# uap Documentation

***Release 0.1.1***

**Christoph Kämpf, Michael Specht**

**Dec 12, 2018**

# Contents

**Authors:**

Christoph Kämpf, Michael Specht, Sven-Holger Puppel, Alexander Scholz, Gero Doose, Kristin Reiche, Jana Hertel, Jörg Hackermüller

**Description:**

**uap** executes, controls and keeps track of the analysis of large data sets. It enables users to perform robust, consistent, and reproducible data analysis. **uap** encapsulates the usage of (bioinformatic) tools and handles data flow and processing during an analysis. Users can use predefined or self-made analysis steps to create custom analysis. Analysis steps encapsulate best practice usages for bioinformatic software tools. **uap** focuses on the analysis of high-throughput sequencing (HTS) data. But its plugin architecture allows users to add functionality, such that it can be used for any kind of large data analysis.

**Usage:**

**uap** is a command-line tool, implemented in Python. It requires a user-defined configuration file, which describes the analysis, as input.

**Supported Platforms:**

- Unix-like operating systems.
- High Performance Compute (HPC) cluster systems such as , and .
- see *Tested Platforms* for detailed information

**Important Information**

**uap** does **NOT** include all tools necessary for the data analysis. It expects that the required tools are **already installed**.

CHAPTER 1

Table of contents

## 1.1 Introducing uap

### 1.1.1 Core aspects

#### Robustness

- Data is processed in temporary location. If and only if ALL involved processes exited graceful, the output files are copied to the final output directory.
- The final output directory names are suffixed with a hashtag which is based on the commands executed to generate the output data. Data is not easily overwritten and this helps to check for necessary recomputations.
- Processing can be aborted and continued from the command line at any time. Failures during data processing do not lead to unstable state of analysis.
- Errors are reported as early as possible, fail-fast. Tools are checked for availability, and the entire processing pipeline is calculated in advance before jobs are being started or submitted to a cluster.

#### Consistency

- Steps and files are defined in a directed acyclic graph (DAG). The DAG defines dependencies between in- and output files.
- Prior to any execution the dependencies between files are calculated. If a file is newer or an option for a calculation has changed all dependent files are marked for recalculation.

#### Reproducibility

- Comprehensive annotations are written to the output directories. They allow for later investigation of errors or review of executed commands. They contain also versions of used tool, required runtime, memory and CPU usage, etc.

#### Usability

- Single configuration file describes entire processing pipeline.
- Single command-line tool interacts with the pipeline. It can be used to execute, monitor, and analyse the pipeline.

## 1.2 Software Design

**uap** is designed as a plugin architecture. The plugins are called steps because they resemble steps of the analysis.

### 1.2.1 Source and Processing Steps: Building Blocks of the Analysis

There are two different types of steps: source and processing steps. **Source steps** are used to include data from outside the destination path (see *destination_path Section*) into the analysis. **Processing steps** are blueprints that describe how to process input to output data. Processing steps describe what needs to be done on an abstract level. **uap** controls the ordered execution of the steps as defined in the *analysis configuration file*.

### 1.2.2 Runs: Atomic Units of the Analysis

Steps define **runs** which represent the concrete commands for a part of the analysis. You can think of steps as objects and runs as instances like in object-oriented programming. A **run** is an atomic unit of the analysis. It can only succeed or fail entirely. Typically a single run computes data of a single sample. Runs compute output files from input files and provide these output files to subsequent steps via so called **connections**.

### 1.2.3 Connections: Propagation of Data

**Connections** are like tubes that connect steps. A step can have any number of connections. Run have to assign output file(s) to each connection of the step. If a run can not assign files to a connection it has to define it as empty. Downstream steps can access the connections to get the information which run created which file. The names of the connections can be arbitrarily chosen. The name should **not** be just the file format of the contained files but a description of their content. For example an `out/alignment` can contain gzipped SAM and/or BAM files. That's why the file type is often checked in steps and influences the issued commands or set parameters.

### 1.2.4 Analysis as Directed Acyclic Graph

The steps and connections are the building blocks of the analysis graph. Steps are the nodes and connections are the edges of the analysis graph. That graph has to be a directed acyclic graph (DAG). This implies that every step has one or more parent steps, which may in turn have parents themself. The analysis graph is not allowed to contain cycles. Steps without parents have to be source steps. They provide the initial input data, like for example FASTQ files with raw sequencing reads, genome sequences, genome annotations, etc..

## 1.3 Recommended uap Workflow

The recommended workflow to analyse data with **uap** is:

1. Install **uap** (see *Installation of uap*)
2. Optionally: Extend **uap** by adding new steps (see *Add New Functionality*)

3. Write a configuration file to setup your analysis (see *Analysis Configuration File*)
4. Start the analysis locally (see *run-locally*) or submit it to a cluster (see *submit-to-cluster*)
5. Follow the progress of the analysis (see *status*)
6. Share your extensions with the public (send us a pull request via github)

A **finished** analysis leaves the user with:

- *The original input files* (which are, of course, left untouched).
- *The experiment-specific configuration file* (see *Analysis Configuration File*). You should keep this configuration file for later reference and you could even make it publicly available along with your input files for anybody to re-run the entire data analysis or parts thereof.
- *The output files and comprehensive annotations of the analysis*

**(see *Annotation Files*).** These files are stored in the destination path defined in the configuration file.

## 1.4 Quick Start uap

At first, you need to install **uap** (see *Installation of uap*). After successfully finishing the installation of **uap** example analysis can be found in the folder `example-configurations`.

Let's jump head first into **uap** and have a look at some examples:

```
$ cd <uap-path>/example-configurations/
$ ls *.yaml
2007-CD4+_T_Cell_ChIPseq-Barski_et_al_download.yaml
2007-CD4+_T_Cell_ChIPseq-Barski_et_al.yaml
2014-RNA_CaptureSeq-Mercer_et_al_download.yaml
2014-RNA_CaptureSeq-Mercer_et_al.yaml
download_human_gencode_release.yaml
index_homo_sapiens_hg19_genome.yaml
index_mycoplasma_genitalium_ASM2732v1_genome.yaml
```

These example configurations differ in their usage of computational resources. Some example configurations download or work on small datasets and are thus feasible for machines with limited resources. Most examples can be extended by uncommenting additional steps. This might change their computational requirements in such a way that a very powerful stand-alone machine or a cluster system is required. The examples are marked accordingly in the sections below.

---

**Note:** Before **computing an example on a cluster**, you need to uncomment the *cluster Section* and adapt the settings as required. Please check also if the *Cluster Configuration File* fits your cluster system.

---

---

**Note:** The examples contain information where users can obtain **required external/bioinformatics tools**. If **uap** fails due to a missing tool, please check the provided URLs for installation instructions.

---

### 1.4.1 Handle Genomic Data

A usual analysis of High-Throughput Sequencing (HTS) data relies on different publicly available data. Most important is probably the genomic sequence of the species under investigation. That sequence is required to construct the indices (data structures used by read aligners). Other publicly available data sets (such as reference annotations or

the chromosome sizes) might also be required for an analysis. The following configurations showcase how to get or generate that data:

**index_mycoplasma_genitalium_ASM2732v1_genome.yaml** Downloads the *Mycoplasma genitalium* genome, generates the indices for , , , and . This workflow is quite fast because it uses the very small genome of *Mycoplasma genitalium*.

**Max. memory** ~0,5 GB

**Disk usage** ~20 MB

**Run time** minutes

Required tools:

- 
- 
- 
- 
- 
- 

**index_homo_sapiens_hg19_genome.yaml** Downloads chromosome 21 of the *Homo sapiens* genome, generates the indices for , , and . This minimal version should work just fine. Users can uncomment steps to download the complete genome. This would substantially increase the required computational resources. The index creation is commented out due to its high memory consumption (~50-60 GB), if working with the whole genome.

**Max. memory** ~2 GB

**Disk usage** ~240 MB

**Run time** several minutes

Downloads *Homo sapiens* chromosome 21, generates the indices for , , and (and if you uncomment it). This workflow requires substantial computational resources due to the size of the human genome. The index creation is commented out due to its high memory consumption. Please make sure to only run it on well equipped machines.

Required tools:

- 
- 
- 
- 
- 
- 
- (if uncommented)

**download_human_gencode_release.yaml** Downloads the human Gencode main annotation v19 and a subset for long non-coding RNA genes. This workflow only downloads files from the internet and and thus should work on any machine.

**Max. memory** depends on your machine

**Disk usage** ~1,2 GB

**Run time** depends on your internet connection

Required tools:

- 
- 
- 
- 
- 
- 

Let's have a look at the *Mycoplasma genitalium* example workflow by checking its uap_status:

```
$ cd <uap-path>/example-configurations/
$ uap index_mycoplasma_genitalium_ASM2732v1_genome.yaml status
[uap] Set log level to ERROR
[uap][ERROR]: index_mycoplasma_genitalium_ASM2732v1_genome.yaml: Destination path␣
↪does not exist: genomes/bacteria/Mycoplasma_genitalium/
```

Oops, the `destination_path` does not exist (see *destination_path Section*). Create it and start again:

```
$ mkdir -p genomes/bacteria/Mycoplasma_genitalium/
$ uap index_mycoplasma_genitalium_ASM2732v1_genome.yaml status

Waiting tasks
-------------
[w] bowtie2_index/Mycoplasma_genitalium_index-download
[w] bwa_index/Mycoplasma_genitalium_index-download
[w] fasta_index/download
[w] segemehl_index/Mycoplasma_genitalium_genome-download

Ready tasks
-----------
[r] M_genitalium_genome/download

tasks: 5 total, 4 waiting, 1 ready
```

A list with all runs and their respective state should be displayed. A run is always in one of these states:

- `[r]eady`
- `[w]aiting`
- `[q]ueued`
- `[e]xecuting`
- `[f]inished`

If the command still fails, please check that the tools defined in `index_mycoplasma_genitalium_ASM2732v1_genome.yaml` are available in your environment (see *tools Section*). If you really want to download and index the genome tell **uap** to start the workflow:

```
$ uap index_mycoplasma_genitalium_ASM2732v1_genome.yaml run-locally
```

**uap** should have created a symbolic link named `index_mycoplasma_genitalium_ASM2732v1_genome.yaml-out` pointing to the `destination_path`. The content should look something like that:

```
$ tree --charset=ascii
.
|-- bowtie2_index
|   |-- Mycoplasma_genitalium_index-download-cMQPtBxs
|   |   |-- Mycoplasma_genitalium_index-download.1.bt2
|   |   |-- Mycoplasma_genitalium_index-download.2.bt2
|   |   |-- Mycoplasma_genitalium_index-download.3.bt2
|   |   |-- Mycoplasma_genitalium_index-download.4.bt2
|   |   |-- Mycoplasma_genitalium_index-download.rev.1.bt2
|   |   `-- Mycoplasma_genitalium_index-download.rev.2.bt2
|   `-- Mycoplasma_genitalium_index-download-ZsvbSjtK
|       |-- Mycoplasma_genitalium_index-download.1.bt2
|       |-- Mycoplasma_genitalium_index-download.2.bt2
|       |-- Mycoplasma_genitalium_index-download.3.bt2
|       |-- Mycoplasma_genitalium_index-download.4.bt2
|       |-- Mycoplasma_genitalium_index-download.rev.1.bt2
|       `-- Mycoplasma_genitalium_index-download.rev.2.bt2
|-- bwa_index
|   `-- Mycoplasma_genitalium_index-download-XRyj5AnJ
|       |-- Mycoplasma_genitalium_index-download.amb
|       |-- Mycoplasma_genitalium_index-download.ann
|       |-- Mycoplasma_genitalium_index-download.bwt
|       |-- Mycoplasma_genitalium_index-download.pac
|       `-- Mycoplasma_genitalium_index-download.sa
|-- fasta_index
|   `-- download-HA439DGO
|       `-- Mycoplasma_genitalium.ASM2732v1.fa.fai
|-- M_genitalium_genome
|   `-- download-5dych7Xj
|-- Mycoplasma_genitalium.ASM2732v1.fa
|-- segemehl_index
|   |-- Mycoplasma_genitalium_genome-download-2UKxxupJ
|   |   |-- download-segemehl-generate-index-log.txt
|   |   `-- Mycoplasma_genitalium_genome-download.idx
|   `-- Mycoplasma_genitalium_genome-download-zgtEpQmV
|       |-- download-segemehl-generate-index-log.txt
|       `-- Mycoplasma_genitalium_genome-download.idx
`-- temp
```

Congratulation you've finished your first **uap** workflow!

