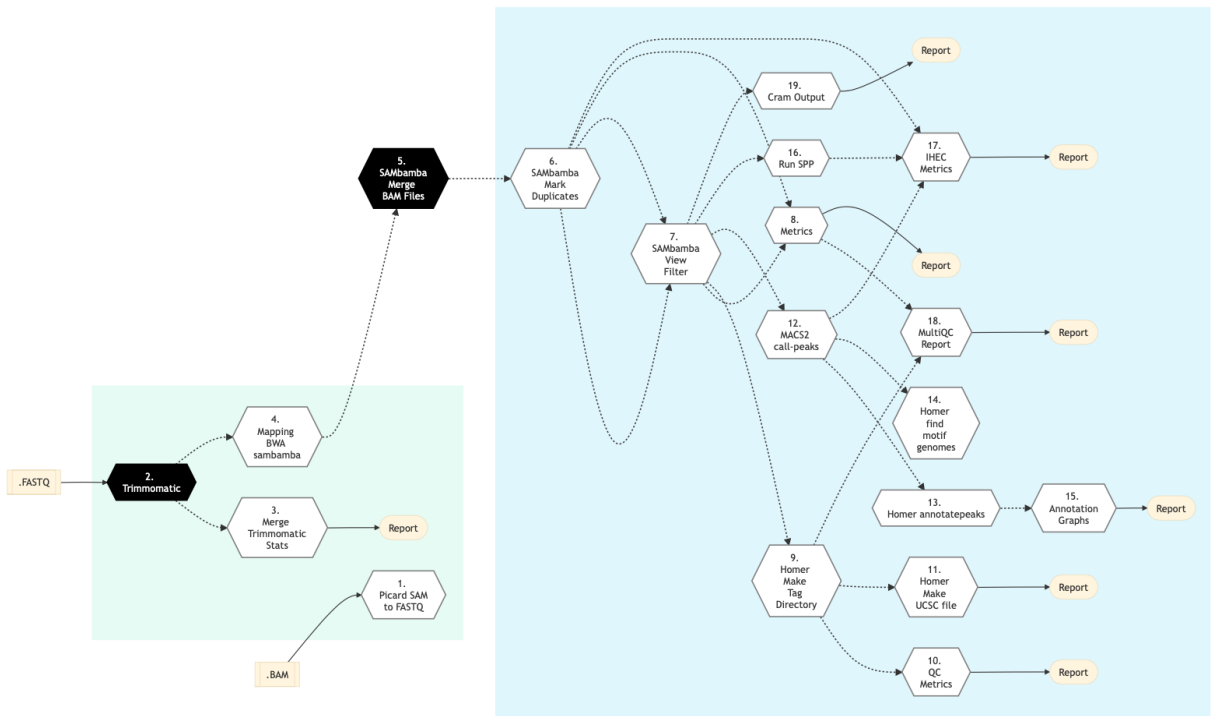
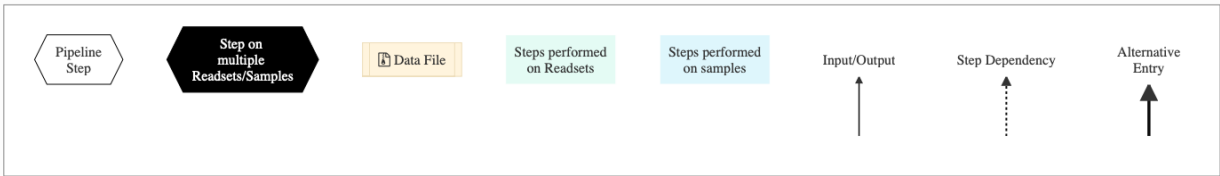


Fig. 7: Figure: Schema of ChIP Sequencing -t atacseq protocol



## **Pipeline Steps**

The table below shows various steps that constitute the ChIP sequencing pipeline:

	ChIP Sequencing Steps	ChIP Sequencing (atacseq)
1.	<i>Picard Sam to Fastq</i>	<i>Picard Sam to Fastq</i>
2.	<i>Trimmomatic</i>	<i>Trimmomatic</i>
3.	<i>Merge Trimmomatic Stats</i>	<i>Merge Trimmomatic Stats</i>
4.	<i>Mapping BWA Mem Sambamba</i>	<i>Mapping BWA Mem Sambamba</i>
5.	<i>SAMBamba Merge BAM</i>	<i>SAMBamba Merge BAM</i>
6.	<i>SAMBamba Mark Duplicates</i>	<i>SAMBamba Mark Duplicates</i>
7.	<i>SAMBamba View Filter</i>	<i>SAMBamba View Filter</i>
8.	<i>Metrics</i>	<i>Metrics</i>
9.	<i>Homer Make Tag Directory</i>	<i>Homer Make Tag Directory</i>
10.	<i>QC Metrics</i>	<i>QC Metrics</i>
11.	<i>Homer Make UCSC file</i>	<i>Homer Make UCSC file</i>
12.	<i>MACS2 call peak</i>	<i>MACS2 ATAC-seq call peak</i>
13.	<i>Homer annotate peaks</i>	<i>Homer annotate peaks</i>
14.	<i>Homer find motifs genome</i>	<i>Homer find motifs genome</i>
15.	<i>Annotation Graphs</i>	<i>Annotation Graphs</i>
16.	<i>Run SPP</i>	<i>Run SPP</i>
17.	<i>Differential Binding</i>	<i>IHEC Metrics</i>
18.	<i>IHEC Metrics</i>	<i>MultiQC Report</i>
19.	<i>MultiQC Report</i>	<i>CRAM Output</i>
20.	<i>CRAM Output</i>	



## Step Details

Following are the various steps that are part of GenPipes ChIP-Seq genomic analysis pipeline:

### **Picard Sam to Fastq**

If FASTQ files are not already specified in the Readset file, then this step converts SAM/BAM files from the input Readset into FASTQ format. Otherwise, it does nothing.

### **Trimmomatic**

Raw reads quality trimming and removing of Illumina adapters is performed using [Trimmomatic Process](#). If an adapter FASTA file is specified in the config file (section 'trimmomatic', param 'adapter\_fasta'), it is used first. Else, 'Adapter1' and 'Adapter2' columns from the readset file are used to create an adapter FASTA file, given then to Trimmomatic. For PAIRED\_END readsets, readset adapters are reversed-complemented and swapped, to match Trimmomatic Palindrome strategy. For SINGLE\_END readsets, only Adapter1 is used and left unchanged.

If available, trimmomatic step in Hi-C analysis takes FASTQ files from the readset file as input. Otherwise, it uses the FASTQ output file generated from the previous [Picard Sam to Fastq](#) step conversion of the BAM files.

### **Merge Trimmomatic Stats**

The trim statistics per Readset file are merged at this step.

### **Mapping BWA Mem Sambamba**

This step takes as input files trimmed FASTQ files, if available. Otherwise it takes FASTQ files from the readset. If readset is not supplied then it uses FASTQ output files from the previous [Picard Sam to Fastq](#) conversion of BAM files.

### **SAMBamba Merge BAM**

BAM readset files are merged into one file per sample. Merge is done using [Sambamba](#).

If available, the aligned and sorted BAM output files from previous [Mapping BWA Mem Sambamba](#) step are used as input. Otherwise, BAM files from the readset file is used as input.

### **SAMBamba Mark Duplicates**

Mark duplicates. Aligned reads per sample are duplicates if they have the same 5' alignment positions (for both mates in the case of paired-end reads). All but the best pair (based on alignment score) will be marked as a duplicate in the BAM file. Marking duplicates is done using [SAMBamba](#).

### **SAMBamba View Filter**

Filter unique reads by mapping quality using [SAMBamba](#).

### **Metrics**

The number of raw/filtered and aligned reads per sample are computed at this stage.

### **Homer Make Tag Directory**

The Homer Tag directories, used to check for quality metrics, are computed at this step.

### **QC Metrics**

Sequencing quality metrics as tag count, tag auto-correlation, sequence bias and GC bias are generated.

### **Homer Make UCSC files**

Wiggle Track Format files are generated from the aligned reads using Homer. The resulting files can be loaded in browsers like [IGV](#) or [UCSC](#).

### MACS2 call peak

Peaks are called using the [MACS2](#) software. Different calling strategies are used for narrow and broad peaks. The mfold parameter used in the model building step is estimated from a peak enrichment diagnosis run. The estimated mfold lower bound is 10 and the estimated upper bound can vary between 15 and 100.

The default mfold parameter of MACS2 is [10,30].

### MACS2 ATAC-seq call peak

Assay for Transposon Accessible Chromatin (ATAC-seq) analysis enables measurement of chromatin structure modifications (nucleosome free regions) on gene regulation. This step involves calling peaks using the [MACS2](#) software. Different calling strategies are used for narrow and broad peaks. The mfold parameter used in the model building step is estimated from a peak enrichment diagnosis run. The estimated mfold lower bound is 10 and the estimated upper bound can vary between 15 and 100.

The default mfold parameter of MACS2 is [10,30].

### Homer annotate peaks

The peaks called previously are annotated with [HOMER tool](#) using RefSeq annotations for the reference genome. Gene ontology and genome ontology analysis are also performed at this stage.

### Homer find motifs genome

In this step, De novo and known motif analysis per design are performed using [HOMER](#).

### Annotation Graphs

This step focuses on peak location statistics. The following peak location statistics are generated per design: proportions of the genomic locations of the peaks. The locations are: Gene (exon or intron), Proximal ([0;2] kb upstream of a transcription start site), Distal ([2;10] kb upstream of a transcription start site), 5d ([10;100] kb upstream of a transcription start site), Gene desert ( $\geq 100$  kb upstream or downstream of a transcription start site), Other (anything not included in the above categories); The distribution of peaks found within exons and introns; The distribution of peak distance relative to the transcription start sites (TSS); the Location of peaks per design.

### Run SPP

This step runs spp to estimate NSC and RSC ENCODE metrics. For more information - see quality enrichment of ChIP sequence data, [phantompeakqualtools](#).

### Differential Binding

Differential Binding step is meant for processing DNA data enriched for genomic loci, including ChIP- seq data enriched for sites where specific protein binding occurs, or histone marks are enriched. It uses [DiffBind](#) package that helps in identifying sites that are differentially bound between sample groups.

GenPipes ChIPseq pipeline performs differential binding based on the provided treatments and controls as per the particular comparison specified in the design file. The differential analysis results are generated separately for each specified comparison in the [Design File](#), with correctly specified treatments (2) and controls (1) samples.

Samples with '0' (zero) are ignored during the comparison. For details regarding how to specify sample group membership in the design file, refer to [Design File Format details](#).

For comparison, at least two samples for each group must be specified. If two samples per group are not specified, the differential binding step will be skipped during the pipeline run.

#### **Warning:** Incorrect Design File Format

If the specified design file does not follow the specified [Design File Format](#) for ChIPseq pipeline, the differential binding step will be skipped during the pipeline run.

The output of differential analysis containing differentially bound peaks are saved as a TSV. In addition, for each comparison done using [DiffBind](#), an html report is also generated for QC differential analysis.

### IHEC Metrics

This step generates IHEC's [standard metrics](#).

### MultiQC Report

A quality control report for all samples is generated. For more detailed information see [MultiQC documentation](#).

### CRAM Output

Generate long term storage version of the final alignment files in CRAM format. Using this function will include the original final BAM file into the removable file list.

## More Information

For the latest implementation and usage details, see [ChIP-Seq Pipeline README](#). Here is some more information about ChIP sequencing pipeline that you may find interesting.

- [ChIP-Seq and Beyond](#).
- [ChIP-Seq Technology and Workflow](#).
- [Schematic representation of major methods to detect functional elements in DNA](#).
- [ChIP Sequencing and ATAC Sequencing](#)

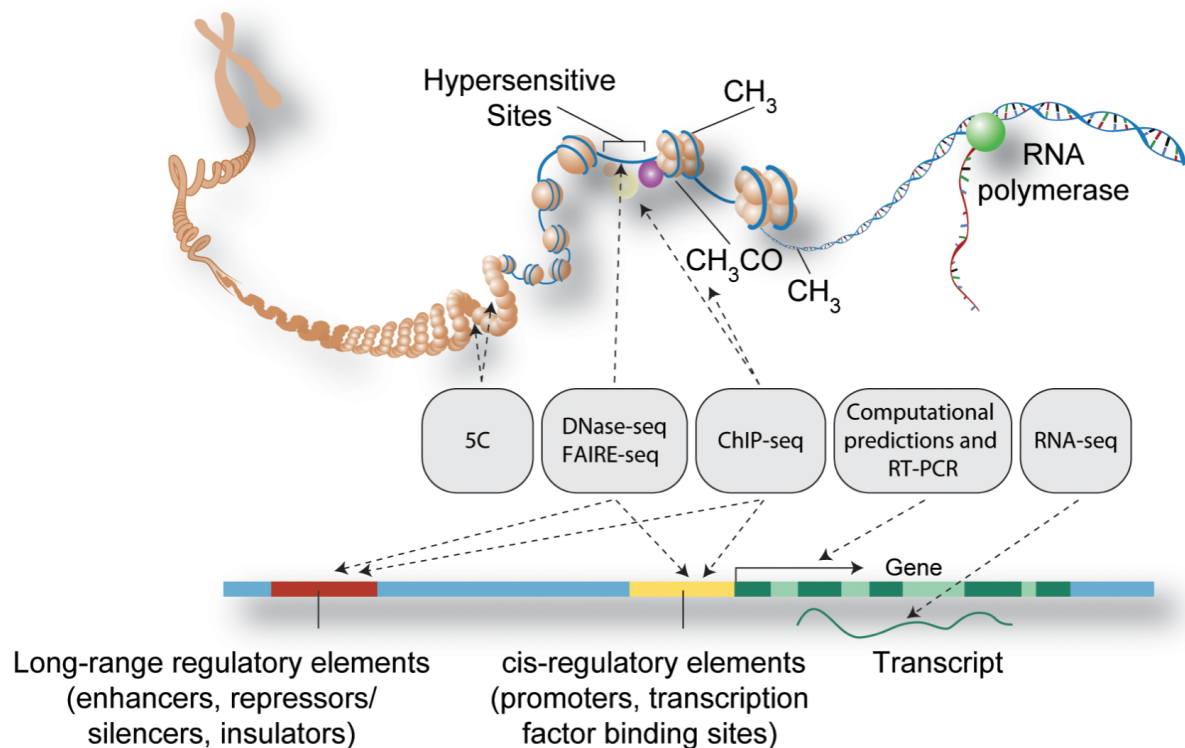


Fig. 8: Figure: Schematic representation of major methods used to detect functional elements in DNA (Source PLOS)

## CoV Sequencing Pipeline

CoV Sequencing Pipeline can be used to effectively amplify and detect SARS-CoV-2 RNA in samples such that it enables reliable results from even low copy numbers. It helps to assay clean characteristic target peaks of defined sizes, allowing for direct detection of the presence of viral genome from the Coronaviridae family. Sequencing provides confirmation for the species as well as phylogenetic information for the specific strain discrimination.

The design of this pipeline is directed against SARS-CoV-2 and other coronaviruses as well. By amplifying conserved regions of other coronaviruses in a sample, along with mutation tolerant panels, it can provide additional insights and pinpoint sequence variability, thus offering a powerful solution for more in-depth research and surveillance of the rapidly evolving virus.

CoVSeQ pipeline is designed as part of the [partnership for Québec SARS-CoV-2 sequencing](#). It is funded by the CanCOGeN initiative through Genome Canada and from the Ministère de la santé et des services sociaux du Québec. For more details, see [CoVSeQ website](#).

- *[Introduction](#)*
- *[Version](#)*
- *[Usage](#)*
- *[Example Run](#)*
- *[Pipeline Schema](#)*
- *[Pipeline Steps](#)*
- *[Step Details](#)*
- *[More information](#)*

---

## Introduction

TBD some high level introduction about Pipeline steps et all and how it is different from other available CoV Seq pipelines (if any). Refer to Ed and Hector.

---

## Version

3.6.2

For the latest implementation and usage details refer to DNA Sequencing implementation [README file](#) file.

---

## Usage

```
covseq.py [-h] [--help] [-c CONFIG [CONFIG ...]] [-s STEPS]
          [-o OUTPUT_DIR] [-j {pbs,batch,daemon,slurm}] [-f]
          [--no-json] [--report] [--clean]
          [-l {debug,info,warning,error,critical}] [--sanity-check]
          [--container {wrapper, singularity} <IMAGE PATH>]
          [--genpipes_file GENPIPES_FILE]
          [-r READSETS] [-v]
```

## Optional Arguments

`-r READSETS, --readsets READSETS`

readset file

`-h` show this help message **and** exit

`--help` show detailed description of pipeline **and** steps

`-c CONFIG [CONFIG ...], --config CONFIG [CONFIG ...]`

config INI-style **list** of files; config parameters  
are overwritten based on files order

`-s STEPS, --steps STEPS` step **range** e.g. '1-5', '3,6,7', '2,4-8'

`-o OUTPUT_DIR, --output-dir OUTPUT_DIR`

output directory (default: current)

`-j {pbs,batch,daemon,slurm}, --job-scheduler {pbs,batch,daemon,slurm}`

job scheduler **type** (default: slurm)

`-f, --force` force creation of jobs even **if** up to date (default:  
false)

`--no-json` do **not** create JSON file per analysed sample to track  
the analysis status (default: false i.e. JSON file  
will be created)

`--report` create '**pandoc**' command to merge **all** job markdown  
report files **in** the given step **range** into HTML, **if**  
they exist; **if** `--report` **is** set, `--job-scheduler`,  
`--force`, `--clean` options **and** job up-to-date status  
are ignored (default: false)

`--clean` create '**rm**' commands **for** **all** job removable files **in**  
the given step **range**, **if** they exist; **if** `--clean` **is**

(continues on next page)

(continued from previous page)

```
set, --job-scheduler, --force options and job up-to-
date status are ignored (default: false)
```

```
-l {debug,info,warning,error,critical}, --log {debug,info,warning,error,critical}
```

```
log level (default: info)
```

```
--sanity-check      run the pipeline in `sanity check mode` to verify
                    all the input files needed for the pipeline to run
                    are available on the system (default: false)
```

```
--container {wrapper, singularity} <IMAGE PATH>
```

```
run pipeline inside a container providing a container
image path or accessible singularity hub path
```

```
-v, --version      show the version information and exit
```

```
-g GENPIPES_FILE, --genpipes_file GENPIPES_FILE
```

```
Commands for running the pipeline are output to this
file pathname. The data specified to pipeline command
line is processed and pipeline run commands are
stored in GENPIPES_FILE, if this option is specified
. Otherwise, the output will be redirected to stdout
. This file can be used to actually "run the
GenPipes Pipeline".
```

## Example Run

Use the following commands to execute CoVSeq sequencing pipeline:

```
covseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/covseq/covseq.base.ini $MUGQIC_PIPELINES_
HOME/pipelines/covseq/covseq.beluga.ini -g covseqCommands_mugqic.sh
```

```
bash covseqCommands_mugqic.sh
```

**Warning:** While issuing the pipeline run command, use ``-g GENPIPES_FILE`` option (see example above) instead of using the ``> GENPIPES_FILE`` option supported by GenPipes so far, as shown below:

```
[genpipes_seq_pipeline].py -t mugqic -c $MUGQIC_PIPELINES_HOME/pipelines/[genpipes_
seq_pipeline]/[genpipes_seq_pipeline].base.ini $MUGQIC_PIPELINES_HOME/pipelines/
[genpipes_seq_pipeline]/[genpipes_seq_pipeline].guillimin.ini -r readset.[genpipes_
seq_pipeline].txt -s 1-6 > [genpipes_seq_pipeline]_commands_mugqic.sh
```

```
bash [genpipes_seq_pipeline]_commands_mugqic.sh
```

``> scriptfile`` should be considered deprecated and ``-g scriptfile`` option is recommended instead.

Please note that redirecting commands to a script `> genpipe\_script.sh` is still supported for now. **But going forward, this mechanism might be dropped in a future GenPipes release.**

You can download the test dataset for this pipeline [here](#).

**Note:** Check with Paul/Hector if this pipeline requires a test dataset and whether one is available. Then update/edit the test dataset link above accordingly.

## Pipeline Schema

Figure below shows the schema of the CovSeq sequencing protocol.

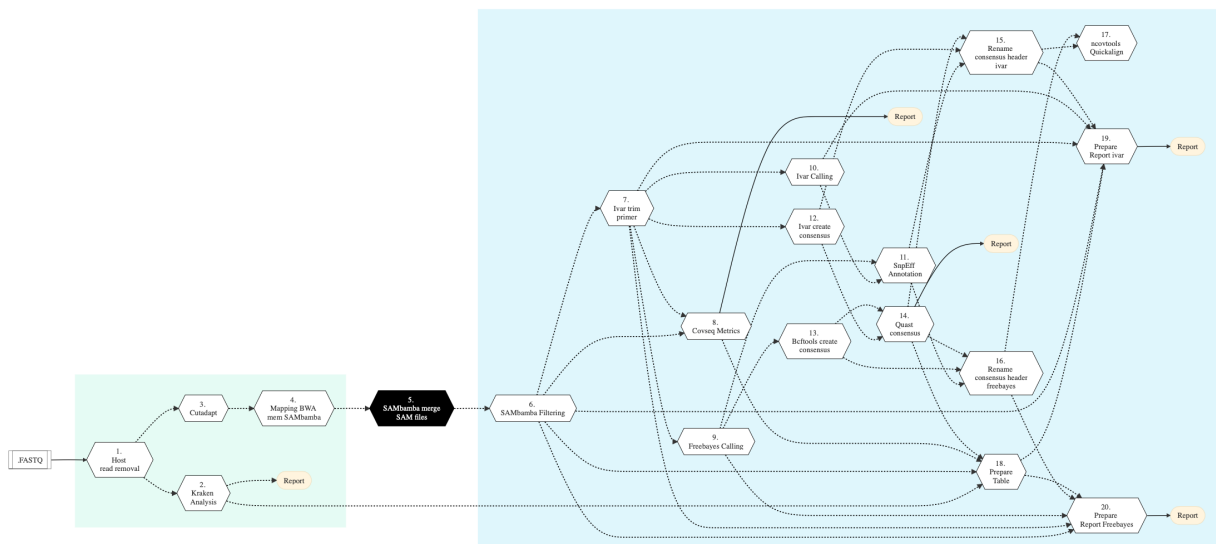
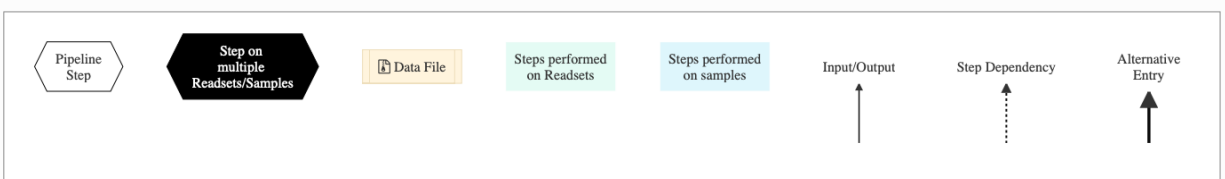


Fig. 9: Figure: Schema of CoVSeq Sequencing protocol



## Pipeline Steps

The table below shows various steps that constitute the CoVSeq genomic analysis pipeline.



	<i>SARS-CoV-2 Sequencing Steps</i>
1.	<i>Host Reads Removal</i>
2.	<i>Kraken Analysis</i>
3.	<i>Cutadapt</i>
4.	<i>Mapping BWA Mem Sambamba</i>
5.	<i>Sambamba Merge SAM Files</i>
6.	<i>Sambamba Filtering</i>
7.	<i>iVar Trim Primers</i>
8.	<i>CoVSeq Metrics</i>
9.	<i>Freebayes Calling</i>
10.	<i>iVar Calling</i>
11.	<i>SNPEff Annotate</i>
12.	<i>iVar Create Consensus</i>
13.	<i>BCFTools Create Consensus</i>
14.	<i>QUAST Consensus Metrics</i>
15.	<i>Rename Consensus Header ivar</i>
16.	<i>Rename Consensus Header freebayes</i>
17.	<i>ncovtools Quickalign</i>
18.	<i>Prepare Table</i>
19.	<i>Prepare Report ivar</i>
20.	<i>Prepare Report Freebayes</i>

## Step Details

Following are the various steps that are part of GenPipes CoVSeq genomic analysis pipeline:

### Host Reads Removal

In this step, the filtered reads are aligned to a reference genome. The alignment is done per sequencing readset using the [BWA Software](#) and [BWA-Mem Algorithm](#). BWA output BAM files are then sorted by coordinate using [Sambamba](#).

Input files for this step include:

- Trimmed FASTQ files, if available
- If no FASTQ files then FASTQ files from supplied readset is used.
- If no readset is supplied then FASTQ output from Picard SAM to FASTQ conversion of BAM files is used.

### Kraken Analysis

[Kraken](#) is an ultra fast and highly accurate mechanism for assigning taxonomic labels to metagenomic DNA sequences. It achieves high sensitivity and speed by utilizing exact alignments of k-mers and a novel classification algorithm. This step performs Kraken analysis using output of the Sambamba processing in [previous step](#).

### Cutadapt

[Cutadapt processing](#) cleans data by finding and removing adapter sequences, primers, poly-A tails and other types of unwanted sequence from high throughput sequencing reads obtained after Kraken analysis. In this step, quality trimming of raw reads and removing of adapters is performed by giving 'Adapter1' and 'Adapter2' columns from the readset file to Cutadapt. For PAIRED\_END readsets, both adapters are used. For SINGLE\_END readsets, only Adapter1 is used and left unchanged.

To trim the front of the read, use adapter\_5p\_fwd and adapter\_5p\_rev (For PAIRED\_END only) in cutadapt section of the .ini file.

This step takes as input files:

1. FASTQ files from the readset file, if available.
2. Otherwise, FASTQ output files from previous Picard SAM to FASQ conversion of BAM files is used.

### Mapping BWA Mem Sambamba

This step takes as input files trimmed FASTQ files, if available. Otherwise it takes FASTQ files from the readset. If readset is not supplied then it uses FASTQ output files from the previous Picard SAM to FASTQ conversion of BAM files.

Here, the filtered reads are aligned to a reference genome. The alignment is done per sequencing readset. [BWA Software](#) is used for alignment with [BWA-Mem Algorithm](#). BWA output BAM files are then sorted by coordinate using [Sambamba](#).

### Sambamba Merge SAM Files

This step uses [Sambamba-Merge Tool](#) to merge several BAM files into one. SAM headers are merged automatically similar to how it is done in Picard merging tool.

### Sambamba Filtering

In this step, raw BAM files are filtered using [Sambamba](#) and and `awk` command is run to filter by insert size.

### iVar Trim Primers

iVar uses primer positions supplied in a BED file to soft clip primer sequences from an aligned and sorted BAM file. Following this, the reads are trimmed based on a quality threshold(Default: 20). To do the quality trimming, iVar uses a sliding window approach(Default: 4). The window slides from the 5' end to the 3' end and if at any point the average base quality in the window falls below the threshold, the remaining read is soft clipped.

In this step, primer sequences are removed from individual BAM files using iVar.

### CoVSeq Metrics

In this step, multiple metrics are computed from sequencing:

- DNA Sample [Qualimap](#) to facilitate quality control of alignment sequencing
- [Sambamba-flagstat](#) for obtaining flag statistics from BAM file
- [BED Tools GenomeCov](#) is used for computing histograms(default), per-base reports and BEDGRAPH summaries of feature coverage of aligned sequences for a given genome.
- [Picard HS Metrics](#) are picked from SAM/BAM files. Only those metrics are collected that are specific for sequence datasets generated through hybrid-selection. Hybrid-selection (HS) is the most commonly used technique to capture exon-specific sequences for targeted sequencing experiments such as exome sequencing.

### Freebayes Calling

Freebayes is a haplotype-based variant detector designed to find small polymorphisms, specifically SNPs (single-nucleotide polymorphisms), indels (insertions and deletions), MNPs (multi-nucleotide polymorphisms), and complex events (composite insertion and substitution events) smaller than the length of a short-read sequencing alignment.

This method avoids one of the core problems with alignment-based variant detection— that identical sequences may have multiple possible alignments. See [Freebayes details here](#).

### iVar Calling

In this step, iVar is used for creating a trimmed BAM file after trimming aligned reads in the input BAM file using primer positions specified in the BED input file. Besides the [Freebayes Calling](#) tool, ivar calling is also used in covseq pipeline as part of the latest release.

### SNPEff Annotate

This step uses [SNPEff](#) to annotate the data.

### iVar Create Consensus

In this step, iVar is used to create consensus. Also, it removes primer sequences to individual BAM files using [fgbio](#).

### BCFTools Create Consensus

BCFtools consensus is created in this step. BCFtools is a set of utilities that manipulate variant calls in the Variant Call Format (VCF) and its binary counterpart BCF

### QUAST Consensus Metrics

In this step, QUAST is used to compare and evaluate assemblies to rule out misassemblies.

### Rename Consensus Header ivar

[Consensus sequence](#) is the calculated order of most frequent residues found at each position in a sequence alignment. This information is important when considering sequence-dependent enzymes such as RNA Polymerase which is important for SAR-CoV-2 studies. In this step, [header sequence](#) can be modified in various ways as specified in rename type parameter: Multipart header, Replace word, Replace interval, and Add prefix/suffix.

### Rename Consensus Header freebayes

Two variant calling tools are used in covseq pipeline - ivar and freebayes. In this step, the consensus header rename for freebayes is done.

### ncovtools Quickalign

Uses quickalign to provides summary statistics, which can be used to determine the sequencing quality and evolutionary novelty of input genomes (e.g. number of new mutations and indels).

It uses ivar consensus as well as freebayes consensus to arrive at the alignment decisions.

### Prepare Table

Gathers all analysis data for quast, kraken and other metrics and module details.

### Prepare Report ivar

Prepare ivar analysis report.

### Prepare Report Freebayes

Prepare [Freebayes](#) analysis report.

---

## More information

For the latest implementation and usage details refer to CoVSeq Pipeline implementation [README.md](#).

- [CoVSeq Cost Effective Workflow](#)
- [CoVSeq Genome Analysis and Visualization](#)

## DNA Sequencing Pipeline

DNA sequencing is the process of determining the sequence of nucleotides in a section of DNA. Next-generation sequencing (NGS), also known as high-throughput sequencing, allows for sequencing of DNA and RNA much more quickly and cheaply than the previously used Sanger sequencing, and as such revolutionized the study of genomics and molecular biology. The enormous amount of short reads generated by the new DNA sequencing technologies call for the development of fast and accurate read alignment programs.

Over the past decade, [long-read](#), single-molecule DNA sequencing technologies have emerged as powerful players in genomics. With the ability to generate reads tens to thousands of kilobases in length with an accuracy approaching that of short-read sequencing technologies, these platforms have proven their ability to resolve some of the most challenging regions of the human genome, detect previously inaccessible structural variants and generate some of the first telomere-to-telomere assemblies of whole chromosomes.

Burrows-Wheeler Alignment tool, [BWA](#), is a new read alignment package that is based on backward search with Burrows-Wheeler Transform (BWT), to efficiently align short sequencing reads against a large reference sequence such as the human genome, allowing mismatches and gaps. BWA is 10–20 times [faster than](#) Mapping and Assembly with Quality, [MAQ](#), while achieving similar accuracy. In addition, BWA outputs alignment in the new standard SAM (Sequence Alignment/Map) format. Variant calling and other downstream analyses after the alignment can be achieved with the open source [SAMtools software package](#).

DNA Sequencing GenPipes pipeline has been implemented by optimizing the Genome Analysis Toolkit, [GATK](#) best practices standard operating protocols. This procedure entails trimming raw reads derived from whole-genome or exome data followed by alignment to a known reference, post-alignment refinements, and variant calling. Trimmed reads are aligned to a reference by the Burrows-Wheeler Aligner (BWA), [bwa-mem](#). Refinements of mismatches near insertions and deletions (indels) and base qualities are performed using GATK indels realignment and base recalibration to improve read quality after alignment. Processed reads are marked as fragment duplicates using Picard MarkDuplicates and single-nucleotide polymorphisms and small indels are identified using either GATK haplotype callers or [SAMtools mpileup](#).

The [Genome in a Bottle dataset](#) was used to select steps and parameters minimizing the false-positive rate and maximizing the true-positive variants to achieve a sensitivity of 99.7%, precision of 99.1%, and F1 score of 99.4%. Finally, additional annotations are incorporated using [dbNSFP](#) and / or Gemini and QC metrics are collected at various stages and visualized using [MultiQC](#).

Latest release 3.2.0 of GenPipes supports a new DNA sequencing protocol option called 'sv' besides the [mugqic](#), [mpileup](#) and [light](#) options. This is useful for structural variation detection.

**Structural Variation (SVs)** are the genetic variations in the structure of chromosome with different types of rearrangements. They comprise millions of nucleotides of heterogeneity within every genome, and are likely to make an important contribution to genetic diversity and disease susceptibility. In the genomics community, substantial efforts have been devoted to improving understanding of the roles of SVs in genome functions relating to diseases and researchers are working actively to develop effective algorithms to reliably identify various types of SVs such as deletions, insertions, duplications and inversions. GenPipes supports SV detection option in the DNA Sequencing Pipeline.

- [Introduction](#)
- [Version](#)
- [Usage](#)
- [Example Run](#)
- [Pipeline Schema](#)
- [Pipeline Steps](#)
- [Step Details](#)
- [More information](#)

## Introduction

The standard MUGQIC DNA-Seq pipeline uses BWA to align reads to the reference genome. Treatment and filtering of mapped reads approaches as INDEL realignment, mark duplicate reads, recalibration and sort are executed using Picard and GATK. Samtools MPILEUP and bcftools are used to produce the standard SNP and indels variants file (VCF). Additional SVN annotations mostly applicable to human samples include mappability flags, dbSNP annotation and extra information about SVN by using published databases. The SnpEff tool is used to annotate variants using an integrated database of functional predictions from multiple algorithms (SIFT, Polyphen2, LRT and MutationTaster, PhyloP and GERP++, etc.) and to calculate the effects they produce on known genes. Refer to the list of [SnpEff effects](#).

A summary html report is automatically generated by the pipeline. This report contains description of the sequencing experiment as well as a detailed presentation of the pipeline steps and results. Various Quality Control (QC) summary statistics are included in the report and additional QC analysis is accessible for download directly through the report. The report includes also the main references of the software and methods used during the analysis, together with the full list of parameters that have been passed to the pipeline main script.

The DNA sequencing pipeline supports two trimmers: [Trimmomatic](#) and [Skewer](#).

GenPipes DNA sequencing pipeline offers four different protocol options:

1. The default protocol based on the GATK variant caller, haplotype caller, (-t mugqic)
2. Another based on the mpileup/bcftools caller (-t mpileup).
3. Light option (-t light)
4. Structural Variation Detection, sv option (-t sv)

See [More information](#) section below for details.

---

## Version

3.6.2

For the latest implementation and usage details refer to DNA Sequencing implementation [README file](#) file.

---

## Usage

```
dnaseq.py [-h] [--help] [-c CONFIG [CONFIG ...]] [-s STEPS]
          [-o OUTPUT_DIR] [-j {pbs,batch,daemon,slurm}] [-f]
          [--no-json] [--report] [--clean]
          [-l {debug,info,warning,error,critical}] [--sanity-check]
          [--container {wrapper, singularity} <IMAGE PATH>]
          [--genpipes_file GENPIPES_FILE]
          [-t {mugqic,mpileup,light,sv}] [-r READSETS] [-v]
```

### Optional Arguments

```
-t {mugqic,mpileup,light, sv}, --type {mugqic,mpileup,light,sv}
```

DNAseq analysis **type**

```
-r READSETS, --readsets READSETS
```

readset file

```
-h
```

show this help message **and** exit

```
--help
```

show detailed description of pipeline **and** steps

```
-c CONFIG [CONFIG ...], --config CONFIG [CONFIG ...]
```

config INI-style **list** of files; config parameters  
are overwritten based on files order

```
-s STEPS, --steps STEPS
```

step **range** e.g. '1-5', '3,6,7', '2,4-8'

```
-o OUTPUT_DIR, --output-dir OUTPUT_DIR
```

output directory (default: current)

```
-j {pbs,batch,daemon,slurm}, --job-scheduler {pbs,batch,daemon,slurm}
```

job scheduler **type** (default: slurm)

<code>-f, --force</code>	force creation of jobs even <b>if</b> up to date (default: false)
<code>--no-json</code>	do <b>not</b> create JSON file per analysed sample to track the analysis status (default: false i.e. JSON file will be created)
<code>--report</code>	create ' <b>pandoc</b> ' command to merge <b>all</b> job markdown report files <b>in</b> the given step <b>range</b> into HTML, <b>if</b> they exist; <b>if</b> <code>--report</code> <b>is</b> set, <code>--job-scheduler</code> , <code>--force</code> , <code>--clean</code> options <b>and</b> job up-to-date status are ignored (default: false)
<code>--clean</code>	create ' <b>rm</b> ' commands <b>for</b> <b>all</b> job removable files <b>in</b> the given step <b>range</b> , <b>if</b> they exist; <b>if</b> <code>--clean</code> <b>is</b> set, <code>--job-scheduler</code> , <code>--force</code> options <b>and</b> job up-to-date status are ignored (default: false)
<code>-l {debug,info,warning,error,critical}, --log {debug,info,warning,error,critical}</code>	log level (default: info)
<code>--sanity-check</code>	run the pipeline in 'sanity check mode' to verify all the input files needed for the pipeline to run are available on the system (default: false)
<code>--container {wrapper, singularity} &lt;IMAGE PATH&gt;</code>	run pipeline inside a container providing a container image path <b>or</b> accessible singularity hub path
<code>-v, --version</code>	show the version information <b>and</b> exit
<code>-g GENPIPES_FILE, --genpipes_file GENPIPES_FILE</code>	Commands <b>for</b> running the pipeline are output to this file pathname. The data specified to pipeline command line <b>is</b> processed <b>and</b> pipeline run commands are stored <b>in</b> GENPIPES_FILE, <b>if</b> this option <b>is</b> specified . Otherwise, the output will be redirected to stdout . This file can be used to actually " <b>run the GenPipes Pipeline</b> ".