Go on and try to run some more workflows. Most examples require the human genome so you might turn your head towards the `index_homo_sapiens_hg19_genome.yaml` workflow from her:

```
$ uap index_homo_sapiens_hg19_genome.yaml status
[uap] Set log level to ERROR
[uap][ERROR]: Output directory (genomes/animalia/chordata/mammalia/primates/homo_
↪sapiens/hg19/chromosome_sizes) does not exist. Please create it.
$ mkdir -p genomes/animalia/chordata/mammalia/primates/homo_sapiens/hg19/chromosome_
↪sizes
$ uap index_homo_sapiens_hg19_genome.yaml run-locally
<Analysis starts>
```

Again you need to create the output folder (you get the idea). Be aware that by default only the smallest chromosome, chromsome 21, is downloaded and indexed. This reduces required memory and computation time. You can uncomment the download steps for the other chromosomes and the index for the complete genome will be created.

### 1.4.2 Sequencing Data Analysis

Now that you possess the genome sequences, indices, and annotations let's have a look at some example analysis.

#### General Steps

The analysis of high-throughput sequencing (HTS) data usually start with some basic steps.

1. Conversion of the raw sequencing data to, most likely, fastq(.gz) files
2. Removal of adapter sequences from the sequencing reads
3. Alignment of the sequencing reads onto the reference genome

These basic steps can be followed up with a lot of different analysis steps. The following analysis examples illustrate how to perform the basic as well as some more specific steps.

#### RNAseq Example – Reanalysing Data from

RNAseq analysis often aims at the discovery of differentially expressed (known) transcripts. Therefore mappped reads for at least two different samples have to be available.

1. Differential Expression Analysis
   (d) Get annotation set (for e.g. genes, transcripts, . . . )
   (e) Count the number of reads overlapping the annotation
   (f) Perform statistical analysis, based on counts

Another common analysis performed with RNAseq data is the identification of novel tarnscripts. This approach is useful to identify tissue-specific transcipts.

2. *De novo* Transcript Assembly
   (d) Apply transcript assembly tool on mapped reads

**`2014-RNA_CaptureSeq-Mercer_et_al_download.yaml`** Downloads the data published in the paper .

**Max. memory** ~? GB

**Disk usage** ~12 GB

**Run time** minutes (depending on your internet connection)

Required tools:

- 
- 

**`2014-RNA_CaptureSeq-Mercer_et_al.yaml`** The downloaded FASTQ files get analysed by and . The reads are afterwards mapped to the human genome with . The mapped reads are afterwards sorted by position using . is used to count the mapped reads for every exon of the annotation. is used to perform *de novo* transcript assembly. The usage of is **disabled** by default. But it can be enabled and combined with *de novo* transcript assembly employing our **s2c** python script.

**Max. memory** ~? GB

**Disk usage** ~3 GB

**Run time** several hours

- 

- 
- 
- 
- 
- 
- 
- (if uncommented)
- 

---

**Note:** Before computing `2014-RNA_CaptureSeq-Mercer_et_al.yaml` please make sure that, the following examples were executed:

- `index_homo_sapiens_hg19_genome.yaml`
- `download_human_gencode_release.yaml`

---

### ChIPseq Example – Reanalysing Data from

ChIPseq analysis aims at the discovery of genomic loci at which protein(s) of interest were bound. The experiment is an enrichment procedure using specific antibodies. The enrichment detection is normally performed by so called peak calling programs. The data is prone to duplicate reads from PCR due to relatively low amounts of input DNA. So these steps follow the basic ones:

4. Duplicate removal
5. Peak calling

The analysis of data published in the paper is contained in these files:

**`2007-CD4+_T_Cell_ChIPseq-Barski_et_al_download.yaml`** Downloads the data published in the paper .

**Max. memory** ~? GB

**Disk usage** ~17 GB

**Run time** depends on your internet connection

Downloads the data published in the paper .

Required tools:

- 
- 

**`2007-CD4+_T_Cell_ChIPseq-Barski_et_al.yaml`** At first the downloaded FASTQ files are grouped by sample. All files per sample are merged. Sequencing quality is controlled by and . Adapter sequences are removed from the reads before they are mapped to the human genome. Reads are mapped with , , and . Again mapping with is disabled by default due to its high resource requirements. Library complexity is estimated using . After the mapping duplicate reads are removed using . Finally enriched regions are detected with .

**Max. memory** ~? GB

**Disk usage** ~51 GB

**Run time** ~several hours (on a cluster), ~1 day (on a single machine)

Required tools:

- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 

**Hint:** The usage of can differ a lot between systems. On Ubuntu systems it can be called like this:

```
$ picard-tools --version
```

If you use it as recommended at , it is called like this:

```
$ java -jar /path/to/picard.jar -h
```

Please check how to use it on your system and adjust the example configuration accordingly (see *tools Section*).

**Note:** Before computing `2007-CD4+_T_Cell_ChIPseq-Barski_et_al.yaml` please make sure that, the following examples were executed:

- `index_homo_sapiens_hg19_genome.yaml`
- `download_human_gencode_release.yaml`

#### Create Your Own Workflow

You finished to check out the examples? Go and try to create your own workflow If you are fine with what you saw Although writing the configuration may seem a bit complicated, the trouble pays off later because further interaction with the pipeline is quite simple. The structure and content of the configuration files is very detailed described on another page (see *Analysis Configuration File*).

## 1.5 Installation of uap

### 1.5.1 Prerequisites

The installation requires , and . So, please install it if its not already installed.:

```
$ sudo apt-get install python-virtualenv git graphviz
```

**uap** does **NOT** include any tools necessary for the data analysis. It is expected that the required tools are **already installed**.

### 1.5.2 Downloading the Software

Download the software from like this:

```
$ git clone https://github.com/yigbt/uap.git
```

### 1.5.3 Setting Up Python Environment

After cloning the repository, change into the created directory and run the bootstrapping script `bootstrap.sh`:

```
$ cd uap
$ ./bootstrap.sh
```

The script creates the required Python environment (which will be located in `./python_env/`). Afterwards it installs , , and into the freshly created environment. There is no harm in accidentally running this script multiple times.

### 1.5.4 Making uap Globally Available

**uap** can be used globally. On Unix-type operating systems it is advised to add the installation path to your `$PATH` variable. Therefore change into the **uap** directory and execute:

```
$ echo ""PATH=$PATH:$(pwd)" >> ~/.bashrc
$ source ~/.bashrc
OR
$ echo ""PATH=$PATH:$(pwd)" >> ~/.bash_profile
$ source ~/.bash_profile
```

## 1.6 Analysis Configuration File

**uap** requires a file which contains all information about the data analysis. These files are called configuration files.

A configuration file describes a complete analysis. Configurations consist of four sections (let's just call them sections, although technically, they are keys):

**Mandatory Sections**

- `destination_path` – points to the directory where the result files, annotations and temporary files are written to
- `constants` – defines constants for later use (define repeatedly used values as constants to increase readability of the following sections)
- `steps` – defines the source and processing steps and their order
- `tools` – defines all tools used in the analysis and how to determine their versions (for later reference)

**Optional Sections**

- `cluster` – if **uap** is required to run on a HPC cluster some default parameters can be set her

Please refer to the definition for the correct notation used in that file.

### 1.6.1 Sections of a Configuration File

#### `destination_path` Section

The value of `destination_path` is the directory where **uap** is going to store the created files.

```
destination_path: "/path/to/workflow/output"
```

#### `constants` Section

This section is the place where repeatedly used constants should be defined. For instance absolute paths to the genome index files can be defined as constant.

```
- &genome_faidx
    genomes/animalia/chordata/mammalia/primates/homo_sapiens/hg19/samtools_faidx/
↪hg19_all_chr_UCSC-download-B7ceRp9K/hg19_all_chr_UCSC-download.fasta.fai
```

Later on the value can be reused by typing `*genome_faidx`. **There are no restrictions about what can be defined here.**

#### `steps` Section

The `steps` section is the core of the analysis file, because it defines when steps are executed and how they depend on each other. All available steps are described in detail in the steps documentation: *Available steps*. The `steps` section contains an entry (technically a key) for every step. Every step name **must** be unique.

---

**Note:** Please be aware that the , the YAML parser used by uap, does not complain about keys with the same name. But drops one of the duplicates without giving an error.

---

There are two ways to name a step to allow multiple steps of the same type and still ensure unique naming:

```
steps:
    # here, the step name is unchanged, it's a cutadapt step which is also
    # called 'cutadapt'
    cutadapt:
        ... # options following

    # here, we also insert a cutadapt step, but we give it a different name:
    # 'clip_adapters'
    clip_adapters (cutadapt):
        ... # options following
```

Now let's have a look at the two different types of steps which constitute an **uap** analaysis.

#### Source Steps

Source steps are the only steps which are allowed to use or create data outside the `destination_path`. Feature of source steps:

- they provide the input files for the following steps
- they can start processes e.g. to download files or demultiplex reads
- they do not depend on previous steps
- they are the root nodes of the analysis graph

If you want to work with fastq files, you should use the `fastq_source` step to import the required files. Such a step definition would look like this:

```
steps:
    input_step (fastq_source):
    pattern: /Path/to/fastq-files/*.gz
    group: ([SL]\w+)_R[12]-00[12].fastq.gz
    sample_id_prefix: MyPrefix
    first_read: '_R1'
    second_read: '_R2'
    paired_end: True
```

The options of the `fastq_source` step are described at *Available steps*. The `group` option takes a regular expression (regexp). You can test your regular expression at .

### Processing Steps

Processing steps depend upon one or more preceding steps. They use their output files and process them. Output files of processing steps are automatically named and saved by **uap**. A complete list of available options per step can be found at *Available steps* or by using the *steps Subcommand*.