## Example Run

Use the following commands to execute MUGQIC DNA sequencing pipeline:

```
dnaseq.py -t mugqic -c $MUGQIC_PIPELINES_HOME/pipelines/dnaseq/dnaseq.base.ini $MUGQIC_
↳ PIPELINES_HOME/pipelines/dnaseq/dnaseq.guillimin.ini -r readset.dnaseq.txt -s 1-29 -g_
↳ dnaseqCommands_mugqic.sh

bash dnaseqCommands_mugqic.sh
```

Use the following commands to execute the Mpileup DNA sequencing pipeline:

```
dnaseq.py -t mpileup -c $MUGQIC_PIPELINES_HOME/pipelines/dnaseq/dnaseq.base.ini $MUGQIC_
↳ PIPELINES_HOME/pipelines/dnaseq/dnaseq.guillimin.ini -r readset.dnaseq.txt -s 1-33 -g_
↳ dnaseqCommands_mpileup.sh

bash dnaseqCommands_mpileup.sh
```

**Warning:** While issuing the pipeline run command, use ``-g GENPIPES_FILE`` option (see example above) instead of using the ``> GENPIPES_FILE`` option supported by GenPipes so far, as shown below:

```
[genpipes_seq_pipeline].py -t mugqic -c $MUGQIC_PIPELINES_HOME/pipelines/[genpipes_
↳ seq_pipeline]/[genpipes_seq_pipeline].base.ini $MUGQIC_PIPELINES_HOME/pipelines/
↳ [genpipes_seq_pipeline]/[genpipes_seq_pipeline].guillimin.ini -r readset.[genpipes_
↳ seq_pipeline].txt -s 1-6 > [genpipes_seq_pipeline]_commands_mugqic.sh

bash [genpipes_seq_pipeline]_commands_mugqic.sh
```

``> scriptfile`` should be considered deprecated and ``-g scriptfile`` option is recommended instead. Please note that redirecting commands to a script ``> genpipe_script.sh`` is still supported for now. **But going forward, this mechanism might be dropped in a future GenPipes release.**

You can download the test dataset for this pipeline [here](#).

---

## Pipeline Schema

Figure below shows the schema of the DNA sequencing protocol - MUGQIC type.

The following figure shows the pipeline schema for **Mpileup** type of DNA sequencing protocol.

The following figure shows the pipeline schema for 'Light' type of DNA sequencing protocol.

The following figure shows the pipeline schema for **Structural Variation** type of DNA sequencing protocol.

Refer to the Pipeline steps below, for details.

---



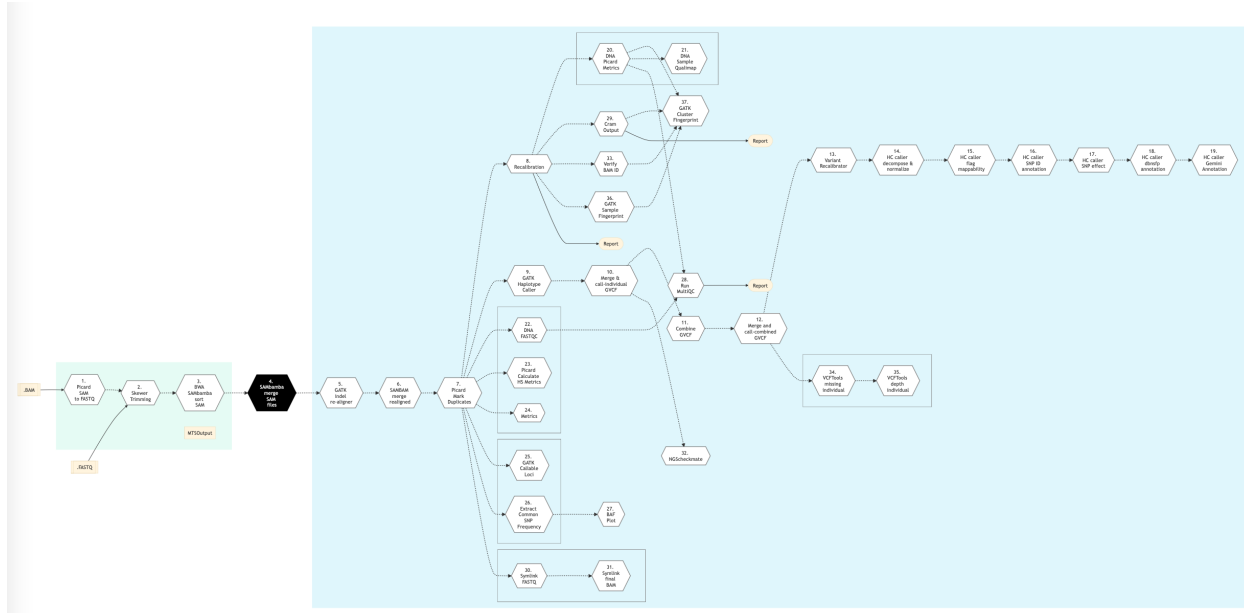


Fig. 10: Figure: Schema of MUGQIC DNA Sequencing protocol

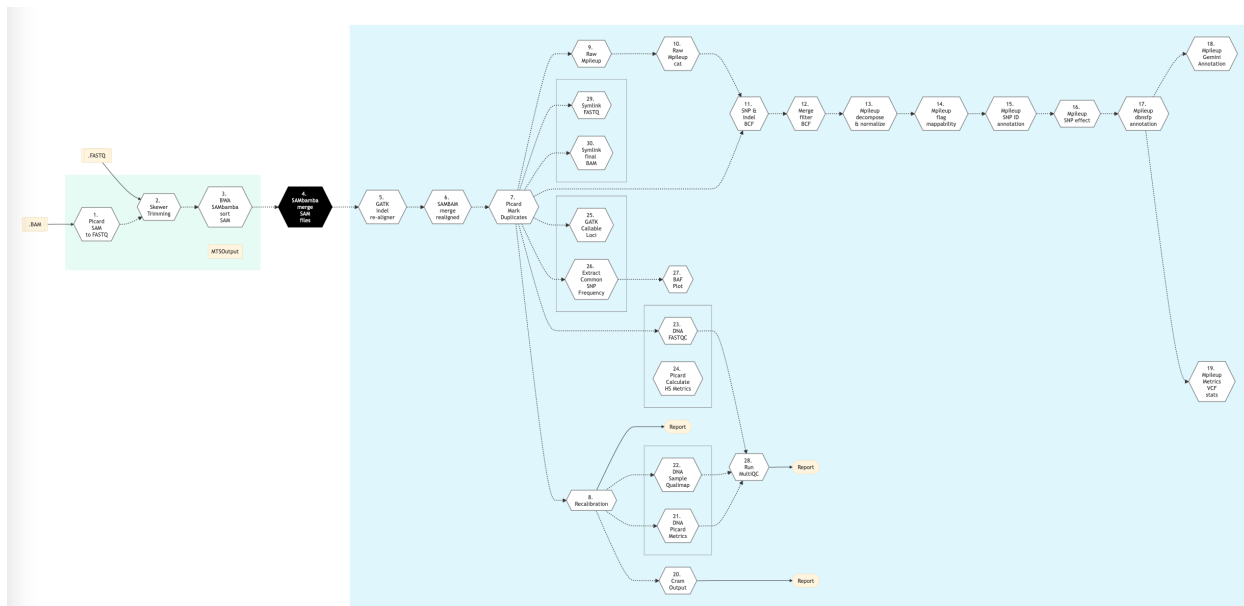
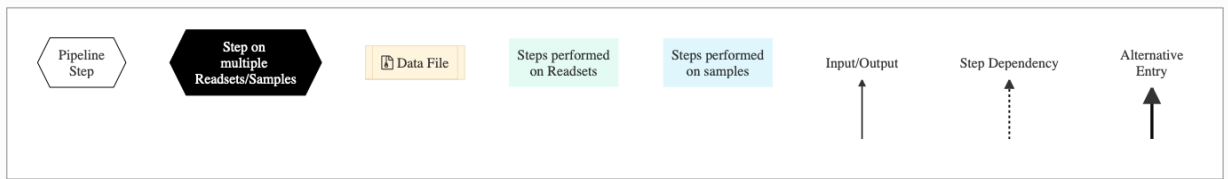


Fig. 11: Figure: Schema of Mpileup DNA Sequencing protocol

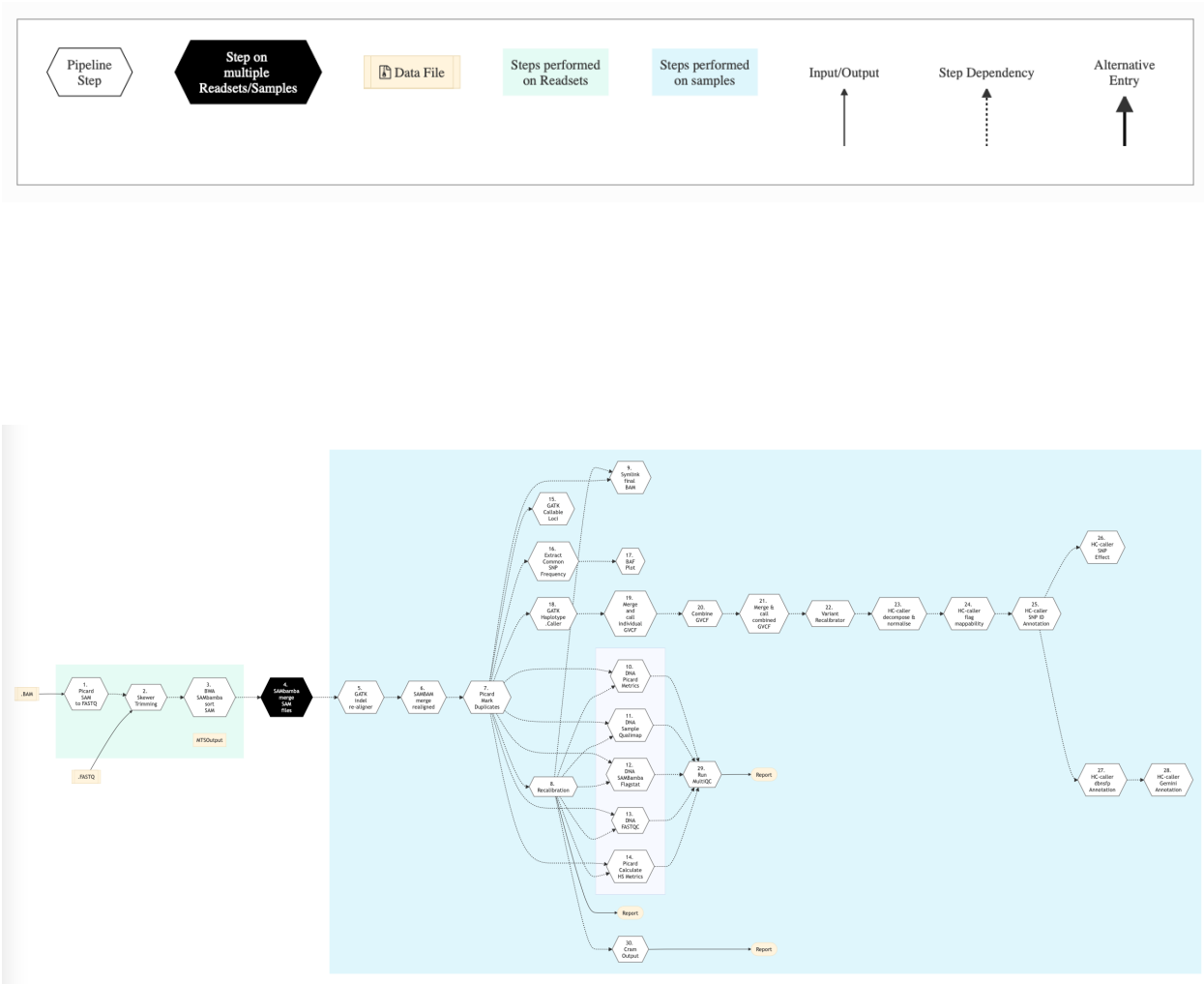
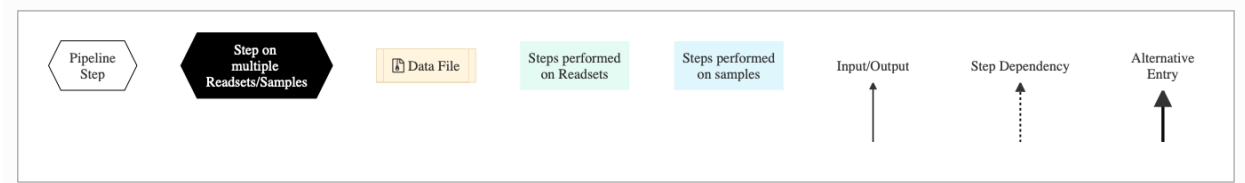


Fig. 12: Figure: Schema of Light DNA Sequencing protocol



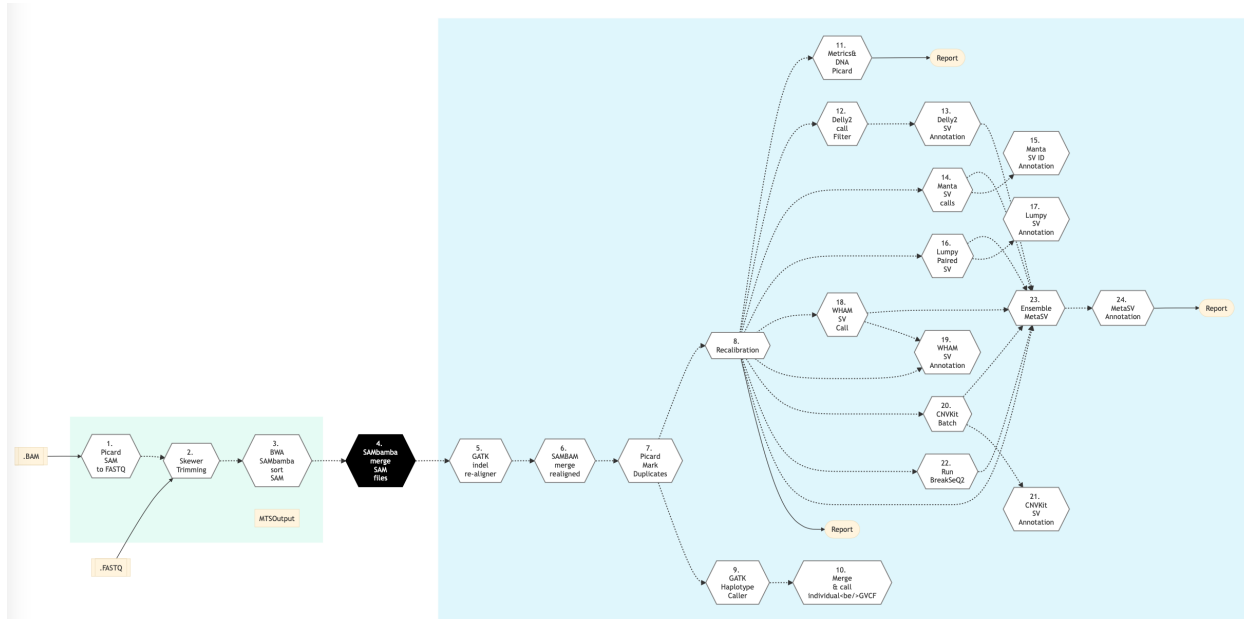
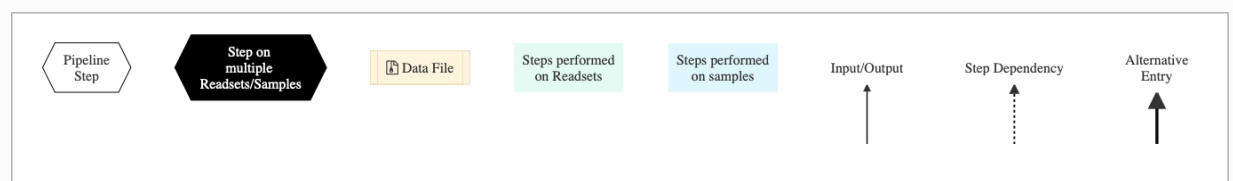


Fig. 13: Figure: Schema of Structural Variations (SV) DNA Sequencing protocol



## Pipeline Steps

The table below shows various steps that constitute the MUGQIC and Mpileup types of DNA Sequencing for genomic analysis pipelines.

	<i>MUGQIC DNA Sequencing Steps</i>	<i>MPileup DNA Sequencing Steps</i>	<i>Light DNA Sequencing Steps</i>	<i>Structural Variations Steps</i>
1.	<i>Picard SAM to FastQ</i>	<i>Picard SAM to FastQ</i>	<i>Picard SAM to FastQ</i>	<i>Picard SAM to FastQ</i>
2.	<i>Skewer Trimming</i>	<i>Skewer Trimming</i>	<i>Skewer Trimming</i>	<i>Skewer Trimming</i>
3.	<i>BWA SAMbamba Sort SAM</i>	<i>BWA SAMbamba Sort SAM</i>	<i>BWA SAMbamba Sort SAM</i>	<i>BWA SAMbamba Sort SAM</i>
4.	<i>SAMBAM Merge SAM Files</i>	<i>SAMBAM Merge SAM Files</i>	<i>SAMBAM Merge SAM Files</i>	<i>SAMBAM Merge SAM Files</i>
5.	<i>GATK Indel Re-aligner</i>	<i>GATK Indel Re-aligner</i>	<i>GATK Indel Re-aligner</i>	<i>GATK Indel Re-aligner</i>
6.	<i>SAMBAM Merge Realigned</i>	<i>SAMBAM Merge Realigned</i>	<i>SAMBAM Merge Realigned</i>	<i>SAMBAM Merge Realigned</i>
7.	<i>Picard Mark Duplicates</i>	<i>Picard Mark Duplicates</i>	<i>Picard Mark Duplicates</i>	<i>Picard Mark Duplicates</i>
8.	<i>Recalibration</i>	<i>Recalibration</i>	<i>Recalibration</i>	<i>Recalibration</i>
9.	<i>GATK Haplotype Caller</i>	<i>Raw MPileup</i>	<i>Sym Link Final BAM</i>	<i>GATK Haplotype Caller</i>
10.	<i>Merge and call individual GVCF</i>	<i>Compress Raw MPileup</i>	<i>Metrics DNA Picard</i>	<i>Merge and call individual GVCF</i>
11.	<i>Combine GVCF</i>	<i>SNP and indel BCF</i>	<i>Metrics DNA Sample Quality Map</i>	<i>Metrics DNA Picard</i>
12.	<i>Merge and call combined GVCF</i>	<i>Merge Filter BCF</i>	<i>Metrics DNA SAM-BAM Flag Stats</i>	<i>Delly2 Call Filter</i>
13.	<i>Variant Recalibrator</i>	<i>MPileup decompose and normalize</i>	<i>Metrics DNA FastQC</i>	<i>Delly2 SV Annotation</i>
14.	<i>Haplotype caller decompose and normalize</i>	<i>MPileup Flag Mappability</i>	<i>Picard Calculate HS Metrics</i>	<i>Manta SV Calls</i>
15.	<i>Haplotype caller flag mappability</i>	<i>Mpileup SNP ID annotation</i>	<i>GATK Callable Loci</i>	<i>Manta SV Annotation</i>

continues on next page

Table 2 – continued from previous page

	<i>MUGQIC DNA Sequencing Steps</i>	<i>MPileup DNA Sequencing Steps</i>	<i>Light DNA Sequencing Steps</i>	<i>Structural Variations Steps</i>
16.	<i>Haplotype caller SNP ID annotation</i>	<i>MPileup SNP Effect</i>	<i>Extract Common SNP Frequencies</i>	<i>Lumpy Paired SV</i>
17.	<i>Haplotype caller SNP Effect</i>	<i>MPileup dbNSFP annotation</i>	<i>BAF Plot</i>	<i>Lumpy SV Annotation</i>
18.	<i>Haplotype caller dbNSFP annotation</i>	<i>MPileup Gemini annotation</i>	<i>GATK Haplotype Caller</i>	<i>Wham SV Call</i>
19.	<i>Haplotype caller Gemini annotation</i>	<i>MPileup Metrics VCF stats</i>	<i>Merge and call individual GVCF</i>	<i>Wham SV Annotation</i>
20.	<i>Metrics DNA Picard</i>	<i>CRAM Output</i>	<i>Combine GVCF</i>	<i>CNVkit Batch</i>
21.	<i>Metrics DNA Sample Quality Map</i>	<i>Metrics DNA Picard</i>	<i>Merge and call combined GVCF</i>	<i>CNVkit SV Annotation</i>
22.	<i>Metrics DNA FastQC</i>	<i>Metrics DNA Sample Quality Map</i>	<i>Variant Recalibrator</i>	<i>Run BreakSeq2</i>
23.	<i>Picard Calculate HS Metrics</i>	<i>Metrics DNA FastQC</i>	<i>Haplotype caller decompose and normalize</i>	<i>Ensemble MetaSV</i>
24.	<i>Metrics</i>	<i>Picard Calculate HS Metrics</i>	<i>Haplotype caller flag mappability</i>	<i>MetaSV Annotation</i>
25.	<i>GATK Callable Loci</i>	<i>GATK Callable Loci</i>	<i>Haplotype caller SNP ID annotation</i>	
26.	<i>Extract Common SNP Frequencies</i>	<i>Extract Common SNP Frequencies</i>	<i>Haplotype caller SNP Effect</i>	
27.	<i>BAF Plot</i>	<i>BAF Plot</i>	<i>Haplotype caller dbNSFP annotation</i>	
28.	<i>Run MultiQC</i>	<i>Run MultiQC</i>	<i>Haplotype caller Gemini annotation</i>	
29.	<i>CRAM Output</i>	<i>Sym Link FastQ</i>	<i>Run MultiQC</i>	
30.	<i>Sym Link FastQ</i>	<i>Sym Link Final BAM</i>	<i>CRAM Output</i>	
31.	<i>Sym Link Final BAM</i>			

continues on next page

Table 2 – continued from previous page

	<i>MUGQIC DNA Sequencing Steps</i>	<i>MPileup DNA Sequencing Steps</i>	<i>Light DNA Sequencing Steps</i>	<i>Structural Variations Steps</i>
32.	<i>Metrics NGSCheckmate</i>			
33.	<i>Metrics Verify BAM ID</i>			
34.	<i>Metrics VCFTools Missing Individual</i>			
35.	<i>Metrics VCFTools Depth Individual</i>			
36.	<i>Metrics GATK Sample Fingerprint</i>			
37.	<i>Metrics GATK Cluster Fingerprint</i>			

## Step Details

Following are the various steps that are part of GenPipes DNA Sequencing genomic analysis pipeline:

### Picard SAM to FastQ

Convert SAM/BAM files from the input readset file into FASTQ format if FASTQ files are not already specified in the readset file. Do nothing otherwise.

### Sym Link FastQ

TBD-GenPipes-Dev

### Trimmomatic

If available, this step takes FastQ file from the readset as input. Otherwise, FastQ output files from the previous *Picard SAM to FastQ* step, where BAM files are converted to FastQ format, are utilized in this step.

Raw reads quality trimming and removing of Illumina adapters is performed using *Trimmomatic*. If an adapter FASTA file is specified in the config file (section ‘trimmomatic’, param ‘adapter\_fasta’), it is used first. Else, ‘Adapter1’ and ‘Adapter2’ columns from the readset file are used to create an adapter FASTA file, given then to Trimmomatic. For PAIRED\_END readsets, readset adapters are reversed-complemented and swapped, to match Trimmomatic Palindrome strategy. For SINGLE\_END readsets, only Adapter1 is used and left unchanged.

### Merge Trimmomatic Stats

The trim statistics per readset file are merged in this step.

### GATK Indel Re-aligner

Insertion and deletion realignment is performed on regions where multiple base mismatches are preferred over indels by the aligner since it can appear to be less costly by the algorithm. Such regions will introduce false positive variant calls which may be filtered out by realigning those regions properly. Realignment is done using *GATK Software*. The reference genome is divided by a number regions given by the nb\_jobs parameter.

**Fix Mate by Coordinate**

This step fixes the read mates. Once local regions are realigned, the read mate coordinates of the aligned reads need to be recalculated since the reads are realigned at positions that differ from their original alignment. Fixing the read mate positions is done using `BVATools`_``.

**Picard Mark Duplicates**

This step marks duplicates. Aligned reads per sample are duplicates if they have the same 5' alignment positions (for both mates in the case of paired-end reads). All but the best pair (based on alignment score) will be marked as a duplicate in the BAM file. Marking duplicates is done using [Picard](#).

**Recalibration**

This step is used to recalibrate base quality scores of sequencing-by-synthesis reads in an aligned BAM file. After recalibration, the quality scores in the QUAL field in each read in the output BAM are more accurate in that the reported quality score is closer to its actual probability of mismatching the reference genome. Moreover, the recalibration tool attempts to correct for variation in quality with machine cycle and sequence context, and by doing so, provides not only more accurate quality scores but also more widely dispersed ones.

**Sym Link Final BAM**

TBD-GenPipes-Dev

**Metrics DNA Picard**

TBD-GenPipes-Dev

**Metrics DNA Sample Quality Map**

TBD-GenPipes-Dev

**Metrics DNA SAMBAM Flag Stats**

TBD-GenPipes-Dev

**Metrics DNA FastQC**

TBD-GenPipes-Dev

**Picard Calculate HS Metrics**

Compute on target percent of hybridization based capture happens in this step.

**GATK Callable Loci**

Computes the callable region or the genome as a bed track.

**Extract Common SNP Frequencies**

Extracts allele frequencies of possible variants across the genome.

**BAF Plot**

Plots DepthRatio and B allele frequency of previously extracted alleles.

**GATK Haplotype Caller**

GATK Haplotype Caller is used for SNPs and small indels.

**Merge and call individual GVCF**

Merges the gvcfs of haplotype caller and also generates a per sample vcf containing genotypes.

**Combine GVCF**

Combine the per sample gvcfs of haplotype caller into one main file for all sample.

**Merge and call combined GVCF**

Merges the combined gvcfs and also generates a general vcf containing genotypes.

### **Variant Recalibrator**

This step involves GATK variant recalibrator. The purpose of the variant recalibrator is to assign a well-calibrated probability to each variant call in a call set. You can then create highly accurate call sets by filtering based on this single estimate for the accuracy of each call. The approach taken by variant quality score recalibration is to develop a continuous, covarying estimate of the relationship between SNP call annotations (QD, MQ, HaplotypeScore, and ReadPosRankSum, for example) and the probability that a SNP is a true genetic variant versus a sequencing or data processing artifact. This model is determined adaptively based on “true sites” provided as input, typically HapMap 3 sites and those sites found to be polymorphic on the Omni 2.5M SNP chip array. This adaptive error model can then be applied to both known and novel variation discovered in the call set of interest to evaluate the probability that each call is real. The score that gets added to the INFO field of each variant is called the VQSLOD. It is the log odds ratio of being a true variant versus being false under the trained Gaussian mixture model. Using the tranche file generated by the previous step the ApplyRecalibration walker looks at each variant’s VQSLOD value and decides which tranche it falls in. Variants in tranches that fall below the specified truth sensitivity filter level have their filter field annotated with its tranche level. This will result in a call set that simultaneously is filtered to the desired level but also has the information necessary to pull out more variants for a higher sensitivity but a slightly lower quality level.

### **Haplotype caller decompose and normalize**

TBD-GenPipes-Dev

### **Haplotype caller flag mappability**

Mappability annotation applied to haplotype caller vcf. An in-house database identifies regions in which reads are confidently mapped to the reference genome.

### **Haplotype caller SNP ID annotation**

dbSNP annotation applied to haplotype caller vcf. The .vcf files are annotated for dbSNP using the software SnpSift (from the [SNPEff Suite](#)).

### **Haplotype caller SNP Effect**

Variant effect annotation applied to haplotype caller vcf. The .vcf files are annotated for variant effects using the [SNPEff Suite](#) software. SnpEff annotates and predicts the effects of variants on genes (such as amino acid changes).

### **Haplotype caller dbNSFP annotation**

Additional SVN annotations applied to haplotype caller vcf. Provides extra information about SVN by using numerous published databases. Applicable to human samples. Databases available include Biomart (adds GO annotations based on gene information) and dbNSFP (an integrated database of functional annotations from multiple sources for the comprehensive collection of human non-synonymous SNPs. It compiles prediction scores from four prediction algorithms (SIFT, Polyphen2, LRT and MutationTaster), three conservation scores (PhyloP, GERP++ and SiPhy) and other function annotations).

### **Haplotype caller Gemini annotation**

TBD-GenPipes-Dev

### **Haplotype caller metrics VCF stats**

Metrics SNV applied to haplotype caller vcf. Multiple metrics associated to annotations and effect prediction are generated at this step: change rate by chromosome, changes by type, effects by impact, effects by functional class, counts by effect, counts by genomic region, SNV quality, coverage, InDel lengths, base changes, transition-transversion rates, summary of allele frequencies, codon changes, amino acid changes, changes per chromosome, change rates.

### **Run MultiQC**

TBD-GenPipes-Dev

### **Raw MPileup**



Full pileup (optional). A raw mpileup file is created using samtools mpileup and compressed in gz format. One packaged mpileup file is created per sample/chromosome.

### **Compress Raw MPileup**

Merge mpileup files per sample/chromosome into one compressed gzip file per sample.

### **SNP and indel BCF**

Mpileup and Variant calling. Variants (SNPs and INDELs) are called using [SAMTools software package](#) mpileup. bcftools view is used to produce binary bcf files.

### **Merge Filter BCF**

bcftools is used to merge the raw binary variants files created in the snpAndIndelBCF step. The output of bcftools is fed to varfilter, which does an additional filtering of the variants and transforms the output into the VCF (.vcf) format. One vcf file contain the SNP/INDEL calls for all samples in the experiment.

### **MPileup decompose and normalize**

TBD-GenPipes-Dev

### **MPileup Flag Mappability**

Mappability annotation applied to mpileup vcf. An in-house database identifies regions in which reads are confidently mapped to the reference genome.

### **Mpileup SNP ID annotation**

dbSNP annotation applied to [MPileup](#) vcf. The .vcf files are annotated for dbSNP using the software SnpSift (from the [SNPEff Suite](#)).

### **MPileup SNP Effect**

Variant effect annotation applied to mpileup vcf. The .vcf files are annotated for variant effects using the SnpEff software. SnpEff annotates and predicts the effects of variants on genes (such as amino acid changes).

### **MPileup dbNSFP annotation**

Additional SVN annotations applied to mpileup vcf. Provides extra information about SVN by using numerous published databases. Applicable to human samples. Databases available include Biomart (adds GO annotations based on gene information) and dbNSFP (an integrated database of functional annotations from multiple sources for the comprehensive collection of human non-synonymous SNPs. It compiles prediction scores from four prediction algorithms (SIFT, Polyphen2, LRT and MutationTaster), three conservation scores (PhyloP, GERP++ and SiPhy) and other function annotations).

### **MPileup Gemini annotation**

TBD-GenPipes-Dev

### **MPileup Metrics VCF stats**

Metrics SNV applied to mpileup caller vcf. Multiple metrics associated to annotations and effect prediction are generated at this step: change rate by chromosome, changes by type, effects by impact, effects by functional class, counts by effect, counts by genomic region, SNV quality, coverage, InDel lengths, base changes, transition-transversion rates, summary of allele frequencies, codon changes, amino acid changes, changes per chromosome, change rates.

### **SAMBAM Mark Duplicates**

In this step duplicates are marked. Aligned reads per sample are duplicates if they have the same 5' alignment positions (for both mates in case of paired-end reads). All but the best pair (based on alignment score) will be marked as duplicates in the BAM file. Marking duplicates is done using [Picard](#).

### **Delly2 Call Filter**

This step uses normal and tumor final BAMs as input and generates a binary variant call format (BCF) file as output. It utilizes [Delly2](#), an integrated structural variant (SV) prediction method that can discover genotype and visualize deletions, tandem duplications, inversions and translocations at single-nucleotide resolution in short-read massively parallel sequencing data. It uses paired-ends, split-reads and read-depth to sensitively and accurately delineate genomic rearrangements throughout the genome. Structural variants can be annotated using [Delly-sansa](#) and visualized using [Delly-maze](#) or [Delly-suave](#).

### **Delly2 SV Annotation**

SV Annotation step utilizes the BCF file generated in previous [Delly2 Call Filter](#) step and performs genome annotation at various levels. At the nucleotide level it tries to identify the physical location of the SV dna sequences. Next, at the protein level the annotation process tries to determine the possible functions of the SV genes. Lastly, at the process-level annotation, it tries to identify the pathways and processes in which different SV genes interact, assembling an efficient functional annotation. For more details on annotation see [Genome Annotations](#).

### **Manta SV Calls**

[Manta](#) calls structural variants (SVs) and indels from mapped paired-end sequencing reads. It is optimized for analysis of germline variation in small sets of individuals and somatic variation in tumor/normal sample pairs. Manta discovers, assembles and scores large-scale SVs, medium-sized indels and large insertions within a single efficient workflow.

Manta accepts input read mappings from BAM or CRAM files and reports all SV and indel inferences in VCF 4.1 format

**\*\* Manta SV Annotation\*\***

This step uses the VCF file generated in previous step and performs SV annotations to compares types and breakpoints for candidate SVs from different callsets and enables fast comparison of SVs to genomic features such as genes and repetitive regions, as well as to previously established SV datasets.

### **Lumpy Paired SV**

This step uses [Lumpy](#) for structural variant discovery in the given input file. The output is available in BAM format.

Comprehensive discovery of structural variation (SV) from whole genome sequencing data requires multiple detection signals including read-pair, split-read, read-depth and prior knowledge. Owing to technical challenges, extant SV discovery algorithms either use one signal in isolation, or at best use two sequentially. Lumpy is a novel SV discovery framework that naturally integrates multiple SV signals jointly across multiple samples. It yields improved sensitivity, especially when SV signal is reduced owing to either low coverage data or low intra-sample variant allele frequency.

### **Lumpy SV Annotation**

This step performs Lumpy SV Annotation for mapping and characterization of SVs.

### **Wham SV Call**

[Wham](#) (Whole-genome Alignment Metrics) provides a single, integrated framework for both structural variant calling and association testing, thereby bypassing many of the difficulties that currently frustrate attempts to employ SVs in association testing. This step returns a VCF file.

### **Wham SV Annotation**

This step uses the VCF file generated in the previous step [Wham SV Call](#) and performs SV annotations.

### **CNVkit Batch**

A copy number variation ([CNV](#)) is when the number of copies of a particular gene varies from one individual to the next. Copy-number variation (CNV) is a large category of structural variation, which includes insertions, deletions and duplications. For copy number variation analysis, GenPipes DNA Sequencing pipeline (-t sv option) uses CNVkit that allows for CNV calling on single samples (e.g., tumor samples).

CNVkit provides an advantageous way to run the entire pipeline using the batch option to run various stages in copy number calling pipeline such as:

- Create target/anti-target bed files
- Gather read depths for those regions
- Compile a copy number reference
- Correct biases in tumor samples while calculating copy ratios
- Mark copy number segments

### CNVkit SV Annotation

This step performs CNVkit SV annotation.

### Run BreakSeq2

In this step, [BreakSeq2](#) is used to combine DNA double-strand breaks (DSBs) labeling with next generation sequencing (NGS) to map chromosome breaks with improved sensitivity and resolution. It is an ultra fast and accurate nucleotide-resolution analysis of structural variants.

### Ensemble MetaSV

[MetaSV](#) uses highly effective ensemble approach for calling SVs. It is an integrated SV caller which leverages multiple orthogonal SV signals for high accuracy and resolution. MetaSV proceeds by merging SVs from multiple tools for all types of SVs. It also analyzes soft-clipped reads from alignment to detect insertions accurately since existing tools underestimate insertion SVs. Local assembly in combination with dynamic programming is used to improve breakpoint resolution. Paired-end and coverage information is used to predict SV genotypes.

### MetaSV Annotation

This step uses output from previous step and performs SV annotations.

### Metrics

This step computes metrics and generates coverage tracks per sample. Multiple metrics are computed at this stage:

- Number of raw reads,
- Number of filtered reads,
- Number of aligned reads,
- Number of duplicate reads,
- Median, mean and standard deviation of insert sizes of reads after alignment,
- Percentage of bases covered at X reads (%\_bases\_above\_50 means the % of exons bases which have at least 50 reads)
- Whole genome or targeted percentage of bases covered at X reads (%\_bases\_above\_50 means the % of exons bases which have at least 50 reads).

A TDF (.tdf) coverage track is also generated at this step for easy visualization of coverage in the IGV browser.

### Metrics NGSCheckmate

[NGSCheckMate](#) is a software package for identifying next generation sequencing (NGS) data files from the same individual. It analyzes various types of NGS data files including (but not limited to) whole genome sequencing (WGS), whole exome sequencing (WES), RNA-seq, ChIP-seq, and targeted sequencing of various depths. Data types can be mixed (e.g. WES and RNA-seq, or RNA-seq and ChIP-seq). It takes BAM (reads aligned to the genome), VCF (variants) or FASTQ (unaligned reads) files as input. NGSCheckMate uses depth-dependent correlation models of allele fractions of known single-nucleotide polymorphisms (SNPs) to identify samples from the same individual.

This step takes as input the file containing all vcfs in project output.

### Metrics Verify BAM ID

In this step, [VerifyBAMID](#) software is used to verify whether the reads in particular file match previously known genotypes for an individual (or group of individuals), and checks whether the reads are contaminated as a mixture of two samples.

### **Metrics VCFTools Missing Individual**

This step uses [VCFtools](#) and `-missing_indv` option to generate a file reporting the *missingness* factor in the analysis on a per-individual basis. It takes bgzipped .vcf file as input and creates .imiss flat file indicating missingness.

### **Metrics VCFTools Depth Individual**

This step uses [VCFtools](#) and `-depth` option to generate a file containing the mean depth per individual. It takes as input bgzipped .vcf file and creates a .idepth flat file.

### **Metrics GATK Sample Fingerprint**

[CrosscheckFingerprints](#) (Picard) functionality in GATK toolkit is used to cross-check readgroups, libraries, samples, or files to determine if all data in the set of input files appears to come from the same individual. In this step, sample SAM/BAM or VCF file is taken as input and a fingerprint file is generated using [CrosscheckFingerprints](#) (Picard) in GATK. It checks the sample identity of the sequence/genotype data in the provided file (SAM/BAM or VCF) against a set of known genotypes in the supplied genotype file (in VCF format).

### **Metrics GATK Cluster Fingerprint**

In this step, `ClusterCrosscheckMetrics` function from GATK is used as a follow-up step to running [CrosscheckFingerprints](#) created in the [Metrics GATK Sample Fingerprint](#) step earlier. There are cases where one would like to identify a few groups out of a collection of many possible groups (say to link a bam to it's correct sample in a multi-sample vcf). In this step, sample SAM/BAM or VCF file is taken as input and a fingerprint file is generated.

### **Skewer Trimming**

TBD-GenPipes-Dev

### **BWA SAMbamba Sort SAM**

The input for this step is the trimmed FASTQ files if available. Otherwise, it uses the FASTQ files from the readset. If those are not available then it uses FASTQ output files from the previous 'Picard SAM to FASTQ' step where BAM files are converted to FASTQ format.

In this step, filtered reads are aligned to a reference genome. The alignment is done per sequencing readset. The alignment software used is [BWA](#) with algorithm: `bwa mem`. BWA output BAM files are then sorted by coordinate using [SAMBAMBA](#).

### **SAMBAM Merge SAM Files**

This step takes as input files:

- Aligned and sorted BAM output files from previous `bwa_mem_picard_sort_sam` step if available
- Else, BAM files from the readset file

In this step, BAM readset files are merged into one file per sample. Merge is done using [Picard](#).

### **SAMBAM Merge Realigned**

In this step, BAM files of regions of realigned reads are merged per sample using [SAMBAMBA](#). **CRAM Output**

Generate long term storage version of the final alignment files in CRAM format. Using this function will include the original final BAM file into the removable file list.

## More information

For the latest implementation and usage details refer to DNA Sequencing implementation [README.md](#) file.

- [Three-stage quality control strategies for DNA sequencing](#)
  - [NGS Mapping, errors and quality control](#)
  - [dbNSFP: a lightweight database of human nonsynonymous SNPs and their functional predictions](#)
  - [DNA-Seq Pipeline: some interesting information.](#)
  - [Manta - Rapid Detection of Structural Variants, Using Manta for filtering and annotations.](#)
- 

## DNA Sequencing (High Coverage) Pipeline

DNA Sequencing (High Coverage) is another GenPipes pipeline that is optimized for deep coverage samples. It is based on [DNA Sequencing Pipeline](#).

- [Introduction](#)
  - [Version](#)
  - [Usage](#)
  - [Example Run](#)
  - [Pipeline Schema](#)
  - [Pipeline Steps](#)
  - [Step Details](#)
  - [More information](#)
- 

## Introduction

The DnaSeq High Coverage Pipeline is based on the [DNA Sequencing Pipeline](#) and follows the same initial pipeline steps. The difference begins at the Mark Duplicates step. Since this data is high coverage Mark Duplicates step is not run. Recalibration is not run either because typically, these datasets are targeted with amplicons or custom capture which render recalibration useless. Also variant calling is done only using frequency. Bayesian callers are not used because these typically don't fare well with the high coverage.

---

## Version

3.6.2

For latest pipeline implementation details visit [README.md](#).

---

## Usage

```
dnaseq_high_coverage.py [-h] [--help] [-c CONFIG [CONFIG ...]]
                        [-s STEPS] [-o OUTPUT_DIR]
                        [-j {pbs,batch,daemon,slurm}] [-f] [--no-json]
                        [--report] [--clean]
                        [-l {debug,info,warning,error,critical}]
                        [--sanity-check]
                        [--container {wrapper, singularity} <IMAGE PATH>]
                        [--genpipes_file GENPIPES_FILE]
                        [-r READSETS] [-v]
                        [-t {mugqic,mpileup,light,sv}]
```

### Optional Arguments

`-t {mugqic,mpileup,light, sv}, --type {mugqic,mpileup,light,sv}`

DNaseq High Coverage analysis **type**

`-r READSETS, --readsets READSETS`

readset file

`-h` show this help message **and** exit

`--help` show detailed description of pipeline **and** steps

`-c CONFIG [CONFIG ...], --config CONFIG [CONFIG ...]`

config INI-style **list** of files; config parameters  
are overwritten based on files order

`-s STEPS, --steps STEPS` step **range** e.g. '1-5', '3,6,7', '2,4-8'

`-o OUTPUT_DIR, --output-dir OUTPUT_DIR`

output directory (default: current)

`-j {pbs,batch,daemon,slurm}, --job-scheduler {pbs,batch,daemon,slurm}`

job scheduler **type** (default: slurm)

<code>-f, --force</code>	force creation of jobs even <b>if</b> up to date (default: false)
<code>--no-json</code>	do <b>not</b> create JSON file per analysed sample to track the analysis status (default: false i.e. JSON file will be created)
<code>--report</code>	create ' <b>pandoc</b> ' command to merge <b>all</b> job markdown report files <b>in</b> the given step <b>range</b> into HTML, <b>if</b> they exist; <b>if</b> <code>--report</code> <b>is</b> set, <code>--job-scheduler</code> , <code>--force</code> , <code>--clean</code> options <b>and</b> job up-to-date status are ignored (default: false)
<code>--clean</code>	create ' <b>rm</b> ' commands <b>for</b> <b>all</b> job removable files <b>in</b> the given step <b>range</b> , <b>if</b> they exist; <b>if</b> <code>--clean</code> <b>is</b> set, <code>--job-scheduler</code> , <code>--force</code> options <b>and</b> job up-to-date status are ignored (default: false)
<code>-l {debug,info,warning,error,critical}, --log {debug,info,warning,error,critical}</code>	log level (default: info)
<code>--sanity-check</code>	run the pipeline in 'sanity check mode' to verify all the input files needed for the pipeline to run are available on the system (default: false)
<code>--container {wrapper, singularity} &lt;IMAGE PATH&gt;</code>	run pipeline inside a container providing a container image path <b>or</b> accessible singularity hub path
<code>-v, --version</code>	show the version information <b>and</b> exit
<code>-g GENPIPES_FILE, --genpipes_file GENPIPES_FILE</code>	Commands <b>for</b> running the pipeline are output to this file pathname. The data specified to pipeline command line <b>is</b> processed <b>and</b> pipeline run commands are stored <b>in</b> GENPIPES_FILE, <b>if</b> this option <b>is</b> specified . Otherwise, the output will be redirected to stdout . This file can be used to actually " <b>run the GenPipes Pipeline</b> ".

## Example Run

Following instructions are meant to be run on C3G deployed GenPipes on Beluga server. It uses human genome data available at Beluga server. Use the following command on Beluga server to run DNA Sequencing (high coverage) pipeline:

```
dnaseq_high_coverage.py -c
$MUGQIC_PIPELINES_HOME/pipelines/dnaseq_high_coverage/dnaseq_high_coverage.base.ini
$MUGQIC_PIPELINES_HOME/pipelines/dnaseq_high_coverage/dnaseq_high_coverage.beluga.ini
-j slurm -s 1-15 -g dna_high_cov_commands.sh

bash dna_high_cov_commands.sh
```

**Warning:** While issuing the pipeline run command, use ``-g GENPIPES_FILE`` option (see example above) instead of using the ``> GENPIPES_FILE`` option supported by GenPipes so far, as shown below:

```
[genpipes_seq_pipeline].py -t mugqic -c $MUGQIC_PIPELINES_HOME/pipelines/[genpipes_
↪ seq_pipeline]/[genpipes_seq_pipeline].base.ini $MUGQIC_PIPELINES_HOME/pipelines/
↪ [genpipes_seq_pipeline]/[genpipes_seq_pipeline].guillimin.ini -r readset.[genpipes_
↪ seq_pipeline].txt -s 1-6 > [genpipes_seq_pipeline]_commands_mugqic.sh

bash [genpipes_seq_pipeline]_commands_mugqic.sh
```

``> scriptfile`` should be considered deprecated and ``-g scriptfile`` option is recommended instead.

Please note that redirecting commands to a script ``> genpipe_script.sh`` is still supported for now. **But going forward, this mechanism might be dropped in a future GenPipes release.**

## Pipeline Schema

Figure below shows the schema of the DNA Sequencing (High Coverage) sequencing protocol.

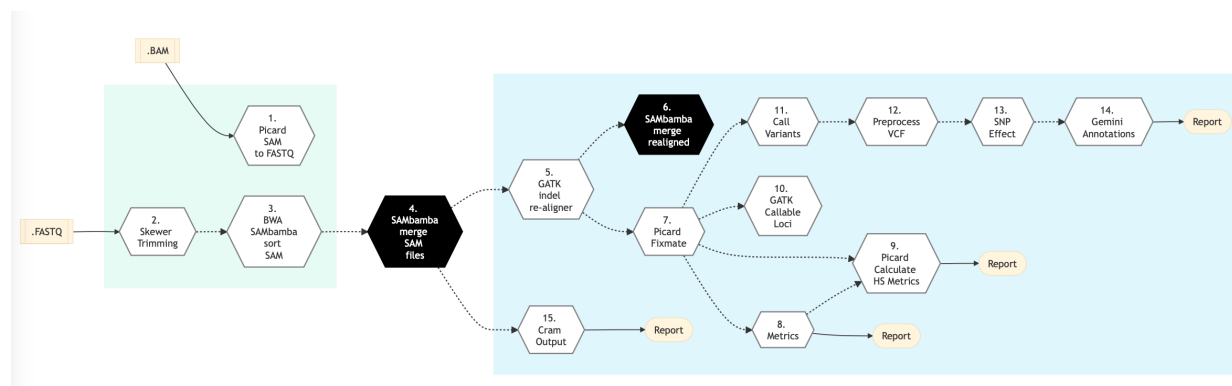
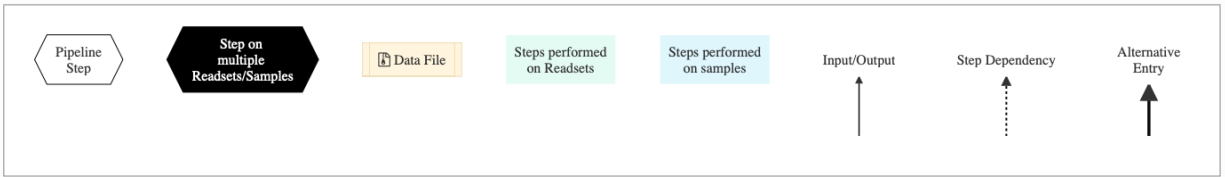


Fig. 14: Figure: Schema of DNA Sequencing (High Coverage) Sequencing protocol





## Pipeline Steps

The table below shows various steps that constitute the DNA Sequencing (High Coverage) genomic analysis pipelines.

	<i>DNA (High Coverage) sequencing Steps</i>
1.	<i>Picard SAM to FASTQ</i>
2.	<i>Skewer Trimming</i>
3.	<i>BWA SAMbamba Sort SAM</i>
4.	<i>SAMBAM Merge SAM Files</i>
5.	<i>GATK Indel Realigner</i>
6.	<i>SAMBAM Merge Realigned</i>
7.	<i>Picard Fixmate</i>
8.	<i>Metrics</i>
9.	<i>Picard Calculate HS Metrics</i>
10	<i>GATK callable loci</i>
11.	<i>Call Variants</i>
12.	<i>Pre-process VCF</i>
13.	<i>SNP Effect</i>
14.	<i>Gemini Annotations</i>
15.	<i>CRAM Output</i>

## Step Details

Following are the various steps that are part of GenPipes DNA (High Coverage) Sequencing genomic analysis pipeline:

### **Picard SAM to FASTQ**

This step converts SAM/BAM files from the input readset file into FASTQ format if FASTQ files are not already specified in the readset file. Otherwise, it does no processing.

### **Trimmomatic**

This step takes as input files:

- FASTQ files from the readset file if available
- Else, FASTQ output files from previous `picard_sam_to_fastq` conversion of BAM files

Raw reads quality trimming and removing of Illumina adapters is performed using [Trimmomatic - Flexible sequencing data trimmer](#). If an adapter FASTA file is specified in the config file (section 'trimmomatic', param 'adapter\_fasta'), it is used first. Else, 'Adapter1' and 'Adapter2' columns from the readset file are used to create an adapter FASTA file, given then to Trimmomatic. For PAIRED\_END readsets, readset adapters are reversed-complemented and swapped, to match Trimmomatic Palindrome strategy. For SINGLE\_END readsets, only Adapter1 is used and left unchanged.

### **Merge Trimmomatic Stats**

The trim statistics per readset are merged at this step.

### **GATK Indel Realigner**

Insertion and deletion realignment is performed on regions where multiple base mismatches are preferred over indels by the aligner since it can appear to be less costly by the algorithm. Such regions will introduce false positive variant calls which may be filtered out by realigning those regions properly. Realignment is done using [GATK Toolkit](#). The reference genome is divided by a number regions given by the `nb_jobs` parameter.

### **Picard Fixmate**

After realignment, some read pairs might become unsynchronized. This step verifies that the information is correct between the two pairs and if it is not, it fixes any inaccuracies to avoid problems during the following steps.

### **Metrics**

Compute metrics and generate coverage tracks per sample. Multiple metrics are computed at this stage: Number of raw reads, Number of filtered reads, Number of aligned reads, Number of duplicate reads, Median, mean and standard deviation of insert sizes of reads after alignment, percentage of bases covered at X reads (`%_bases_above_50` means the % of exons bases which have at least 50 reads) whole genome or targeted percentage of bases covered at X reads (`%_bases_above_50` means the % of exons bases which have at least 50 reads). A TDF (.tdf) coverage track is also generated at this step for easy visualization of coverage in the IGV browser.

### **Picard Calculate HS Metrics**

Compute on target percent of hybridization based capture.

### **GATK callable loci**

Computes the callable region or the genome as a bed track.

### **Call Variants**

This step uses VarScan caller for insertions and deletions.

### **Pre-process VCF**

Pre-process vcf for loading into a annotation database - [Gemini](#). Processes include normalization and decomposition of MNPs by [Vt](#) and vcf FORMAT modification for correct loading into Gemini.

### **SNP Effect**

This step performs Variant effect annotation. The .vcf files are annotated for variant effects using the SnpEff software. SnpEff annotates and predicts the effects of variants on genes (such as amino acid changes).

### Gemini Annotations

This step loads functionally annotated vcf file into a mysql lite annotation database as part of [Gemini processing](#).

### Skewer Trimming

TBD-GenPipes-Dev

### BWA SAMbamba Sort SAM

The input for this step is the trimmed FASTQ files if available. Otherwise, it uses the FASTQ files from the readset. If those are not available then it uses FASTQ output files from the previous ‘Picard SAM to FASTQ’\_ step where BAM files are converted to FASTQ format.

In this step, filtered reads are aligned to a reference genome. The alignment is done per sequencing readset. The alignment software used is [BWA](#) with algorithm: bwa mem. BWA output BAM files are then sorted by coordinate using [SAMBAMBA](#).

### SAMBAM Merge SAM Files

This step takes as input files:

- Aligned and sorted BAM output files from previous bwa\_mem\_picard\_sort\_sam step if available
- Else, BAM files from the readset file

In this step, BAM readset files are merged into one file per sample. Merge is done using [Picard](#).

### SAMBAM Merge Realigned

In this step, BAM files of regions of realigned reads are merged per sample using [SAMBAMBA](#). **CRAM Output**

Generate long term storage version of the final alignment files in CRAM format. Using this function will include the original final BAM file into the removable file list.

## More information

For the latest implementation and usage details refer to DNA Sequencing (High Coverage) [README.md](#) file.

- [Gemini Annotations Paper](#)

## HiC Sequencing Pipeline

The HiC-Seq workflow aligns reads using [HiCUP](#). It creates tag directories, produces interaction matrices, and identifies compartments and significant interactions using [Homer](#). It identifies topologically associating domains using [TopDom](#) and [RobusTAD](#) (bioRxiv 293175). It also creates “.hic” files using [JuiceBox](#) and metrics reports using [MultiQC](#). The HiC-Seq workflow can also process capture Hi-C data with the flag “-t capture” using [CHICAGO](#).

- [Introduction](#)
- [Version](#)
- [Usage](#)

- *Example Run*
- *Pipeline Schema*
- *Pipeline Steps*
- *Step Details*
- *More information*

---

## Introduction

**Hi-C experiments** allow researchers to understand chromosomal folding and structure using proximity ligation techniques. This pipeline analyzes both Hi-C experimental data (-t hic) and capture Hi-C data (-t capture).

### Hi-C

The Hi-C pipeline, selected using the “-t hic” parameter, starts by trimming adapters and low quality bases. It then maps the reads to a reference genome using **HiCUP**. HiCUP first truncates chimeric reads, maps reads to the genome, then it filters Hi-C artifacts and removes read duplicates. Samples from different lanes are merged and a tag directory is created by Homer, which is also used to produce the interaction matrices and compartments. TopDom is used to predict topologically associating domains (TADs) and Homer is used to identify significant interactions.

### Hi-C capture

The capture Hi-C pipeline, selected using the “-t capture” parameter, starts by trimming adapters and low quality bases. It then maps the reads to a reference genome using HiCUP. HiCUP first truncates chimeric reads, maps reads to the genome, then it filters Hi-C artifacts and removes read duplicates. Samples from different lanes are merged and CHiCAGO is then used to filter capture-specific artifacts and call significant interactions. This pipeline also identifies enrichment of regulatory features when provided with ChIP-Seq marks. It can also return bed interactions with significant baits (bait\_intersect step) or with captured interactions (capture\_intersect step).

An example of the Hi-C report for an analysis on public data (GM12878 Rao. et al.) is available for illustration purposes only: [Hi-C report](#).

---

## Version

3.6.2

For the latest implementation and usage details refer to Hi-C Sequencing implementation [README.md](#) file.

---

## Usage

```
hicseq.py [-h] [--help] [-c CONFIG [CONFIG ...]] [-s STEPS]
          [-o OUTPUT_DIR] [-j {pbs,batch,daemon,slurm}] [-f]
          [--no-json] [--report] [--clean]
          [-l {debug,info,warning,error,critical}] [--sanity-check]
          [--container {wrapper, singularity} <IMAGE PATH>
```

(continues on next page)

(continued from previous page)

```

[--genpipes_file GENPIPES_FILE]
-e {DpnII,HindIII,NcoI,MboI,Arima} [-t {hic,capture}]
[-r READSETS] [-v]

```

**Optional Arguments**

```

-e {DpnII,HindIII,NcoI,MboI,Arima},
--enzyme {DpnII,HindIII,NcoI,MboI,Arima}

Restriction Enzyme used to generate Hi-C library (default:
↳DpnII)

```

```

-t {hic,capture},
--type {hic,capture}      Hi-C experiment type (default hic)

```

```

-r READSETS, --readsets READSETS

readset file

```

```

-h                        show this help message and exit

```

```

--help                  show detailed description of pipeline and steps

```

```

-c CONFIG [CONFIG ...], --config CONFIG [CONFIG ...]

config INI-style list of files; config parameters
are overwritten based on files order

```

```

-s STEPS, --steps STEPS  step range e.g. '1-5', '3,6,7', '2,4-8'

```

```

-o OUTPUT_DIR, --output-dir OUTPUT_DIR

output directory (default: current)

```

```

-j {pbs,batch,daemon,slurm}, --job-scheduler {pbs,batch,daemon,slurm}

job scheduler type (default: slurm)

```

```

-f, --force             force creation of jobs even if up to date (default:
false)

```

```

--no-json               do not create JSON file per analysed sample to track
the analysis status (default: false i.e. JSON file
will be created)

```

```

--report               create 'pandoc' command to merge all job markdown
report files in the given step range into HTML, if
they exist; if --report is set, --job-scheduler,
--force, --clean options and job up-to-date status
are ignored (default: false)

```

<code>--clean</code>	create 'rm' commands for all job removable files in the given step range, if they exist; if <code>--clean</code> is set, <code>--job-scheduler</code> , <code>--force</code> options and job up-to-date status are ignored (default: false)
<code>-l {debug,info,warning,error,critical}, --log {debug,info,warning,error,critical}</code>	log level (default: info)
<code>--sanity-check</code>	run the pipeline in 'sanity check mode' to verify all the input files needed for the pipeline to run are available on the system (default: false)
<code>--container {wrapper, singularity} &lt;IMAGE PATH&gt;</code>	run pipeline inside a container providing a container image path or accessible singularity hub path
<code>-v, --version</code>	show the version information and exit
<code>-g GENPIPES_FILE, --genpipes_file GENPIPES_FILE</code>	Commands for running the pipeline are output to this file pathname. The data specified to pipeline command line is processed and pipeline run commands are stored in GENPIPES_FILE, if this option is specified. Otherwise, the output will be redirected to stdout. This file can be used to actually "run the GenPipes Pipeline".

## Example Run

You can download [Hi-C test dataset](#) and run the following commands:

```
hicseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/hicseq/hicseq.base.ini $MUGQIC_PIPELINES_
↪HOME/pipelines/hicseq/hicseq.guillimn.ini -r readsets.HiC010.tsv -s 1-15 -e MboI -j_
↪pbs -g hicseqScript_SRR1658581.txt

bash hicseqScript_SRR1658581.txt
```

**Warning:** While issuing the pipeline run command, use ``-g GENPIPES_FILE`` option (see example above) instead of using the ``> GENPIPES_FILE`` option supported by GenPipes so far, as shown below:

```
[genpipes_seq_pipeline].py -t mugqic -c $MUGQIC_PIPELINES_HOME/pipelines/[genpipes_
↪seq_pipeline]/[genpipes_seq_pipeline].base.ini $MUGQIC_PIPELINES_HOME/pipelines/
↪[genpipes_seq_pipeline]/[genpipes_seq_pipeline].guillimin.ini -r readset.[genpipes_
↪seq_pipeline].txt -s 1-6 > [genpipes_seq_pipeline]_commands_mugqic.sh
```

```
bash [genpipes_seq_pipeline]_commands_mugqic.sh
```

`> scriptfile` should be considered deprecated and `-g scriptfile` option is recommended instead. Please note that redirecting commands to a script `> genpipe\_script.sh` is still supported for now. **But going forward, this mechanism might be dropped in a future GenPipes release.**

## Pipeline Schema

Figure below shows the schema of the Hi-C protocol.

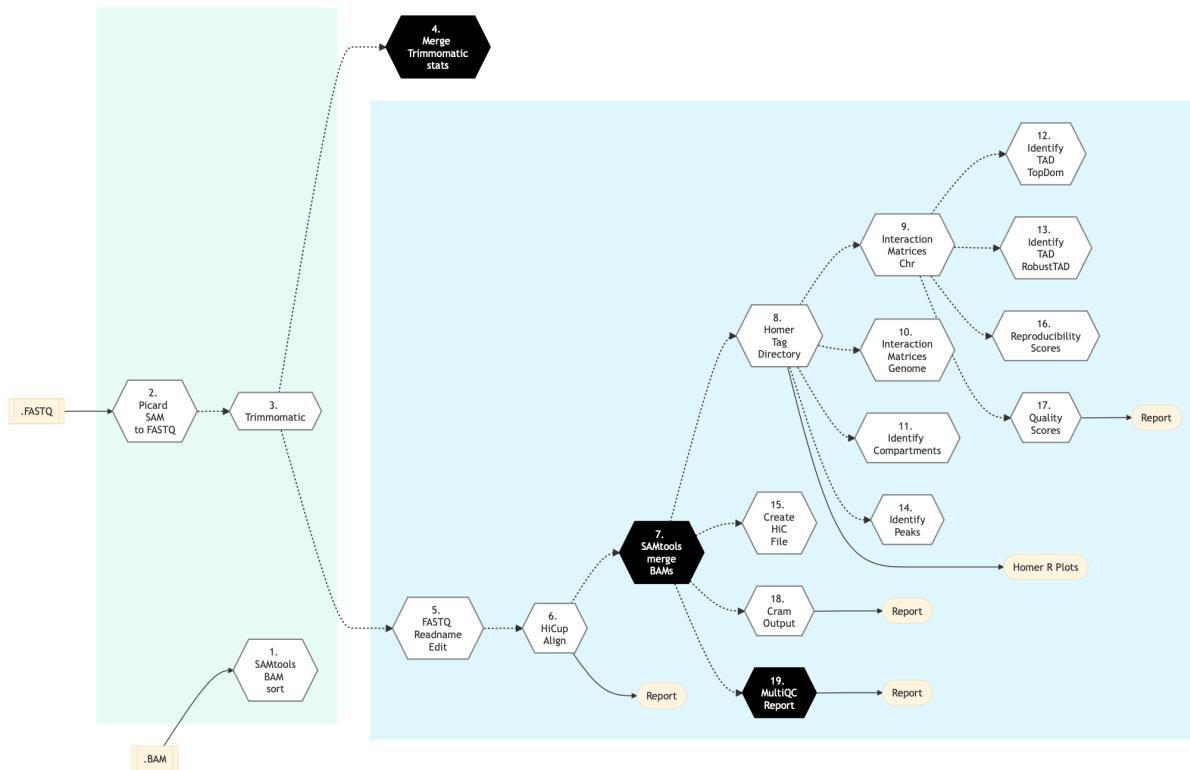
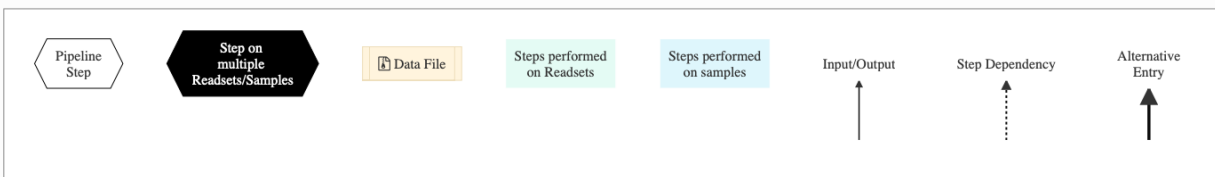


Fig. 15: Figure: Schema of Hi-C Sequencing protocol



The following figure shows the pipeline schema for capture Hi-C protocol.

The following figure shows the pipeline schema for capture Hi-C protocol.

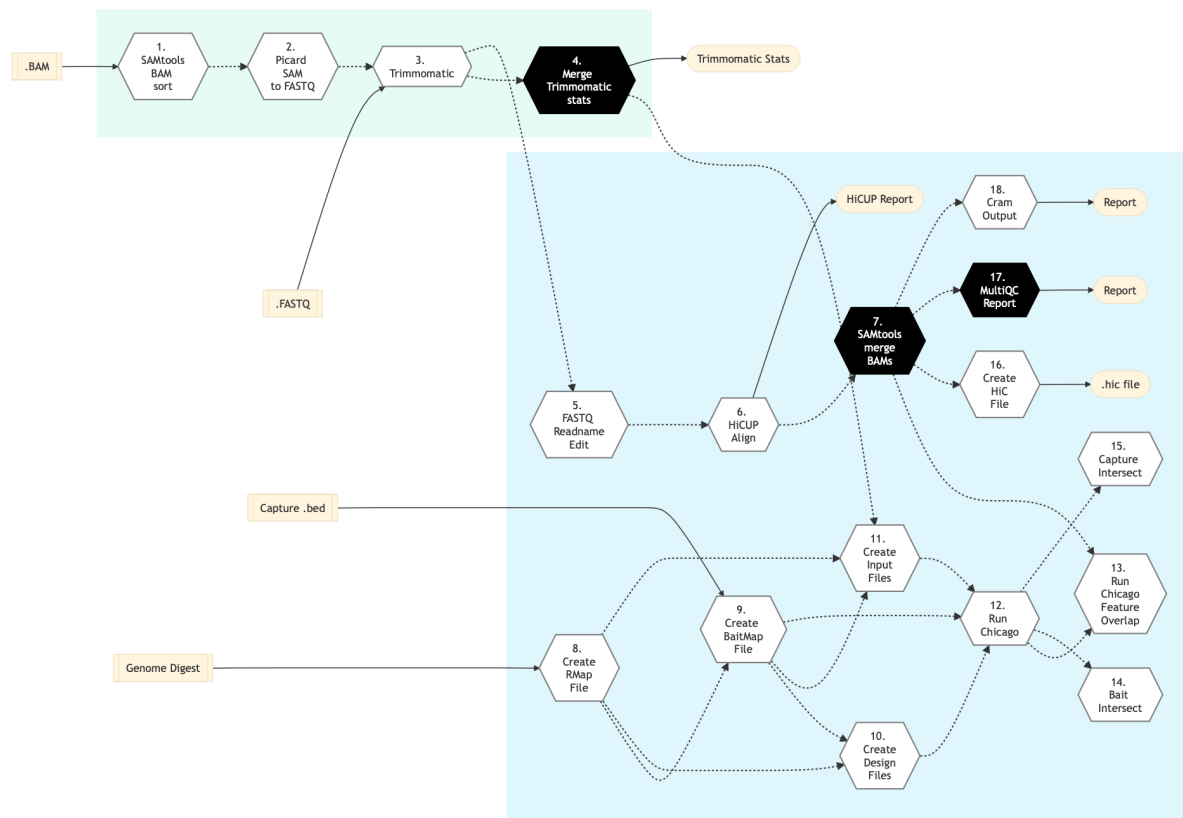
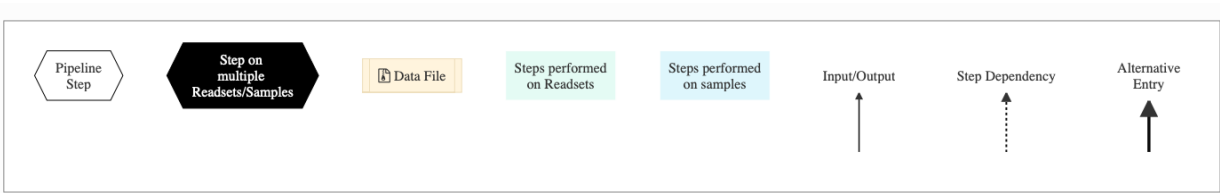


Fig. 16: Figure: Schema of Hi-C capture protocol





## Pipeline Steps

The table below shows various steps that constitute the Hi-C and Hi-C capture genomic analysis pipelines.

	<i>Hi-C sequencing Steps</i>	<i>Hi-C Capture sequencing Steps</i>
1.	<i>Samtools Bam Sort</i>	<i>Samtools Bam Sort</i>
2.	<i>Picard Sam to Fastq</i>	<i>Picard Sam to Fastq</i>
3.	<i>Trimmomatic</i>	<i>Trimmomatic</i>
4.	<i>Merge Trimmomatic Stats</i>	<i>Merge Trimmomatic Stats</i>
5.	<i>Fastq ReadName Edit</i>	<i>Fastq ReadName Edit</i>
6.	<i>Hicup Align</i>	<i>Hicup Align</i>
7.	<i>Samtools Merge Bams</i>	<i>Samtools Merge Bams</i>
8.	<i>Homer Tag Directory</i>	<i>Create RMAP File</i>
9.	<i>Interaction Matrices Chr</i>	<i>Create Baitmap File</i>
10.	<i>Interaction Matrices Genome</i>	<i>Create Design Files</i>
11.	<i>Identify Compartments</i>	<i>Create Input Files</i>
12.	<i>Identify TADs TopDom</i>	<i>Run Chicago</i>
13.	<i>Identify TADs RobustTAD</i>	<i>RunChicago FeatureOverlap</i>
14.	<i>Identify Peaks</i>	<i>Bait Intersect</i>
15.	<i>Create Hic File</i>	<i>Capture Intersect</i>
16.	<i>Reproducibility Scores</i>	<i>Create Hic File</i>
17.	<i>Quality Scores</i>	<i>Multiqc Report</i>
18.	<i>CRAM Output</i>	<i>CRAM Output</i>
19.	<i>Multiqc Report</i>	

## Step Details

Following are the various steps that are part of Hi-C genomic analysis pipelines:

### Samtools Bam Sort

Sorts [BAM](#) files by readname prior to *Picard Sam to Fastq* step, in order to minimize memory consumption. If BAM file is small and the memory requirements are reasonable, this step can be skipped.

### Picard Sam to Fastq

If FASTQ files are not already specified in the Readset file, then this step converts [SAM / BAM](#) files from the input Readset into FASTQ format. Otherwise, it does nothing.

### Trimmomatic

Raw reads quality trimming and removing of Illumina adapters is performed using [Trimmomatic Process](#). If an adapter FASTA file is specified in the config file (section 'trimmomatic', param 'adapter\_fasta'), it is used first. Else, 'Adapter1' and 'Adapter2' columns from the readset file are used to create an adapter FASTA file, given then to Trimmomatic. For PAIRED\_END readsets, readset adapters are reversed-complemented and swapped, to match Trimmomatic Palindrome strategy. For SINGLE\_END readsets, only Adapter1 is used and left unchanged.

If available, trimmomatic step in Hi-C analysis takes FASTQ files from the readset file as input. Otherwise, it uses the FASTQ output file generated from the previous *Picard Sam to Fastq* step conversion of the BAM files.

### Merge Trimmomatic Stats

The trim statistics per Readset file are merged at this step.

### Fastq ReadName Edit

Removes the added /1 and /2 by *Picard Sam to Fastq* transformation to avoid issues with downstream software like [Homer](#).

### HiCUP Align

Paired-end Hi-C reads are truncated, mapped and filtered using HiCUP. The resulting BAM file is filtered for Hi-C artifacts and duplicated reads. It is ready for use as input for downstream analysis. For more detailed information about the HiCUP process visit [HiCUP Project Page](#).

### Samtools Merge Bams

BAM readset files are merged into one file per sample. Merge is done using [samtools](#). This step takes as input files the aligned [BAM / SAM](#) files from the *Hicup Align* step.

### Homer Tag Directory

The BAM file produced by HiCUP is used to create a tag directory using [Homer](#) for further analysis that includes interaction matrix generation, compartments and identifying significant interactions. For more details, visit [Homer](#).

### Interaction Matrices Chr

In this step, intra-chromosomal interaction matrices are produced by [Homer Matrices](#) at resolutions defined in the ini config file and plotted by [HiCPlotter](#).

### Interaction Matrices Genome

Genome-wide interaction matrices are produced by Homer at resolutions defined in the ini config file. See [Homer Matrices](#) for details.

### Identify Compartments

Genomic compartments are identified using Homer at resolutions defined in the ini config file. For details, see [Homer Compartments](#).

### Identify TADs TopDom

Topological Associating Domains (TADs) are identified using [TopDom](#) at resolutions defined in the ini config file. For details, see [TopDom](#).

### **Identify TADs RobusTAD**

In this step, topological associating Domain (TAD) scores are calculated using RobusTAD for every bin in the genome. RobusTAD is resolution-independent and will use the first resolution in “resolution\_TADs” under [identify\_TADs] in the ini file. For details, see [latest README.md for RobusTAD](#).

### **Identify Peaks**

This step uses [Homer Peaks](#) to identify significant intra-Chromosomal interactions or **peaks**.

### **Create HiC File**

In this step, a .hic file is created per sample in order to visualize in [JuiceBox](#) , WashU epigenome browser or as input for other tools.

### **Create Baitmap File**

Here, the baitmap file for Chicago capture analysis is created using the created [RMAP](#) file and the probe capture bed file.

### **Create Design Files**

In this step, the design files ([NPerBin file](#) (.npb), nbaitspersbin file (.nbpb), [Proximal Other End](#) (proxOE) file (.poe)) for Chicago capture analysis are created using the [RMAP](#) file and the baitmap file.

### **Create Input Files**

Here, the input file (sample.chinput) for Chicago capture analysis is created using the [RMAP](#) file, the baitmap file and the hicup aligned BAM.

### **Run Chicago**

Chicago is run on capture data. Chicago will filter capture hic artifacts and identify significant interactions. It will output data as a bed file and will also output SeqMonk and WashU tracks. For more detailed information about the Chicago, including how to interpret the plots, visit [Chicago documentation](#).

### **RunChicago FeatureOverlap**

This step runs the feature enrichment of Chicago significant interactions. See [Chicago documentation](#) for details.

### **Bait Intersect**

With a bed file as input, for example a bed of [GWAS](#) SNPs or features of interest, this method returns the lines in the bed file that intersect with the baits that have significant interactions. Input bed must have 4 columns (<chr> <start> <end> <annotation>) and must be tab separated.

### **Capture Intersect**

With bed file as input, for example a bed of GWAS SNPs or features of interest, this method returns the lines in the bed file that intersect with the captured ends (“Other Ends”) that have significant interactions. Input bed must have 4 columns (<chr> <start> <end> <annotation>) and must be tab separated.

### **Create RMAP File**

A [RMAP](#) file for Chicago Capture Analysis is created using the HiCUP digestion file.

### **Multiqc Report**

A quality control report for all samples is generated. For more detailed information see [MultiQC documentation](#).

### **Reproducibility Scores**

In this step, R package [hicrep](#) is used for calculating the inter-chromosomal reproducibility score. Hicrep considers spatial features in Hi-C data, such as domain structure and distance-dependence, so that there are no misleading results in Hi-C data reproducibility while studying genome-wide chromatin interactions.

The novel reproducibility measure in hicrep can assess pairwise differences between Hi-C matrices under a wide range of settings, and can be used to determine optimal sequencing depth. Compared to existing approaches, it consistently shows higher accuracy in distinguishing subtle differences in reproducibility and depicting interrelationships of cell lineages than existing approaches.

Pairwise reproducibility scores are calculated for each chromosome pair in each sample pair, using hicrep at resolutions (bin size) defined in [Interaction Matrices Chr](#) step. Other parameters are defined in [reproducibility\\_scores](#) section of the hicseq.base.ini file.

All the scores are finally merged together and an output .csv file is created using the parameters specified during the analysis. Chromosome number, reproducibility scores, standard deviation and smoothing value are used in the analysis. In order to compare samples, ensure that the smoothing value and the sequencing depth are similar across samples.

Down sampling of samples can be performed using the down\_sampling parameter in the .ini config file. Correlation matrices and wight matrices can be saved using the following initialization values in the .ini config file:

```
corr=TRUE
weights=TRUE
```

### Quality Scores

In this step, Quality score per chromosome for each sample is calculated using QuSAR-QC at all resolutions, sequencing depth (coverage) and down\_sampling value (coverage) defined in quality\_scores step of .ini config file. [QUSAR-QC](#) is a part of the hifive hic-seq analysis suite.