### Reserved Keywords for Steps

**_depends:**

Dependencies are defined via the `_depends` key which may either be `null`, a step name, or a list of step names.

```
steps:
    # the source step which depends on nothing
    fastq_source:
        # ...

    run_folder_source:
        # ...

    # the first processing step, which depends on the source step
    cutadapt:
        _depends: [fastq_source, run_folder_source]

    # the second processing step, which depends on the cutadapt step
    fix_cutadapt:
        _depends: cutadapt
```

**_connect:**

Normally steps connected with `_depends` do pass data along by defining so called connections. If the name of an output connection matches the name of an input connection of a succeeding step the data gets passed on automatically. But, sometimes the user wants to force the connection of differently named

connections. This can be done with the `_connect` keyword. A common usage is to connect downloaded data with a *Processing Steps*.

```
  steps:
      # Source step to download i.e. sequence of chr1 of some species
      chr1 (raw_url_source):
          ...

      # Download chr2 sequence
      chr2 (raw_url_source):
          ...

      merge_fasta_files:
          _depends:
              - chr1
              - chr2
          # Equivalent to:
          # _depends: [chr1, chr2]
          _connect:
              in/sequence:
                  - chr1/raw
                  - chr2/raw
          # Equivalent to:
          # _connect:
          #     in/sequence: [chr1/raw, chr2/raw]

The examples shows how the ``raw_url_source`` output connection ``raw`` is
connected to the input connection ``sequence`` of the ``merge_fasta_files``
step.
```

**_BREAK:**

If you want to cut off entire branches of the step graph, set the `_BREAK` flag in a step definition, which will force the step to produce no runs (which will in turn give all following steps nothing to do, thereby effectively disabling these steps):

```
steps:
    fastq_source:
        # ...

    cutadapt:
        _depends: fastq_source

    # this step and all following steps will not be executed
    fix_cutadapt:
        _depends: cutadapt
        _BREAK: true
```

**_volatile:**

Steps can be marked with `_volatile: yes`. This flag tells **uap** that the output files of the marked step are only intermediate results.

```
steps:
    # the source step which depends on nothing
    fastq_source:
        # ...
```

(continues on next page)

(continued from previous page)

```
    # this steps output can be deleted if all depending steps are finished
    cutadapt:
        _depends: fastq_source
        _volatile: yes
        # same as:
        # _volatile: True

    # if fix_cutadapt is finished the output files of cutadapt can be
    # volatilized
    fix_cutadapt:
        _depends: cutadapt
```

If all steps depending on the intermediate step are finished **uap** tells the user that he can free disk space. The message is output if the *status* is checked and looks like this:

```
Hint: You could save 156.9 GB of disk space by volatilizing 104 output files.
Call 'uap <project-config>.yaml volatilize --srsly' to purge the files.
```

**uap** is going to replace the output files by placeholder files if the user executes the *volatilize* command.

**_cluster_submit_options**

This string contains the entire submit options which will be set in the submit script. This option allows to overwrite the values set in *default_submit_options*.

**_cluster_pre_job_command**

This string contains command(s) that are executed **BEFORE uap** is started on the cluster. This option allows to overwrite the values set in *default_pre_job_command*.

**_cluster_post_job_command**

This string contains command(s) that are executed **AFTER uap** did finish on the cluster. This option allows to overwrite the values set in *default_post_job_command*.

**_cluster_job_quota**

This option defines the number of jobs of the same type that can run simultaneously on a cluster. This option allows to overwrite the values set in *default_job_quota*.

### `tools` Section

The `tools` section lists all programs required for the execution of a particular analysis. An example tool configuration looks like this:

```
tools:

    # you don't have to specify a path if the tool can be found in $PATH
    cat:
        path: cat
        get_version: --version
        module_load:

    # you have to specify a path if the tool can not be found in $PATH
    some-tool:
        path: /path/to/some-tool
        get_version: --version
```

(continues on next page)

(continued from previous page)

```
    pigz:
        path: pigz
        get_version: --version
        exit_code: 0
```

**uap** uses the `path`, `get_version`, and `exit_code` information to control the availability of a tool. This is particularly useful on cluster systems were software can be dynamically loaded and unloaded. **uap** logs the version of every used tool. If `get_version` and `exit_code` is not set, **uap** tries to determine the version by calling the program without command-line arguments. `get_version` is the command line argument (e.g. `--version`) required to get the version information. `exit_code` is the value returned by `echo $?` after trying to determine the version e.g. by running `pigz --version`. If not set `exit_code` defaults to 0.

**uap** can use the module system if you are working on a cluster system (e.g. or ). The configuration for `pigz` would change a bit:

```
tools:

    pigz:
        path: pigz
        get_version: --version
        exit_code: 0
        module_load: /path/to/modulecmd python load pigz
        module_unload: /path/to/modulecmd python unload pigz
```

As you can see you need to get the `/path/to/modulecmd`. So let's investigate what happens when a module is loaded or unloaded:

```
$ module load <module-name>
$ module unload <module-name>
```

As far as I know is `module` neither a command nor an alias. It is a BASH function. So use `declare -f` to find out what it is actually doing:

```
$ declare -f module
```

The output should look like this:

```
module ()
    {
        eval `/usr/local/modules/3.2.10-1/Modules/$MODULE_VERSION/bin/modulecmd bash
↪$*`
    }
```

An other possible output is:

```
module ()
    {
        eval $($LMOD_CMD bash "$@");
        [ $? = 0 ] && eval $(${LMOD_SETTARG_CMD:-:} -s sh)
    }
```

In this case you have to look in `$LMOD_CMD` for the required path:

```
$ echo $LMOD_CMD
/usr/local/modules/3.2.10-1/Modules/$MODULE_VERSION/bin/modulecmd
```

You can use this path to assemble the `module_load` and `module_unload` options for `pigz`. Just replace the `$MODULE_VERSION` with the current version of the module system.

```
tools:

    pigz:
        path: pigz
        get_version: --version
        exit_code: 0
        module_load: /usr/local/modules/3.2.10-1/Modules/$MODULE_VERSION/bin/
↪modulecmd python load pigz
        module_unload: /usr/local/modules/3.2.10-1/Modules/$MODULE_VERSION/bin/
↪modulecmd python unload pigz
```

**Note:** Use `python` instead of `bash` for loading modules via **uap**. Because the module is loaded from within a python environment and not within a BASH shell.

#### `cluster` Section

The `cluster` section is required only if the analysis is executed on a system using a cluster engine like or . This section interacts tightly with the An example `cluster` section looks like this:

```
cluster:
    default_submit_options: "-pe smp #{CORES} -cwd -S /bin/bash -m as -M me@example.
↪com -l h_rt=1:00:00 -l h_vmem=2G"
    default_pre_job_command: "echo 'Started the run!'"
    default_post_job_command: "echo 'Finished the run!'"
    default_job_quota: 5
```

**default_submit_options**

This is the default submit options string which replaces the *#{SUBMIT_OPTIONS}* placeholder in the *submit script template*. It is **mandatory** to set this value.

**default_pre_job_command**

This string contains the default commands which will be executed **BEFORE uap** is started on the cluster. It will replace the *#{PRE_JOB_COMMAND}* placeholder in the *submit script template*. If mutliple commands shall be executed separate those with `\n`. It is **optional** to set this value.

**default_post_job_command**

This string contains the default commands which will be executed **AFTER uap** is started on the cluster. It will replace the *#{POST_JOB_COMMAND}* placeholder in the *submit script template*. If mutliple commands shall be executed separate those with `\n`. It is **optional** to set this value.

**default_job_quota:**

This option defines the number of jobs of the same type that can run simultaneously on a cluster. The number influences the way **uap** sets the job dependencies of submitted jobs. It is **optional** to set this value, if the value is not provided it is set to *5*.

### 1.6.2 Example Configurations

Example configurations can be found in **uap**'s `example-configurations` folder. More information about these examples can be found in *Quick Start uap*.

## 1.7 Cluster Configuration File

The cluster configuration file resides at:

```
$ ls -la $(dirname $(which uap))/cluster/cluster-specific-commands.yaml
```

This YAML file contains a dictionary for every cluster type. An example file is shown here:

```
# Configuration for a UGE cluster engine
uge:
    # Command to get version information
    identity_test: ['qstat', '-help']
    # The expected output of identity_test for this cluster engine
    identity_answer: 'UGE'
    # Command to submit job
    submit: 'qsub'
    # Command to check job status
    stat: 'qstat'
    # Relative path to submit script template
    # The path has to be relative to:
    # $ dirname $(which uap)
    template: 'cluster/submit-scripts/qsub-template.sh'
    # way to define job dependencies
    hold_jid: '-hold_jid'
    # Separator for job dependencies
    hold_jid_separator: ';'
    # Option to set job names
    set_job_name: '-N'
    # Option to set path of stderr file
    set_stderr: '-e'
    # Option to set path of stdout file
    set_stdout: '-o'
    # Regex to extract Job ID after submission
    parse_job_id: 'Your job (\d+)'

# Configuration for a SLURM cluster engine
slurm:
    identity_test: ['sbatch', '--version']
    identity_answer: 'slurm'
    submit: 'sbatch'
    stat: 'squeue'
    template: 'cluster/submit-scripts/sbatch-template.sh'
    hold_jid: '--dependency=afterany:%s'
    hold_jid_separator: ':'
    set_job_name: '--job-name=%s'
    set_stderr: '-e'
    set_stdout: '-o'
    parse_job_id: 'Submitted batch job (\d+)'
```

Let's browse over the options which need to be set per cluster engine:

**`identity_test:`** Command used to determine if **uap** has been started on a system running a cluster engine e.g. `sbatch --version`.

**`identity_answer:`** **uap** checks if the output of the `identity_test` command starts with this value e.g. `slurm`. If that is true the cluster type has been detected.

**`submit:`** Command to submit a job onto the cluster e.g. `sbatch`.

**stat:** Command to check the status of jobs on the cluster e.g. `squeue`.

**template:** Path to the submit script template which has to be used for this cluster type e.g. `cluster/submit-scripts/sbatch-template.sh`.

**hold_jid:** Option given to the `submit` command to define dependencies between jobs e.g. `--dependency=afterany:%s`. Placeholder `%s` gets replaced with the jobs this job depends on if present.

**hold_jid_separator:** Separator used to concatenate multiple jobs for `hold_jid` e.g. `:`.

**set_job_name:** Option given to the `submit` command to set the job name e.g. `--job-name=%s`. `%s` is replaced by the job name if present.

**set_stderr:** Option given to the `submit` command to set the name of the stderr file e.g. `-e`.

**set_stdout:** Option given to the `submit` command to set the name of the stdout file e.g. `-o`.

**parse_job_id:** Python regular expression whose first parenthesized subgroup represents the cluster job ID e.g. `Submitted batch job (\d+)`.

### 1.7.1 Submit Script Template

The submit script template contains a lot of placeholders which are replaced if a job is submitted to the cluster with the actual commands.

The submit script templates reside at:

```
$ ls $(dirname $(which uap))/cluster/submit-scripts/*
qsub-template.sh
sbatch-template.sh
```

Feel free to add your own templates. The templates need to contain the following placeholders:

**#{SUBMIT_OPTIONS}** Will be replaced with the steps `_cluster_submit_options` value (see _cluster_submit_options), if present, or the `default_submit_options` value.

**#{PRE_JOB_COMMAND}** Will be replaced with the steps `_cluster_pre_job_command` value (see _cluster_pre_job_command), if present, or the `default_pre_job_command` value.

**#{COMMAND}** Will be replaced with `uap <project-config>.yaml run-locally <run ID>`.

**#{POST_JOB_COMMAND}** Will be replaced with the steps `_cluster_post_job_command` value (see _cluster_post_job_command), if present, or the `default_post_job_command` value.

The submit script template is required by *submit-to-cluster* for job submission to the cluster.