Quality Assessment of Spatial Arrangement Reproducibility (QuASAR) transformation and scoring uses the consistency between the raw interaction matrix and the correlation matrix of distance-corrected signal to determine sample quality. For all interactions less than or equal to 100 times the resolution of size, the correlation value is calculated.

In order to determine the quality score for a given chromosome, the weighted average of the correlation values (weighted by the raw interaction signal) minus the unweighted correlation signal average is calculated. A genome-wide value is calculated by summing all numerators and denominators across all chromosomes prior to dividing and subtracting score components. The replicate score is calculated by finding the correlation between the weighted correlation matrices for two samples.

For details, see [hifive suite](#).

### CRAM Output

Generate long term storage version of the final alignment files in CRAM format. Using this function will include the original final BAM file into the removable file list.

---

## More information

For the latest implementation and usage details refer to Hi-C Sequencing implementation [README.md](#) file.

- Comprehensive Mapping of Long-Range Interactions Reveals Folding Principles of the Human Genome - [Paper introducing Hi-C](#).
  - A high-resolution map of the three-dimensional chromatin interactome in human cells. [Defining target gene using Hi-C](#).
  - [Measuring the reproducibility of Hi-C data](#).
-

## Illumina Run Processing Pipeline

Genomic Analyzers create massive amounts of data. The Illumina Run Processing Pipeline transforms primary imaging output from the Genome Analyzer into discrete aligned strings of bases. A package of integrated algorithms perform the core primary data transformation steps: image analysis, intensity scoring, base calling, and alignment. It also converts Base Call (BCL) files into **FASTQ** files that are needed for higher genomic analysis such as *ChIP Sequencing*.

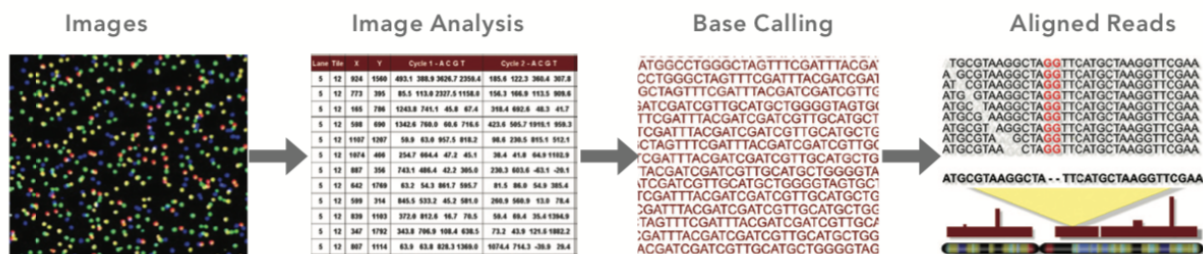


Fig. 17: Figure: Pipeline data transformation steps (Source: Illumina)

- [Introduction](#)
- [Version](#)
- [Usage](#)
- [Example Run](#)
- [Pipeline Schema](#)
- [Pipeline Steps](#)
- [Step Details](#)
- [More Information](#)

## Introduction

The standard MUGQIC Illumina Run Processing pipeline uses the **Illumina bcl2fastq** software to convert and demultiplex the base call files (BCL) to FASTQ files. The pipeline runs some QCs on the raw data, on the FASTQ and on the alignment. It performs the core primary data transformation steps: image analysis, intensity scoring, base calling, and alignment.

1. **Image Analysis:** interpreting the image data to identify distinct clusters of genes
2. **Base Calling:** profiles for each cluster are used to call bases. Obtaining the quality of each base call is crucial for downstream analysis.
3. **Alignment:** entire set of called sequence reads are aligned to create a final sequence output file optimized for SNP identification.

## Sample Files

This pipeline uses two input sample sheets.

1. [Casava Sheet](#)

## 2. Nanuq Run Sheet

You can see samples of Casava Sheet and Nanuq Run Sheet and also refer to [Casava User Guide](#) for more details.

## Version

3.6.2

For the latest implementation and usage details refer to Illumina Sequencing implementation [README file](#) file.

## Usage

```
illumina_run_processing.py [-h] [--help] [-c CONFIG [CONFIG ...]]
                           [-s STEPS] [-o OUTPUT_DIR]
                           [-j {pbs,batch,daemon,slurm}] [-f]
                           [--no-json] [--report] [--clean]
                           [-l {debug,info,warning,error,critical}]
                           [--sanity-check]
                           [--container {wrapper, singularity} <IMAGE PATH>]
                           [--genpipes_file GENPIPES_FILE]
                           [-d RUN_DIR] [--lane LANE_NUMBER]
                           [-r READSETS] [-i CASAVA_SHEET_FILE]
                           [-x FIRST_INDEX] [-y LAST_INDEX]
                           [-m NUMBER_OF_MISMATCHES] [-w] [-v]
```

### Optional Arguments

`-d RUN_DIR, --run RUN_DIR`

run directory

`--lane LANE_NUMBER`      lane number

`-r READSETS, --readsets READSETS`

nanuq readset file. The default file is `'run.nanuq.csv'` in the output folder. Will be automatically downloaded if not present.

`-i CASAVA_SHEET_FILE`    illumina casava sheet. The default file is `'SampleSheet.nanuq.csv'` in the output folder. Will be automatically downloaded if not present

`-x FIRST_INDEX`            first index base to use for demultiplexing (inclusive). The index from the sample sheet will be adjusted according to that value.

-y LAST_INDEX	last index base to use <b>for</b> demultiplexing (inclusive)
-m NUMBER_OF_MISMATCHES	number of index mismatches allowed <b>for</b> demultiplexing (default 1). Barcode collisions are always checked.
-w, --force-download	force the download of the samples sheets (default: false)
-r READSETS, --readsets READSETS	readset file
-h	show this help message <b>and</b> exit
--help	show detailed description of pipeline <b>and</b> steps
-c CONFIG [CONFIG ...], --config CONFIG [CONFIG ...]	config INI-style <b>list</b> of files; config parameters are overwritten based on files order
-s STEPS, --steps STEPS	step <b>range</b> e.g. '1-5', '3,6,7', '2,4-8'
-o OUTPUT_DIR, --output-dir OUTPUT_DIR	output directory (default: current)
-j {pbs,batch,daemon,slurm}, --job-scheduler {pbs,batch,daemon,slurm}	job scheduler <b>type</b> (default: slurm)
-f, --force	force creation of jobs even <b>if</b> up to date (default: false)
--no-json	do <b>not</b> create JSON file per analysed sample to track the analysis status (default: false i.e. JSON file will be created)
--report	create ' <b>pandoc</b> ' command to merge <b>all</b> job markdown report files <b>in</b> the given step <b>range</b> into HTML, <b>if</b> they exist; <b>if</b> --report <b>is</b> set, --job-scheduler, --force, --clean options <b>and</b> job up-to-date status are ignored (default: false)
--clean	create ' <b>rm</b> ' commands <b>for</b> <b>all</b> job removable files <b>in</b> the given step <b>range</b> , <b>if</b> they exist; <b>if</b> --clean <b>is</b> set, --job-scheduler, --force options <b>and</b> job up-to-date status are ignored (default: false)



```
-l {debug,info,warning,error,critical}, --log {debug,info,warning,error,critical}
```

log level (default: info)

```
--sanity-check      run the pipeline in `sanity check mode` to verify
                    all the input files needed for the pipeline to run
                    are available on the system (default: false)
```

```
--container {wrapper, singularity} <IMAGE PATH>
```

run pipeline inside a container providing a container image path **or** accessible singularity hub path

```
-v, --version      show the version information and exit
```

```
-g GENPIPES_FILE, --genpipes_file GENPIPES_FILE
```

Commands **for** running the pipeline are output to this file pathname. The data specified to pipeline command line **is** processed **and** pipeline run commands are stored **in** GENPIPES\_FILE, **if** this option **is** specified. Otherwise, the output will be redirected to stdout. This file can be used to actually "run the GenPipes Pipeline".

## Example Run

Use the following commands to execute Illumina Sequencing Pipeline:

```
illumina_run_processing.py <Add options - info not available in README file> >illumina_
↪cmd.sh
```

```
bash illumina_cmd.sh
```

**Warning:** While issuing the pipeline run command, use ``-g GENPIPES_FILE`` option (see example above) instead of using the ``> GENPIPES_FILE`` option supported by GenPipes so far, as shown below:

```
[genpipes_seq_pipeline].py -t mugqic -c $MUGQIC_PIPELINES_HOME/pipelines/[genpipes_
↪seq_pipeline]/[genpipes_seq_pipeline].base.ini $MUGQIC_PIPELINES_HOME/pipelines/
↪[genpipes_seq_pipeline]/[genpipes_seq_pipeline].guillimin.ini -r readset.[genpipes_
↪seq_pipeline].txt -s 1-6 > [genpipes_seq_pipeline]_commands_mugqic.sh
```

```
bash [genpipes_seq_pipeline]_commands_mugqic.sh
```

``> scriptfile`` should be considered deprecated and ``-g scriptfile`` option is recommended instead.

Please note that redirecting commands to a script ``> genpipe_script.sh`` is still supported for now. **But going forward, this mechanism might be dropped in a future GenPipes release.**

You can download the test dataset for this pipeline [here](#).

Pipeline Schema

Figure below shows the schema of the Illumina Sequencing workflow.

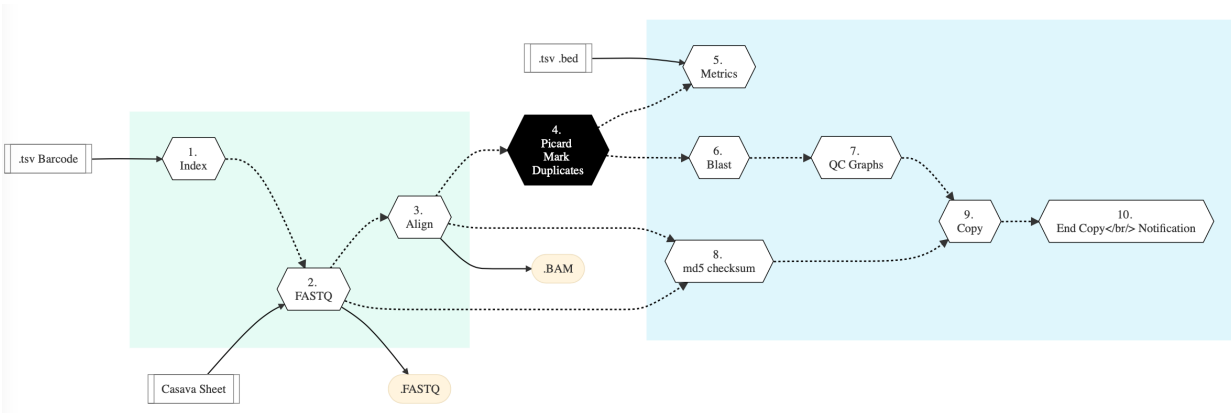
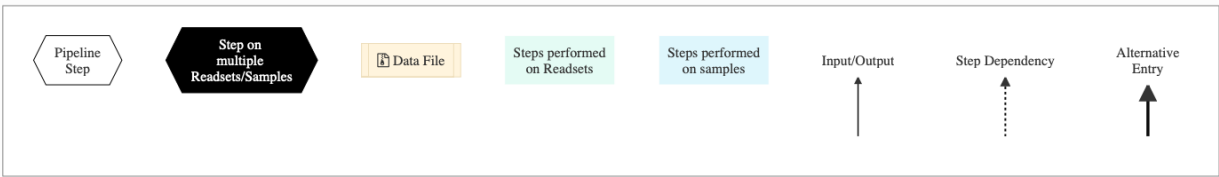


Fig. 18: Figure: Schema of Illumina Sequencing protocol



## Pipeline Steps

The table below shows various steps that are part of Illumina Sequencing Pipeline:

	<i>Illumina Sequencing Steps</i>
1.	<i>Index</i>
2.	<i>FASTQ Step</i>
3.	<i>Align</i>
4.	<i>Picard Mark Duplicates</i>
5.	<i>Metrics</i>
6.	<i>BLAST Step</i>
7.	<i>QC Graphs Step</i>
8.	<i>MD5 Step</i>
9.	<i>Copy Step</i>
10.	<i>End Copy Notification</i>

## Step Details

Following are the various steps that are part of GenPipes Illumina Sequencing genomic analysis pipeline:

### Index

This step generates a file with all the indexes found in the index-reads of the run. The input barcode file is a two columns tsv file. Each line has a `barcode_sequence` and the corresponding `barcode_name`. This file can be generated by a [LIMS](#).

The output is a tsv file named `RUNFOLDER_LANENUMBER.metrics` that will be saved in the output directory. This file has four columns, the barcode/index sequence, the index name, the number of reads and the number of reads that have passed the filter.

### FASTQ Step

This step launches FASTQ generation from Illumina raw data using BCL2FASTQ conversion software. The index base mask is calculated according to the sample and run configuration; and also according the mask parameters received (first/last index bases). The Casava sample sheet is generated with this mask. The default number of mismatches allowed in the index sequence is 1 and can be overridden with an command line argument. No demultiplexing occurs when there is only one sample in the lane.

An optional notification command can be launched to notify the start of the fastq generation with the calculated mask.

### Align

In this step, reads are aligned from the fastq file, sort the resulting .bam and create an index of that .bam. An basic alignment is performed on a sample when the `SampleRef` field of the Illumina sample sheet match one of the regexp in the configuration file and the corresponding genome (and indexes) are installed. [STAR Software](#) is used as a splice-junctions aware aligner when the sample `library_source` is cDNA or contains RNA; otherwise `BWA_mem` is used to align the reads.

For details, refer to [STAR User Guide](#).

### Picard Mark Duplicates

Runs Picard mark duplicates on the sorted BAM file.

### Metrics

This step runs a series of multiple metrics collection jobs and the output bam from mark duplicates.

- **Picard CollectMultipleMetrics:** A collection of picard metrics that runs at the same time to save on I/O. - `CollectAlignmentSummaryMetrics`, - `CollectInsertSizeMetrics`, - `QualityScoreDistribution`, - `MeanQualityByCycle`, - `CollectBaseDistributionByCycle`
- **BVATools DepthOfCoverage:** Using the specified BED Files in the sample sheet, calculate the coverage of each target region.
- **Picard CalculateHsMetrics:** Calculates a set of Hybrid Selection specific metrics from the BAM file. The bait and interval list is automatically created from the specified BED Files.

### BLAST Step

Runs BLAST on a subsample of the reads of each sample to find the 20 most frequent hits. The `runBlast.sh` tool from MUGQIC Tools is used. The number of reads to subsample can be configured by sample or for the whole lane. The output will be in the `Blast_sample` folder, under the `Unaligned` folder.

### QC Graphs Step

Generate some QC Graphics and a summary XML file for each sample using [BVATools](#).

Files are created in a 'qc' subfolder of the fastq directory. Examples of output graphic:

- Per cycle qualities, sequence content and sequence length;
- Known sequences (adapters);
- Abundant Duplicates

### MD5 Step

Create md5 checksum files for the fastq, BAM and BAI using the system 'md5sum' utility. One checksum file is created for each file.

### Copy Step

Copy processed files to another place where they can be served or loaded into a [LIMS](#). The destination folder and the command used can be set in the configuration file. An optional notification can be sent before the copy. The command used is in the configuration file.

### End Copy Notification

Send an optional notification to notify that the copy is finished. The command used is in the configuration file. This step is skipped when no command is provided.

## More Information

- [Illumina Genome Analyzer Data Analysis Software](#).

## Methylation Sequencing Pipeline

**DNA Methylation** is an important epigenetic system related to gene expression. It is involved in embryonic development, genomic imprinting, X chromosome inactivation and cell differentiation. Since methylation takes part in many normal cellular functions, aberrant methylation of DNA may be associated with a wide spectrum of human diseases, including cancer. Bisulfite sequencing approaches are currently considered a “gold standard” allowing comprehensive understanding of the methylome landscape. Whole-genome bisulfite sequencing (WGBS), provides single-base resolution of methylated cytosines across the entire genome.

GenPipes methylation sequencing workflow is adapted from the [Bismark pipeline](#). It aligns paired-end reads with [Bowtie 2](#) default mode. Duplicates are removed with [Picard](#), and methylation calls are extracted using [Bismark](#). [Wiggle](#) tracks for both read coverage and methylation profile are generated for visualization. Variant calls can be extracted from the whole-genome bisulfite sequencing (WGBS) data directly using [Bis-SNP caller](#). Bisulfite conversion rates are estimated with lambda genome or from human non-CpG methylation directly. Several metrics based on IHEC requirements are also calculated. Methylation sequencing can also process capture data if provided with a capture [BED](#) file.

Both paired-end reads as well as single-end reads are supported by this pipeline. Paired-end reading improves the ability to identify the relative positions of various reads in the genome making it much more effective than single-end reading for resolving structural rearrangements such as gene insertions, deletions or inversions and assembly of repetitive regions. Single-end reads are much less expensive in terms of resource consumption and time needed for analysis and can be used for experiments that do not require higher degrees of accuracy offered by paired-reads.

- [Introduction](#)
- [Version](#)
- [Usage](#)
- [Example Run](#)
- [Pipeline Schema](#)
- [Pipeline Steps](#)
- [Step Details](#)
- [More information](#)

## Introduction

The standard MUGQIC Methyl-Seq pipeline uses [Bismark](#) to align reads to the reference genome. Treatment and filtering of mapped reads approaches as mark duplicate reads, recalibration and sort are executed using [Picard](#) and [GATK](#). [Samtools](#) [MPILEUP](#) and [BCFtool](#) are used to produce the standard SNP and indels variants file (VCF). Additional SVN annotations mostly applicable to human samples include mappability flags, [dbSNP](#) annotation and extra information about SVN by using published databases. The [SNPeff](#) tool is used to annotate variants using an integrated database of functional predictions from multiple algorithms ([SIFT](#), [Polyphen 2](#), [LRT](#) and [MutationTaster](#), [PhyloP](#) and [GERP++](#), etc.) and to calculate the effects they produce on known genes.

A summary html report is automatically generated by the pipeline. This report contains description of the sequencing experiment as well as a detailed presentation of the pipeline steps and results. Various Quality Control (QC) summary statistics are included in the report and additional QC analysis is accessible for download directly through the report. The report includes also the main references of the software and methods used during the analysis, together with the full list of parameters that have been passed to the pipeline main script.

---

## Version

3.6.2

For the latest implementation and usage details refer to WGS or Methylation Sequencing implementation [README file](#).

---

## Usage

```
methylseq.py [-h] [--help] [-c CONFIG [CONFIG ...]] [-s STEPS]
              [-o OUTPUT_DIR] [-j {pbs,batch,daemon,slurm}] [-f]
              [--no-json] [--report] [--clean]
              [-l {debug,info,warning,error,critical}] [--sanity-check]
              [--container {wrapper, singularity} <IMAGE PATH>]
              [--genpipes_file GENPIPES_FILE]
              [-d DESIGN] [-r READSETS]
              [-v]
```

## Optional Arguments

```
-t {mugqic,mpileup,light, sv}, --type {mugqic,mpileup,light,sv}
```

Methylseq analysis **type**

```
-d DESIGN, --design DESIGN
```

design file

```
-r READSETS, --readsets READSETS
```

readset file

```
-h
```

show this help message **and** exit

```
--help
```

show detailed description of pipeline **and** steps

```
-c CONFIG [CONFIG ...], --config CONFIG [CONFIG ...]
```

config INI-style **list** of files; config parameters  
are overwritten based on files order

<code>-s STEPS, --steps STEPS</code>	step <b>range</b> e.g. '1-5', '3,6,7', '2,4-8'
<code>-o OUTPUT_DIR, --output-dir OUTPUT_DIR</code>	output directory (default: current)
<code>-j {pbs,batch,daemon,slurm}, --job-scheduler {pbs,batch,daemon,slurm}</code>	job scheduler <b>type</b> (default: slurm)
<code>-f, --force</code>	force creation of jobs even <b>if</b> up to date (default: false)
<code>--no-json</code>	do <b>not</b> create JSON file per analysed sample to track the analysis status (default: false i.e. JSON file will be created)
<code>--report</code>	create ' <b>pandoc</b> ' command to merge <b>all</b> job markdown report files <b>in</b> the given step <b>range</b> into HTML, <b>if</b> they exist; <b>if</b> --report <b>is</b> set, --job-scheduler, --force, --clean options <b>and</b> job up-to-date status are ignored (default: false)
<code>--clean</code>	create ' <b>rm</b> ' commands <b>for</b> <b>all</b> job removable files <b>in</b> the given step <b>range</b> , <b>if</b> they exist; <b>if</b> --clean <b>is</b> set, --job-scheduler, --force options <b>and</b> job up-to-date status are ignored (default: false)
<code>-l {debug,info,warning,error,critical}, --log {debug,info,warning,error,critical}</code>	log level (default: info)
<code>--sanity-check</code>	run the pipeline in 'sanity check mode' to verify all the input files needed for the pipeline to run are available on the system (default: false)
<code>--container {wrapper, singularity} &lt;IMAGE PATH&gt;</code>	run pipeline inside a container providing a container image path <b>or</b> accessible singularity hub path
<code>-v, --version</code>	show the version information <b>and</b> exit
<code>-g GENPIPES_FILE, --genpipes_file GENPIPES_FILE</code>	Commands <b>for</b> running the pipeline are output to this file pathname. The data specified to pipeline command line <b>is</b> processed <b>and</b> pipeline run commands are stored <b>in</b> GENPIPES_FILE, <b>if</b> this option <b>is</b> specified

(continues on next page)

(continued from previous page)

```
. Otherwise, the output will be redirected to stdout
. This file can be used to actually "run the
GenPipes Pipeline".
```

## Example Run

Use the following commands to execute the methylation pipeline:

```
methylseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/methylseq/methylseq.base.ini $MUGQIC_
↳ PIPELINES_HOME/pipelines/methylseq/methylseq.guillimin.ini -r readset.methylseq.txt -s_
↳ 1-14 -g methylseq.sh

bash methylseq.sh
```

**Warning:** While issuing the pipeline run command, use ``-g GENPIPES_FILE`` option (see example above) instead of using the ``> GENPIPES_FILE`` option supported by GenPipes so far, as shown below:

```
[genpipes_seq_pipeline].py -t mugqic -c $MUGQIC_PIPELINES_HOME/pipelines/[genpipes_
↳ seq_pipeline]/[genpipes_seq_pipeline].base.ini $MUGQIC_PIPELINES_HOME/pipelines/
↳ [genpipes_seq_pipeline]/[genpipes_seq_pipeline].guillimin.ini -r readset.[genpipes_
↳ seq_pipeline].txt -s 1-6 > [genpipes_seq_pipeline]_commands_mugqic.sh

bash [genpipes_seq_pipeline]_commands_mugqic.sh
```

``> scriptfile`` should be considered deprecated and ``-g scriptfile`` option is recommended instead. Please note that redirecting commands to a script ``> genpipe_script.sh`` is still supported for now. **But going forward, this mechanism might be dropped in a future GenPipes release.**

You can download the test dataset for this pipeline *in Reference section*.

## Pipeline Schema

Figure below shows the schema of the WGS or Methylation sequencing pipeline.

## Pipeline Steps

The table below shows various steps that constitute the WGS or Methylation Sequencing genomic analysis pipelines.



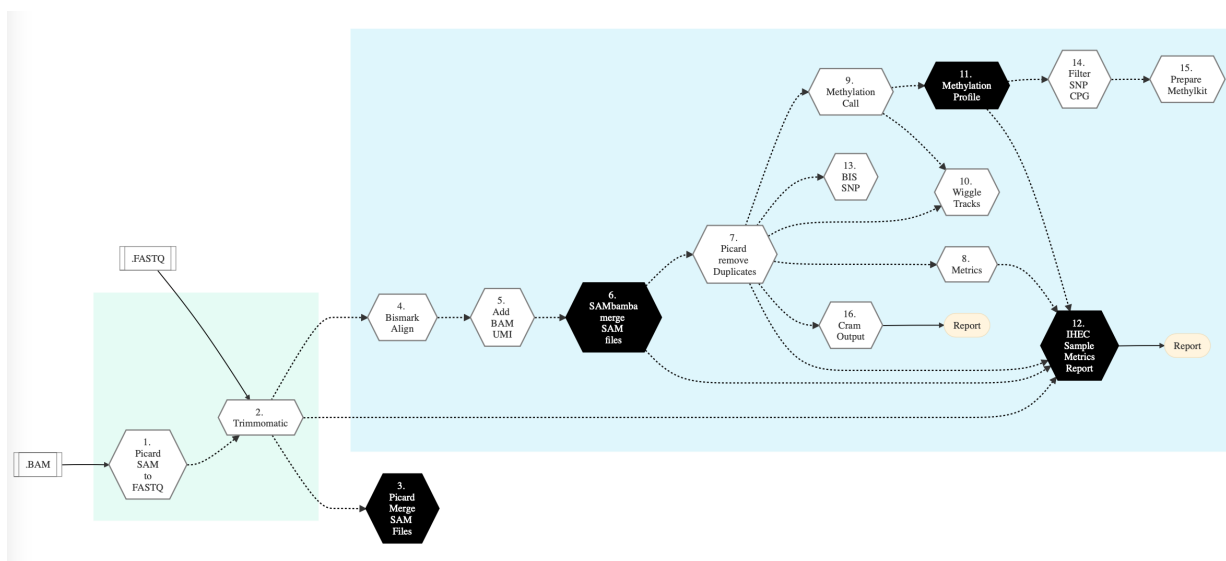
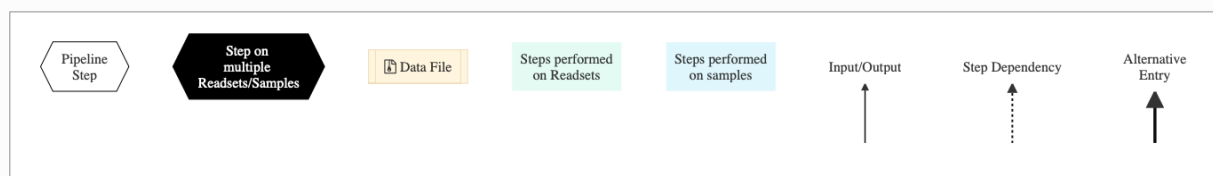


Fig. 19: Figure: Schema of WGS or Methylation Sequencing pipeline



	<i>Methylation sequencing Steps</i>	
1.	<i>Picard SAM to FastQ</i>	
2.	<i>Trimmomatic</i>	
3.	<i>Merge Trimmomatic Statistics</i>	
4.	<i>Bismark Aligner Step</i>	
5.	<i>UMI BAM Tag Processing</i>	
6.	<i>Sambamba Merge SAM Files</i>	
7.	<i>Picard remove duplicates</i>	
8.	<i>Compute Metrics</i>	
9.	<i>Methylation Call</i>	
10.	<i>Wiggle Tracks</i>	
11.	<i>Methylation Profile</i>	
12.	<i>IHEC Sample Metrics Report</i>	

## Step Details

### Picard SAM to FastQ

In this step, if FASTQ files are not already specified in the readset file, then it converts SAM/BAM files from the input readset file into FASTQ format. Otherwise, nothing is done.

### Trimmomatic

This step takes as input files:

- FASTQ files from the readset file if available
- Else, FASTQ output files from previous `picard_sam_to_fastq` conversion of BAM files

Raw reads quality trimming and removing of Illumina adapters is performed using [Trimmomatic Tool](#). If an adapter FASTA file is specified in the config file (section 'trimmomatic', param 'adapter\_fasta'), it is used first. Else, 'Adapter1' and 'Adapter2' columns from the readset file are used to create an adapter FASTA file, given then to Trimmomatic. For PAIRED\_END readsets, readset adapters are reversed-complemented and swapped, to match Trimmomatic Palindrome strategy. For SINGLE\_END readsets, only Adapter1 is used and left unchanged.

### Merge Trimmomatic Statistics

The trim statistics per readset are merged at this step.

### Bismark Aligner Step

In this step, reads are aligned using the [Bismark Tool](#).

### UMI BAM Tag Processing

This step uses [fgbio tool](#) to add read UMI tag to individual BAM files.

### Sambamba Merge SAM Files

[Sambamba](#) is a powerful filtering tool used for extracting information from SAM/BAM files.

In this step, BAM readset files are merged into one file per sample. Merge is done using [Picard Tool](#).

As input, this step takes the aligned and sorted BAM output files from previous step, *UMI BAM Tag Processing*, if available. Otherwise, it uses BAM files from the readset file as input.

### Picard remove duplicates

This step removes duplicates. Aligned reads per sample are duplicates if they have the same 5' alignment positions (for both mates in the case of paired-end reads). All but the best pair (based on alignment score) will be removed as a duplicate in the BAM file. Removing duplicates is done using [Picard Tool](#).

### Compute Metrics

Besides computing metrics, this step also generate coverage tracks per sample. Multiple metrics are computed at this stage: Number of raw reads, Number of filtered reads, Number of aligned reads, Number of duplicate reads, Median, mean and standard deviation of insert sizes of reads after alignment, percentage of bases covered at X reads (`%_bases_above_50` means the % of exons bases which have at least 50 reads) whole genome or targeted percentage of bases covered at X reads (`%_bases_above_50` means the % of exons bases which have at least 50 reads). A TDF (.tdf) coverage track is also generated at this step for easy visualization of coverage in the IGV browser.

### Methylation Call

The script in this pipeline step reads in a bisulfite read alignment file produced by the Bismark bisulfite mapper and extracts the methylation information for individual cytosines. The methylation extractor outputs result files for cytosines

in CpG, CHG and CHH context. It also outputs [bedGraph](#), a coverage file from positional methylation data and cytosine methylation report.

### Wiggle Tracks

This step generates wiggle tracks suitable for multiple browsers, to show coverage and methylation data. When using GRCh37 build of Human genome, to be compatible with the UCSC Genome Browser we only keep chromosomes 1-22, X, Y and MT, and we also rename them by prefixing “chr” to the chromosome name (e.g. “1” becomes “chr1”), and changing the mitochondrial chromosome from “MT” to “chrM”, and keeping the GRCh37 coordinates.

### Methylation Profile

This step involves generation of a CpG methylation profile by combining both forward and reverse strand Cs.

It also generates all the methylation metrics:

- CpG stats,
- [pUC19](#) CpG stats,
- lambda conversion rate,
- median CpG coverage,
- GC bias

### IHEC Sample Metrics Report

In this step, computed metrics which fit the IHEC standards, are retrieved. It also build a tsv report table for IHEC.

### Bis SNP Processing

Identifying SNPs is important for accurate quantification of methylation levels and for identification of allele-specific epigenetic events such as imprinting. This step uses bisulfite SNP caller, [Bis-SNP](#), for correctly identifying SNPs using the default high-stringency settings at an average 30× genomic coverage.

### Filter SNP CpGs

This step involves SNP CpGs filtering. The goal here is to use a known variants db (VARIANT CALL FORMAT (VCF) file provided via the ini file) to filter the CpG sites produced earlier during the step [Methylation Profile](#).

### Prepare for MethylKit Differential Analysis

This step enables comparison of methylation profile across samples. It prepares inputs that are required for the subsequent step of MethylKit Differential Analysis.

### MethylKit Differential Analysis

This step involves running of [MethylKit](#) to get DMCs & DMRs for different designed comparisons.

### CRAM Output

Generate long term storage version of the final alignment files in CRAM format. Using this function will include the original final BAM file into the removable file list.

## More information

For the latest implementation and usage details refer to WGS or Methylation Sequencing implementation [README.md](#) file.

- [DNA Methylation Detection: Bisulphite genomic sequencing analysis.](#)
- [Strategies for analyzing bisulfite sequencing data.](#)
- [Analysis and Visualization Tool for Targeted Amplicon Bisulfite Sequencing on Ion Torrent Sequencers.](#)
- [Cytosine and Guanine - CpGs.](#)

## Nanopore Pipeline

Structural Variations (SV) are genomic alterations including insertions, deletions, duplications, inversions, and translocation. They account for approximately 1% of the differences among human genomes and play a significant role in phenotypic variation and disease susceptibility.

Nanopore sequencing technology can generate long sequence reads and provides more accurate SV identification in terms of both sequencing and data analysis. For SV analysis, several new aligners and SV callers have been developed to leverage the long-read sequencing data. Assembly based approaches can also be used for SV identification. [Minimap2](#) aligner offers high speed and relatively balanced performance for calling both insertions as well as deletions.

The Nanopore sequencing technology commercialized by [Oxford Nanopore Technologies \(ONT\)](#).

- [Introduction](#)
- [Version](#)
- [Usage](#)
- [Example Run](#)
- [Pipeline Schema](#)
- [Pipeline Steps](#)
- [Step Details](#)
- [More Information](#)

---

## Introduction

The Nanopore is used to analyze long reads produced by the [Oxford Nanopore Technologies \(ONT\)](#) sequencers. Currently, the pipeline uses [Minimap2](#) to align reads to the reference genome. Additionally, it produces a QC report that includes an interactive dashboard for each readset with data from the basecalling summary file as well as the alignment. A step aligning random reads to the [NCBI nucleotide](#) database and reporting the species of the highest hits is also done as QC.

Once the QC and alignments have been produced, Picard is used to merge readsets coming from the same sample. Finally, SVIM is used to detect Structural Variants (SV) including deletions, insertions and translocations. For a full summary of the types of SVs detected, please consult the following [site](#).

The SV calls produced by SVIM are saved as VCFs for each sample, which can then be used in downstream analyses. No filtering is performed on the SV calls.

This pipeline currently does not perform base calling and requires both FASTQ and a sequencing\_summary file produced by a ONT supported basecaller (we recommend [Guppy](#)). Additionally, the testing and development of the pipeline were focused on genomics applications, and functionality has not been tested for transcriptomic or epigenomic datasets. Beyond the QC dashboards for each readset, there is currently no implemented reporting step in this pipeline.

For more information on using ONT data for structural variant detection, as well as an alternative approach, please consult [Oxford Nanopore Technologies SV Pipeline GitHub repository](#).

For information on the structure and contents of the Nanopore readset file, please consult [Nanopore Readsets details](#).

## Version

3.6.2

For the latest implementation and usage details refer to Nanopore Sequencing implementation [README file](#) file.

## Usage

```
nanopore.py [-h] [--help] [-c CONFIG [CONFIG ...]]
            [-s STEPS] [-o OUTPUT_DIR]
            [-j {pbs,batch,daemon,slurm}] [-f]
            [--no-json] [--report] [--clean]
            [-l {debug,info,warning,error,critical}]
            [--sanity-check]
            [--container {wrapper, singularity} <IMAGE FILE>]
            [--genpipes_file GENPIPES_FILE]
            [-r READSETS] [-v]
```

### Optional Arguments

`-r READSETS, --readsets READSETS`

readset file

`-h` show this help message **and** exit

`--help` show detailed description of pipeline **and** steps

`-c CONFIG [CONFIG ...], --config CONFIG [CONFIG ...]`

config INI-style **list** of files; config parameters are overwritten based on files order

`-s STEPS, --steps STEPS` step **range** e.g. '1-5', '3,6,7', '2,4-8'

`-o OUTPUT_DIR, --output-dir OUTPUT_DIR`

output directory (default: current)

-j {pbs,batch,daemon,slurm}, --job-scheduler {pbs,batch,daemon,slurm}	job scheduler <b>type</b> (default: slurm)
-f, --force	force creation of jobs even <b>if</b> up to date (default: false)
--no-json	do <b>not</b> create JSON file per analysed sample to track the analysis status (default: false i.e. JSON file will be created)
--report	create ' <b>pandoc</b> ' command to merge <b>all</b> job markdown report files <b>in</b> the given step <b>range</b> into HTML, <b>if</b> they exist; <b>if</b> --report <b>is</b> set, --job-scheduler, --force, --clean options <b>and</b> job up-to-date status are ignored (default: false)
--clean	create ' <b>rm</b> ' commands <b>for</b> <b>all</b> job removable files <b>in</b> the given step <b>range</b> , <b>if</b> they exist; <b>if</b> --clean <b>is</b> set, --job-scheduler, --force options <b>and</b> job up-to-date status are ignored (default: false)
-l {debug,info,warning,error,critical}, --log {debug,info,warning,error,critical}	log level (default: info)
--sanity-check	run the pipeline in `sanity check mode` to verify all the input files needed for the pipeline to run are available on the system (default: false)
--container {wrapper, singularity} <IMAGE PATH>	run pipeline inside a container providing a container image path <b>or</b> accessible singularity hub path
-v, --version	show the version information <b>and</b> exit
-g GENPIPES_FILE, --genpipes_file GENPIPES_FILE	Commands <b>for</b> running the pipeline are output to this file pathname. The data specified to pipeline command line <b>is</b> processed <b>and</b> pipeline run commands are stored <b>in</b> GENPIPES_FILE, <b>if</b> this option <b>is</b> specified . Otherwise, the output will be redirected to stdout . This file can be used to actually " <b>run the GenPipes Pipeline</b> ".

## Example Run

Use the following commands to execute Nanopore Sequencing Pipeline:

```
nanopore.py <Add options - info not available in README file -g nanopore_cmd.sh
bash nanopore_cmd.sh
```

**Warning:** While issuing the pipeline run command, use ``-g GENPIPES_FILE`` option (see example above) instead of using the ``> GENPIPES_FILE`` option supported by GenPipes so far, as shown below:

```
[genpipes_seq_pipeline].py -t mugqic -c $MUGQIC_PIPELINES_HOME/pipelines/[genpipes_
↪ seq_pipeline]/[genpipes_seq_pipeline].base.ini $MUGQIC_PIPELINES_HOME/pipelines/
↪ [genpipes_seq_pipeline]/[genpipes_seq_pipeline].guillimin.ini -r readset.[genpipes_
↪ seq_pipeline].txt -s 1-6 > [genpipes_seq_pipeline]_commands_mugqic.sh

bash [genpipes_seq_pipeline]_commands_mugqic.sh
```

``> scriptfile`` should be considered deprecated and ``-g scriptfile`` option is recommended instead.

Please note that redirecting commands to a script ``> genpipe_script.sh`` is still supported for now. **But going forward, this mechanism might be dropped in a future GenPipes release.**

You can download the test dataset for this pipeline [here](#).

## Pipeline Schema

The following figure shows the schema for Nanopore sequencing pipeline:

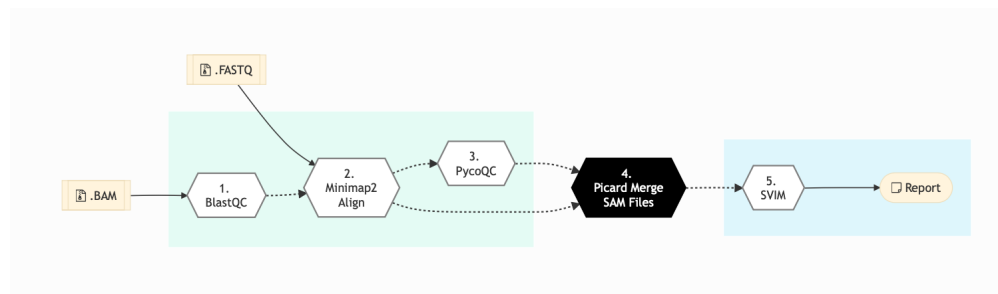
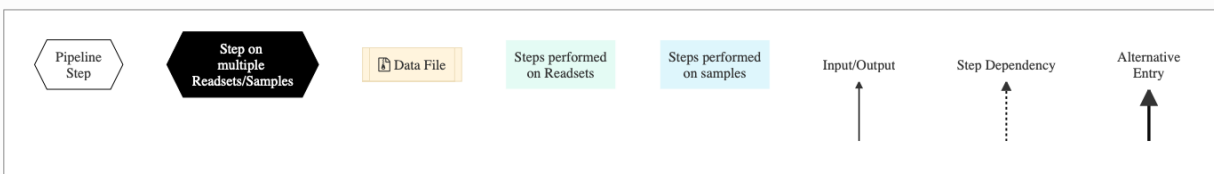


Fig. 20: Figure: Schema of Nanopore Sequencing protocol



## Pipeline Steps

The table below shows various steps that are part of Nanopore Sequencing Pipeline:

	<i>Nanopore Sequencing Steps</i>
1.	<i>BlastQC</i>
2.	<i>Minimap2 Align</i>
3.	<i>pycoQC</i>
4.	<i>Picard Merge SAM Files</i>
5.	<i>Structural Variant Identification using Mapped Long Reads</i>

---

## Step Details

Following are the various steps that are part of GenPipes Nanopore genomic analysis pipeline:

### BlastQC

In this step, [Blast-QC utility](#) is used for sequence alignment and identification. It performs a basic QC test by aligning 1000bp of randomly selected reads to the [NCBI Nucleotide Database](#) in order to detect potential contamination.

### Minimap2 Align

[Minimap2 Align Program](#) is a fast, general purpose sequencing alignment program that maps DNA and long mRNA sequences against a large reference database. It can be used for Nanopore sequencing for mapping 1kb genomic reads at an error rate of 15% (e.g., PacBio or Oxford Nanopore genomic reads), among other uses.

In this step, minimap2 to align the Fastq reads that passed the minimum QC threshold to the provided reference genome. By default, it aligns to the human genome reference (GRCh38) with Minimap2.

### pycoQC

In this step, [pycoQC Software](#) is used produce an interactive quality report based on the summary file and alignment outputs. PycoQC relies on the sequencing\_summary.txt file generated by Guppy. If needed, it can also generate a summary file from basecalled FAST5 files. PycoQC computes metrics and generates interactive QC plots for Oxford Nanopore technologies sequencing data.

### Picard Merge SAM Files

BAM readset files are merged into one file per sample in this step. Using aligned and sorted BAM output files from [Minimap2 Align](#) step, it performs the merge using [Picard](#).

### Structural Variant Identification using Mapped Long Reads

In this step, Structural Variant Identification using Mapped Long Reads ([SVIM methodology](#)), is used to perform structural variant (SV) analysis on each sample.



## More Information

- Evaluating nanopore sequencing data processing pipelines for structural variation identification.
- [Minimap2](#): Pairwise alignment for nucleotide sequences.
- Basecalling using [Guppy](#).

## RNA Sequencing Pipeline

This pipeline aligns reads with [STAR](#) 2-passes mode, assembles transcripts with [Cufflinks](#), and performs differential expression with [Cuffdiff](#). In parallel, gene-level expression is quantified using [htseq-count](#), which produces raw read counts that are subsequently used for differential gene expression with both [DESeq process](#) and [edgeR algorithms](#). Several common quality metrics (e.g., ribosomal RNA content, expression saturation estimation) are also calculated through the use of [RNA-SeQC](#) and in-house scripts. Gene Ontology terms are also tested for over-representation using [Goseq](#). Expressed short single-nucleotide variants (SNVs) and indels calling is also performed by this pipeline, which optimizes GATK best practices to reach a sensitivity of 92.8%, precision of 87.7%, and F1 score of 90.1%.

- [Introduction](#)
- [Version](#)
- [Usage](#)
- [Example Run](#)
- [Pipeline Schema](#)
- [Pipeline Steps](#)
- [Step Details](#)
- [More information](#)

## Introduction

The standard MUGQIC RNA-Seq pipeline is based on the use of the [STAR aligner](#) to align reads to the reference genome. These alignments are used during downstream analysis to determine genes and transcripts differential expression. The [Cufflinks suite](#) is used for the transcript analysis whereas [DESeq](#) and [edgeR](#) are used for the gene analysis. The RNAseq pipeline requires to provide a design file which will be used to define group comparison in the differential analyses. The differential gene analysis is followed by a Gene Ontology (GO) enrichment analysis. This analysis use the [goseq approach](#). The goseq is based on the use of non-native GO terms (see details in the section 5 of the corresponding vignette).

Finally, a summary html report is automatically generated by the pipeline at the end of the analysis. This report contains description of the sequencing experiment as well as a detailed presentation of the pipeline steps and results. Various Quality Control (QC) summary statistics are included in the report and additional QC analysis is accessible for download directly through the report. The report includes also the main references of the software tools and methods used during the analysis, together with the full list of parameters that have been passed to the pipeline main script.

See [More information](#) section below for details.

## Version

3.6.2

For the latest implementation and usage details refer to RNA Sequencing implementation [README file](#) file.

---

## Usage

```
usage: rnaseq.py [-h] [--help] [-c CONFIG [CONFIG ...]] [-s STEPS]
                [-o OUTPUT_DIR] [-j {pbs,batch,daemon,slurm}] [-f]
                [--no-json] [--report] [--clean]
                [-l {debug,info,warning,error,critical}] [--sanity-check]
                [--container {wrapper, singularity} <IMAGE PATH>]
                [--genpipes_file GENPIPES_FILE]
                [-d DESIGN]
                [-t {cufflinks,stringtie}] [-r READSETS] [-v]
```

### Optional Arguments

`-t {cufflinks,stringtie}, --type {cufflinks,stringtie}`

Type of RNA-seq assembly method (default stringtie,  
faster than cufflinks)

`-d DESIGN, --design DESIGN`

design file

`-r READSETS, --readsets READSETS`

readset file

`-h` show this help message **and** exit

`--help` show detailed description of pipeline **and** steps

`-c CONFIG [CONFIG ...], --config CONFIG [CONFIG ...]`

config INI-style **list** of files; config parameters  
are overwritten based on files order

`-s STEPS, --steps STEPS` step **range** e.g. '1-5', '3,6,7', '2,4-8'

`-o OUTPUT_DIR, --output-dir OUTPUT_DIR`

output directory (default: current)

<code>-j {pbs,batch,daemon,slurm}, --job-scheduler {pbs,batch,daemon,slurm}</code>	job scheduler <b>type</b> (default: slurm)
<code>-f, --force</code>	force creation of jobs even <b>if</b> up to date (default: false)
<code>--no-json</code>	do <b>not</b> create JSON file per analysed sample to track the analysis status (default: false i.e. JSON file will be created)
<code>--report</code>	create ' <b>pandoc</b> ' command to merge <b>all</b> job markdown report files <b>in</b> the given step <b>range</b> into HTML, <b>if</b> they exist; <b>if</b> <code>--report <b>is</b> set</code> , <code>--job-scheduler</code> , <code>--force</code> , <code>--clean</code> options <b>and</b> job up-to-date status are ignored (default: false)
<code>--clean</code>	create ' <b>rm</b> ' commands <b>for</b> <b>all</b> job removable files <b>in</b> the given step <b>range</b> , <b>if</b> they exist; <b>if</b> <code>--clean <b>is</b> set</code> , <code>--job-scheduler</code> , <code>--force</code> options <b>and</b> job up-to-date status are ignored (default: false)
<code>-l {debug,info,warning,error,critical}, --log {debug,info,warning,error,critical}</code>	log level (default: info)
<code>--sanity-check</code>	run the pipeline in 'sanity check mode' to verify all the input files needed for the pipeline to run are available on the system (default: false)
<code>--container {wrapper, singularity} &lt;IMAGE PATH&gt;</code>	run pipeline inside a container providing a container image path <b>or</b> accessible singularity hub path
<code>-v, --version</code>	show the version information <b>and</b> exit
<code>-g GENPIPES_FILE, --genpipes_file GENPIPES_FILE</code>	Commands <b>for</b> running the pipeline are output to this file pathname. The data specified to pipeline command line <b>is</b> processed <b>and</b> pipeline run commands are stored <b>in</b> GENPIPES_FILE, <b>if</b> this option <b>is</b> specified . Otherwise, the output will be redirected to stdout . This file can be used to actually " <b>run the GenPipes Pipeline</b> ".

## Example Run

You can download [RNA Sequencing Pipeline test dataset](#) and use the following command to execute the RNA Sequencing genomics pipeline:

```
rnaseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/rnaseq/rnaseq.base.ini $MUGQIC_PIPELINES_
↪HOME/pipelines/rnaseq/rnaseq.cedar.ini workshop.ini -r readset.rnaseq.txt -d design.
↪rnaseq.txt -s 1-24 -j slurm -g commands.txt

bash commands.txt
```

**Warning:** While issuing the pipeline run command, use ``-g GENPIPES_FILE`` option (see example above) instead of using the ``> GENPIPES_FILE`` option supported by GenPipes so far, as shown below:

```
[genpipes_seq_pipeline].py -t mugqic -c $MUGQIC_PIPELINES_HOME/pipelines/[genpipes_
↪seq_pipeline]/[genpipes_seq_pipeline].base.ini $MUGQIC_PIPELINES_HOME/pipelines/
↪[genpipes_seq_pipeline]/[genpipes_seq_pipeline].guillimin.ini -r readset.[genpipes_
↪seq_pipeline].txt -s 1-6 > [genpipes_seq_pipeline]_commands_mugqic.sh

bash [genpipes_seq_pipeline]_commands_mugqic.sh
```

``> scriptfile`` should be considered deprecated and ``-g scriptfile`` option is recommended instead. Please note that redirecting commands to a script ``> genpipe_script.sh`` is still supported for now. **But going forward, this mechanism might be dropped in a future GenPipes release.**

This set of commands is meant for running GenPipes on C3 data center. For more details, you can refer to [GenPipes RNA Sequencing Workshop 2018 presentation](#).

---

## Pipeline Schema

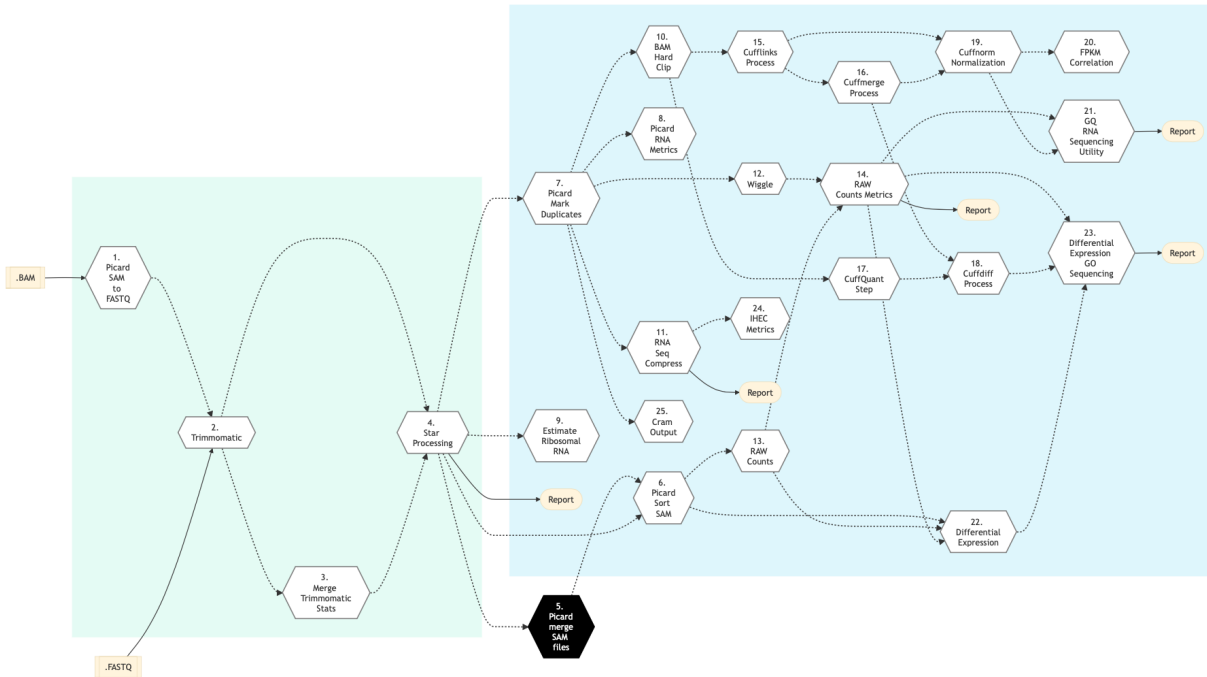
Figure below shows the schema of the RNA sequencing protocol (cufflinks).

The following figure shows the schema of the RNA sequencing protocol (stringtie).

---

## Pipeline Steps

The table below lists various steps of GenPipes RNA Sequencing Pipeline:



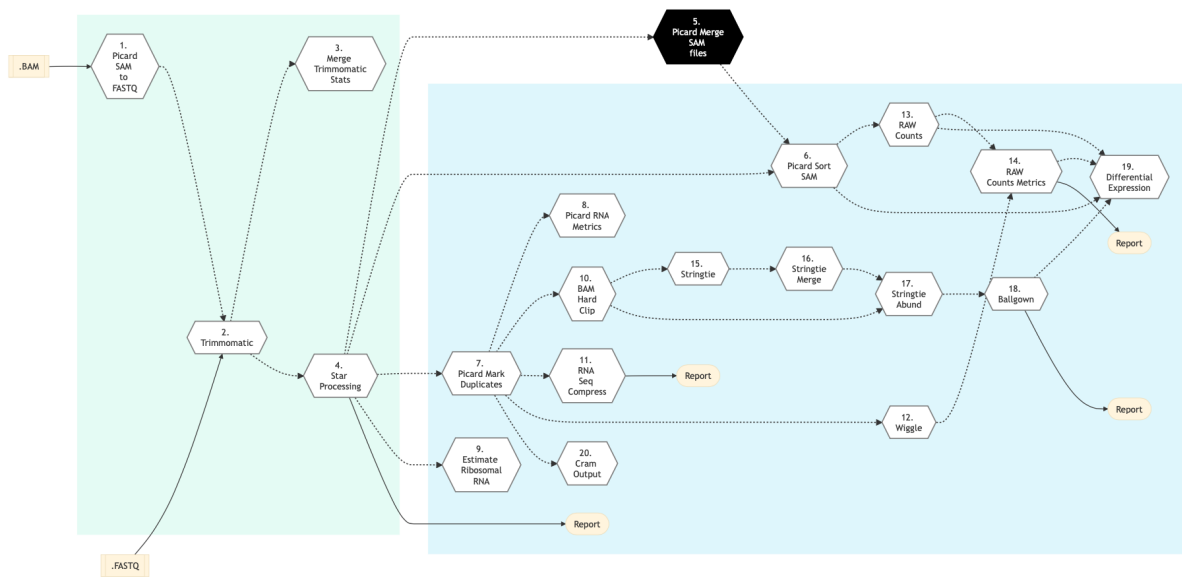
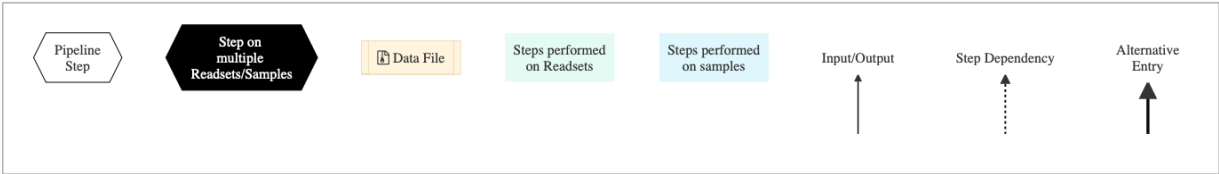


Fig. 22: Figure: Schema of RNA Sequencing pipeline (Stringtie)



	<i>RNA Sequencing (Cufflinks)</i>	<i>RNA Sequencing (Stringtie)</i>
1.	<i>Picard SAM to FastQ</i>	<i>Picard SAM to FastQ</i>
2.	<i>Trimmomatic</i>	<i>Trimmomatic</i>
3.	<i>Merge Trimmomatic Stats</i>	<i>Merge Trimmomatic Stats</i>
4.	<i>Star Processing</i>	<i>Star Processing</i>
5.	<i>Picard Merge SAM Files</i>	<i>Picard Merge SAM Files</i>
6.	<i>Picard Sort SAM</i>	<i>Picard Sort SAM</i>
7.	<i>Picard Mark Duplicates</i>	<i>Picard Mark Duplicates</i>
8.	<i>Picard RNA Metrics</i>	<i>Picard RNA Metrics</i>
9.	<i>Estimate Ribosomal RNA</i>	<i>Estimate Ribosomal RNA</i>
10.	<i>BAM Hard Clip</i>	<i>BAM Hard Clip</i>
11.	<i>RNA Seq Compress</i>	<i>RNA Seq Compress</i>
12.	<i>Wiggle</i>	<i>Wiggle</i>
13.	<i>Raw Counts</i>	<i>Raw Counts</i>
14.	<i>Raw Counts Metrics</i>	<i>Raw Counts Metrics</i>
15.	<i>Cufflinks Process</i>	<i>Stringtie</i>
16.	<i>Cuffmerge Process</i>	<i>Stringtie Merge</i>
17.	<i>Cuffquant Step</i>	<i>Stringtie Assemble Transcriptome</i>
18.	<i>Cuffdiff Process</i>	<i>Ballgown Gene Expression</i>
19.	<i>Cuffnorm Normalization</i>	<i>Differential Expression</i>
20.	<i>FPKM Correlation</i>	<i>CRAM Output</i>
21.	<i>GQ RNA Sequencing Utility</i>	
<b>1.6. GenPipes User Guide</b>		<b>143</b>
22.	<i>Differential Expression</i>	
23.	<i>Differential Expression GO se-</i>	

## Step Details

### Picard SAM to FastQ

In this step, if FASTQ files are not already specified in the readset file, then it converts SAM/BAM files from the input readset file into FASTQ format. Otherwise, nothing is done.

#### Trimmomatic

This step takes as input files:

- FASTQ files from the readset file if available
- Else, FASTQ output files from previous `picard_sam_to_fastq` conversion of BAM files

Raw reads quality trimming and removing of Illumina adapters is performed using [Trimmomatic tool](#). If an adapter FASTA file is specified in the config file (section 'trimmomatic', param 'adapter\_fasta'), it is used first. Else, 'Adapter1' and 'Adapter2' columns from the readset file are used to create an adapter FASTA file, given then to Trimmomatic. For PAIRED\_END readsets, readset adapters are reversed-complemented and swapped, to match Trimmomatic Palindrome strategy. For SINGLE\_END readsets, only Adapter1 is used and left unchanged.

### Merge Trimmomatic Stats

The trim statistics per readset are merged at this step.

### Star Processing

The filtered reads are aligned to a reference genome. The alignment is done per readset of sequencing using the STAR software. It generates a Binary Alignment Map file (.bam).

This step takes as input files:

- Trimmed FASTQ files if available
- Else, FASTQ files from the readset file if available
- Else, FASTQ output files from previous `picard_sam_to_fastq` conversion of BAM files

### Picard Merge SAM Files

BAM readset files are merged into one file per sample. Merge is done using [Picard Tool](#).

### Picard Sort SAM

The alignment file is reordered (QueryName) using [Picard Tool](#). The QueryName-sorted BAM files will be used to determine raw read counts.

### Picard Mark Duplicates

Mark duplicates. Aligned reads per sample are duplicates if they have the same 5' alignment positions (for both mates in the case of paired-end reads). All but the best pair (based on alignment score) will be marked as a duplicate in the BAM file. Marking duplicates is done using [Picard package](#).

### Picard RNA Metrics

Computes a series of quality control metrics using both `CollectRnaSeqMetrics` and `CollectAlignmentSummaryMetrics` functions metrics are collected using [Picard package](#).

### Estimate Ribosomal RNA

This step uses readset BAM files and `bwa mem` to align reads on the rRNA reference fasta and count the number of read mapped. The filtered reads are aligned to a reference fasta file of ribosomal sequence. The alignment is done per



sequencing readset. The alignment software used is [BWA package](#) with algorithm: bwa mem. BWA output BAM files are then sorted by coordinate using [Picard package](#).

### **BAM Hard Clip**

Generate a hardclipped version of the BAM for the toxedo suite which does not support this official SAM feature.

### **RNA Seq Compress**

Computes a series of quality control metrics using [RNA SeQC processing](#).

### **Wiggle**

Generate wiggle tracks suitable for multiple browsers.

### **Raw Counts**

Count reads in feature using [HT Seq Count](#).

### **Raw Counts Metrics**

Create rawcount matrix, zip the wiggle tracks and create the saturation plots based on standardized read counts.

### **Cufflinks Process**

Compute RNA-Seq data expression using [cufflinks package](#). Warning: It needs to use a hard clipped bam file while Tuxedo tools do not support official soft clip SAM format.

### **Cuffmerge Process**

Merge assemblies into a master transcriptome reference using [cuffmerge package](#).

### **Cuffquant Step**

Compute expression profiles (abundances.cxb) using [cuffquant](#). Warning: It needs to use a hard clipped bam file while Tuxedo tools do not support official soft clip SAM format.

### **Cuffdiff Process**

[Cuffdiff package](#) is used to calculate differential transcript expression levels and test them for significant differences.

### **Cuffnorm Normalization**

This step performs global normalization of RNA-Sequence expression levels using [Cuffnorm algorithm](#).

### **FPKM Correlation**

Compute the pearson correlation matrix of gene and transcripts FPKM. FPKM data are those estimated by cuffnorm.

### **GQ RNA Sequencing Utility**

Exploratory analysis using the [gqSeqUtils R package](#).

### **Differential Expression**

Performs differential gene expression analysis using [DESEQ package](#) and [EDGER package](#). Merge the results of the analysis in a single csv file.

### **Differential Expression GO sequencing**

Gene Ontology analysis for RNA-Seq using the [Bioconductor's R package goseq](#). Generates GO annotations for differential gene expression analysis.

### **IHEC Metrics**

Generate IHEC's standard metrics.

### **Stringtie**

Assemble transcriptome using [Stringtie assembler](#).

### Stringtie Assemble Transcriptome

This step assembles transcriptome and compute RNA-seq expression using [Stringtie assembler](#).

### Stringtie Merge

Merge assemblies into a master transcriptome reference using [Stringtie assembler](#).

### Ballgown Gene Expression

[Ballgown tool](#) is used to calculate differential transcript and gene expression levels and test them for significant differences.

### CRAM Output

Generate long term storage version of the final alignment files in CRAM format. Using this function will include the original final BAM file into the removable file list.

---

## More information

For the latest implementation and usage details refer to [README.md](#).

- [RNA Sequencing Workshop 2019 \(Slides\)](#)
- 

## De-Novo RNA Sequencing Pipeline

RNA Sequencing is a technique that allows [transcriptome studies](#) based on high throughput next-generation gene sequencing (NGS). De novo sequencing refers to sequencing a novel genome where there is no reference sequence available for alignment. Sequence reads are assembled as contigs, and the coverage quality of de novo sequence data depends on the size and continuity of the contigs (i.e., the number of gaps in the data).

De-Novo RNASeq pipeline is adapted from the [Trinity-Trinotate suggested workflow](#). It reconstructs transcripts from short reads, predicts proteins, and annotates, leveraging several databases. Quantification is computed using [RSEM Tool](#), and differential expression is tested in a manner identical to the RNA-seq pipeline. We observed that the default parameters of the Trinity suite are very conservative, which could result in the loss of low-expressed but biologically relevant transcripts. To provide the most complete set of transcripts, the pipeline was designed with lower stringency during the assembly step in order to produce every possible transcript and not miss low-expressed messenger RNA. A stringent filtration step is included afterward in order to provide a set of transcripts that make sense biologically.

- [Introduction](#)
  - [Version](#)
  - [Usage](#)
  - [Example Run](#)
  - [Pipeline Schema](#)
  - [Pipeline Steps](#)
  - [Step Details](#)
  - [More information](#)
-

## Introduction

The standard MUGQIC RNA-Seq *De Novo* Assembly pipeline uses the [Trinity](#) software suite to reconstruct transcriptomes from RNA-Seq data without using any reference genome or transcriptome. First, reads are trimmed with [Trimmomatic](#) and normalized in order to reduce memory requirement and decrease assembly runtime, using the Trinity normalization utility inspired by the [Diginorm](#) algorithm.

Then, the transcriptome is assembled on normalized reads using the Trinity assembler. Trinity creates a Trinity.fasta file with a list of contigs representing the transcriptome isoforms. Those transcripts are grouped in components mostly representing genes. Components and transcripts are functionally annotated using the [Trinotate](#) suite. Gene abundance estimation for each sample has been performed using [RSEM Tool](#) (RNA-Seq by Expectation-Maximization). Differential gene expression analysis is performed using [DESeq2](#) and [edgeR](#) Bioconductor packages.

The [DESeq2](#) and [edgeR](#) methods model **count data** by a negative binomial distribution. The parameters of the distribution (mean and dispersion) are estimated from the data, i.e. from the read counts in the input files. Both methods compute a measure of read abundance, i.e. expression level (called *base mean* or *mean of normalized counts* in [DESeq2](#), and *concentration* in [edgeR](#)) for each gene and apply a hypothesis test to each gene to evaluate differential expression. In particular, both methods determine a p-value and a log2 fold change (in expression level) for each gene. The Log2 FC of [edgeR](#) is reported in the differential gene results file, one file per design.

The log2fold change is the logarithm (to basis 2) of the fold change condition from condition A to B (mutation or treatment are the most common conditions). A “fold change” between conditions A and B at a gene or transcript is normally computed as the ratio at gene or transcript of the base mean of scaled counts for condition B to the base mean of scaled counts for condition A. Counts are scaled by a size factor in a step called normalization (if the counts of non-differentially expressed genes in one sample are, on average, twice as high as in another, the size factor for the first sample should be twice that of the other sample). Each column of the count table is then divided by the size factor for this column and the count values are brought to a common scale, making them comparable. See the [edgeR vignette](#) for additional information on normalization approaches used in the pipeline.

The differential gene analysis is followed by a Gene Ontology (GO) enrichment analysis. This analysis use the [goseq approach](#). The [goseq](#) is based on the use of non-native GO terms resulting from trinotate annotations (see details in the section 5 of the [corresponding vignette](#)).

Thus a high quality contigs assembly is created by extracting all transcripts having a functional annotation as defined by trinotate, the Top BLASTX hit and TmHMM annotations are used by default.

Finally, different exploratory data analysis (EDA) techniques are applied to filtered isoforms expression levels. Main goals of expression level EDA are the detection of outliers, potential mislabeling, to explore the homogeneity of biological replicates and to appreciate the global effects of the different experimental variables.

An HTML summary report is automatically generated by the pipeline. This report contains description of the sequencing experiment as well as a detailed presentation of the pipeline steps and results. Various Quality Control (QC) summary statistics are included in the report and additional QC analysis is accessible for download directly through the report. The report includes also the main references of the software and methods used during the analysis, together with the full list of parameters that have been passed to the pipeline main script.

## Version

3.6.2

For the latest implementation and usage details refer to RNA Sequencing implementation [README file](#) file.

---

## Usage

```
usage: rnaseq_denovo_assembly.py [-h] [--help] [-c CONFIG [CONFIG ...]]
                                [-s STEPS] [-o OUTPUT_DIR]
                                [-j {pbs,batch,daemon,slurm}] [-f]
                                [--no-json] [--report] [--clean]
                                [-l {debug,info,warning,error,critical}]
                                [--sanity-check]
                                [--container {wrapper, singularity} <IMAGE PATH>]
                                [--genpipes_file GENPIPES_FILE]
                                [-d DESIGN] [-t {cufflinks,stringtie}]
                                [-r READSETS] [-v]
```

### Optional Arguments

`-t {cufflinks,stringtie}, --type {cufflinks,stringtie}`

Type of RNA-seq Denovo assembly method  
(default stringtie, faster than cufflinks)

`-d DESIGN, --design DESIGN`

design file

`-r READSETS, --readsets READSETS`

readset file

`-h` show this help message **and** exit

`--help` show detailed description of pipeline **and** steps

`-c CONFIG [CONFIG ...], --config CONFIG [CONFIG ...]`

config INI-style **list** of files; config parameters  
are overwritten based on files order

`-s STEPS, --steps STEPS` step **range** e.g. '1-5', '3,6,7', '2,4-8'

`-o OUTPUT_DIR, --output-dir OUTPUT_DIR`

output directory (default: current)

<code>-j {pbs,batch,daemon,slurm}, --job-scheduler {pbs,batch,daemon,slurm}</code>	job scheduler <b>type</b> (default: slurm)
<code>-f, --force</code>	force creation of jobs even <b>if</b> up to date (default: false)
<code>--no-json</code>	do <b>not</b> create JSON file per analysed sample to track the analysis status (default: false i.e. JSON file will be created)
<code>--report</code>	create ' <b>pandoc</b> ' command to merge <b>all</b> job markdown report files <b>in</b> the given step <b>range</b> into HTML, <b>if</b> they exist; <b>if</b> <code>--report <b>is</b> set</code> , <code>--job-scheduler</code> , <code>--force</code> , <code>--clean</code> options <b>and</b> job up-to-date status are ignored (default: false)
<code>--clean</code>	create ' <b>rm</b> ' commands <b>for</b> <b>all</b> job removable files <b>in</b> the given step <b>range</b> , <b>if</b> they exist; <b>if</b> <code>--clean <b>is</b> set</code> , <code>--job-scheduler</code> , <code>--force</code> options <b>and</b> job up-to-date status are ignored (default: false)
<code>-l {debug,info,warning,error,critical}, --log {debug,info,warning,error,critical}</code>	log level (default: info)
<code>--sanity-check</code>	run the pipeline in 'sanity check mode' to verify all the input files needed for the pipeline to run are available on the system (default: false)
<code>--container {wrapper, singularity} &lt;IMAGE PATH&gt;</code>	run pipeline inside a container providing a container image path <b>or</b> accessible singularity hub path
<code>-v, --version</code>	show the version information <b>and</b> exit
<code>-g GENPIPES_FILE, --genpipes_file GENPIPES_FILE</code>	Commands <b>for</b> running the pipeline are output to this file pathname. The data specified to pipeline command line <b>is</b> processed <b>and</b> pipeline run commands are stored <b>in</b> GENPIPES_FILE, <b>if</b> this option <b>is</b> specified . Otherwise, the output will be redirected to stdout . This file can be used to actually " <b>run the GenPipes Pipeline</b> ".

## Example Run

Use the following commands to execute the *De Novo* sequencing pipeline:

```
rnaseq_denovo_assembly.py -c $MUGQIC_PIPELINES_HOME/pipelines/rnaseq_denovo_assembly/
↪rnaseq_denovo_assembly.base.ini $MUGQIC_PIPELINES_HOME/pipelines/rnaseq_denovo_
↪assembly/rnaseq_denovo_assembly.guillimin.ini -r readset.rnaseq.txt -d design.rnaseq.
↪txt -s 1-23 -g rnaseqDeNovoCommands.sh

bash rnaseqDeNovoCommands.sh
```

**Warning:** While issuing the pipeline run command, use ``-g GENPIPES_FILE`` option (see example above) instead of using the ``> GENPIPES_FILE`` option supported by GenPipes so far, as shown below:

```
[genpipes_seq_pipeline].py -t mugqic -c $MUGQIC_PIPELINES_HOME/pipelines/[genpipes_
↪seq_pipeline]/[genpipes_seq_pipeline].base.ini $MUGQIC_PIPELINES_HOME/pipelines/
↪[genpipes_seq_pipeline]/[genpipes_seq_pipeline].guillimin.ini -r readset.[genpipes_
↪seq_pipeline].txt -s 1-6 > [genpipes_seq_pipeline]_commands_mugqic.sh

bash [genpipes_seq_pipeline]_commands_mugqic.sh
```

``> scriptfile`` should be considered deprecated and ``-g scriptfile`` option is recommended instead.

Please note that redirecting commands to a script ``> genpipe_script.sh`` is still supported for now. **But going forward, this mechanism might be dropped in a future GenPipes release.**

You can download the test dataset for this pipeline [here](#).

---

## Pipeline Schema

Figure below shows the schema of RNA Sequencing *De Novo* Assembly pipeline.

---

## Pipeline Steps

The table below lists various steps that constitute the RNA Sequencing *De Novo* Assembly.



	RNA Sequencing De Novo Assembly Steps
1.	<i>Picard SAM to FastQ</i>
2.	<i>Trimmomatic Step</i>
3.	<i>Merge Trimmomatic Stats</i>
4.	<i>InSilico Read Normalization of Readsets</i>
5.	<i>InSilico Read Normalization (All)</i>
6.	<i>Trinity Step</i>
7.	<i>Exonerate FASTA Split</i>
8.	<i>BLASTX Trinity UniProt</i>
9.	<i>BLASTX Trinity UniProt Merge</i>
10.	<i>TransDecoder Step</i>
11.	<i>HMMER Biosequence Analysis Step</i>
12.	<i>RNAmmer Method</i>
13.	<i>BLAST Transdecoder UniProt</i>
14.	<i>SignalP Method</i>
15.	<i>TMHMM Method</i>
16.	<i>Trinotate Step</i>
17.	<i>Align and estimate Abundance Prep Reference</i>
18.	<i>Align and estimate Abundance</i>
19.	<i>Exploratory Analysis with gqSeqUtils R package</i>
20.	<i>Differential Expression</i>
21.	<i>Filter Annotated Components</i>
22.	<i>Exploratory Analysis with subset of filtered transcripts</i>
23.	<i>GOSEQ using filtered transcripts</i>



## Step Details

### Picard SAM to FastQ

Convert SAM/BAM files from the input readset file into FASTQ format if FASTQ files are not already specified in the readset file. Do nothing otherwise.

### Trimmomatic

Raw reads quality trimming and removing of Illumina adapters is performed using [Trimmomatic Tool](#). If an adapter FASTA file is specified in the config file (section 'trimmomatic', param 'adapter\_fasta'), it is used first. Else, 'Adapter1' and 'Adapter2' columns from the readset file are used to create an adapter FASTA file, given then to Trimmomatic. For PAIRED\_END readsets, readset adapters are reversed-complemented and swapped, to match Trimmomatic Palindrome strategy. For SINGLE\_END readsets, only Adapter1 is used and left unchanged.

This step takes as input files:

- FASTQ files from the readset file if available
- Else, FASTQ output files from previous picard\_sam\_to\_fastq conversion of BAM files

### Merge Trimmomatic Stats

The trim statistics per readset are merged at this step.

### InSilico Read Normalization of Readsets

Normalize each readset, using the Trinity normalization utility.

### InSilico Read Normalization (All)

Merge all normalized readsets together and normalize the result, using the Trinity normalization utility.

### Trinity

Create a de novo assembly from normalized readsets using [Trinity tool](#).

### Exonerate FASTA Split

Split the Trinity assembly FASTA into chunks for further parallel BLAST annotations.

### BLASTX Trinity UniProt

Annotate Trinity FASTA chunks with Swiss-Prot and UniRef databases using [BLAST Tool](#).

### BLASTX Trinity UniProt Merge

Merge blastx Swiss-Prot and UniRef chunks results.

### TransDecoder Step

Identifies candidate coding regions within transcript sequences using [Transdecoder](#).

### HMMER Biosequence Analysis Step

Identifies protein domains using [HMMER Biosequence Analysis](#)

### RNAmmer Method

Identify potential rRNA transcripts using [RNAmmer Transcriptome](#).

### BLAST Transdecoder UniProt

Search Transdecoder-predicted coding regions for sequence homologies on UniProt using [BLASTP](#).

### SignalP Method

Predict the presence of signal peptides and the location of their cleavage sites in proteins using [SignalP](#).

### TMHMM Method

This step involves prediction of transmembrane helices regions in proteins using [TMHMM](#).

### Trinotate

This step performs transcriptome functional annotation and analysis using [Trinotate Annotation suite](#). If functional annotation data is integrated into a SQLite database and a whole annotation report is created.

### Align and estimate Abundance Prep Reference

Index Trinity FASTA file for further abundance estimation using [Trinity align\\_and\\_estimate\\_abundance.pl](#) utility.

### Align and estimate Abundance

Estimate transcript abundance using RNA-Seq by Expectation-Maximization (RSEM) via [Trinity align\\_and\\_estimate\\_abundance.pl](#) utility.

### Exploratory Analysis with gqSeqUtils R package

This step performs exploratory analysis using the gqSeqUtils R package.

### Differential Expression

This step performs differential gene expression analysis using [DESeq package](#) and [edgeR package](#). Merge the results of the analysis in a single csv file. Also, this pipeline step performs Gene Ontology analysis for RNA-Seq denovo Assembly using the Bioconductor's R package [goseq](#). It generates GO annotations for differential genes and isoforms expression analysis, based on associated GOTERMS generated by [Trinotate Step](#) earlier.

### Filter Annotated Components

This step filters high quality contigs based on values in trinotate annotations. It also recreates a high quality contigs fasta file and run Assembly statistics using the gqSeqUtils R package.