## 1.8 Command-Line Usage of uap

**uap** uses Python's . Therefore, **uap** provides help information on the command-line:

```
$ uap -h
usage: uap [-h] [-v] [--version]
           [<project-config>.yaml]
           {fix-problems,render,run-locally,status,steps,submit-to-cluster,run-info,
↪volatilize}
           ...
```

(continues on next page)

(continued from previous page)

```
This script starts and controls 'uap' analysis.

positional arguments:
  <project-config>.yaml
                        Path to YAML file that contains the pipeline configuration.
                        The content of that file needs to follow the documentation.

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbose         Increase output verbosity
  --version             Display version information.

subcommands:
  Available subcommands.

  {fix-problems,render,run-locally,status,steps,submit-to-cluster,run-info,volatilize}
    fix-problems        Fixes problematic states by removing stall files.
    render              Renders DOT-graphs displaying information of the analysis.
    run-locally         Executes the analysis on the local machine.
    status              Displays information about the status of the analysis.
    steps               Displays information about the steps available in uap.
    submit-to-cluster   Submits the jobs created by uap to a cluster
    run-info            Displays information about certain source or processing runs.
    volatilize          Saves disk space by volatilizing intermediate results

For further information please visit http://uap.readthedocs.org/en/latest/
For citation use ...
```

Almost all subcommands require a YAML configuration file (see *Analysis Configuration File*) **except** for `uap steps`, which works independent of an analysis configuration file.

Everytime **uap** is started with a *analysis configuration file* the following actions happen:

1. Configuration file is read
2. Tools given in the tools section are checked
3. Input files are checked
4. State of all runs are calculated

If any of these steps fail, **uap** will exit and print an error message.

**uap** will create a symbolic link, if it does not exist already, pointing to the *destination path* called `<project-config>.yaml-out`. The symbolic link is created in the directory containing the `<project-config>.yaml`.

There are a couple of global command line parameters which are valid for all scripts (well, actually, it's only one):

**`--even-if-dirty` or short `--even`:** If this parameter appears **uap** will work even if uncommited changes to its source code are detected. **uap** would otherwise immediately stop working. If you specify this flag, the repositories state is recorded in all annotation files created by this process. A full Git diff is *included* as well.

### 1.8.1 Subcommands

Here an overview of all the available subcommands are given.

### `steps` Subcommand

The `steps` subcommand lists all available *source* and *processing* steps:

```
$ uap steps -h
usage: uap [<project-config>.yaml] steps [-h] [--even-if-dirty] [--show STEP]

This script displays by default a list of all steps the pipeline can use.

optional arguments:
  -h, --help       show this help message and exit
  --even-if-dirty  This option must be set if the local git repository
                   contains uncommited changes.
                   Otherwise uap will not run.
  --show STEP      Show the details of a specific step.
```

### `status` Subcommand

The `status` subcommand lists all runs of an analysis. A run is describes the concrete processing of a sample by a step. Samples are usually defined at the source steps and are then propagated through the analysis. Here is the help message:

```
$ uap <project-config>.yaml status -h
usage: uap [<project-config>.yaml] status [-h] [--even-if-dirty]
                                          [--cluster CLUSTER] [--summarize]
                                          [--graph] [--sources]
                                          [-r [RUN [RUN ...]]]

This script displays by default information about all runs of the pipeline as
configured in '<project-config>.yaml'. But the displayed information can be
narrowed down via command line options.
IMPORTANT: Hints given by this script are just valid if the jobs were
submitted to the cluster.

optional arguments:
  -h, --help            show this help message and exit
  --even-if-dirty       This option must be set if the local git repository
                        contains uncommited changes.
                        Otherwise uap will not run.
  --cluster CLUSTER     Specify the cluster type. Default: [auto].
  --summarize           Displays summarized information of the analysis.
  --graph               Displays the dependency graph of the analysis.
  --sources             Displays only information about the source runs.
  -r [RUN [RUN ...]], --run [RUN [RUN ...]]
                        The status of these runs are displayed.
```

At any time, each run is in one of the following states:

- `[w]aiting` – the run is waiting for input files to appear, or its input files are not up-to-date regarding their respective dependencies
- `[r]eady` – all input files are present and up-to-date regarding their upstream input files (and so on, recursively), the run is ready and can be started
- `[q]ueued` – the run is currently queued and will be started "soon" (only available if you use a compute cluster)
- `[e]xecuting` – the run is currently running on this or another machine

- `[f]inished` – all output files are in place and up-to-date

Here is an example output:

```
$ uap <project-config>.yaml status
Waiting tasks
-------------
[w] fasta_index/download
[w] segemehl_index/Mycoplasma_genitalium_genome-download

Ready tasks
-----------
[r] bowtie2_index/Mycoplasma_genitalium_index-download
[r] bwa_index/Mycoplasma_genitalium_index-download

Finished tasks
--------------
[f] M_genitalium_genome/download

tasks: 5 total, 2 waiting, 2 ready, 1 finished
```

To get a more concise summary, specify `--summarize`:

```
$ uap <project-config>.yaml status --summarize
Waiting tasks
-------------
[w]   1 fasta_index
[w]   1 segemehl_index

Ready tasks
-----------
[r]   1 bowtie2_index
[r]   1 bwa_index

Finished tasks
--------------
[f]   1 M_genitalium_genome

tasks: 5 total, 2 waiting, 2 ready, 1 finished
```

... or print a fancy ASCII art graph with `--graph`:

```
$ uap <project-config>.yaml status --graph
M_genitalium_genome (raw_url_source) [1 finished]
└─┬─┬─┬─bowtie2_index (bowtie2_generate_index) [1 ready]
  └─┬─┬─bwa_index (bwa_generate_index) [1 ready]
    └─┬─fasta_index (samtools_faidx) [1 waiting]
      └─segemehl_index (segemehl_generate_index) [1 waiting]
```

Detailed information about a specific task can be obtained by specifying the run ID on the command line:

```
$ uap index_mycoplasma_genitalium_ASM2732v1_genome.yaml status -r \
  bowtie2_index/Mycoplasma_genitalium_index-download
[uap] Set log level to ERROR
output_directory: genomes/bacteria/Mycoplasma_genitalium/bowtie2_index/Mycoplasma_
↪genitalium_index-download-ZsvbSjtK
output_files:
  out/bowtie_index:
```

(continues on next page)

(continued from previous page)

```
    Mycoplasma_genitalium_index-download.1.bt2: &id001
    - genomes/bacteria/Mycoplasma_genitalium/Mycoplasma_genitalium.ASM2732v1.fa
    Mycoplasma_genitalium_index-download.2.bt2: *id001
    Mycoplasma_genitalium_index-download.3.bt2: *id001
    Mycoplasma_genitalium_index-download.4.bt2: *id001
    Mycoplasma_genitalium_index-download.rev.1.bt2: *id001
    Mycoplasma_genitalium_index-download.rev.2.bt2: *id001
private_info: {}
public_info: {}
run_id: Mycoplasma_genitalium_index-download
state: FINISHED
```

This is the known data for run `bowtie2_index/Mycoplasma_genitalium_index-download`. It contains information about the output folder, the output files and the input files they depend on as well as the run ID and the run state.

Source steps can be viewed separately by specifying `--sources`:

```
$ uap <project-config>.yaml status --sources
[uap] Set log level to ERROR
M_genitalium_genome/download
```

### `run-info` Subcommand

The `run-info` subcommand displays the commands issued for a given run. The output looks like a BASH script, but might not be functional. This is due to the fact that output redirections for some commands are missing in the BASH script. The output includes also the information as shown by the `status -r <run-ID>` subcommand.

An example output showing the download of the *Mycoplasma genitalium* genome:

```
$ uap index_mycoplasma_genitalium_ASM2732v1_genome.yaml run-info --even -r M_
↪genitalium_genome/download
[uap] Set log level to ERROR
#!/usr/bin/env bash

# M_genitalium_genome/download -- Report
# ======================================
#
# output_directory: genomes/bacteria/Mycoplasma_genitalium/M_genitalium_genome/
↪download-7RncJ4tr
# output_files:
#   out/raw:
#     genomes/bacteria/Mycoplasma_genitalium/Mycoplasma_genitalium.ASM2732v1.fa: []
# private_info: {}
# public_info: {}
# run_id: download
# state: FINISHED
#
# M_genitalium_genome/download -- Commands
# ========================================

# 1. Group of Commands -- 1. Command
# ----------------------------------

curl ftp://ftp.ncbi.nih.gov/genomes/genbank/bacteria/Mycoplasma_genitalium/latest_
↪assembly_versions/GCA_000027325.1_ASM2732v1/GCA_000027325.1_ASM2732v1_genomic.fna.gz
```

(continues on next page)

(continued from previous page)

```
# 2. Group of Commands -- 1. Command
# ----------------------------------

../tools/compare_secure_hashes.py --algorithm md5 --secure-hash␣
↪f02c78b5f9e756031eeaa51531517f24 genomes/bacteria/Mycoplasma_genitalium/M_
↪genitalium_genome/download-7RncJ4tr/L9PXBmbPKlemghJGNM97JwVuzMdGCA_000027325.1_
↪ASM2732v1_genomic.fna.gz

# 3. Group of Commands -- 1. Pipeline
# -----------------------------------

pigz --decompress --stdout --processes 1 genomes/bacteria/Mycoplasma_genitalium/M_
↪genitalium_genome/download-7RncJ4tr/L9PXBmbPKlemghJGNM97JwVuzMdGCA_000027325.1_
↪ASM2732v1_genomic.fna.gz | dd bs=4M of=/home/hubert/develop/uap/example-
↪configurations/genomes/bacteria/Mycoplasma_genitalium/Mycoplasma_genitalium.
↪ASM2732v1.fa
```

This subcommand enables the user to manually run parts of the analysis without **uap**. That can be helpful for debugging steps during development.

### `run-locally` Subcommand

The `run-locally` subcommand runs all non-finished runs (or a specified subset) sequentially on the local machine. The execution can be cancelled at any time, it won't put your project in a unstable state. However, if the `run-locally` subcommand receives a signal, the currently executing job will continue to run and the corresponding run will be reported as `executing` by calling `status` subcommand for five more minutes ( should be fine and exit gracefully but *doesn't just yet*). After that time, you will be warned that a job is marked as being currently run but no activity has been seen for a while, along with further instructions about what to do in such a case (don't worry, it shouldn't happen by accident).

Specify a set of run IDs to execute only those runs. Specify the name of a step to execute all ready runs of that step.

This subcommands usage information:

```
$ uap index_mycoplasma_genitalium_ASM2732v1_genome.yaml run-locally -h
usage: uap [<project-config>.yaml] run-locally [-h] [--even-if-dirty]
                                               [run [run ...]]

This command  starts 'uap' on the local machine. It can be used to start:
 * all runs of the pipeline as configured in <project-config>.yaml
 * all runs defined by a specific step in <project-config>.yaml
 * one or more steps
To start the complete pipeline as configured in <project-config>.yaml execute:
$ uap <project-config>.yaml run-locally
To start a specific step execute:
$ uap <project-config>.yaml run-locally <step_name>
To start a specific run execute:
$ uap <project-config>.yaml run-locally <step/run>
The step_name is the name of an entry in the 'steps:' section as defined in '<project-
↪config>.yaml'. A specific run is defined via its run ID 'step/run'. To get a list␣
↪of all run IDs please run:
$ uap <project-config>.yaml status

positional arguments:
  run              These runs are processed on the local machine.
```

(continues on next page)

(continued from previous page)

```
optional arguments:
  -h, --help       show this help message and exit
  --even-if-dirty  This option must be set if the local git repository
                   contains uncommited changes.
                   Otherwise uap will not run.
```

**Note:** Why is it safe to cancel the pipeline? The pipeline is written in a way which expects processes to fail or cluster jobs to disappear without notice. This problem is mitigated by a design which relies on file presence and file timestamps to determine whether a run is finished or not. Output files are automatically written to temporary locations and later moved to their real target directory, and it is not until the last file rename operation has finished that a run is regarded as finished.