### Exploratory Analysis with subset of filtered transcripts

In this step, exploratory analysis is performed by the gqSeqUtils R package using a subset of filtered transcripts.

### GOSEQ using filtered transcripts

In this step, differential expression and [goseq](#) analysis is performed based on filtered transcripts and genes.

---

## More information

You can find more information about RNA Sequencing *De Novo* Assembly Pipeline in the following references:

- Grabherr MG, Haas BJ, Yassour M, et al. Full-length transcriptome assembly from RNA-Seq data without a reference genome - [Trinity-Trinotate](#).
- Chin CS, Alexander DH, Marks P, et al. Non-hybrid, finished microbial genome assemblies from long-read SMRT sequencing data - [suggested workflow](#).
- Trinity RNA sequencing utilities [Workshop Slides](#).

## RNA Sequencing (Light) Pipeline

This is a lightweight RNA Sequencing Expression analysis pipeline based on [Kallisto technique](#). It is used for quick Quality Control (QC) in gene sequencing studies.

- [Introduction](#)
- [Version](#)
- [Usage](#)
- [Example Run](#)
- [Pipeline Schema](#)
- [Pipeline Steps](#)
- [More Information](#)

### Introduction

The central computational problem in RNA-seq remains the efficient and accurate assignment of short sequencing reads to the transcripts they originated from and using this information to infer gene expressions. Conventionally, read assignment is carried out by aligning sequencing reads to a reference genome, such that relative gene expressions can be inferred by the alignments at annotated gene loci. These alignment-based methods are conceptually simple, but the read-alignment step can be time-consuming and computationally intensive.

Alignment-free RNA quantification tools have significantly increased the speed of RNA-seq analysis. The alignment-free pipelines are orders of magnitude faster than alignment-based pipelines, and they work by breaking sequencing reads into k-mers and then performing fast matches to pre-indexed transcript databases. To achieve fast transcript quantification without compromising quantification accuracy, different sophisticated algorithms were implemented in addition to k-mer counting, such as pseudo-alignments by [Kallisto technique](#) and quasi-mapping along with GC and sequence-bias corrections using [Salmon](#).

RNA Sequencing Light is a lightweight pipeline that performs quick QC and removes a major computation bottleneck in RNA Sequence analysis. Kallisto is two orders of magnitude faster than previous approaches and achieves similar accuracy. Kallisto pseudo-aligns reads to a reference, producing a list of transcripts that are compatible with each read while avoiding alignment of individual bases. For details refer to *More Information* section below.

### Version

3.1.5

### Usage

```
rnaseq_light.py [-h] [--help] [-c CONFIG [CONFIG ...]] [-s STEPS]
                [-o OUTPUT_DIR] [-j {pbs,batch,daemon,slurm}] [-f]
                [--no-json] [--report] [--clean]
                [-l {debug,info,warning,error,critical}] [-d DESIGN]
                [--sanity-check]
                [--container {wrapper, singularity} <IMAGE PATH>]
                [--genpipes_file GENPIPES_FILE] [-d DESIGN]
                [-t {cufflinks,stringtie}] [-r READSETS] [-v]
```

### Optional Arguments

<code>-t {cufflinks,stringtie}, --type {cufflinks,stringtie}</code>	Type of RNA-seq (light) assembly method (default stringtie, faster than cufflinks)
<code>-d DESIGN, --design DESIGN</code>	design file
<code>-r READSETS, --readsets READSETS</code>	readset file
<code>-h</code>	show this help message <b>and</b> exit
<code>--help</code>	show detailed description of pipeline <b>and</b> steps
<code>-c CONFIG [CONFIG ...], --config CONFIG [CONFIG ...]</code>	config INI-style <b>list</b> of files; config parameters are overwritten based on files order
<code>-s STEPS, --steps STEPS</code>	step <b>range</b> e.g. '1-5', '3,6,7', '2,4-8'
<code>-o OUTPUT_DIR, --output-dir OUTPUT_DIR</code>	output directory (default: current)
<code>-j {pbs,batch,daemon,slurm}, --job-scheduler {pbs,batch,daemon,slurm}</code>	job scheduler <b>type</b> (default: slurm)
<code>-f, --force</code>	force creation of jobs even <b>if</b> up to date (default: false)
<code>--no-json</code>	do <b>not</b> create JSON file per analysed sample to track the analysis status (default: false i.e. JSON file will be created)
<code>--report</code>	create ' <b>pandoc</b> ' command to merge <b>all</b> job markdown report files <b>in</b> the given step <b>range</b> into HTML, <b>if</b> they exist; <b>if</b> --report <b>is set</b> , --job-scheduler, --force, --clean options <b>and</b> job up-to-date status are ignored (default: false)
<code>--clean</code>	create ' <b>rm</b> ' commands <b>for all</b> job removable files <b>in</b> the given step <b>range</b> , <b>if</b> they exist; <b>if</b> --clean <b>is</b> <b>set</b> , --job-scheduler, --force options <b>and</b> job up-to- date status are ignored (default: false)

```
-l {debug,info,warning,error,critical}, --log {debug,info,warning,error,critical}
```

log level (default: info)

```
--sanity-check      run the pipeline in `sanity check mode` to verify
                    all the input files needed for the pipeline to run
                    are available on the system (default: false)
```

```
--container {wrapper, singularity} <IMAGE PATH>
```

run pipeline inside a container providing a container image path **or** accessible singularity hub path

```
-v, --version      show the version information and exit
```

```
-g GENPIPES_FILE, --genpipes_file GENPIPES_FILE
```

Commands **for** running the pipeline are output to this file pathname. The data specified to pipeline command line **is** processed **and** pipeline run commands are stored **in** GENPIPES\_FILE, **if** this option **is** specified. Otherwise, the output will be redirected to stdout. This file can be used to actually "run the GenPipes Pipeline".

## Example Run

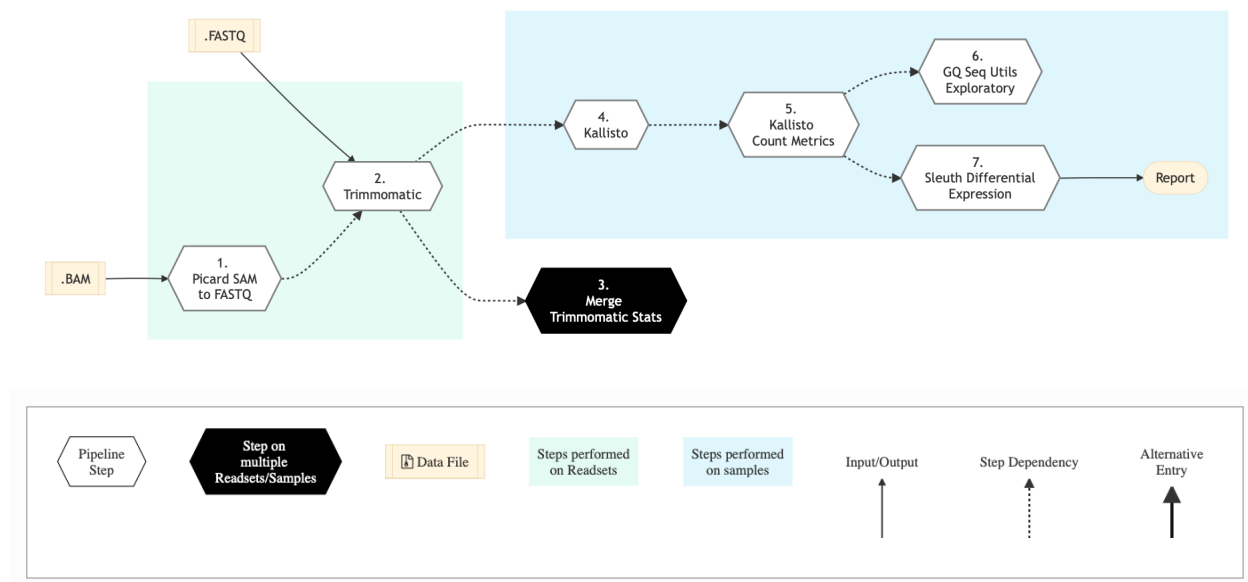
```
rnaseq_light.py -c $MUGQIC_PIPELINES_HOME/pipelines/rnaseq_light/rnaseq_light.base.ini
↪ $MUGQIC_PIPELINES_HOME/pipelines/rnaseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/rnaseq/
↪ rnaseq.base.ini $MUGQIC_PIPELINES_HOME/pipelines/rnaseq/rnaseq.guillimin.ini -r_
↪ readset.rnaseq.txt -d design.rnaseq.txt -s 1-25 -g rnaseqCommands.sh

bash rnaseqCommands.sh
```

You can download the test dataset for this pipeline in the [TestData Reference section](#)

## Pipeline Schema

Refer to the diagram below for the schema of RNA Sequencing (Light) Pipeline.



## Pipeline Steps

### Step Details

#### Picard SAM to FASTQ

Convert SAM/BAM files from the input readset file into FASTQ format if FASTQ files are not already specified in the readset file. Do nothing otherwise.

#### Trimmomatic

Raw reads quality trimming and removing of Illumina adapters is performed using [Trimmomatic](#). If an adapter FASTA file is specified in the config file (section 'trimmomatic', param 'adapter\_fasta'), it is used first. Else, 'Adapter1' and 'Adapter2' columns from the readset file are used to create an adapter FASTA file, given then to Trimmomatic. For PAIRED\_END readsets, readset adapters are reversed-complemented and swapped, to match Trimmomatic Palindrome strategy. For SINGLE\_END readsets, only Adapter1 is used and left unchanged.

This step takes as input files:

1. FASTQ files from the readset file if available
2. Else, FASTQ output files from previous picard\_sam\_to\_fastq conversion of BAM files

## Merge Trimmomatic Stats

The trim statistics per readset are merged at this step.

## Kallisto

Run Kallisto on fastq files for a fast estimate of abundance.

## Kallisto Count Matrix

TBD-GenPipes-Dev

## GQ Seq Utils Exploratory

Exploratory analysis using the gqSeqUtils R package adapted for RnaSeqLight.

## Sleuth Differential Expression

Performs differential gene expression analysis using [Sleuth](#). Analysis are performed both at a transcript and gene level, using two different tests: LRT and WT.

## More Information

- Kallisto, a new [ultra-fast RNA Sequencing technique](#)
- Limitations of alignment-free tools in [RNA sequencing quantification](#)

## Tumor Pair Sequencing Pipeline

**Warning:** A new updated Tumor Pair is back in the release 3.6.0

Please note that Tumor Pair Sequencing Pipeline was not available in GenPipes Release 3.2.0 until 3.6.0.

The documentation below refers to the pipeline corresponding to GenPipes Release 3.1.5.

We are in the process of updating the documentation corresponding to latest release 3.6.0. Thank you for your patience.

Human genome comprises of a set of chromosome pairs. One chromosome in each pair, called homolog, is derived from each parent. It is typically referred to as diploid whereas the set of chromosomes from a single parent is called haploid genome. For a given gene on a given chromosome, there is a comparable, if not identical, gene on the other chromosome in the pair, known as an allele. Large structural alterations in chromosomes can change the number of copies of affected genes on those chromosomes. This is one of the key reasons for causing cancer. In cancer cells, instead of having a homologous pair of alleles for a given gene, there may be deletions or duplications of those genes.

Such alterations leads to unequal contribution of one allele over the other, altering the copy number of a given allele. These variations in copy number indicated by the ratio of cancer cell copy number to normal cell copy number can provide information regarding the structure and history of cancer. However, when DNA is extracted, there is a mix of

cancer and normal cells and this information regarding absolute copy number per cancer cell is lost in DNA extraction process. Hence it must be inferred.

Inferring absolute copy number is difficult for [three reasons](#):

- cancer cells are nearly always intermixed with an unknown fraction of normal cells; the measure for this is tumor purity
- the actual quantity of DNA in the cancer cells after gross numerical and structural chromosomal changes is unknown; the measure for this is tumor ploidy
- the cancer cell population may be heterogeneous, possibly because of ongoing mutations and changes

Tumor purity and ploidy have a substantial impact on next-gen sequence analyses of tumor samples and may alter the biological and clinical interpretation of results.

GenPipes Tumor Analysis pipeline is designed to detect somatic variants from a tumor and match normal sample pair more accurately.

- [Introduction](#)
- [Version](#)
- [Usage](#)
- [Example Run](#)
- [Pipeline Schema](#)
- [Pipeline Steps](#)
- [Step Details](#)
- [More information](#)

---

## Introduction

GenPipes Tumor Pair workflow consumes BAM files. It inherits the BAM processing protocol from DNA-seq implementation, for retaining the benchmarking optimizations, but differs in alignment refinement and mutation identification. It achieves this by maximizing the information, utilizing both tumor and normal samples together.

The pipeline is based on an ensemble approach, which was optimized using both the [DREAM3 challenge](#) and the CEPH mixture datasets to select the best combination of callers for both SNV and structural variation detection. For SNVs, multiple callers such as ‘GATK MuTect2’, [VarScan 2](#), [BCFTools](#), [VarDict](#) were combined to achieve a sensitivity of 97.5%, precision of 98.8%, and F1 score of 98.1% for variants found in 2 callers.

Similarly, SVs were identified using multiple callers such as [DELLY](#), [LUMPY](#), [WHAM](#), [CNVKit](#), and [SvABA](#) combined using [MetaSV](#) to achieve a sensitivity of 84.6%, precision of 92.4%, and F1 score of 88.3% for duplication variants found in the DREAM3 dataset. The pipeline also integrates specific cancer tools to estimate tumor purity and tumor ploidy of sample pair normaltumor.

Additional annotations are incorporated to the SNV calls using [dbNSFP](#) and/or [Gemini](#), and QC metrics are collected at various stages and visualized using [MultiQC](#).

GenPipes Tumor Pair pipeline has three protocol options: sv, ensemble, or fastpass. For details refer to [Pipeline Schema](#) section below.

---



## Version

3.6.2

For the latest implementation and usage details refer to Tumor Pair Pipeline implementation [README file](#) file.

## Usage

```
python tumor_pair.py [-h] [--help] [-c CONFIG [CONFIG ...]] [-s STEPS]
                    [-o OUTPUT_DIR] [-j {pbs,batch,daemon,slurm}] [-f]
                    [--no-json] [--report] [--clean]
                    [-l {debug,info,warning,error,critical}] [--sanity-check]
                    [--container {wrapper, singularity} <IMAGE PATH>]
                    [--genpipes_file GENPIPES_FILE]
                    [-p PAIRS] [-t {sv,ensemble,fastpass}] [-r READSETS] [-v]
```

### Optional Arguments

`-p PAIRS, --pairs PAIRS`

pairs file

`-t {sv,ensemble,fastpass}, --type {sv,ensemble,fastpass}`

tumor pair analysis **type**

`-r READSETS, --readsets READSETS`

readset file

`-h` show this help message **and** exit

`--help` show detailed description of pipeline **and** steps

`-c CONFIG [CONFIG ...], --config CONFIG [CONFIG ...]`

config INI-style **list** of files; config parameters are overwritten based on files order

`-s STEPS, --steps STEPS` step **range** e.g. '1-5', '3,6,7', '2,4-8'

`-o OUTPUT_DIR, --output-dir OUTPUT_DIR`

output directory (default: current)

`-j {pbs,batch,daemon,slurm}, --job-scheduler {pbs,batch,daemon,slurm}`

job scheduler **type** (default: slurm)

<code>-f, --force</code>	force creation of jobs even <b>if</b> up to date (default: false)
<code>--no-json</code>	do <b>not</b> create JSON file per analysed sample to track the analysis status (default: false i.e. JSON file will be created)
<code>--report</code>	create ' <b>pandoc</b> ' command to merge <b>all</b> job markdown report files <b>in</b> the given step <b>range</b> into HTML, <b>if</b> they exist; <b>if</b> <code>--report</code> <b>is</b> set, <code>--job-scheduler</code> , <code>--force</code> , <code>--clean</code> options <b>and</b> job up-to-date status are ignored (default: false)
<code>--clean</code>	create ' <b>rm</b> ' commands <b>for</b> <b>all</b> job removable files <b>in</b> the given step <b>range</b> , <b>if</b> they exist; <b>if</b> <code>--clean</code> <b>is</b> set, <code>--job-scheduler</code> , <code>--force</code> options <b>and</b> job up-to-date status are ignored (default: false)
<code>-l {debug,info,warning,error,critical}, --log {debug,info,warning,error,critical}</code>	log level (default: info)
<code>--sanity-check</code>	run the pipeline in 'sanity check mode' to verify all the input files needed for the pipeline to run are available on the system (default: false)
<code>--container {wrapper, singularity} &lt;IMAGE PATH&gt;</code>	run pipeline inside a container providing a container image path <b>or</b> accessible singularity hub path
<code>-v, --version</code>	show the version information <b>and</b> exit
<code>-g GENPIPES_FILE, --genpipes_file GENPIPES_FILE</code>	Commands <b>for</b> running the pipeline are output to this file pathname. The data specified to pipeline command line <b>is</b> processed <b>and</b> pipeline run commands are stored <b>in</b> GENPIPES_FILE, <b>if</b> this option <b>is</b> specified . Otherwise, the output will be redirected to stdout . This file can be used to actually "run the GenPipes Pipeline".

## Example Run

Use the following commands to execute MUGQIC DNA sequencing pipeline:

```
python tumor_pair.py -c $MUGQIC_PIPELINES_HOME/pipelines/dnaseq/dnaseq.base.ini $MUGQIC_
↳ PIPELINES_HOME/pipelines/tumor_pair/tumor_pair.base.ini $MUGQIC_PIPELINES_HOME/
↳ pipelines/tumor_pair/tumor_pair.guillimin.ini -r readset.tumorPair.txt -p pairs.csv -s_
↳ 1-44 -g tumor_pairCommands.sh

bash tumor_pairCommands.sh
```

**Warning:** While issuing the pipeline run command, use ``-g GENPIPES_FILE`` option (see example above) instead of using the ``> GENPIPES_FILE`` option supported by GenPipes so far, as shown below:

```
[genpipes_seq_pipeline].py -t mugqic -c $MUGQIC_PIPELINES_HOME/pipelines/[genpipes_
↳ seq_pipeline]/[genpipes_seq_pipeline].base.ini $MUGQIC_PIPELINES_HOME/pipelines/
↳ [genpipes_seq_pipeline]/[genpipes_seq_pipeline].guillimin.ini -r readset.[genpipes_
↳ seq_pipeline].txt -s 1-6 > [genpipes_seq_pipeline]_commands_mugqic.sh

bash [genpipes_seq_pipeline]_commands_mugqic.sh
```

``> scriptfile`` should be considered deprecated and ``-g scriptfile`` option is recommended instead.

Please note that redirecting commands to a script ``> genpipe_script.sh`` is still supported for now. **But going forward, this mechanism might be dropped in a future GenPipes release.**

where, p pairs : format - patient\_name,normal\_sample\_name,tumor\_sample\_name

You can download the test dataset for this pipeline [here](#).

## Pipeline Schema

Pair Pipeline. You can refer to the latest [pipeline implementation](#).

There are three options for Tumor Pair Pipeline: sv, ensemble and fastpass.

Figure below shows the schema of the Tumor Pair Pipeline (sv) option. See [here](#) to download a high resolution image of [Tumor Pair Sequencing Pipeline \(sv\)](#) to download a high resolution image of the same.

Figure below shows the schema of the Tumor Pair Pipeline (ensemble) option. See [here](#) to download a high resolution image of [Tumor Pair Sequencing Pipeline \(ensemble\)](#) to download a high resolution image of the same.

Figure below shows the schema of the Tumor Pair Pipeline (fastpass) option. See [here](#) to download a high resolution image of [Tumor Pair Sequencing Pipeline \(fastpass\)](#) to download a high resolution image of the same.

tumor\_pair.py -t sv

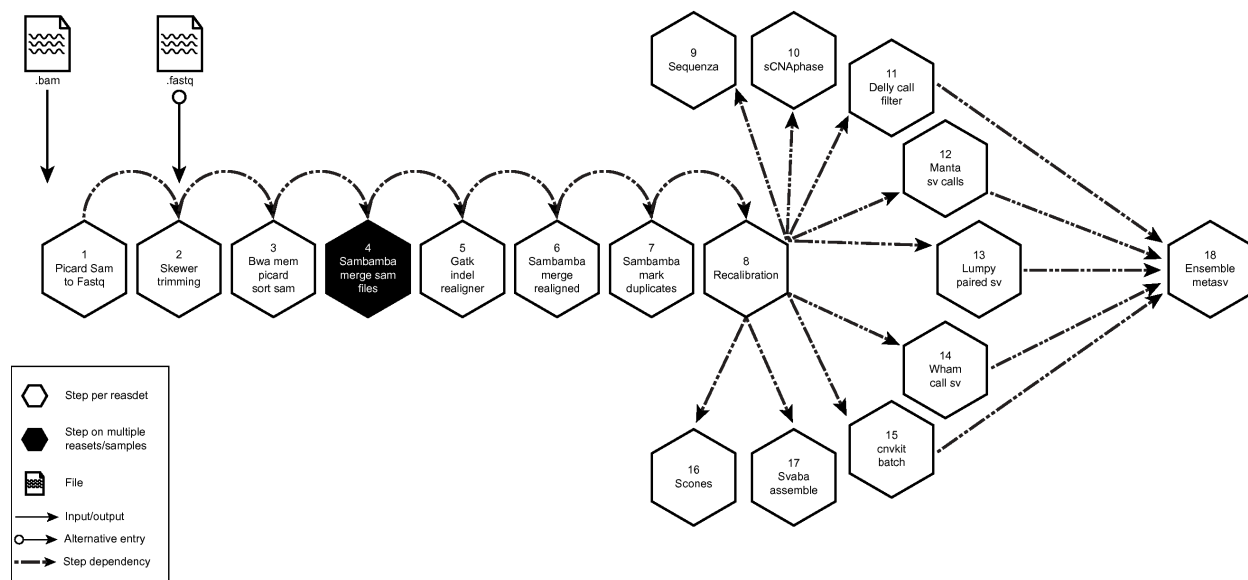


Fig. 24: Figure: Schema of Tumor Pair Pipeline (sv)

tumor\_pair.py -t ensemble

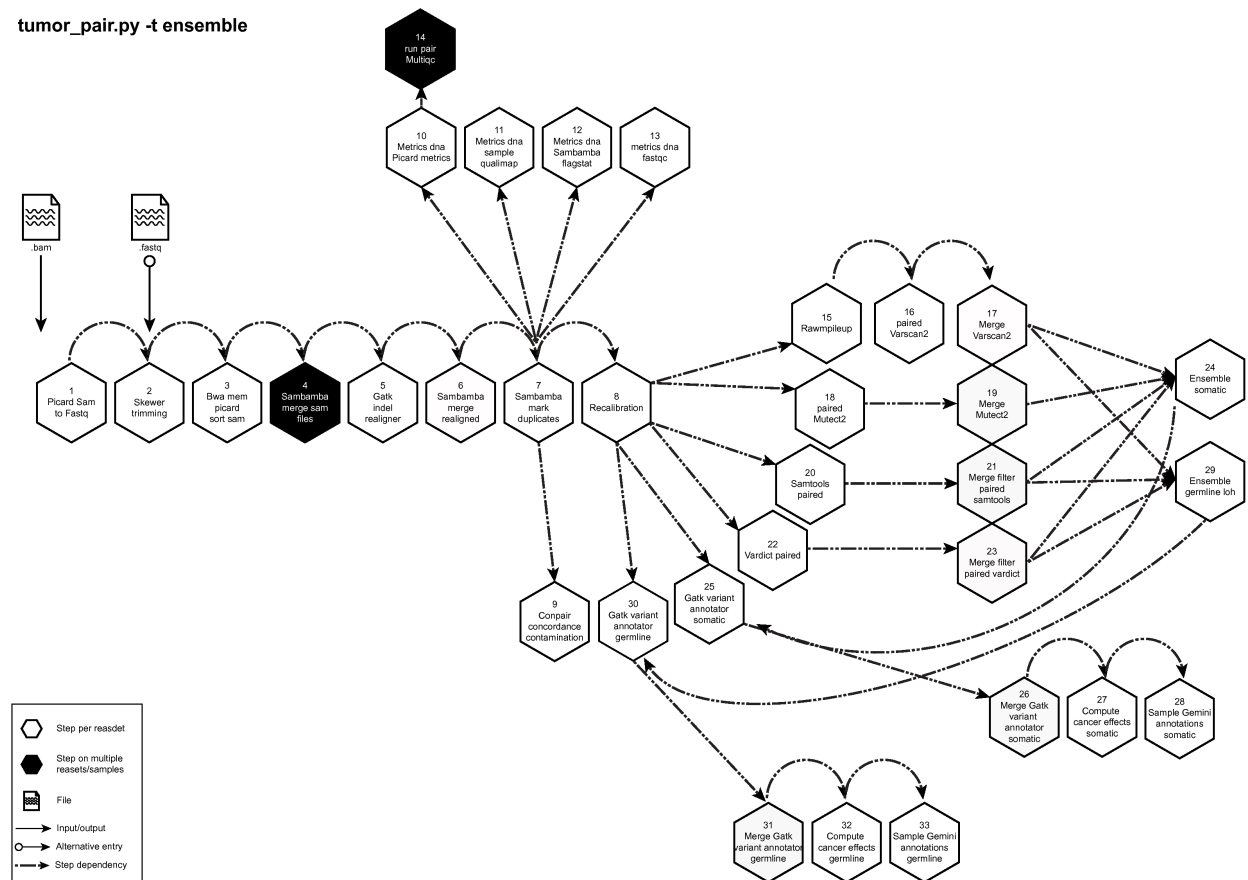


Fig. 25: Figure: Schema of Tumor Pair Pipeline (ensemble)

tumor\_pair.py -t fastpass

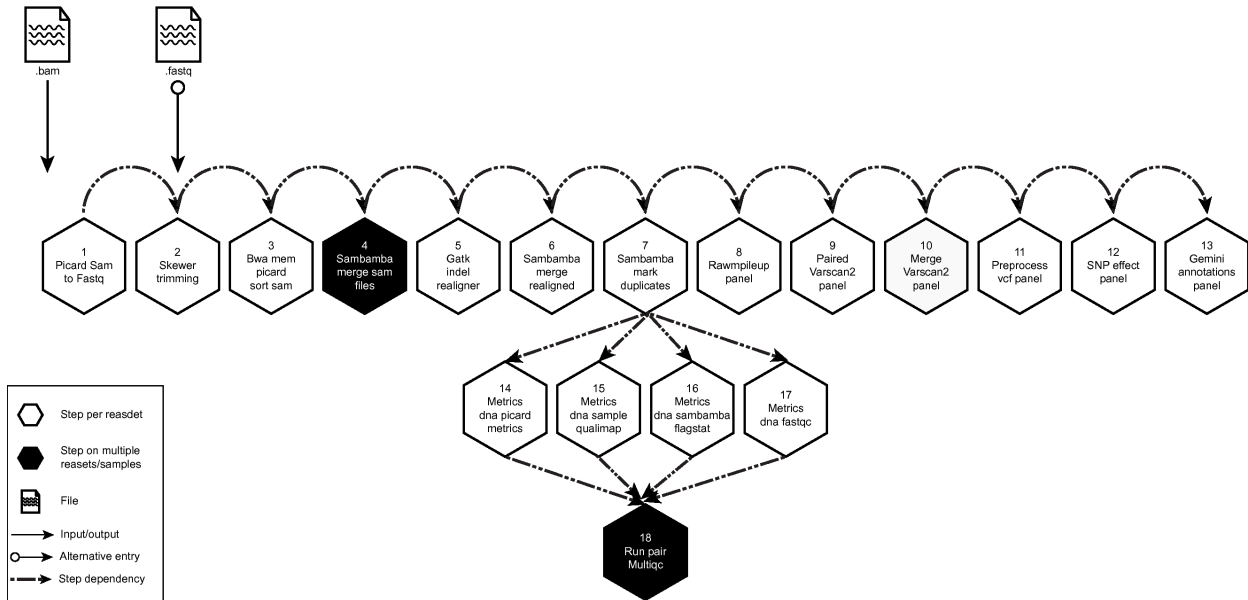


Fig. 26: Figure: Schema of Tumor Pair Pipeline (fastpass)

## Pipeline Steps

The table below shows various steps that constitute the Tumor Pair Pipeline.

	<i>Tumor Pair Pipeline Steps</i>
1.	<i>Picard SAM to FastQ</i>
2.	<i>Trimmomatic</i>
3.	<i>Merge Trimmomatic Stats</i>
4.	<i>BWA Picard Sort</i>
5.	<i>SamBamba Merge Files</i>
6.	<i>GATK InDel Realigner</i>
7.	<i>SamBamba Merge Realigned</i>

continues on next page

Table 3 – continued from previous page

	<i>Tumor Pair Pipeline Steps</i>
8.	<i>SamBamba Mark Duplicates</i>
9.	<i>Recalibration</i>
10.	<i>Conpair Concorance Contamination</i>
11.	<i>Raw Mpileup Panel</i>
12.	<i>Paired VarScan 2</i>
13.	<i>Merge VarScan 2 Panel</i>
14.	<i>PreProcess VCF Panel</i>
15.	<i>SNP Effect Panel</i>
16.	<i>Gemini Annotations Panel</i>
17.	<i>Metrics</i>
18.	<i>Picard Calculate HS Metrics</i>
19.	<i>GATK callable Loci</i>
20.	<i>Extract Common SNP Frequency</i>
21.	<i>BAF Plot</i>
22.	<i>Raw Mpileup</i>
23.	<i>Paired Var Scan 2</i>

continues on next page

Table 3 – continued from previous page

	<i>Tumor Pair Pipeline Steps</i>
24.	<i>Merge Var Scan 2</i>
25.	<i>Paired Mutect2</i>
26.	<i>Merge Mutect2</i>
27.	<i>SAM Tools Paired</i>
28.	<i>Merge Filter Paired SAM Tools</i>
29.	<i>VarDict Paired</i>
30.	<i>Merge Filter Paired VarDict</i>
31.	<i>Ensemble Somatic</i>
32.	<i>GATK Variant Annotator Somatic</i>
33.	<i>Merge GATK Variant Annotator Somatic</i>
34.	<i>Compute Cancer Effects Somatic</i>
35.	<i>Combine Tumor Pairs Somatic</i>
36.	<i>All Pairs Compute Effects Somatic</i>
37.	<i>Gemini Annotations Somatic</i>
38.	<i>Ensemble Germline Loh</i>
39.	<i>GATK Variant Annotator Germline</i>

continues on next page

Table 3 – continued from previous page

	<i>Tumor Pair Pipeline Steps</i>
40.	<i>Merge GATK Variant Annotator Germline</i>
41.	<i>Compute Cancer Effects Germline</i>
42.	<i>Combine Tumor Pairs Germline</i>
43.	<i>All Pairs Compute Effects Germline</i>
44.	<i>Gemini Annotations Germline</i>
45.	<i>CRAM Output</i>

## Step Details

Following are the various steps that are part of GenPipes Tumor Pair Pipeline:

### Picard SAM to FastQ

This step converts SAM/BAM files from the input readset file into FASTQ format, if FASTQ files are not already specified in the readset file. Otherwise, it does nothing.

### Trimmomatic

This step takes as input files:

- FASTQ files from the readset file if available
- Else, FASTQ output files from previous picard\_sam\_to\_fastq conversion of BAM files

Raw reads quality trimming and removing of Illumina adapters is performed using [Trimmomatic Method](#). If an adapter FASTA file is specified in the config file (section 'trimmomatic', param 'adapter\_fasta'), it is used first. Else, 'Adapter1' and 'Adapter2' columns from the readset file are used to create an adapter FASTA file, given then to Trimmomatic. For PAIRED\_END readsets, readset adapters are reversed-complemented and swapped, to match Trimmomatic Palindrome strategy. For SINGLE\_END readsets, only Adapter1 is used and left unchanged.

### Merge Trimmomatic Stats

The trim statistics per readset are merged at this step.

### BWA Picard Sort SAM

This step takes as input files:

1. Trimmed FASTQ files if available
2. Else, FASTQ files from the readset file if available
3. Else, FASTQ output files from previous picard\_sam\_to\_fastq conversion of BAM files



The filtered reads are aligned to a reference genome. The alignment is done per sequencing readset. The alignment software used is [BWA](#) with algorithm: bwa mem. BWA output BAM files are then sorted by coordinate using [Picard](#).

### SamBamba Merge Files

This step takes as input files:

- Aligned and sorted BAM output files from previous *BWA Picard Sort* step, if available
- Else, BAM files from the readset file

BAM readset files are merged into one file per sample. Merge is done using [Picard](#).

### GATK InDel Realigner

Insertion and deletion realignment is performed on regions where multiple base mismatches are preferred over indels by the aligner since it can appear to be less costly by the algorithm. Such regions will introduce false positive variant calls which may be filtered out by realigning those regions properly. Realignment is done using [GATK](#). The reference genome is divided by a number regions given by the nb\_jobs parameter.

Note: modified to use both normal and tumor BAMs to reduce FPs around indels

### SamBamba Merge Realigned

BAM files of regions of realigned reads are merged per sample using [SamBamba](#).

### Sambamba Mark Duplicates

Mark duplicates. Aligned reads per sample are duplicates if they have the same 5' alignment positions (for both mates in the case of paired-end reads). All but the best pair (based on alignment score) will be marked as a duplicate in the BAM file. Marking duplicates is done using [Picard](#).

### Recalibration

This step re-calibrates base quality scores of sequencing-by-synthesis reads in an aligned BAM file. After recalibration, the quality scores in the QUAL field in each read in the output BAM are more accurate in that the reported quality score is closer to its actual probability of mismatching the reference genome. Moreover, the recalibration tool attempts to correct for variation in quality with machine cycle and sequence context, and by doing so, provides not only more accurate quality scores but also more widely dispersed ones.

### Conpair Concorance Contamination

[Conpair](#) is a concordance and contamination estimator for tumor–normal pairs. This step is a quality control process to ensure the normal sample and cancer sample come from the same patient. It estimates this by determining the amount of normal sample in the tumor and the amount of tumor in normal sample.

### Raw Mpileup Panel

Full pileup (optional). A raw Mpileup file is created using samtools Mpileup and compressed in gzipped format. One packaged Mpileup file is created per sample/chromosome.

### Paired VarScan 2

This step involves variant calling and somatic mutation/CNV detection for next-generation sequencing data. For details see - [VarScan 2 Somatic mutation and copy number alteration discovery in cancer by exome sequencing](#).

This is a test.

### Merge VarScan 2 Panel

Merge Mpileup files per sample/chromosome into one compressed gzip file per sample.

### PreProcess VCF Panel

Preprocess VCF for loading into a annotation database called [Gemini](#). Processes include normalization and decomposition of MNPs by *Vt* and VCF modification for correct loading into [Gemini](#).

### **SNP Effect Panel**

Variant effect annotation. The .vcf files are annotated for variant effects using the SnpEff software. SnpEff annotates and predicts the effects of variants on genes (such as amino acid changes).

### **Gemini Annotations Panel**

Load functionally annotated VCF file into a MySQL lite annotation database [Gemini](#).

### **Metrics**

Compute metrics and generate coverage tracks per sample. Multiple metrics are computed at this stage: Number of raw reads, Number of filtered reads, Number of aligned reads, Number of duplicate reads, Median, mean and standard deviation of insert sizes of reads after alignment, percentage of bases covered at X reads (%\_bases\_above\_50 means the % of exons bases which have at least 50 reads) whole genome or targeted percentage of bases covered at X reads (%\_bases\_above\_50 means the % of exons bases which have at least 50 reads). A TDF (.tdf) coverage track is also generated at this step for easy visualization of coverage in the IGV browser.

### **Picard Calculate HS Metrics**

Compute on target percent of hybridization based capture.

### **GATK callable Loci**

Computes the callable region or the genome as a bed track.

### **Extract Common SNP Frequency**

Extracts allele frequencies of possible variants across the genome.

### **BAF Plot**

Plots DepthRatio and B allele frequency of previously extracted alleles.

### **Raw Mpileup**

Full pileup (optional). A raw Mpileup file is created using samtools Mpileup and compressed in gzipped format. One packaged Mpileup file is created per sample/chromosome.

### **Paired Var Scan 2**

Variant calling and somatic mutation/CNV detection for next-generation sequencing data. Uses [VarScan 2](#) for Somatic mutation and copy number alteration discovery in cancer by exome sequencing [VarScan 2](#) thresholds based on [DREAM3 results](#) generated by SC INFO field remove to prevent collision with Samtools output during ensemble.

### **Merge Var Scan 2**

Merge Mpileup files per sample/chromosome into one compressed gzip file per sample.

### **Paired Mutect2**

[GATK MuTect2 Overview](#) caller for SNVs and Indels.

### **Merge Mutect2**

Merge SNVs and indels for [GATK MuTect2 Overview](#) Replace TUMOR and NORMAL sample names in VCF to the exact tumor/normal sample names Generate a somatic VCF containing only PASS variants

### **SAM Tools Paired**

Samtools caller for SNVs and Indels using version 0.1.19.

### **Merge Filter Paired SAM Tools**

In this step, [bcftools](#) is used to merge the raw binary variants files created in the snpAndIndelBCF step. The output of bcftools is fed to varfilter, which does an additional filtering of the variants and transforms the output into

the VCF (.vcf) format. One VCF file contain the SNP/INDEL calls for all samples in the experiment. Additional somatic filters are performed to reduce the number of FPs: 1. vcfliibs vcfsampledif tags each variant with <tag>={[germline](#),`somatic`,`loh`\_} to specify the type of variant given the genotype difference between the two samples. 2. bcftools filter is used to retain only variants with CLR>=15 and have STATUS=somatic from VCFsamplediffer 3. bcftools filter is used to retain only variants that have STATUS=germline or STATUS=loh from vcfsamplediffer

### **VarDict Paired**

In this step, [VarDict](#) caller is used for SNVs and Indels. Note: variants are filtered to remove instance where REF == ALT and REF modified to 'N' when REF is AUPAC nomenclature

### **Merge Filter Paired VarDict**

The fully merged VCF is filtered using following steps: 1. Retain only variants designated as somatic by VarDict: either StrongSomatic or LikelySomatic 2. Somatics identified in step 1 must have PASS filter.