### `submit-to-cluster` Subcommand

The `submit-to-cluster` subcommand determines which runs still need to be executed and which supported cluster engine is available. It submits a job for every run to the cluster if a cluster engine could be detected. Dependencies are passed to cluster engine in a way that jobs that depend on other jobs won't get scheduled until their dependencies have been satisfied. For more information read about the *cluster configuration* and the *submit script template*. Each submitted job calls **uap** with the `run-locally` subcommand on the executing cluster node.

Here is the usage information:

```
$ uap index_mycoplasma_genitalium_ASM2732v1_genome.yaml submit-to-cluster -h
usage: uap [<project-config>.yaml] submit-to-cluster [-h] [--even-if-dirty]
                                                     [--cluster CLUSTER]
                                                     [run [run ...]]

This script submits all runs configured in <project-config>.yaml to a cluster.
The configuration for the available cluster types is stored at
/<path-to-uap>/cluster/cluster-specific-commands.yaml.
The list of runs can be narrowed down to specific steps. All runs of the
specified step will be submitted to the cluster. Also, individual runs IDs
(step/run) can be used for submission.

positional arguments:
  run                Submit only these runs to the cluster.

optional arguments:
  -h, --help         show this help message and exit
  --even-if-dirty    This option must be set if the local git repository
                     contains uncommited changes.
                     Otherwise uap will not run.
  --cluster CLUSTER  Specify the cluster type. Default: [auto].
```

### `fix-problems` Subcommand

The `fix-problems` subcommand removes temporary files written by **uap** if they are not required anymore.

Here is the usage information:

```
$ uap <project-config>.yaml fix-problems -h
usage: uap [<project-config>.yaml] fix-problems [-h] [--even-if-dirty]
                                                [--cluster CLUSTER]
                                                [--details] [--srsly]

optional arguments:
  -h, --help         show this help message and exit
  --even-if-dirty    This option must be set if the local git repository
                     contains uncommited changes.
                     Otherwise uap will not run.
  --cluster CLUSTER  Specify the cluster type. Default: [auto].
  --details          Displays information about the files causing problems.
  --srsly            Delete problematic files.
  usage: uap [<project-config>.yaml] fix-problems [-h] [--even-if-dirty]
                                                  [--cluster CLUSTER]
                                                  [--details] [--srsly]

  optional arguments:
    -h, --help         show this help message and exit
    --even-if-dirty    Must be set if the local git repository contains
                       uncommited changes. Otherwise the pipeline will not start.
    --cluster CLUSTER  Specify the cluster type (sge, slurm), defaults to auto.
    --details          Displays information about problematic files which need
                       to be deleted to fix problem.
    --srsly            Deletes problematic files.
```

**uap** writes temporary files to indicate if a job is queued or executed. Sometimes (especially on the compute cluster) jobs fail, without even starting **uap**. This leaves the temporary file, written on job submission, indicating that a run was queued on the cluster without process (because it already failed). The `status` subcommand will inform the user if `fix-problems` needs to be executed to clean up the mess. The hint given by `status` would look like:

```
Warning: There are 10 tasks marked as queued, but they do not seem to be queued
Hint: Run 'uap <project-config>.yaml fix-problems --details' to see the details.
Hint: Run 'uap <project-config>.yaml fix-problems --srsly' to fix these problems
      (that is, delete all problematic ping files).
```

Be nice and do as you've told. Now you are able to resubmit your runs to the cluster. You've fixed the problem, haven't you?

### `volatilize` Subcommand

The `volatilize` subcommand is useful to reduce the required disk space of your analysis. It works only if the *_volatile* keyword is set in the *analysis configuration file* for. As already mentioned there, steps marked as `_volatile` compute their output files as normal but can be replaced by placeholder files if their dependent steps are finished.

This subcommand provides usage information:

```
$ uap <project-config>.yaml volatilize -h

usage: uap [<project-config>.yaml] volatilize [-h] [--even-if-dirty]
                                              [--details] [--srsly]

Save disk space by volatilizing intermediate results. Only steps marked with '_
↪volatile: True' are considered.
```

(continues on next page)

(continued from previous page)

```
optional arguments:
  -h, --help       show this help message and exit
  --even-if-dirty  This option must be set if the local git repository
                   contains uncommited changes.
                   Otherwise uap will not run.
  --details        Shows which files can be volatilized.
  --srsly          Replaces files marked for volatilization with a placeholder.
```

After running `volatilize --srsly` the output files of the volatilized step are replaced by placeholder files. The placeholder files have the same name as the original files suffixed with `.volatile.placeholder.yaml`.

#### `render` Subcommand

The `render` subcommand generates graphs using graphviz. The graphs either show the complete analysis or the execution of a single run. At the moment `--simple` only has an effect in combination with `--steps`.

This subcommand provides usage information:

```
$ uap <project-config>.yaml render -h
usage: uap [<project-config>.yaml] render [-h] [--even-if-dirty] [--files]
                                          [--steps] [--simple]
                                          [--orientation {left-to-right,right-to-left,
↪top-to-bottom}]
                                          [run [run ...]]

'render' generates DOT-graphs. Without arguments
it takes the annotation file of each run and generates a graph,
showing details of the computation.

positional arguments:
  run                   Render only graphs for these runs.

optional arguments:
  -h, --help            show this help message and exit
  --even-if-dirty       This option must be set if the local git repository
                        contains uncommited changes.
                        Otherwise uap will not run.
  --files               Renders a graph showing all files of the analysis.
                        [Not implemented yet!]
  --steps               Renders a graph showing all steps of the analysis and
                        their connections.
  --simple              Simplify rendered graphs.
  --orientation {left-to-right,right-to-left,top-to-bottom}
                        Defines orientation of the graph.
                        Default: 'top-to-bottom'
```

## 1.9 Add New Functionality

### 1.9.1 Implement New Steps

**uap** can be easily extended by implementing new *source* or *processing* steps. This requires basic python programming skills. New steps are added to **uap** by placing a single Python file into one of these folders in the **uap** installation directory:

**include/sources** Place source step files here

**include/steps** Place processing step files here

Let's talk about how to implement such **uap** steps.

### Step 1: Import Statements and Logger

At the beginning of every step please import the required modules and create a logger object.

```
# First import standard libraries
import os
from logging import getLogger

# Secondly import third party libraries
import yaml

# Thirdly import local application files
from abstract_step import AbstractStep # or AbstractSourceStep

# Get application wide logger
logger = getLogger("uap_logger")
```

Essential imports are the `from logging import getLogger` and `from abstract_step import ...`. The former is necessary to get access to the application wide logger and the latter to be able to inherit either from `AbstractStep` or `AbstractSourceStep`.

### Step 2: Class Definition

Now you need to define a class (which inherits either from `AbstractStep` or `AbstractSourceStep`) and its `__init__` method.

```
class ConcatenateFiles(AbstractStep):
    # Overwrite initialisation
    def __init__(self, pipeline):
        # Call super classes initialisation
        super(ConcatenateFiles, self).__init__(pipeline)

..
```

The new class needs to be derived from either `AbstractStep`, for processing steps, or `AbstractSourceStep`, for source steps.

### Step 3: `__init__` Method

The `__init__` method is the place where you should declare:

**Tools via `self.require_tool('tool_name')`:** Steps usually require tools to perform their task. Each tool that is going to be used by a step needs to be requested via the method `require_tool('tool_name')`. **uap** tests the existence of the required tools whenever it constructs the directed acyclic graph (DAG) of the analysis. The test is based on the information provided in the tools section of the *analysis configuration*. An entry for `tool_name` has to exist and to provide information to verify the tools accessibility.

**Connections via `add_connection(...)`:** Connections are defined by the method `add_connection(...)`. They are used to transfer data from one step to another. If a step defines an output connection `out/something`

and a subsequent step defines an input connection named `in/something`, then the files beloging to `out/something` will be available via the connection `in/something`.

Please name connection in a way that they describe the data itself and **NOT** the data type. For instance, use `in/genome` over `in/fasta`. The data type of the received input data should be checked by the steps to make sure to execute the correct commands.

**TODO**: Reanimate the constraints feature. It would often save some lines of code to be able to define constraints on the connections.

**Options via `self.add_option()`:** Options allow to influence the commands executed by a step. It is advisable to provide as many meaningful options as possible to keep steps flexible. Steps can have any number of options. Options are defined via the method `add_option()`.

The `add_option()` method allows to specify various information about the option. The method parameters are these:

1. **`key`** name of the option (if possible include the name of the tool this option influences e.g. `dd-blocksize` to set `dd` blocksize)
2. **`option_type`** The option type has to be at least one of `int`, `float`, `str`, `bool`, `list`, or `dict`.
3. **`optional` (Boolean)** Defines if the option is mandatory (`False`) or optional (`True`).
4. **`choices`** List of valid values for the option.
5. **`default`** Defines the default value for the option.
6. **`description`** The description of the functionality of the option.

```
..

        # Define connections
        self.add_connection('in/text')
        self.add_connection('out/text')

        # Request tools
        self.require_tool('cat')

        # Options for workflow
        self.add_option('concatenate_all_files', bool, optional=False,
                        default=False, description="Concatenate all files from "
                        "all runs, if 'True'.")

        # Options for 'cat' (see manpage)
        self.add_option('show-all', bool, optional=True,
                        description="Show all characters")

        self.add_option('number-nonblank', int, optional=True,
                        description="number nonempty output lines, "
                        "overrides --number")

        self.add_option('show-ends', bool, optional=True,
                        description="display $ at end of each line")

        self.add_option("number", int, optional=True,
                        description="number all output lines")

        self.add_option("squeeze-blank", bool, optional=True,
                        description="suppress repeated empty output lines")
```

(continues on next page)

(continued from previous page)

```
        self.add_option("show-tabs", bool, optional=True,
                        description="display TAB characters as ^I")

        self.add_option("show-nonprinting", bool, optional=True,
                         description="use ^ and M- notation, except for "
                         "LFD and TAB")

..
```

### Step 4: `runs` Method

The `runs` method is where all the work is done. This method gets handed over a dictionary of dictionaries. The keys of the first dictionary are the run IDs (often resembling the samples). The values of the first dictionary is another dictionary. The keys of that second dictionary are the connections e.g. "in/text" and the values are the corresponding files belonging to that connection.

Let's inspect all the run IDs, connections, and input files we got from our upstream steps. And let's tore all files we received in a list for later use.

```
..

    def runs(self, run_ids_connections_files):
        all_files = list()
        # Let's inspect the run_ids_connections_files data structure
        for run_id in run_ids_connections_files.keys():
            logger.info("Run ID: %s" % run_id)
            for connection in run_ids_connections_files[run_id].keys():
                logger.info("Connection: %s" % connection)
                for in_file in run_ids_connections_files[run_id][connection]:
                    logger.info("Input file: %s" % in_file)
                    # Collect all files
                    all_files.append(in_file)

..
```

It comes in handy to assemble a list with all options for `cat` here.

```
..

     # List with options for 'cat'
     cat_options = ['show-all', 'number-nonblank', 'show-ends', 'number',
                    'squeeze-blank', 'show-tabs', 'show-nonprinting']

     # Get all options which were set
     set_options = [option for option in cat_options if \
                    self.is_option_set_in_config(option)]

     # Compile the list of options
     cat_option_list = list()
     for option in set_options:
         # bool options look different than ...
         if isinstance(self.get_option(option), bool):
             if self.get_option(option):
                 cat_option_list.append('--%s' % option)
         # ... the rest ...
```

(continues on next page)

(continued from previous page)

```
        else:
            cat_option_list.append('--%s' % option)
            # ... make sure to cast the values to string
            cat_option_list.append(str(self.get_option(option)))

..
```

What should happen if we are told to concatenate all files from all input runs? We have to create a single run with a new run ID 'all_files'. The run consists of a `exec_group` that runs the `cat` command.