### **Ensemble Somatic**

Apply [Bcbio.variations](#) ensemble approach for [GATK MuTect2 Overview](#), [VarDict](#), Samtools and [VarScan 2](#) calls Filter ensemble calls to retain only calls overlapping 2 or more callers

### **GATK Variant Annotator Somatic**

Add VCF annotations to ensemble VCF: Standard and Somatic annotations.

### **Merge GATK Variant Annotator Somatic**

Merge annotated somatic VCFs.

### **Compute Cancer Effects Somatic**

Variant effect annotation. The .vcf files are annotated for variant effects using the SnpEff software. SnpEff annotates and predicts the effects of variants on genes (such as amino acid changes). Modified arguments to consider paired cancer data.

### **Combine Tumor Pairs Somatic**

Combine numerous ensemble VCFs into one VCF for [Gemini](#) annotations.

### **All Pairs Compute Effects Somatic**

Variant effect annotation. The .vcf files are annotated for variant effects using the SnpEff software. SnpEff annotates and predicts the effects of variants on genes (such as amino acid changes). Modified arguments to consider paired cancer data. Applied to all tumor pairs.

### **Gemini Annotations Somatic**

Load functionally annotated VCF file into a MySQL lite annotation database [Gemini](#).

### **Ensemble Germline Loh**

Apply [Bcbio.variations](#) ensemble approach for [VarDict](#), Samtools and [VarScan 2](#) calls Filter ensemble calls to retain only calls overlapping 2 or more callers.

### **GATK Variant Annotator Germline**

Add VCF annotations to ensemble VCF: most importantly the AD field.

### **Merge GATK Variant Annotator Germline**

Merge annotated germline and LOH VCFs.

### **Compute Cancer Effects Germline**

Variant effect annotation. The .vcf files are annotated for variant effects using the SnpEff software. SnpEff annotates and predicts the effects of variants on genes (such as amino acid changes). Modified arguments to consider paired cancer data.

### Combine Tumor Pairs Germline

Combine numerous ensemble VCFs into one VCF for [Gemini](#) annotations.

### All Pairs Compute Effects Germline

Variant effect annotation. The .vcf files are annotated for variant effects using the SnpEff software. SnpEff annotates and predicts the effects of variants on genes (such as amino acid changes). Modified arguments to consider paired cancer data. Applied to all tumor pairs.

### Gemini Annotations Germline

Load functionally annotated VCF file into a MySQL lite annotation database [Gemini](#).

### CRAM Output

Generate long term storage version of the final alignment files in CRAM format. Using this function will include the original final BAM file into the removable file list.

---

## More information

For the latest implementation and usage details refer to the latest [pipeline implementation](#).

- MuTect2 Tool for calling somatic SNVs and indels via local assembly of haplotypes - [See here](#).
- A [three-caller pipeline](#) for variant analysis of cancer whole-exome sequencing data.

## 1.6.3 Latest GenPipes Release Note

GenPipes is a flexible Python-based framework that facilitates the development and deployment of multi-step workflows optimized for high-performance computing clusters and the cloud.

GenPipes comes with 12 pre-built, validated, scalable pipelines for various genomics applications, including RNA sequencing, chromatin immunoprecipitation sequencing, DNA sequencing, methylation sequencing, Hi-C, capture Hi-C, metagenomics, and SARS-CoV-2 genome sequencing pipeline. The software is available under a GPLv3 open source license and is continuously updated to follow recent advances in genomics and bioinformatics. The framework has already been configured on several servers, and a Docker image is also available to facilitate additional installations.

The latest release of GenPipes is 3.6.2. You can refer to the latest GenPipes Release note for details regarding what has changed. See [Release Notes](#) section of this documentation for details regarding prior GenPipes releases.

## 1.7 GenPipes Tutorials

GenPipes is a flexible Python-based framework that facilitates the development and deployment of multi-step genomic workflows, optimized for High-Performance Computing (HPC) clusters and the cloud. It offers open sourced 12 validated and scalable pipelines for various genomics applications.

There are multiple ways to deploy and run GenPipes. For details, see [GenPipes Deployment Guide](#).

The GenPipes tutorials listed below demonstrate how GenPipes can be deployed and run for first time users. Please note that the tutorials correspond to the GenPipes deployment [type](#). If you are deploying GenPipes on the cloud, then refer to the *GenPipes in the Cloud* tutorial below. If you are accessing GenPipes which is pre-deployed on Compute Canada servers, use the first tutorial listed below:

## 1.7.1 GenPipes Tutorial

GenPipes bioinformatics pipelines developed at the Canadian Centre for Computational Genomics (C3G), as part of the GenAP project, are available for public use. They include a wide array of pipelines, including RNA-Seq, ChIP-Seq, WGS, exome sequencing, Bisulfite sequencing, Hi-C, capture Hi-C, Metagenomics and SARS-CoV-2 genome sequencing pipeline. This document explains how to use the pipelines using Hi-C analysis pipeline and the ChIP-Seq pipeline, as examples.

### Setting up the environment

#### Abacus, Compute Canada users

Software and scripts used by GenPipes are already installed on several Compute Canada servers including Guillimin, Mammouth, Cedar and Graham, and will soon be installed on Beluga. To access the tools, you will need to add the tool path to your **bash\_profile**. The bash profile is a hidden file in your home directory that sets up your environment every time you log in.

You can also use your bashrc file. For more information on the differences between the .bash\_profile and the .bashrc profile, consult [this page](#).

```
## open bash_profile:
nano $HOME/.bash_profile
```

paste the following lines of code and save the file and Exit (Control + X):

```
## GenPipes/MUGQIC genomes and modules
export MUGQIC_INSTALL_HOME=/cvmfs/soft.mugqic/CentOS6
module use $MUGQIC_INSTALL_HOME/modulefiles
module load mugqic/python/2.7.14
module load mugqic/genpipes/<latest_version>
export JOB_MAIL=<my.name@my.email.ca>
export RAP_ID=<my-rap-id>
```

You will need to replace text within < >, with your information.

To find out the latest GenPipes version use the output of:

```
module avail 2>&1 | grep mugqic/genpipes
```

**Note:** previous versions of GenPipes were named mugqic\_pipelines and are still available for use.

**JOB\_MAIL** is the email to which the notifications are sent after each job. We advise you to create a separate email for jobs since you can receive hundreds of emails per pipeline. You can also de-activate the email sending option by removing the “-M \$JOB\_MAIL” option from the .ini files (discussed below).

**RAP\_ID** is the Resource Allocation Project ID from Compute Canada; It is usually in the format: rrg-lab-xy OR def-lab

When you make changes to your bash\_profile, you will need to log out then log in again, or type in the following command:

```
source $HOME/.bash_profile
```

By adding those lines to your bash profile, you are now ready to use our pipelines. This also gives you access to hundreds of bioinformatics tools pre-installed by our team. To explore the available tools, you can type:

```
module avail muggic/
```

For a full list of available modules, you can visit our [module page](#).

To load a tool, for example samtools, you can use:

```
# module add muggic/<tool>/<version>
module add muggic/samtools/1.4.1
# Now samtools 1.4.1 is available to use. To check:
samtools
```

You also have access to pre-installed genomes available in: `$MUGQIC_INSTALL_HOME/genomes/species/` To check all the available species, type:

```
ls $MUGQIC_INSTALL_HOME/genomes/species
```

All genome-related files, including indices for different aligners and annotation files can be found in:

```
$MUGQIC_INSTALL_HOME/genomes/species/<species_scientific_name>.<assembly>/
## so for Homo Sapiens hg19 assembly, that would be:
ls $MUGQIC_INSTALL_HOME/genomes/species/Homo_sapiens.hg19/
```

For a list of available genomes, you can visit our [genome page](#).

## Usage:

Now that your variables are set, you can launch any pipeline using: `<pipeline_name>.py` To check the help information for our hicseq (Hi-C analysis) and our chipseq pipelines, try:

```
hicseq.py -h
chipseq.py -h
```

All our pipelines use the same framework and work in similar ways; each with its own output of course. We will focus on two pipelines to demonstrate how the framework works.

To use most of our pipelines you will need two types of files; a **configuration file** that stores all the parameters used by the pipeline (extension `.ini`) and a **readset file** that stores all the information about your samples.

## Configuration File:

GenPipes pipelines are multi-step pipelines that run several tools, each with its own parameter inputs. All those parameters are stored in configuration files with **.ini** extension. Those files have a structure similar to Microsoft Windows INI files, where parameters are divided within sections.

---

### Note: What is a “configuration file” or an “ini” file and why do we need it?

An ini file is a file that contains parameters needed to run a pipeline. Our genome alignment pipeline contains over 20 steps, each involving over 5 parameters per step. Imagine having to type all 100 parameters to run a pipeline! For simplicity, all the parameters are stored in an “ini” file (extension `.ini`) that accompanies the pipeline. Try opening an ini file in a text editor and look at its content!

Each pipeline has several configuration/ini files in:

`$MUGQIC_PIPELINES_HOME/pipelines/<pipeline_name>/<pipeline_name>.*.ini` For hicseq, that would be:

```
ls $MUGQIC_PIPELINES_HOME/pipelines/hicseq/hicseq.*.ini
```

For chipseq, that would be:

```
ls $MUGQIC_PIPELINES_HOME/pipelines/chipseq/chipseq.*.ini
```

You will find a **<pipeline\_name>.base.ini** as well as an ini file for particular servers like Guillimin (**<pipeline\_name>.guillimin.ini**). The base.ini file has all the parameters needed by the pipeline but is optimized for usage on our own server, Abacus. To use the pipeline on Guillimin, you will need to use both base.ini and guillimin.ini, as such:

```
hicseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/hicseq/hicseq.base.ini $MUGQIC_PIPELINES_
↪HOME/pipelines/hicseq/hicseq.guillimin.ini ...
```

To change different parameters in the ini files, you can create your own ini file and overwrite the required parameters. For example, to change the number of threads for trimmomatic and hicup, I can create my own ini file: hicseq.test.ini and in it I can include the parameters to be changed:

```
[trimmomatic]

threads=2

[hicup_align]

threads=4
```

then add my ini file after the other ini files:

```
hicseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/hicseq/hicseq.base.ini $MUGQIC_PIPELINES_
↪HOME/pipelines/hicseq/hicseq.guillimin.ini hicseq.test.ini...
```

For different species, we have custom ini files stored in **\$MUGQIC\_PIPELINES\_HOME/resources/genomes/config/**. The genome default for our pipelines is human. To use other species, you can either create a custom .ini file or you can use the .ini files provided in **\$MUGQIC\_PIPELINES\_HOME/resources/genomes/config/** if your species of interest is available.

To run the hicseq pipeline on mouse mm9, for example, you can do the following:

```
hicseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/hicseq/hicseq.base.ini $MUGQIC_PIPELINES_
↪HOME/pipelines/hicseq/hicseq.guillimin.ini $MUGQIC_PIPELINES_HOME/resources/genomes/
↪config/Mus_musculus.mm9.ini ...
```

## Readset File:

The readset file is a **tab-separated** file that contains the following information:

**Sample:** must contain letters A-Z, numbers 0-9, hyphens (-) or underscores (\_) only; BAM files will be merged into a file named after this value; mandatory.

## Note: Sample

The definition of a sample in the context of GenPipes is the “input” biological sample, i.e. the sample on which processing such as IP, IgG assay (ChIPSeq Pipeline) or nothing (input) was performed. This is in contrast to sample being defined as the “sample sent for sequencing”.

**Readset:** a unique readset name with the same allowed characters as above; mandatory.

**Library:** optional. **RunType:** PAIRED\_END or SINGLE\_END; mandatory. **Run:** mandatory. **Lane:** mandatory. **Adapter1:** sequence of the forward trimming adapter **Adapter2:** sequence of the reverse trimming adapter **QualityOffset:** quality score offset integer used for trimming; optional. **BED:** relative or absolute path to BED file; optional. **FASTQ1:** relative or absolute path to first FASTQ file for paired-end readset or single FASTQ file for single-end readset; mandatory if BAM value is missing. **FASTQ2:** relative or absolute path to second FASTQ file for paired-end readset; mandatory if RunType value is “PAIRED\_END”. **BAM:** relative or absolute path to BAM file which will be converted into FASTQ files if they are not available; mandatory if FASTQ1 value is missing, ignored otherwise.

Example:

```
Sample Readset Library RunType Run Lane Adapter1 Adapter2 QualityOffset BED FASTQ1_
↳FASTQ2 BAM
sampleA readset1 lib0001 PAIRED_END run100 1 AGATCGGAAGAGCACACGTCTGAACTCCAGTCA_
↳AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT 33 path/to/file.bed path/to/readset1.paired1.fastq.
↳gz path/to/readset1.paired2.fastq.gz path/to/readset1.bam
sampleA readset2 lib0001 PAIRED_END run100 2 AGATCGGAAGAGCACACGTCTGAACTCCAGTCA_
↳AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT 33 path/to/file.bed path/to/readset2.paired1.fastq.
↳gz path/to/readset2.paired2.fastq.gz path/to/readset2.bam
sampleB readset3 lib0002 PAIRED_END run200 5 AGATCGGAAGAGCACACGTCTGAACTCCAGTCA_
↳AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT 33 path/to/file.bed path/to/readset3.paired1.fastq.
↳gz path/to/readset3.paired2.fastq.gz path/to/readset3.bam
sampleB readset4 lib0002 PAIRED_END run200 6 AGATCGGAAGAGCACACGTCTGAACTCCAGTCA_
↳AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT 33 path/to/file.bed path/to/readset4.paired1.fastq.
↳gz path/to/readset4.paired2.fastq.gz path/to/readset4.bam
```

If some optional information is missing, leave its position empty. **Sample vs Readset:**

Readsets refer to replicates that belong to a particular sample. If a sample was divided over 3 lanes, each lane output would be a readset of that sample. Most pipelines merge readsets and run the analysis based on samples. You can think of readsets as technical replicates while Samples as biological replicates.

---

### Note: What is a “Readset file” and why do we need it?

A readset file is another file that accompanies our pipelines. While the configuration files contains information about the parameters needed by the tools in the pipeline, the readset file contains information about the samples to be analyzed. In the Readset file, you list each readset used for the analysis, which samples are to be merged and where your fastq files or bam files are located.

---

### Creating a Readset File:

If you have access to Abacus, we provide a script `$MUGQIC_PIPELINES_HOME/utils/nanuq2mugqic_pipelines.py` that can access your Nanuq data, creates symlinks to the data on Abacus and creates the Readset file for you.

If your data is on nanuq but you do not have access to Abacus, there is a helper script `$MUGQIC_PIPELINES_HOME/utils/csvToreadset.R` that takes a csv file downloadable from nanuq and creates the Readset file. However, you will have to download the data from Nanuq yourself.

If your data is not on nanuq, you will have to manually create the Readset file. You can use a template and enter your samples manually. Remember that it is a tab separated file. There is a helper `$MUGQIC_PIPELINES_HOME/utils/mugqicValidator.py` script that can validate the integrity of your readset file.



## Design File:

Certain pipelines where samples are compared against other samples, like `chipseq.py` and `rnaseq.py`, require a design file that describes which samples are to be compared. We will discuss this later during an example.

---

### Note: What is a “Design file” and why do we need it?

A Design file is another file that accompanies some of our pipelines, where sample comparison is part of the pipeline. Unlike the configuration file and the Readset file, the Design file is not required by every pipeline. To check whether the pipeline you are interested in requires a Design file and to understand the format of the file, read the specific help pages for your pipeline of interest.

---

## Running GenPipes on Compute Canada Servers:

Guillimin, unlike Cedar, Graham and now Mammouth (mp2b), use the PBS scheduler. To use GenPipes on Guillimin, don't forget to add the “`-j pbs`” option (default is `-j Slurm`).

See example below for more details.

### Example run:

#### hicseq Test Dataset:

Let's now run the pipeline using a test dataset. We will use the first 2 million reads from HIC010 from Rao et al. 2014 (SRR1658581.sra). This is an in situ Hi-C experiment of GM12878 using MboI restriction enzyme.

We will start by downloading the dataset from [HERE](#). In the downloaded zip file, you will find the two fastq read files in folder “rawData” and will find the readset file (`readsets.HiC010.tsv`) that describes that dataset.

We will run this analysis on guillimin as follows:

```
hicseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/hicseq/hicseq.base.ini $MUGQIC_PIPELINES_
↪HOME/pipelines/hicseq/hicseq.guillimin.ini -r readsets.HiC010.tsv -s 1-15 -e MboI >_
↪hicseqScript_SRR1658581.txt
```

`-c` defines the ini configuration files `-r` defines the readset file `-s` defines the steps of the pipeline to execute. To check pipeline steps use `hicseq -h` `-e` defines the restriction enzyme used in the HiC library

The pipelines do not run the commands directly; they output them as text commands. So we need to redirect them into a file using “`>`”. In this case, `hicseqScript_SRR1658581.txt` is the script that contains the analysis commands.

This command works for servers using a SLURM scheduler like Cedar, Mammouth or Graham. For the PBS scheduler, used by Guillimin, you need to add the “`-j pbs`” option, as follows:

```
hicseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/hicseq/hicseq.base.ini $MUGQIC_PIPELINES_
↪HOME/pipelines/hicseq/hicseq.guillimn.ini -r readsets.HiC010.tsv -s 1-15 -e MboI -j_
↪pbs > hicseqScript_SRR1658581.txt
```

To run it, use:

```
bash hicseqScript_SRR1658581.txt
```

You will not see anything happen, but the commands will be sent to the server job queue. **So do not run this more than once per job.** To confirm that the commands have been submitted, wait a minute or two depending on the server and type:

```
showq -u <userID>
```

In case you ran it several times and launched too many commands you do not want, you can use the following line of code to cancel ALL commands:

```
showq -u <userID> | tr "|" " "| awk '{print $1}' | xargs -n1 canceljob
```

Congratulations! you just ran the hicseq pipeline. After the processing is complete, you can access quality control plots in the `homer_tag_directory/HomerQcPlots`. You can find the compartment data in the `compartments` folder, TADs in the `TADs` folder and significant interactions in the `peaks` folder.

For more information about output formats please consult the webpage of the third party tool used.

---

**Note:** The hicseq pipeline also analyzes capture hic data if the “-t capture” flag is used. For more information on the available steps in that pipeline use: **hicseq -h**

---

### Creating a Design File:

Certain pipelines that involve comparing and contrasting samples, need a Design File.

The Design File is a **tab-separated** plain text file with one line per sample and the following columns:

**Sample:** first column; must contain letters A-Z, numbers 0-9, hyphens (-) or underscores (\_) only; the sample name must match a sample name in the readset file; mandatory.

**contrast:** each of the following columns defines an experimental design contrast; the column name defines the contrast name, and the following values represent the sample group membership for this contrast:

- ‘0’ or ’’: the sample does not belong to any group.
- ‘1’: the sample belongs to the control group.
- ‘2’: the sample belongs to the treatment test case group.

Example:

```
Sample Contrast_AB Contrast_AC
sampleA 1 1
sampleB 2 0
sampleC 0 2
sampleD 0 0
```

where Contrast\_AB compares treatment sampleB to control sampleA, while Contrast\_AC compares sampleC to sampleA.

You can add several contrasts per design file.

To see how this works, lets run a ChIP-Seq experiment.

## chipseq Test Dataset:

We will use a subset of the ENCODE data. Specifically, the reads that map to chr22 of the following samples [ENCFF361CSC](#) and [ENCFF837BCE](#). They represent a ChIP-Seq analysis dataset with the CTCF transcription factor and its control input.

We will start by downloading the dataset from [HERE](#)

In the downloaded zip file, you will find the two fastq read files in folder rawData and will find the readset file (readsets.chipseqTest.chr22.tsv) that describes that dataset. You will also find the design file

```
designfile_chipseq.chr22.txt
```

that contains the contrast of interest.

Looking at the content of the Readset file

```
readsets.chipseqTest.tsv
```

we see:

```
Sample Readset Library RunType Run Lane Adapter1 Adapter2 QualityOffset BED FASTQ1_
↳FASTQ2 BAM
ENCFF361CSC_ctrl1 ENCFF361CSC_chr22 SINGLE_END 2965 1 AGATCGGAAGAGCACACGTCTGAACTCCAGTCA_
↳AGATCGGAAGAGCGTCGTAGGGAAAGAGTGT 33 rawData/ENCFF361CSC.chr22.fastq
ENCFF837BCE_ctcf ENCFF837BCE_chr22 SINGLE_END 2962 1 AGATCGGAAGAGCACACGTCTGAACTCCAGTCA_
↳AGATCGGAAGAGCGTCGTAGGGAAAGAGTGT 33 rawData/ENCFF837BCE.chr22.fastq
```

This analysis contains 2 samples with a single readset each. They are both SINGLE\_END runs and have a single fastq file in the “rawData” folder.

Looking at the content of the Design file

```
designfile_chipseq.txt)
```

we see:

```
Sample CTCF_Input,N
ENCFF361CSC_ctrl1 1
ENCFF837BCE_ctcf 2
```

We see a single analysis CTCF\_Input run as Narrow peaks (coded by “N”; you can use “B” for broad peak analysis). This analysis compares CTCF peaks in ENCFF837BCE\_ctcf to its input control peaks identified from ENCFF361CSC\_ctrl.

We will run this analysis on guillimin as follows:

```
chipseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/chipseq/chipseq.base.ini $MUGQIC_
↳PIPELINES_HOME/pipelines/chipseq/chipseq.guillimin.ini -r readsets.chipseqTest.chr22.
↳tsv -d designfile_chipseq.chr22.txt -s 1-15 > chipseqScript.txt
bash chipseqScript.txt
```

The commands will be sent to the job queue and you will be notified once each step is done. If everything runs smoothly, you should get **MUGQICexitStatus:0** or **Exit\_status=0**. If that is not the case, then an error has occurred after which the pipeline usually aborts. To examine the errors, check the content of the **job\_output** folder.

### Available pipelines:

### For more information:

Our pipelines are built around third party tools that the community uses in particular fields. To understand the output of each pipeline, please read the documentation pertaining to the tools that produced the output.

For more information or help with particular pipelines, you can send us an email to: [info@computationalgenomics.ca](mailto:info@computationalgenomics.ca)

Or drop by during our [Open Door](#) slots. We are located at:

740 Dr. Penfield avenue, room 4200 Montréal, QC H3A 1A5

## 1.7.2 Running GenPipes on Google Cloud Platform (GCP)

### Quickstart

The Quickstart uses a “**Try GCP for free**” session. If you already have basic knowledge of GCP, and its shell, you can jump directly to step 4.

1. Create an account on GCP. For more instructions, check out [this page](#).
2. Get acquainted with Google Cloud Shell. For more instructions, check out [this page](#).
3. Create a new project. For more instructions, check out [this page](#).
4. Install GenPipes, as follows:

In your google shell session, run:

```
git clone https://bitbucket.org/mugqic/cloud_deployement.git
cd cloud_deployement/gcp/
gcloud deployment-manager deployments create slurm --config slurm-cluster.yaml
```

From here on, your GenPipes cloud is being deployed and your account is getting billed by Google. Remember to shut down the cluster when the analysis is done. Once this command is done running, a configuration script is started to install SLURM on the cluster. You will be able to monitor the installation after you run the next command.

### Run one of the GenPipes test sets on GCP:

In the Google shell run the following command to log into the login node of the Slurm cluster:

```
gcloud compute ssh login1 --zone=northamerica-northeast1-a
```

You are now on your cloud deployment login node.

The installation is still running and you were welcome by the following message:

---

**Note:** \*\* Slurm is currently being installed/configured in the background. \*\* A terminal broadcast will announce when installation and configuration is complete.

---

Wait for the terminal broadcast this can take up to 10 minutes. Once you have received it or once you log to this node without seeing the warning, you can go to the next step. You can run the GenPipes [tutorial](#) from that location.

Let's use ChIPSeq as an example:

#### 1- Make a folder for the test:

```
mkdir -p chipseq_test
cd chipseq_test
```

## 2- Download dataset and unzip it:

```
wget https://www.computationalgenomics.ca/tutorials/chipseq.zip
unzip chipseq.zip
```

## 3- Download the config file for this Quickstart:

```
wget https://bitbucket.org/mugqic/cloud_deployoment/raw/master/quick_start.ini
```

## 4- Create chipseq pipeline script:

```
bash # You do not need this line if you did a logout login cycle
# The next line generates the pipeline script
chipseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/chipseq/chipseq.base.ini \
$MUGQIC_PIPELINES_HOME/pipelines/chipseq/chipseq.cedar.ini \
quick_start.ini \
-j slurm \
-r readsets.chipseqTest.chr22.tsv \
-d designfile_chipseq.chr22.txt \
-s 1-18 > chipseqScript.sh
```

## 5- Run chipseq pipeline:

```
bash chipseqScript.sh
```

**6- Look at your pipeline progression:** . Use `squeue` command. Your GenPipes analysis is [being run on Slurm](#)

**7- Shut down your Genpipes Cloud installation (and stop being billed):** . After the jobs have run, you can exit the login node:

```
exit
```

You, are now in back on your cloud shell administrative machine. You can shut down your GenPipes cloud cluster.

```
gcloud deployment-manager deployments delete slurm
```

You are not being billed anymore.

---

**Note:** You need to enable the “deployment manager” API on your project. See [this page](#). You also need to make sure that billing is enabled (even for a free try). For more detailed information, check out our [bitbucket repo](#)

---

### 1.7.3 Running GenPipes in a local Containerized Infrastructure

**Note:** The following is work in progress.

This tutorial provides information regarding how to run GenPipes locally within your infrastructure in a containerized environment.

GenPipes pipelines use scheduler's calls (qsub, sbatch) for submitting genomic analysis compute jobs. If you plan to use GenPipes locally using your infrastructure, inside a container, you need to run the GenPipes pipeline python scripts using the "batch mode" option. For local containerized versions of GenPipes, this is the preferred way of running the pipelines, if you don't have access to a scheduler locally such as SLURM or PBS.

This is how you can run GenPipes pipelines such as *DNA Sequencing Pipeline*, refer to the command below:

```
dnaseq.py -c dnaseq.base.ini dnaseq.batch.ini -j batch -r your-readsets.tsv -d your-
design.tsv -s 1-34 -t mugqic > run-in-container-dnaseq-script.sh

bash run-in-container-dnaseq-script.sh
```

Please note, there is a disadvantage to running GenPipes Pipelines without a scheduler. In the batch mode, which is configured using the "-j batch" option, all the jobs would run as a batch, one after another, on a single node. If your server is powerful enough, this might be your preferable option. Otherwise, if you would like to take advantage of GenPipes' job scheduling capabilities, you need to install a job scheduler locally in your infrastructure so that GenPipes can work effectively. We recommend SLURM scheduler for GenPipes.

### 1.7.4 Pipeline Usage Examples

```
chipseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/chipseq/chipseq.base.ini $MUGQIC_PIPELINES_HOME/pipelines/chipseq/
-r readsets.chipseq.txt -d design.chipseq.txt -s 1-19 > chipseqCommands.sh
```

```
rnaseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/rnaseq/rnaseq.base.ini $MUGQIC_PIPELINES_HOME/pipelines/rnaseq/
-r readset.rnaseq.txt -d design.rnaseq.txt -s 1-25 > rnaseqCommands.sh
```

```
dnaseq.py -t mugqic -c $MUGQIC_PIPELINES_HOME/pipelines/dnaseq/dnaseq.base.ini
$MUGQIC_PIPELINES_HOME/pipelines/dnaseq/dnaseq.guillimin.ini -r readset.dnaseq.txt -s 1-29 > dnaseq-
Commands_mugqic.sh
```

```
dnaseq.py -t mpileup -c $MUGQIC_PIPELINES_HOME/pipelines/dnaseq/dnaseq.base.ini
$MUGQIC_PIPELINES_HOME/pipelines/dnaseq/dnaseq.guillimin.ini -r readset.dnaseq.txt -s 1-33 > dnaseq-
Commands_mpileup.sh
```

```
dnaseq_high_coverage.py -c $MUGQIC_PIPELINES_HOME/pipelines/dnaseq_high_coverage/dnaseq_high_coverage.base.ini
$MUGQIC_PIPELINES_HOME/pipelines/dnaseq_high_coverage/dnaseq_high_coverage.guillimin.ini -r read-
set.dnaseq.txt -s 1-15 > dnaseq_high_coverageCommands.sh
```

```
tumor_pair.py -c $MUGQIC_PIPELINES_HOME/pipelines/dnaseq/dnaseq.base.ini
$MUGQIC_PIPELINES_HOME/pipelines/tumor_pair/tumor_pair.base.ini $MUGQIC_PIPELINES_HOME/pipelines/tumor_pair/
-r readset.tumorPair.txt -p pairs.csv -s 1-44 > tumor_pairCommands.sh
```

```
hicseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/hicseq/hicseq.base.ini $MUGQIC_PIPELINES_HOME/pipelines/hicseq/hicseq
-r readsets.hicseq.txt -s 1-16 -t hic -e MboI > hicseqCommands_hic.sh
```

```
hicseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/hicseq/hicseq.base.ini $MUGQIC_PIPELINES_HOME/pipelines/hicseq/hicseq
$TEST_DIR/testdata/hicseq/capture.ini -r readsets.HiC010.tsv -s 1-17 -t capture -e DpnII > hicseqCom-
mands_capture.sh
```

```
rnaseq_light.py -c $MUGQIC_PIPELINES_HOME/pipelines/rnaseq_light/rnaseq_light.base.ini
$MUGQIC_PIPELINES_HOME/pipelines/rnaseq_light/rnaseq_light.guillimin.ini -r readset.rnaseq.txt -d de-
sign.rnaseq.txt -s 1-6 > rnaseqLightCommands.sh
```

```
rnaseq_denovo_assembly.py -c $MUGQIC_PIPELINES_HOME/pipelines/rnaseq_denovo_assembly/rnaseq_denovo_assembly.base.ini
$MUGQIC_PIPELINES_HOME/pipelines/rnaseq_denovo_assembly/rnaseq_denovo_assembly.guillimin.ini -r read-
set.rnaseq.txt -d design.rnaseq.txt -s 1-23 > rnaseqDeNovoCommands.sh
```

```
methyleseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/methyleseq/methyleseq.base.ini
$MUGQIC_PIPELINES_HOME/pipelines/methyleseq/methyleseq.guillimin.ini -r readset.methyleseq.txt -s 1-14 >
methyleseq.sh
```

```
pacbio_assembly.py -c $MUGQIC_PIPELINES_HOME/pipelines/pacbio_assembly/pacbio_assembly.base.ini
$MUGQIC_PIPELINES_HOME/pipelines/pacbio_assembly/pacbio_assembly.guillimin.ini -r readset.pacbio.txt -s
1-12 > pacbioCommands.sh
```

```
ampliconseq.py -c $MUGQIC_PIPELINES_HOME/pipelines/ampliconseq/ampliconseq.base.ini
$MUGQIC_PIPELINES_HOME/pipelines/ampliconseq/ampliconseq.guillimin.ini -r readset.ampliconseq.txt -s
1-32 > ampliconseqCommands.sh
```

## 1.8 GenPipes Support

GenPipes is released as open source software. For details about its license policy, see [GenPipes License](#). The GenPipes developers offer continuous support through [GenPipes Google Forum](#). You can drop an email to the [GenPipes Help Desk](#) for any support issues.

GenPipes is sponsored by [Canadian Centre for Computational Genomics \(C3G\)](#). Besides the Google Forum and Help Desk, you can attend [C3G Open Door Sessions](#). This facility is available only to local (McGill University, Montréal) bioinformaticians, students and researchers. You need to fill a form and RSVP in order to attend a session. For details visit [C3G Open Door link](#).

### 1.8.1 Reporting GenPipes Bug

If you are facing any issues in using or deploying GenPipes, you can report bugs through [GenPipes Support Email](#).

---

#### Note:

- Messages should not be sent directly to our team members. The generic e-mail addresses above are viewable by all of us and facilitate the follow-up of your request.
  - Choose a meaningful subject for your message.
  - Include the pipeline version number in your message (and the commit number if applicable).
  - Provide the following information relevant to the problem encountered: the python command, the bash submission script, the output (job\_outputs//o) file,
  - An error message or code snippet illustrating your request is normally very useful.
-

## 1.9 Contributing to GenPipes

The GenPipes project team welcomes contributions from the community!

We are thrilled to receive your inputs and try to process them as fast as we can.

There are multiple ways in which you, as a user of GenPipes can contribute.

1. GenPipes User feedback and inputs related to usage, documentation or issues, if any.
2. GenPipes Developer inputs in the form of code fixes and new feature requests, forks and code evolutions
3. Technical Writer contributions in the form of improvisations, new content etc.

### 1.9.1 GenPipes Usage related Contributions

For all GenPipes usage related feedback and contributions, please visit *[Get involved with GenPipes](#)*.

### 1.9.2 GenPipes Developer Contributions

Please refer to the *[GenPipes Developers Guide](#)* for guidelines on how to submit code, make feature requests and change requests into GenPipes sources.

### 1.9.3 Technical Writer Contributions

If you are a technical writer and would like to submit your contributions for GenPipes documentation, please refer to the guidelines regarding *[contributing to GenPipes documentation](#)*.

Thank you for your inputs!

## 1.10 Channels

This document contains information for GenPipes contributors related to collaboration means and channels, slack, forum and other similar avenues for active community interaction.

- [GenPipes Google Forum](#)
- [Email GenPipes](#)
- *[GenPipes Support](#)*

To subscribe to [GenPipes Google Forum](#), drop us a mail to [Join GenPipes Google Forum](#)

## 1.11 Citing and Citations

This section gives instructions on how to cite GenPipes Publications and lists examples of citing articles.



### 1.11.1 GenPipes Citation

When citing GenPipes for a publication, please cite the following article in you paper:

#### Citation

Mathieu Bourgey, Rola Dali, Robert Eveleigh, Kuang Chung Chen, Louis Letourneau, Joel Fillon, Marc Michaud, Maxime Caron, Johanna Sandoval, Francois Lefebvre, Gary Leveque, Eloi Mercier, David Bujold, Pascale Marquis, Patrick Tran Van, David Anderson de Lima Morais, Julien Tremblay, Xiaojian Shao, Edouard Henrion, Emmanuel Gonzalez, Pierre-Olivier Quirion, Bryan Caron, Guillaume Bourque, GenPipes: an open-source framework for distributed and scalable genomic analyses, *GigaScience*, Volume 8, Issue 6, June 2019, giz037, <https://doi.org/10.1093/gigascience/giz037>

You can download the citation from the [Download section](#) below. A complete listing of all GenPipes publications is available in the [Publications](#) section under GenPipes Resources Navigation link on the left hand side of this documentation.

### 1.11.2 Citation Examples

1. (June 2019) Colorectal cancer-derived extracellular vesicles induce transformation of fibroblasts into colon carcinoma cells. See [PDF here](#).
2. (May 2018) Whole Genome Sequencing for Mutation Discovery in Rare Neuro-developmental Disorders. [Refer to the PDF here](#).
3. (June 2019) GenPipes: an open-source framework for distributed and scalable genomic analysis. *GigaScience*, Volume 8, Issue 6,.
4. BioRxiv GenPipes Paper [DOI](#).

### 1.11.3 Download GenPipes Citation

- Plain Text.
- Bibtex.
- PDF.

### 1.11.4 Download Method Descriptions

Following are the methods descriptions for the DNA-seq and RNA-seq pipelines. GenPipes users can use these descriptions of methods in their manuscripts.

- Refer to the method description for *DNA Sequencing Pipeline*
- Refer to the method description for *RNA Sequencing Pipeline*

## 1.12 GenPipes Publications

This document contains information on available current publications and other media such as scientific papers, citations, videos etc. that can help new GenPipes users build a better understanding of GenPipes and how it works.

Papers

Videos

Presentations

1. GigaScience Article June 2019: [GenPipes: an open-source framework for distributed and scalable genomic analysis](#).
2. Biorxiv Archives Nov 2018 : [GenPipes Paper](#).
1. How to run C3G-GenPipes on Compute Canada cluster - see video below.
2. [Amplicon Sequencing](#) generates metagenomics output. The [Metagenomics Figures video](#) shows how to build custom OTU figures and tables from C3G's metagenomics output.
3. [Amplicon Sequencing](#) pipeline Trimmomatic16S Step generates 16S analysis data that can be examined and plotted as demonstrated in [16S analysis video](#).
1. ChIP Sequencing Slides.
2. DNA Sequencing Slides.
3. RNA Sequencing Slides.

## 1.13 GenPipes Workshops

This document contains GenPipes training resources from recent workshops. Canadian Centre for Computational Genomics (C3G) facilitates, organizes and collaborates with establishments to assist interested folks in acquisition of knowledge in bioinformatics analysis. For latest upcoming workshop schedules see [C3G Training Page](#).

2019

2018

1. [RNA Sequencing Analysis Workshop Slides](#)
2. [RNA Testdata](#).
1. [Computational Epigenetics Workshop - Slides](#)
2. [ChipSeq Download](#)
3. [WGBS Download](#).

## 1.14 GenPipes Test Datasets

You can execute various GenPipes Pipelines using the following types of data:

- The real data which is generated from your genomic analysis instruments and then measured, sampled and read into various specified bioinformatics data formats.
- Test datasets that are available in the absence of real genomic analysis data.

**Test Dataset** in the context of GenPipes refers to the dataset that needs to be analyzed by one of the GenPipes Pipelines. It can either be real data or sample data that is used to run the pipeline. Test dataset refers to some dataset that user can use to have hands-on on the pipeline. It is typically a smaller datasets (for e.g., one chromosome only and few samples for instance) so that the test runs of the pipelines using sample data get completed quickly say for demonstration purposes.

Test dataset is different from readset file which is input to the pipeline. For other kinds of inputs required for GenPipes pipelines, see [here](#).

In contrast to the test dataset, a Readset File in the context of GenPipes actually describes the dataset (test dataset or real dataset) so that the pipeline can understand the type of data and process it. Readset file is provided as input to almost all the GenPipes pipelines. Readset file contains information about the data to analyze; the path of the raw files, the type of sequencing, the name of the samples, etc.

**Note:** Please remember to use the correct dataset for the respective GenPipes pipelines. The table below lists the test dataset download link for each of the GenPipes pipeline. Do not use the test dataset specified for a different pipeline.