---

**Note:** An `exec_group` is a list of commands which are executed in one go. You might create multiple `exec_group`'s if you need to make sure a set of commands finished before another set is started. An `exec_group` can contain commands and pipelines. They can be added like this:

```
# Add a single command
exec_group.add_command(...)

# Add a pipeline to an exec_group
with exec_group.add_pipeline as pipe:
   ...
   # Add a command to a pipeline
   pipe.add_command(...)
```

---

The result of the concatenation is written to an output file. The run object needs to know about each output file that is going to be created.

---

**Note:** An output file is announced via the run objects `add_output_file(tag, out_path, in_paths)` method. The method parameters are:

1. `tag`: The name of the out connection e.g. 'text' for 'out/text'
2. `out_path`: The name of the output file (best practice is to add the run ID to the file name)
3. `in_paths`: The input files this output file is based on

---

```
..

    # Okay let's concatenate all files we get
    if self.get_option('concatenate_all_files'):
        run_id = 'all_files'

        # New run named 'all_files' is created here
        with self.declare_run(run_id) as run:

            # Create an exec
            with run.new_exec_group() as exec_group:
                # Assemble the cat command
                cat = [ self.get_tool('cat') ]
                # Add the options to the command
                cat.extend( cat_option_list )
                cat.extend( all_files )

                # Now add the command to the execution group
                exec_group.add_command(
```

(continues on next page)

(continued from previous page)

```
                    cat,
                    stdout_path = run.add_output_file(
                        'text',
                        "%s_concatenated.txt" % run_id,
                        all_files)
                )

..
```

What should happen if all files of an input run have to be concatenated? We create a new run for each input run and concatenate all files that belong to the input run.

```
# Concatenate all files from a runs 'in/text' connection
else:
    # iterate over all run IDs ...
    for run_id in run_ids_connections_files.keys():
        input_paths = run_ids_connections_files[run_id]['in/text']
        # ... and declare a new run for each of them.
        with self.declare_run(run_id) as run:
            with run.new_exec_group() as exec_group:
                # Assemble the cat command
                cat = [ self.get_tool('cat') ]
                # Add the options to the command
                cat.extend( cat_option_list )
                cat.extend( input_paths )

                # Now add the command to the execution group
                exec_group.add_command(
                    cat,
                    stdout_path = run.add_output_file(
                        'text',
                        "%s_concatenated.txt" % run_id,
                        input_paths)
                )
```

That's it. You created your first **uap** processing step.

### Step 5: Add the new step to uap

You have to make the new step known to **uap**. Save the complete file into **uap**'s `include/steps` folder. Processing step files are located at **uap**'s `include/steps/` folder and source step files at **uap**'s `include/sources/` folder.

You can control that your step is correctly "installed" if its included in the list of all source and processing steps:

```
$ ls -la $(dirname $(which uap))/include/sources
... Lists all available source step files

$ ls -la $(dirname $(which uap))/include/steps
... Lists all available processing step files
```

You can also use **uap**'s *steps* subcommand to get information about installed steps.

If the step file exists at the correct location that step can be used in an *analysis configuration file*.

A potential example YAML file named `test.yaml` could look like this:

```
destination_path: example-out/test/

steps:
    ##################
    ## Source steps ##
    ##################

    raw_file_source:
        pattern: example-data/text-files/*.txt
        group: (.*).txt

    ######################
    ## Processing steps ##
    ######################

    cat:
        _depends: raw_file_source
        _connect:
            in/text:
                - raw_file_source/raw
        concatenate_all_files: False

tools:
    cat:
        path: cat
        get_version: '--version'
        exit_code: 0
```

You need to create the destination path and some text files matching the pattern `example-data/text-files/*.txt`. Also you see the work of the `_connect` keyword in play. Check the status of the configured analysis:

```
$ uap test.yaml status
Ready runs
----------
[r] cat/Hello_america
[r] cat/Hello_asia
[r] cat/Hello_europe
[r] cat/Hello_world

runs: 4 total, 4 ready
```

### 1.9.2 Best Practices

There are a couple of things you should keep in mind while implementing new steps or modifying existing ones:

- **NEVER** remove files! If files need to be removed report the issue and exit **uap** or force the user to call a specific subcommand. Never delete files without permission by the user.
- Make sure errors already show up in when the steps `runs()` method is called the first time. So, look out for things that may fail in `runs`. Stick to *fail early, fail often*. That way errors show up before submitting jobs to the cluster and wasting precious cluster waiting time is avoided.
- Make sure that all tools which you request inside the `runs()` method are also required by the step via `self.require_tool()`. Use the `__init__()` method to request tools.
- Make sure your disk access is as cluster-friendly as possible (which primarily means using large block sizes and preferably no seek operations). If possible, use pipelines to wrap your commands in `pigz` or `dd` commands.

Make the used block size configurable. Although this is not possible in every case (for example when seeking in files is involved), it is straightforward with tools that read a continuous stream from `stdin` and write a continuous stream to `stdout`.

- Always use `os.path.join(...)` to handle paths.
- Use bash commands like `mkfifo` over python library equivalents like `os.mkfifo()`. The `mkfifo` command is hashed while an `os.mkfifo()` is not.
- Keep your steps as flexible as possible. You don't know what other user might need, so let them decide.

#### Usage of `dd` and `mkfifo`

**uap** relies often on `dd` and FIFOs to process data with fewer disk read-write operations. Please provide a step option to adjust the `dd` blocksize (this option is usually called `dd-blocksize`). Create your steps in a way that they perform the least filesystem operations. Some systems might be very sensitive to huge numbers of read-write operations.

## 1.10 Tested Platforms

So far **uap** has been tested on several operating systems (OS) and cluster engines. The tables below list the combinations of OS and cluster engine we successfully tested.

| `download_human_gencode_release.yaml` | | | | |
|---|---|---|---|---|
| | | | | local machine |
| Cent OS | ✓ | untested | untested | ✓ |
| Fedora | untested | ✓ | untested | ✓ |
| Ubuntu | untested | untested | ✓ | ✓ |

| `index_homo_sapiens_hg19_genome.yaml` | | | | |
|---|---|---|---|---|
| | | | | local machine |
| Cent OS | ✓ | untested | untested | ✓ |
| Fedora | untested | ✓ | untested | ✓ |
| Ubuntu | untested | untested | ✓ | ✓ |

| `index_mycoplasma_genitalium_ASM2732v1_genome.yaml` | | | | |
|---|---|---|---|---|
| | | | | local machine |
| Cent OS | ✓ | untested | untested | ✓ |
| Fedora | untested | ✓ | untested | ✓ |
| Ubuntu | untested | untested | ✓ | ✓ |

| `2007-CD4+_T_Cell_ChIPseq-Barski_et_al_download.yaml` | | | | |
|---|---|---|---|---|
| | | | | local machine |
| Cent OS | ✓ | untested | untested | ✓ |
| Fedora | untested | ✓ | untested | ✓ |
| Ubuntu | untested | untested | ✓ | ✓ |

| 2007-CD4+_T_Cell_ChIPseq-Barski_et_al.yaml | | | | |
|---|---|---|---|---|
| | | | | local machine |
| Cent OS | ✓ | untested | untested | ✓ |
| Fedora | untested | ✓ | untested | ✓ |
| Ubuntu | untested | untested | ✓ | ✓ |

| 2014-RNA_CaptureSeq-Mercer_et_al_download.yaml | | | | |
|---|---|---|---|---|
| | | | | local machine |
| Cent OS | ✓ | untested | untested | ✓ |
| Fedora | untested | untested | untested | untested |
| Ubuntu | untested | untested | ✓ | ✓ |

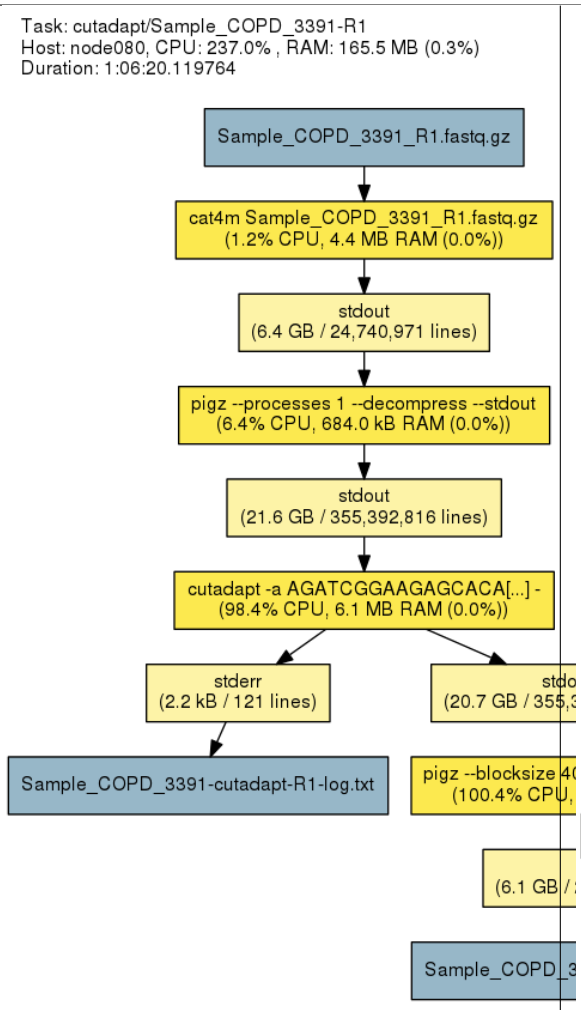
| 2014-RNA_CaptureSeq-Mercer_et_al.yaml | | | | |
|---|---|---|---|---|
| | | | | local machine |
| Cent OS | ✓ | untested | untested | ✓ |
| Fedora | untested | untested | untested | untested |
| Ubuntu | untested | untested | ✓ | ✓ |

## 1.11 Annotation Files

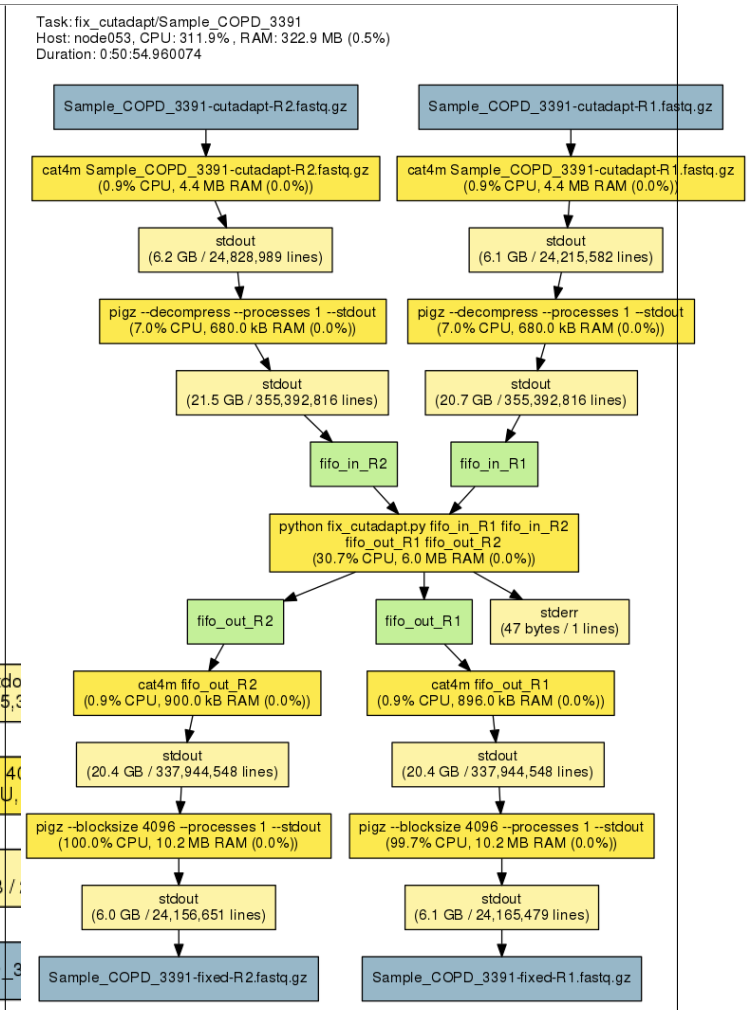
The annotation files contain detailed information about every output file. Also, the Git SHA1 hash of the **uap** repository at the time of data processing is included. The executed commands are listed. Annotation contains information about inter-process streams and output files, including SHA1 checksums, file sizes, and line counts as well.

Upon successful completion of a task, an extensive YAML-formatted annotation is placed next to the output files in a file called `.[task_id]-annotation.yaml`. Also, for every output file, a symbolic link to this file is created: `.[output_filename].annotation.yaml`.

Finally, the annotation is rendered via GraphViz, if available. Rendering can also be done at a later time using annotations as input (see **uap**'s *render* subcommand). The annotation can be used to determine at a later time what exactly happened. Also, annotations may help to identify bottlenecks.




Annotation graph of a `cutadapt` run. CPU and RAM usage for individual processes are shown, file sizes and line counts are shown for output files and inter-process streams.




In this graph, it becomes evident that the `fix_cutadapt.py` process in the middle gets throttled by the following two `pigz` processes, which only run with one core each and therefore cannot compress the results fast enough.

### 1.11.1 known_paths

Contains information about all directories/files used during processing a run. uap calculates the SHA1 hexdigest for each known file with the designation 'output' aka. output/result files.

## 1.12 Available steps

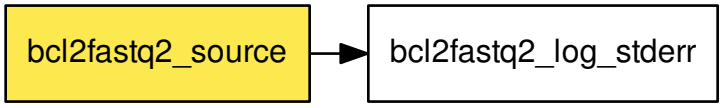
### 1.12.1 Source steps

#### bcl2fastq2_source

**Connections:**

- Output Connection:

- 'out/bcl2fastq2_log_stderr'




**Options:**

- **adapter-stringency** (float, optional) – adapter stringency (=0.9)
- **aggregated-tiles** (str, optional) – tiles aggregation flag determining structure of input files (=AUTO). recognized values: AUTO - Try to detect correct setting. YES - Tiles are aggregated into single input file. NO - There are separate input files for individual tiles
- **barcode-mismatches** (str, optional) – number of allowed mismatches per index multiple entries, comma delimited entries, allowed; each entry is applied to the corresponding index;last entry applies to all remaining indices
- **create-fastq-for-index-reads** (bool, optional) – create FASTQ files also for index reads
- **fastq-compression-level** (int, optional) – Zlib compression level (1-9) used for FASTQ files (=4)
- **find-adapters-with-sliding-window** (bool, optional) – find adapters with simple sliding window algorithm
- **ignore-missing-bcls** (bool, optional) – assume N for missing calls
- **ignore-missing-controls** (bool, optional) – assume 0 for missing controls
- **ignore-missing-filter** (bool, optional) – assume true for missing filters
- **ignore-missing-positions** (bool, optional) – assume [0,i] for missing positions, where i is incremented starting from 0
- **input-dir** (str, optional) – path to input directory (=<runfolder-dir>/Data/Intensities/BaseCalls/)
- **intensities-dir** (str, optional) – path to intensities directory (=<input-dir>/../). If intensities directory is specified, also input directory must be specified.
- **interop-dir** (str, optional) – path to demultiplexing statistics directory (=<runfolder-dir>/InterOp/)
- **loading-threads** (int, optional) – number of threads used for loading BCL data (=4)
- **mask-short-adapter-reads** (int, optional) – smallest number of remaining bases (after masking bases below the minimum trimmed read length) below which whole read is masked (=22)
- **min-log-level** (str, optional) – minimum log level recognized values: NONE, FATAL, ERROR, WARNING, INFO, DEBUG, TRACE. (INFO)
- **minimum-trimmed-read-length** (int, optional) – minimum read length after adapter trimming (=35)
- **no-bgzf-compression** (bool, optional) – Turn off BGZF compression for FASTQ files
- **no-lane-splitting** (bool, optional) – Do not split fastq files by lane.
- **output-dir** (str, required) - **processing-threads** (int, optional) – number of threads used for processing demultipled data (=100% of available CPUs)

- **reports-dir** (str, optional) – path to reporting directory (=<output-dir>/Reports/)
- **runfolder-dir** (str, required) – path to runfolder directory (=./)
- **sample-sheet** (str, optional) – path to the sample sheet(=<runfolder-dir>/SampleSheet.csv)
- **stats-dir** (str, optional) – path to human-readable demultiplexing statistics directory (=<runfolder-dir>/InterOp/)
- **tiles** (str, optional) – Comma-separated list of regular expressions to select only a subset of the tiles available in the flow-cell.Multiple entries allowed, each applies to the corresponding base-calls.For example: * to select all the tiles ending with 5 in all lanes: tiles [0-9][0-9][0-9]5. * to select tile 2 in lane 1 and all the tiles in the other lanes: tiles s_1_0002,s_[2-8]
- **use-bases-mask** (str, optional) – Specifies how to use each cycle.
- **with-failed-reads** (bool, optional) – include non-PF clusters
- **write-fastq-reverse-complement** (bool, optional) – Generate FASTQs containing reverse complements of actual data
- **writing-threads** (int, optional) – number of threads used for writing FASTQ data (=4)

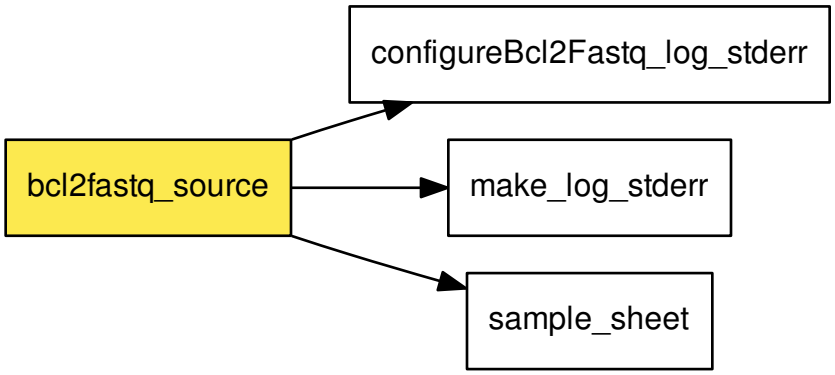
**Required tools:** bcl2fastq, mkdir

This step provides input files which already exists and therefore creates no tasks in the pipeline.

### bcl2fastq_source

**Connections:**

- Output Connection:
  - 'out/configureBcl2Fastq_log_stderr'
  - 'out/make_log_stderr'
  - 'out/sample_sheet'




**Options:**

- **adapter-sequence** (str, optional) - **adapter-stringency** (str, optional) - **fastq-cluster-count** (int, optional) - **filter-dir** (str, optional) - **flowcell-id** (str, optional) - **ignore-missing-bcl** (bool, optional) - **ignore-missing-control** (bool, optional) - **ignore-missing-stats** (bool, optional) - **input-dir** (str, required) – file URL
- **intensities-dir** (str, optional) - **mismatches** (int, optional) - **no-eamss** (str, optional) - **output-dir** (str, optional) - **positions-dir** (str, optional) - **positions-format** (str, optional) - **sample-sheet** (str, required) - **tiles** (str, optional) - **use-bases-mask** (str, optional) – Conversion mask characters:- Y or y: use- N or n: discard- I or i: use for indexingIf not given, the mask will be guessed from theRunInfo.xml file in the run folder.For instance, in a 2x76 indexed paired end run, themask *Y76,I6n,y75n* means: "use all 76 bases from thefirst end, discard the last base of the indexing read,and use only the first 75 bases of the second end".
- **with-failed-reads** (str, optional)**Required tools:** configureBclToFastq.pl, make, mkdir, mv

This step provides input files which already exists and therefore creates no tasks in the pipeline.

### fastq_source

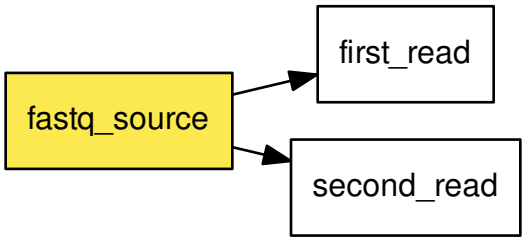
The FastqSource class acts as a source for FASTQ files. This source creates a run for every sample.

Specify a file name pattern in *pattern* and define how sample names should be determined from file names by specifyign a regular expression in *group*.

Sample index barcodes may specified by providing a filename to a CSV file containing the columns *Sample_ID* and *Index* or directly by defining a dictionary which maps indices to sample names.

**Connections:**

- Output Connection:
  - 'out/first_read'
  - 'out/second_read'




**Options:**

- **first_read** (str, required) – Part of the file name that marks all files containing sequencing data of the first read. Example: 'R1.fastq' or '_1.fastq'
- **group** (str, optional) – A regular expression which is applied to found files, and which is used to determine the sample name from the file name. For example, `(Sample_\d+)_R[12].fastq.gz`, when applied to a file called `Sample_1_R1.fastq.gz`, would result in a sample name of `Sample_1`. You can specify multiple capture groups in the regular expression.
- **indices** (str/dict, optional) – path to a CSV file or a dictionary of sample_id: barcode entries.
- **paired_end** (bool, required) – Specify whether the samples are paired end or not.

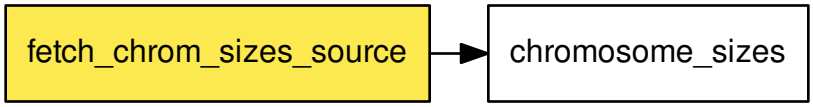
- **pattern** (str, optional) – A file name pattern, for example `/home/test/fastq/Sample_*.fastq.gz`.
- **sample_id_prefix** (str, optional) – This optional prefix is prepended to every sample name.
- **sample_to_files_map** (dict/str, optional) – A listing of sample names and their associated files. This must be provided as a YAML dictionary.
- **second_read** (str, required) – Part of the file name that marks all files containing sequencing data of the second read. Example: 'R2.fastq' or '_2.fastq'

This step provides input files which already exists and therefore creates no tasks in the pipeline.

### fetch_chrom_sizes_source

**Connections:**

- Output Connection:
  - 'out/chromosome_sizes'




**Options:**

- **path** (str, required) – directory to move file to
- **ucsc-database** (str, required) – Name of UCSC database e.g. hg38, mm9

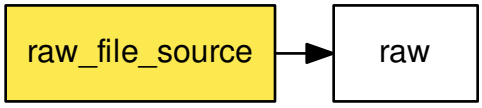
**Required tools:** cp, fetchChromSizes

This step provides input files which already exists and therefore creates no tasks in the pipeline.