<i>GenPipes Pipeline</i>	<i>Test Dataset</i>
<i>HiC Pipeline</i>	<a href="#">Download HiC Pipeline Dataset</a>
<i>Amplicon Seq</i>	<a href="#">Download Amplicon Seq Dataset</a>
<i>ChIP Seq</i>	<a href="#">Download ChIP Seq Dataset</a>
<i>DNA Seq</i>	<a href="#">Download DNA Seq Dataset</a>
<i>RNA Seq</i>	<a href="#">Download RNA Seq Dataset</a>
<i>Methyl Seq</i>	<a href="#">Download Methyl Seq Dataset</a>
<i>PacBio Seq</i>	<a href="#">Download PacBio Seq Dataset</a>
<i>TumorPair Seq</i>	<a href="#">Download TumorPair Seq Dataset</a>

**Warning:** PacBio Sequencing Pipeline is no longer available in GenPipes Release 3.2.0 and beyond.

TumorPair Sequencing Pipeline is under major overhaul and not available *only* in GenPipes Release 3.2.0

### 1.14.1 Test Dataset Usage Examples

For various GenPipes pipelines, you can refer to usage examples and commands for issuing pipeline jobs using various options in the individual pipeline reference guide listed above or a short summary [here](#).

### 1.14.2 Bioinformatic resources

If you are looking for Bioinformatic resources such as available genomes with FASTA sequence, aligner indices and annotation files listed on [Bioinformatics resources](#) C3G website page, you can download those from the public repositories using scripts provided in [GenPipes Repository](#).

You can also download the latest test datasets from Computational Genomics website [download page](#).

### 1.14.3 Latest available datasets

## 1.15 Compute Resources for GenPipes

This document lists compute resources that are available from C3G for GenPipes users.

### 1.15.1 GenPipes Pre-deployed on Compute Canada Servers

Researchers who have access to Compute Canada resources need not deploy GenPipes for genomic analysis. They can simply login and access Compute Canada servers that have pre-installed stable release of GenPipes.

If you are an external user, who does not have access to Compute Canada data centre compute resources, you can apply for your CCDB account. Once you have that account, you can access GenPipes pre-deployed on Compute Canada servers. For details on how to obtain an account and get started with using pre-deployed GenPipes, see [here](#).

### 1.15.2 GenPipes Bioinformatics Modules

C3G, in partnership with Compute Canada, offers and maintains a large set of bioinformatics resources for the community. Below is a list of software currently deployed on several HPC centres, including Guillimin, Cedar, Graham and Mammouth. Several reference genomes are also available.

To access these modules, you need to add the following lines to your .bashrc file.

```
## GenPipes/MUGQIC genomes and modules
export MUGQIC_INSTALL_HOME=/cvmfs/soft.mugqic/CentOS6
module use $MUGQIC_INSTALL_HOME/modulefiles
```

To explore the available tools, you can type:

module avail mugqic/ To load a module, for example samtools, you can use:

```
# module add mugqic/<tool>/<version>
module add mugqic/samtools/1.4.1
# Now samtools 1.4.1 is available to use. To check:
samtools
```

## Available Modules

*Anaconda, ASCAT, Aspera Connect, ATLAS, BCFtools, Beagle, bedtools, bismark, BisSNP, BLAST, boost, Bowtie, Bowtie2, BbreakDancer, butter, bvatools, BWA, Bwakit, CD-HIT, Cell Ranger, Cufflinks, duk, ea-utils, emboss, EPACTS, Exonerate, FastQC, FastTree, FASTX-Toolkit, FLASH, gcc, GEMINI, GEM, Genome Analysis Toolkit, Ghostscript, Gnuplot, HMMER, HOMER, HTSlib, IGV, igvtools, Java, JELLYFISH, kentUtils, KmerGenie, KronaTools, LAPACK, Long Ranger, MACS, MACS2, miRDeep2, mpich, mugqic\_pipelines, mugqic\_R\_packages, mugqic\_tools, MUMmer, MUSCLE, MuTect, NextClip, OpenBLAS, Pandoc, parallel, pbs-drmaa, perl, Picard, pigz, PRINSEQ-lite, Prodigal, Python, qualimap, R\_Bioconductor, Ray, RNAmmer, RNA-SeQC, RSEM, SAMtools, Scalpel, ShortStack, SignalP, SMRT-Analysis, SNAP, SnpEff, Sphinx, SPAdes, SplAdder, STAR, Supernova, Tabix, TMHMM, tools, TopHat, TransDecoder, Trimmomatic, Trinity, Tinotate, UCSC tools, USEARCH, VarScan, VCFtools, Verify-BamID, ViennaRNA, VSEARCH, vt, WebLogo, Celera Assembler*

## List of Modules

### Anaconda

Anaconda is a freemium open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. Versions Available: 2-4.0.0

### ASCAT

A tool for accurate dissection of genome-wide allele-specific copy number in tumors. Versions Available: 2.3

### Aspera Connect

High-performance transfer plug-in Versions Available: 3.3.3

### ATLAS

A data warehouse for integrative bioinformatics Versions Available: 3.10.2

### BCFtools

Utilities for variant calling and manipulating VCFs and BCFs Versions Available: 1.2, 1.3

### Beagle

Beagle is a software package that performs genotype calling, genotype phasing, imputation of ungenotyped markers, and identity-by-descent segment detection. Versions Available: 03May16.862, r13994

## **bedtools**

A software suite for the comparison, manipulation and annotation of genomic features in browser extensible data (BED) and general feature format (GFF) format. Versions Available: 2.17.0, 2.22.1, 2.25.0, 2.26.0

## **bismark**

Bismark is a program to map bisulfite treated sequencing reads to a genome of interest and perform methylation calls in a single step. Versions Available: 0.16.1, 0.16.3, 0.17.0

## **BisSNP**

A bisulfite space genotyper & methylation caller Versions Available: 0.82.2

## **BLAST**

Basic Local Alignment Search Tool Versions Available: 2.2.29+, 2.3.0+

## **boost**

Boost provides free portable peer-reviewed C++ libraries. The emphasis is on portable libraries which work well with the C++ Standard Library. Versions Available: 1.63.0

## **Bowtie**

An ultra fast, memory-efficient short read aligner. Versions Available: 1.0.0, 1.1.2, 1.2.0

## **Bowtie2**

Bowtie 2 is an ultra fast and memory-efficient tool for aligning sequencing reads to long reference sequences. Versions Available: 2.2.3, 2.2.4, 2.2.9

## **BbreakDancer**

A Perl/C++ package that provides genome-wide detection of structural variants from next generation paired-end sequencing reads. Versions Available: 1.1\_2011\_02\_21

### **butter**

A wrapper for Bowtie to produce small RNA-seq alignments where multi-mapped small RNAs tend to be placed near regions of confidently high density. Versions Available: 0.3.3

### **bvatools**

BVATools – Bam and Variant Analysis Tools Versions Available: 1.3, 1.4, 1.5, 1.6

### **BWA**

A software package for mapping low-divergent sequences against a large reference genome, such as the human genome. Versions Available: 0.7.10, 0.7.12

### **Bwakit**

Bwakit is a self-consistent installation-free package of scripts and precompiled binaries, providing an end-to-end solution to read mapping. Versions Available: 0.7.12

### **CD-HIT**

CD-HIT is a very widely used program for clustering and comparing protein or nucleotide sequences. Versions Available: 4.5.4-2011-03-07

### **Cell Ranger**

Cell Ranger is a set of analysis pipelines that processes Chromium single cell 3â€™ RNA-seq output to align reads, generate gene-cell matrices and perform clustering and gene expression analysis. Versions Available: 1.3.0

### **Cufflinks**

Cufflinks assembles transcripts, estimates their abundances, and tests for differential expression and regulation in RNA-Seq samples. Versions Available: 2.2.1

### **duk**

Duk is a fast, accurate, and memory efficient DNA sequence matching tool. It finds whether a query sequence partially or totally matches given reference sequences or not, but it does not give how a query matches a reference sequence. The common application is to group sequencing reads into small manageable chunks for downstream analysis in assessing quality of a sequencing run, which includes contaminant removal (with contaminant sequences known), organelle genome separation, and assembly refinement. Versions Available: 1.1

### **ea-utils**

A command-line tools for processing biological sequencing data. Barcode demultiplexing, adapter trimming, etc. Versions Available: 1.1.2-537

### **emboss**

EMBOSS is ‘The European Molecular Biology Open Software Suite’. EMBOSS is a free Open Source software analysis package specially developed for the needs of the molecular biology (e.g. EMBnet) user community. Versions Available: 6.4.0, 6.6.0

### **EPACTS**

A versatile software pipeline to perform various statistical tests for identifying genome-wide association from sequence data through a user-friendly interface, both to scientific analysts and to method developers. Versions Available: 3.2.6

### **Exonerate**

A generic tool for pairwise sequence comparison. Versions Available: 2.2.0

### **FastQC**

A quality control tool for high throughput sequence data. Versions Available: 0.11.2, 0.11.5

### **FastTree**

FastTree infers approximately-maximum-likelihood phylogenetic trees from alignments of nucleotide or protein sequences. Versions Available: 2.1.7

### **FASTX-Toolkit**

The FASTX-Toolkit is a collection of command line tools for Short-Reads FASTA/FASTQ files preprocessing. Versions Available: 0.0.14

### **FLASH**

FLASH (Fast Length Adjustment of Short reads) is a very fast and accurate software tool to merge paired-end reads from next-generation sequencing experiments. FLASH is designed to merge pairs of reads when the original DNA fragments are shorter than twice the length of reads. The resulting longer reads can significantly improve genome assemblies. Versions Available: 1.2.8, 1.2.11



## **gcc**

The GNU Compiler Collection includes front ends for C, C++, Objective-C, Fortran, Java, Ada, and Go, as well as libraries for these languages (libstdc++, libgccj,...). GCC was originally written as the compiler for the GNU operating system. Versions Available: 4.9.3

## **GEMINI**

Flexible framework for exploring genetic variation in the context of the wealth of genome annotations available for the human genome. Versions Available: 0.18.0, 0.18.2, 0.18.3

## **GEM**

The GEM library strives to be a true ‘next-generation’ tool for handling any kind of sequence data, offering state-of-the-art algorithms and data structures specifically tailored to this demanding task. Versions Available: v1.315

## **Genome Analysis Toolkit**

Developed by the Data Science and Data Engineering group at the Broad Institute, the toolkit offers a wide variety of tools with a primary focus on variant discovery and genotyping. Versions Available: 3.2-2, 3.3-0, 3.5, 3.7

## **Ghostscript**

An interpreter for the PostScript language and for PDF. Versions Available: ‘8.70’

## **Gnuplot**

Gnuplot is a portable command-line driven graphing utility for Linux, OS/2, MS Windows, OSX, VMS, and many other platforms. Versions Available: 4.6.4, 4.6.6

## **HMMER**

HMMER is used for searching sequence databases for sequence homologs, and for making sequence alignments. It implements methods using probabilistic models called profile hidden Markov models (profile HMMs). Versions Available: 2.3.2, 3.1b1, 3.1b2

## **HOMER**

HOMER offers tools and methods for interpreting Next-gen-Seq experiments. In addition to Genome Browser/UCSC visualization support and peak finding [and motif finding of course], HOMER can help assemble data across multiple experiments and look at positional specific relationships between sequencing tags, motifs, and other features. You do not need to use the peak finding methods in this package to use motif finding. Versions Available: 4.7

## **HTSlib**

A C library for reading/writing high-throughput sequencing data Versions Available: 1.2.1, 1.3

## **IGV**

The Integrative Genomics Viewer (IGV) is a high-performance visualization tool for interactive exploration of large, integrated genomic datasets. Versions Available: 2.3.23

## **igvtools**

The igvtools utility provides a set of tools for preprocessing data files. File names must contain an accepted file extension, e.g. test-xyz.bam. Versions Available: 2.3.14, 2.3.67

## **Java**

Java technology is the foundation of most networked applications and is used worldwide to develop and deliver mobile and nested applications, games, web content and enterprise software. Versions Available: openjdk-jdk1.6.0\_38, openjdk-jdk1.7.0\_60, openjdk-jdk1.8.0\_72

## **JELLYFISH**

JELLYFISH is a tool for fast, memory-efficient counting of k-mers in DNA. Versions Available: 2.1.3

## **kentUtils**

UCSC command-line bioinformatic utilities, implemented by Jim Kent Versions Available: 302.1.0

## **KmerGenie**

KmerGenie estimates the best k-mer length for genome de Novo assembly. Versions Available: 1.5692

## **KronaTools**

Krona Tools is a set of scripts to create Krona charts from several Bioinformatics tools as well as from text and XML files. Versions Available: 2.6.1

## **LAPACK**

LAPACK provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. Versions Available: 3.5.0

## **Long Ranger**

Long Ranger is a set of analysis pipelines that processes Chromium sequencing output to align reads and call and phase SNPs, indels, and structural variants. Versions Available: 2.1.2

## **MACS**

Model-based Analysis of ChIP-Seq (MACS) on short reads sequencers such as Genome Analyzer (Illumina / Solexa) Versions Available: 2.0.10.09132012

## **MACS2**

Novel algorithm, named Model-based Analysis of ChIP-Seq (MACS), for identifying transcript factor binding sites. Versions Available: 2.1.0.20140616, 2.1.0.20151222, 2.1.1.20160309

## **miRDeep2**

miRDeep2 is a completely overhauled tool which discovers microRNA genes by analyzing sequenced RNAs. The tool reports known and hundreds of novel microRNAs with high accuracy in seven species representing the major animal clades. The low consumption of time and memory combined with user-friendly interactive graphic output makes miRDeep2 accessible for straightforward application in current research. Versions Available: 0.0.8

## **mpich**

MPICH is a high performance and widely portable implementation of the Message Passing Interface (MPI) standard. Versions Available: 3.1.4

## **mugqic\_pipelines**

MUGQIC pipelines consist of Python scripts which create a list of jobs running Bash commands. Those scripts support dependencies between jobs and smart restart mechanism if some jobs fail during pipeline execution. Jobs can be submitted in different ways: by being sent to a PBS scheduler like Torque or by being run as a series of commands in batch through a Bash script Versions Available: 2.0.1, 2.0.2, 2.1.0, 2.1.1, 2.2.0, 2.2.1

## **mugqic\_R\_packages**

This library implements various -seq downstream analysis, as well as Nozzle-based reporting for mugqic\_pipelines. Versions Available: 1.0.1, 1.0.2, 1.0.3, 1.0.4

## **mugqic\_tools**

Perl, python, R, awk and sh scripts use in several bioinformatics pipelines of the MUGQIC PIPELINE. Versions Available: 2.0.2, 2.0.3, 2.1.0, 2.1.1, 2.1.3, 2.1.4, 2.1.5, 2.1.6, 2.1.7

## **MUMmer**

Ultra-fast alignment of large-scale DNA and protein sequences Versions Available: 3.23

## **MUSCLE**

Program for creating multiple alignments of protein sequences. Versions Available: 3.8.31

## **MuTect**

Reliable and accurate identification of somatic point mutations in next generation sequencing data of cancer genomes Versions Available: 1.1.6

## **NextClip**

Tool for analyzing reads from LMP libraries, generating a comprehensive quality report and extracting good quality trimmed and deduplicated reads Versions Available: b833dd9

## **OpenBLAS**

Optimized BLAS library based on GotoBLAS2 1.13 BSD version Versions Available: 0.2.14, 0.2.17

## **Pandoc**

Universal document converter Versions Available: 1.13.1, 1.15.2

## **parallel**

Shell tool for executing jobs in parallel using one or more computers Versions Available: 20130822

**pbs-drmaa**

DRMAA for Torque/PBS Pro is implementation of Open Grid Forum DRMAA (Distributed Resource Management Application API) specification for submission and control jobs to PBS systems Versions Available: 1.0.18

**perl**

Feature-rich programming language Versions Available: 5.18.2, 5.22.1

**Picard**

Set of tools (in Java) for working with next generation sequencing data in the BAM format Versions Available: 1.118, 1.123, 2.0.1

**pigz**

Replacement for gzip that exploits multiple processors and multiple cores when compressing data Versions Available: 2.3

**PRINSEQ-lite**

Used to filter, reformat, or trim your Genomic and Metagenomic sequence data Versions Available: 0.20.3, 0.20.4

**Prodigal**

Prodigal (Prokaryotic Dynamic Programming Gene finding Algorithm) is a microbial (bacterial and archaeal) gene finding program developed at Oak Ridge National Laboratory and the University of Tennessee. Versions Available: 2.6.3

**Python**

Programming language that lets you work quickly and integrate systems more effectively Versions Available: 2.7.8, 2.7.10\_qiime, 2.7.11, 2.7.12, 2.7.13, 3.4.0, 3.5.2

**qualimap**

Qualimap is a platform-independent application written in Java and R that provides both a Graphical User Interface (GUI) and a command-line interface to facilitate the quality control of alignment sequencing data. Versions Available: 2.2.1

## **R\_Bioconductor**

face (GUI) and a command-line interface to facilitate the quality control of alignment sequencing data Versions Available: 3.1.2\_3.0, 3.2.3\_3.2

## **Ray**

Parallel genome assemblies for parallel DNA sequencing Versions Available: 2.3.1

## **RNAmmmer**

Predicts 5s/8s, 16s/18s, and 23s/28s ribosomal RNA in full genome sequences. Versions Available: 1.2

## **RNA-SeQC**

Java program which computes a series of quality control metrics for RNA-seq data Versions Available: 1.1.7, 1.1.8

## **RSEM**

Accurate quantification of gene and isoform expression from RNA-Seq data Versions Available: 1.2.12

## **SAMtools**

A suite of programs for interacting with high-throughput sequencing data. Versions Available: 0.1.19, 1.0, 1.2, 1.3, 1.3.1

## **Scalpel**

Software package for detecting INDELs (INsertions and DEletions) mutations in a reference genome Versions Available: 0.3.2, 0.5.2

## **ShortStack**

Tool developed to process and analyze small RNA-seq data with respect to a reference genome, and output a comprehensive and informative annotation of all discovered small RNA genes Versions Available: 3.3

## **SignalP**

Predicts the presence and location of signal peptide cleavage sites in amino acid sequences from different organisms Versions Available: 4.1

## **SMRT-Analysis**

Pacbio secondary analysis through a graphical or command-line user interface. Versions Available: 2.3.0.140936.p1, 2.3.0.140936.p2, 2.3.0.140936.p4, 2.3.0.140936.p5

## **SNAP**

General purpose gene finding program suitable for both eukaryotic and prokaryotic genomes Versions Available: ‘2013-11-29’

## **SnpEff**

Variant annotation and effect prediction tool. It annotates and predicts the effects of variants on genes Versions Available: 3.6, 4.0, 4.2

## **Sphinx**

Sphinx is a tool that makes it easy to create intelligent and beautiful documentation of Python projects Versions Available: master

## **SPAdes**

SPAdes “ St. Petersburg genome assembler ” is an assembly toolkit containing various assembly pipelines. Versions Available: 3.10.0

## **SplAdder**

Splicing Adder, a toolbox for alternative splicing analysis based on RNA-Seq alignment data. Briefly, the software takes a given annotation and RNA-Seq read alignments, transforms the annotation into a splicing graph representation, augments the splicing graph with additional information extracted from the read data, extracts alternative splicing events from the graph and quantifies the events. Versions Available: 1.0.0

## **STAR**

Spliced Transcripts Alignment to a Reference. Based on a previously undescribed RNA-seq alignment algorithm that uses sequential maximum mappable seed search in uncompressed suffix arrays followed by seed clustering and stitching procedure. Versions Available: 2.4.0f1, 2.5.0c, 2.5.1b, 2.5.2a, 2.5.2b

## **Supernova**

Supernova is a software package for de Novo assembly from Chromium Linked-Reads that are made from a single whole-genome library from an individual DNA source. Versions Available: 1.1.4

## **Tabix**

Tabix indexes a TAB-delimited genome position file in.tab.bgz and creates an index file ( in.tab.bgz.tbi or in.tab.bgz.csi ) when region is absent from the command-line. Versions Available: 0.2.6

## **TMHMM**

Predicting Transmembrane Protein Topology with a Hidden Markov Model Versions Available: 2.0c

## **tools**

Perl, Python, R, awk and sh scripts use in several bioinformatics pipelines of the MUGQIC PIPELINES repo. Versions Available: 1.10.4

## **TopHat**

TopHat is a fast splice junction mapper for RNA-Seq reads. It aligns RNA-Seq reads to mammalian-sized genomes using the ultra high-throughput short read aligner Bowtie, and then analyzes the mapping results to identify splice junctions between exons. Versions Available: 2.0.13, 2.0.14

## **TransDecoder**

TransDecoder identifies candidate coding regions within transcript sequences, such as those generated by de Novo RNA-Seq transcript assembly using Trinity, or constructed based on RNA-Seq alignments to the genome using Tophat and Cufflinks Versions Available: 2.0.1

## **Trimmomatic**

Trimmomatic performs a variety of useful trimming tasks for Illumina paired-end and single ended data. The selection of trimming steps and their associated parameters are supplied on the command line. Versions Available: 0.32, 0.35, 0.36

## **Trinity**

Trinity assembles transcript sequences from Illumina RNA-Seq data Versions Available: 20140413p1, 2.0.4, 2.1.1, 2.2.0



## Tinotate

A comprehensive annotation suite for functional annotation of transcriptomes, particularly de Novo assembled transcriptomes, from model or non-model organisms. Trinotate makes use of a number of different well referenced methods for functional annotation including homology search to known sequence data (BLAST+/SwissProt), protein domain identification (HMMER/PFAM), protein signal peptide and transmembrane domain prediction (signalP/tmHMM), and leveraging various annotation databases (eggNOG/GO/Kegg databases). Versions Available: 20131110, 2.0.1, 2.0.2

## UCSC tools

UCSC genome browser 'kent' bioinformatic utilities Versions Available: 20140212, v326

## USEARCH

Ultra-fast search for high-identity top hit or hits from sequence files Versions Available: 7.0.1090, 8.1.1861

## VarScan

VarScan is a platform-independent mutation caller for targeted, exome, and whole-genome resequencing data generated on Illumina, SOLiD, Life/PGM, Roche/454 and similar instruments. It can be used to detect different types of variation: Germline variants, multi-sample variants, somatic mutations and somatic copy number alterations Versions Available: 2.3.9

## VCFtools

A program package that can be used to perform the following operations on standard variants (VCF) files: Filter out specific variants Compare files Summarize variants Convert to different file types Validate and merge files Create intersections and subsets of variants Versions Available: 0.1.11, 0.1.14

## VerifyBamID

Verifies whether the reads in particular file match previously known genotypes for an individual (or group of individuals), and checks whether the reads are contaminated as a mixture of two samples. verifyBamID can detect sample contamination and swaps when external genotypes are available. When external genotypes are not available, verifyBamID still robustly detects sample swaps Versions Available: devMaster\_20151216

## ViennaRNA

The ViennaRNA Package consists of a C code library and several stand-alone programs for the prediction and comparison of RNA secondary structures. Versions Available: 2.3.0

## VSEARCH

VSEARCH supports de Novo and reference based chimera detection, clustering, full-length and prefix dereplication, reverse complementation, masking, all-vs-all pairwise global alignment, exact and global alignment searching, shuffling, subsampling and sorting. It also supports FASTQ file analysis, filtering and conversion. Versions Available: 1.11.1

## vt

A tool set for short variant discovery in genetic sequence data. Versions Available: 0.57

## WebLogo

A tool for creating sequence logos from biological sequence alignments. It can be run on the command line as a standalone webserver, as a CGI webapp, or as a python library. Versions Available: 2.8.2, 3.3

## Celera Assembler

A de Novo whole-genome shotgun (WGS) DNA sequence assembler. It reconstructs long sequences of genomic DNA from fragmentary data produced by whole-genome shotgun sequencing Versions Available: 8.1, 8.2, 0.0, 0.2

### 1.15.3 GenPipes Genome resources

C3G, in partnership with Compute Canada, maintains several genomes that are available on several HPC centres including Guillimin, Cedar, Graham and Mammouth. In addition to the fasta sequence, many genomes include aligner indices and annotation files.

To access these genomes, you need to add the following lines to your .bashrc file.

```
## GenPipes/MUGQIC genomes and modules
export MUGQIC_INSTALL_HOME=/cvmfs/soft.mugqic/CentOS6
```

To explore the available genomes, you can type:

```
ls $MUGQIC_INSTALL_HOME/genomes/species/
```

## Available Genomes

### Human

Species: Homo\_sapiens Available builds: GRCh38, GRCh37, hg19

**Mouse**

species: *Mus\_musculus* Available builds: GRCm38, mm10, mm9, NCBIM37

**Rat**

Species: *Rattus\_norvegicus* Available builds: rn5, Rnor\_5.0, Rnor\_6.0

**Monkey**

Species: *Macaca\_mulatta* Available builds: MMUL\_1

**Chimpanzee**

Species: *Pan\_troglodytes* Available builds: panTro4, CHIMP2.1.4

**Baboon**

Species: *Papio\_anubis* Available builds: PapAnu2.0

**Dog**

Species: *Canis\_familiaris* Available builds: CanFam3.1

**Cow**

Species: *Bos\_taurus* Available builds: UMD3.1

**Chicken**

Species: *Gallus\_gallus* Available builds: Galgal4

**Fly**

Species: *Drosophila\_melanogaster* Available builds: BDGP5

## C. Elegans

Species: *Caenorhabditis\_elegans* Available builds: WBcel235

## Yeast

Species: *Saccharomyces\_cerevisiae* Available builds: R64-1-1 Species: *Schizosaccharomyces\_pombe* Available builds: ASM294v2

## Bacteria

Species: *Escherichia\_coli\_str\_k\_12\_substr\_dh10b* Available builds: ASM1942v1 Species: *pseudomonas\_aeruginosa\_pa14* Available builds: Pseu\_aeru\_PA14\_V1 Species: *Pseudomonas\_aeruginosa\_UCBPP\_PA14* Available builds: ASM1462v1

## Plants

Species: *Arabidopsis\_thaliana* Available builds: TAIR10

## 1.16 Release Notes

The first release of GenPipes into open source was made in [2013](#). Since then, the number of GenPipes runs utilizing C3G HPC resources have grown significantly. With the release of version 2.0.0 in 2014, a community of users has run GenPipes to conduct approximately 3,000 analyses processing approximately 100,000 samples.

The current release of GenPipes is 3.6.2.

Following links point to the recent GenPipes release notes.

- GenPipes 3.6.2 released on Nov 11, 2021
- GenPipes 3.6.1 released on Oct 4, 2021
- GenPipes 3.6.0 released on Sept 8, 2021
- GenPipes 3.5.0 released on July 13, 2021
- GenPipes 3.4.0 released on May 3, 2021
- GenPipes 3.3.0 released on Feb 22, 2021
- GenPipes 3.2.0 released on Jan 26, 2021
- GenPipes 3.1.5 released on Jan 16, 2020
- GenPipes 3.1.4 released on Mar 26, 2019
- GenPipes 3.1.3 released on Dec 18, 2018
- GenPipes 3.1.2 released on Nov 22, 2018
- GenPipes 3.1.0 released on Apr 9, 2018
- GenPipes 3.0.0 released on Nov 30, 2017
- GenPipes 2.3.0 released on Feb 28, 2017

- GenPipes 2.2.1 released on Dec 19, 2016
- GenPipes 2.2.0 released on Feb 9, 2016
- GenPipes 2.1.1 released on Apr 14, 2015
- GenPipes 2.1.0 released on Feb 5, 2015
- GenPipes 2.0.2 released on Jan 13, 2015
- GenPipes 2.0.1 released on Dec 17, 2014
- GenPipes 2.0.0 released on Dec 12, 2014

For older GenPipes Release note information, refer to GenPipes [Release Notes Archives](#).

### 1.16.1 Where is the detailed ChangeLog?

A [ChangeLog \(CHANGELOG.md\)](#) is available in the repository.

Since we use git, there are many ways to get the details in many formats. One of our preferred ways is to use a script written by the author of the Ray assembler: Sébastien Boisvert, which lists the commits by tag and author:

<https://raw.githubusercontent.com/sebhtml/ray/master/scripts/dump-ChangeLog.sh>

Enjoy GenPipes! We'd love to hear your feedback and inputs!

## 1.17 Developer Guide

This guide provides information for developers who are interested in contributing to GenPipes sources.

1. Refer to GenPipes sources in BitBucket <https://bitbucket.org/mugqic/genpipes/src>
2. Use a local dev branch in order to make your changes / modifications / enhancements and issue a Pull Request from your local dev branch to GenPipes dev branch.
3. Your PR will be reviewed by GenPipes Dev team reviewers. If the changes are acceptable, those will be merged into the GenPipes/dev branch.
4. For any further queries regarding contributing to GenPipes, refer to [README.md file](#).
5. You may also wish to refer to the GenPipes Coding Guidelines

## 1.18 Troubleshooting Guide

This document contains most frequently encountered issues faced by developers and their fixes.

## 1.19 About this documentation

This documentation is continuously written, corrected, edited, and revamped by members of the GenPipes community. It is edited via text files in the [reStructuredText](#) markup language and then compiled into a static website/offline document using the open source [Sphinx](#) and [ReadTheDocs](#) tools.

You may also want to refer to the [GenPipes New Documentation Architecture Map](#) for more insights on how GenPipes documentation is organized.

Your feedback is valuable to us.

For any documentation feedback, queries or any issues, please refer to [GenPipes Documentation Project](#) and open issues with the following label:

Thank you for your valuable inputs.

---

**Note:** You can contribute to GenPipes documentation.

---

### 1.19.1 How to contribute to GenPipes documentation?

We are happy to receive your contributions!

Before you begin to make edits to GenPipes documentation, we would strongly recommend that you discuss and share your plans through a GitHub issue with the GenPipes documentation owners, especially for more ambitious contributions. This gives other contributors a chance to guide you in the right direction, provide you with relevant feedback and help you find out if someone else is already working on that part of documentation.

Here are some guidelines to contribute to GenPipes documentation:

- Familiarize yourself with the organization and structure of GenPipes documentation in terms of how it is laid out at Read the Docs, TOC and information flow. Also, refer to the [GenPipes documentation map](#) to understand how the content sources are organized.
- Use references within the text instead of duplicating information. Use search feature to identify the keyword and their mapping in the TOC.
- Follow the GenPipes Writing style guidelines while adding or updating content into GenPipes documentation.
- Before you delete any content, make sure it is not relevant in some user context.
- Make sure, any content you add is targeted and tailored to the needs of GenPipes audience - new users or seasoned GenPipes users.

### 1.19.2 References for Sphinx based documentation

- Using Sphinx and ReadTheDocs for building documentation of Python projects - a very informative [reference](#).

## 1.20 GenPipes Documentation Map

GenPipes has several documents to help users. This document describes how GenPipes documentation is laid out and organized to assist the user, be it a new GenPipes user or a seasoned one, as well as documentation contributor (technical writer, content developer, author) point of view.

If you are a GenPipes user, refer to the User's map below that will help you know where to look for things. For documentation contributors there is a separate map which provides a high level overview of how the content sources are organized and find your way to editing or updating them for the end user.

### 1.20.1 GenPipes Documentation User's Map

This map highlights various sections of GenPipes documentation indicating which ones are targeted at new users and those that are meant for seasoned GenPipes users and developers.

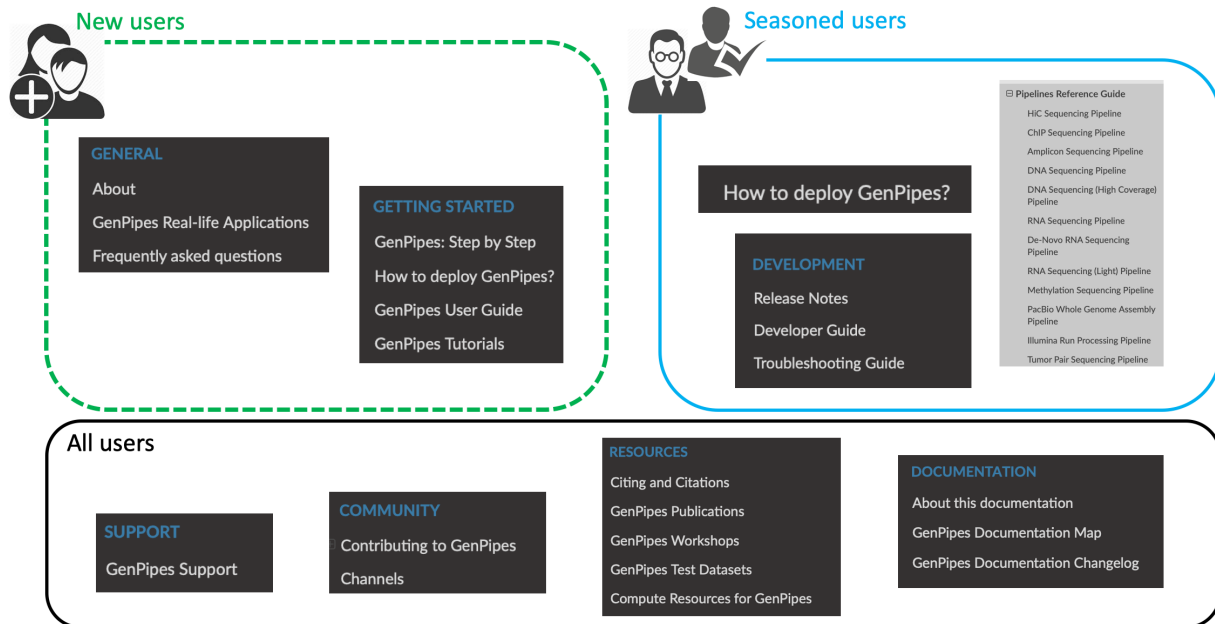


Fig. 27: Figure: GenPipes Documentation Map for Users

### 1.20.2 GenPipes Documentation Contributor's Map

This map is meant for GenPipes documentation contributors. It highlights the GenPipes source code directories and the documentation components that are generated using sources in the specific folders as shown by the arrows.

The tip of the yellow arrow points to the sub-folder that contains restructured text and markdown files used to generate that particular section of the document visible in the navigation bar.

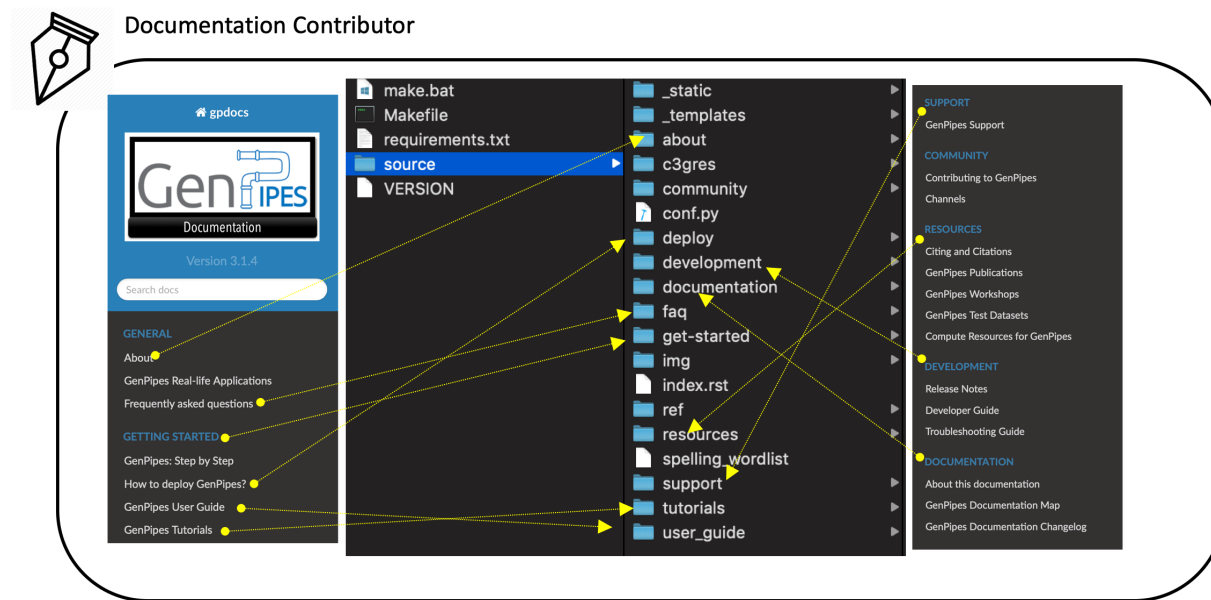


Fig. 28: Figure: GenPipes Documentation Map for Documentation Contributors

## 1.21 GenPipes Documentation Changelog

GenPipes documentation is continually being improved. The current release corresponds to GenPipes version x.y.z. This is the first draft of GenPipes RTD documentation.

### 1.21.1 What's new?

- New improved documentation framework at RTD to address both new and existing GenPipes users.
- Newer infographics oriented as opposed to text only information. Use of rich media in the future to make it more interesting and helpful for new users to find GenPipes information and learn about GenPipes and its usage.

### 1.21.2 Summary of Recent Changes

- Update RN link before retagging 3.6.2 minus epiqc link by shaloo at 2021-12-16 09:21:32
- Merge remote-tracking branch 'origin/gp362' by shaloo at 2021-12-16 09:18:03
- Merge remote-tracking branch 'origin/gp362' by shaloo at 2021-12-16 09:14:53
- Version update to 3.6.2 before retagging by shaloo at 2021-12-16 09:12:16
- Refs #154 Fix version before retagging by shaloo at 2021-12-16 08:44:21
- Remove 3.6.2 reference from release note before retagging by shaloo at 2021-12-16 08:42:49
- Merge remote-tracking branch 'origin/gp361' by shaloo at 2021-12-16 08:20:52
- Merge branch 'gp361' of https://github.com/c3g/GenPipes into gp361 by shaloo at 2021-12-16 08:08:47
- Refs #154 fix linux build issue by shaloo at 2021-12-16 07:53:02



- **Refs #154 epiqc removed from 3.6.1 reference guide** by *shaloo* at 2021-12-16 04:31:10

Refs #154 linux build fix using default latest sphinx and plugins

Refs #154 linux build failure fix in github action

Refs #154 linux pr-action issue fix

### 1.21.3 Commit Tracker

#### Commit

8822aecffc

**Warning:** There were uncommitted changes when this was compiled.

**Warning:** There were untracked files when this was compiled.