### raw_file_source

**Connections:**

- Output Connection:
  - 'out/raw'



**Options:**

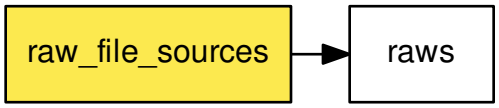
- **group** (str, optional) – A regular expression which is applied to found files, and which is used to determine the sample name from the file name. For example, *(Sample_d+)_R[12].fastq.gz*', when applied to a file called `Sample_1_R1.fastq.gz`, would result in a sample name of `Sample_1`. You can specify multiple capture groups in the regular expression.
- **pattern** (str, optional) – A file name pattern, for example `/home/test/fastq/Sample_*.fastq.gz`.
- **sample_id_prefix** (str, optional) – This optional prefix is prepended to every sample name.
- **sample_to_files_map** (dict/str, optional) – A listing of sample names and their associated files. This must be provided as a YAML dictionary.

This step provides input files which already exists and therefore creates no tasks in the pipeline.

### raw_file_sources

**Connections:**

- Output Connection:
  - 'out/raws'




**Options:**

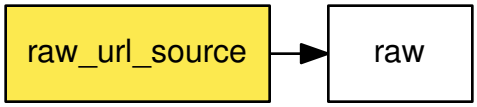
- **group** (str, required) – **This is a LEGACY step.** Do NOT use it, better use the `raw_file_source` step. A regular expression which is applied to found files, and which is used to determine the sample name from the file name. For example, `(Sample_\d+)_R[12].fastq.gz`, when applied to a file called `Sample_1_R1.fastq.gz`, would result in a sample name of `Sample_1`. You can specify multiple capture groups in the regular expression.
- **paired_end** (bool, required) – Specify whether the samples are paired end or not.
- **pattern** (str, required) – A file name pattern, for example `/home/test/fastq/Sample_*.fastq.gz`.
- **sample_id_prefix** (str, optional) – This optional prefix is prepended to every sample name.

This step provides input files which already exists and therefore creates no tasks in the pipeline.

### raw_url_source

**Connections:**

- Output Connection:
  - 'out/raw'




**Options:**

- **dd-blocksize** (str, optional) - default value: 256k
- **filename** (str, optional) – local file name of downloaded file
- **hashing-algorithm** (str, optional) – hashing algorithm to use
  - possible values: 'md5', 'sha1', 'sha224', 'sha256', 'sha384', 'sha512'
- **path** (str, required) – directory to move downloaded file to
- **secure-hash** (str, optional) – expected secure hash of downloaded file
- **uncompress** (bool, optional) – File is uncompressed after download
- **url** (str, required) – Download URL

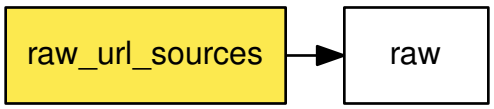
**Required tools:** compare_secure_hashes, cp, curl, dd, mkdir, pigz

This step provides input files which already exists and therefore creates no tasks in the pipeline.

### raw_url_sources

**Connections:**

- Output Connection:
  - 'out/raw'




**Options:**

- **dd-blocksize** (str, optional) - default value: 256k
- **run-download-info** (dict, required) – Dictionary of dictionaries. The keys are the names of the runs. The values are dictionaries whose keys are identical with the options of an 'raw_url_source' source step. An example: <name>: filename: <filename> hashing-algorithm: <hashing-algorithm> path: <path> secure-hash: <secure-hash> uncompress: <uncompress> url: <url>

**Required tools:** compare_secure_hashes, cp, curl, dd, pigz

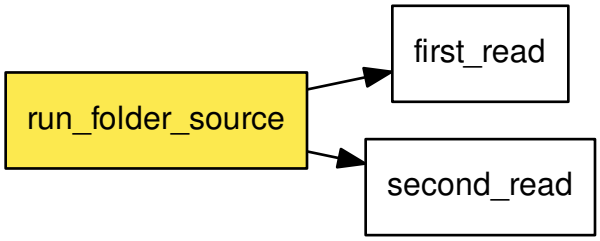
This step provides input files which already exists and therefore creates no tasks in the pipeline.

### run_folder_source

This source looks for fastq.gz files in `[path]/Unaligned/Project_*/Sample_*` and pulls additional information from CSV sample sheets it finds. It also makes sure that index information for all samples is coherent and unambiguous.

**Connections:**

- Output Connection:
  - 'out/first_read'
  - 'out/second_read'




**Options:**

- **first_read** (str, required) – Part of the file name that marks all files containing sequencing data of the first read. Example: '_R1.fastq' or '_1.fastq'
  - default value: _R1
- **paired_end** (bool, optional) – Is the project a paired-end sequencing project?
- **path** (str, required) – Path to the sequencing directories that contain the fastq[.gz] files.
- **project** (str, optional) – Name of the project. If provided, this is appendedto the path string
  - default value: *
- **project_name** (str, optional) – Name of the project. If provided, this is appendedto the path string. This option has the same meaning as 'project',however, the prefix '**Project_**' is not added. If 'project' and 'project_name' are provided, 'project_name' is choosen.
  - default value: *
- **samples** (str, optional) – Pattern for the sample directory names inside path/[**Project_**]project[_name]
  - default value: Sample_*
- **second_read** (str, required) – Part of the file name that marks all files containing sequencing data of the second read. Example: 'R2.fastq.gz' or '_2.fastq'
  - default value: _R2

- **unaligned_included** (bool, optional) – Is the typical Unaligned folder included in path?
  - default value: True

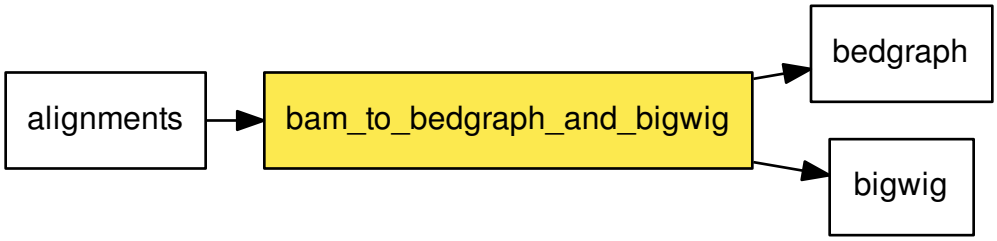
This step provides input files which already exists and therefore creates no tasks in the pipeline.

### 1.12.2 Processing steps

#### bam_to_bedgraph_and_bigwig

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/bedgraph'
  - 'out/bigwig'




**Options:**

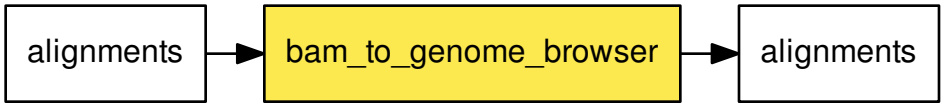
- **chromosome-sizes** (str, required) - **temp-sort-dir** (str, optional)**Required tools:** bedGraphToBigWig, bedtools, sort

**CPU Cores:** 8

#### bam_to_genome_browser

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/alignments'




**Options:**

- **bedtools-bamtobed-color** (str, optional) - **bedtools-bamtobed-tag** (str, optional) - **bedtools-genomecov-3** (bool, optional) - **bedtools-genomecov-5** (bool, optional) - **bedtools-genomecov-max** (int, optional) - **bedtools-genomecov-report-zero-coverage** (bool, required) - **bedtools-genomecov-scale** (float, optional) - **bedtools-genomecov-split** (bool, required) - default value: True
- **bedtools-genomecov-strand** (str, optional) - possible values: '+', '-'
- **chromosome-sizes** (str, required) - **dd-blocksize** (str, optional) - default value: 256k
- **output-format** (str, required) - default value: bigWig - possible values: 'bed', 'bigBed', 'bedGraph', 'bigWig'
- **trackline** (dict, optional) - **trackopts** (dict, optional)**Required tools:** bedGraphToBigWig, bedToBigBed, bedtools, dd, mkfifo, pigz

**CPU Cores:** 8

### bowtie2

Bowtie2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. It is particularly good at aligning reads of about 50 up to 100s or 1,000s of characters, and particularly good at aligning to relatively long (e.g. mammalian) genomes. Bowtie 2 indexes the genome with an FM Index to keep its memory footprint small: for the human genome, its memory footprint is typically around 3.2 GB. Bowtie 2 supports gapped, local, and paired-end alignment modes.

The input reads must come as .f[ast]q[.gz] files.

http://bowtie-bio.sourceforge.net/bowtie2/index.shtml

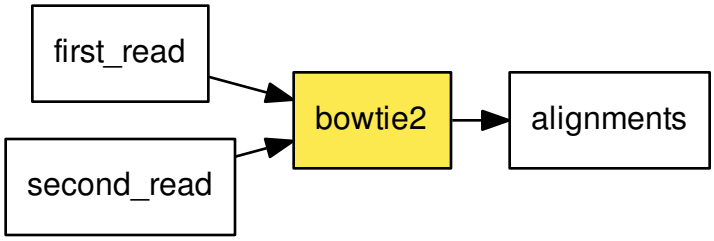
typical command line:

```
bowtie2 [options]* -x <bt2-idx> {-1 <m1> -2 <m2> | -U <r>} -S [<hit>]
```

This step wraps release: bowtie2

**Connections:**

- Input Connection:
  - 'in/first_read'
  - 'in/second_read'
- Output Connection:
  - 'out/alignments'




**Options:**

- **D** (int, optional) – give up extending after <int> failed extends in a row (default=15)
- **L** (int, optional) – length of seed substrings; must be >3, <32 (default=22)
- **N** (int, optional) – Max # mismatches in seed alignment; can be 0 or 1 (default=0)
- **R** (int, optional) – for reads w/ repetitive seeds, try <int> sets of seeds (default=2)
- **all** (bool, optional) – report all alignments; very slow, MAPQ not meaningful
- **compress_add_output** (bool, optional) – Ads -gz to the 4 options above to produce compressed output.
- **dd-blocksize** (str, optional) - default value: 2M
- **dpad** (int, optional) – include <int> extra ref chars on sides of DP table (default=15)
- **end-to-end** (bool, optional) – entire read must align; no clipping. Decide forthis or local option. (default)
- **fast** (bool, optional) – Preset, same as: -D 10 -R 2 -N 0 -L 22 -i S,0,2.50
- **fast-local** (bool, optional) – Preset, same as: -D 10 -R 2 -N 0 -L 22 -i S,1,1.75
- **ff** (bool, optional) – -1, -2 mates align fw/fw
- **fr** (bool, optional) – -1, -2 mates align fw/rev (default)
- **gbar** (int, optional) – disallow gaps within <int> nucs of read extremes (default=4)
- **i** (str, optional) – interval between seed substrings w/r/t read len (default="S,1,1.15")
- **ignore-quals** (bool, optional) – treat all quality values as 30 on Phred scale
- **index** (str, required) – Path to bowtie2 index (not containing file suffixes).
- **int-quals** (bool, optional) – Qualities encoded as space-delimited integers
- **k** (int, optional) – report up to <int> alns per read; MAPQ not meaningful
- **local** (bool, optional) – local alignment; ends might be soft clipped. Decide for this or end-to-end option.
- **ma** (int, optional) – match bonus (0 for end-to-end, 2 for local). (default=0)
- **maxins** (int, optional) – maximum fragment length (default=500)
- **met** (int, optional) – report internal counters & metrics every <int> secs (default=1)
- **met-stderr** (bool, optional) – send metrics to stderr
- **minins** (int, optional) – minimum fragment length (default=0)

- **mm** (bool, optional) – use memory-mapped I/O for index; many 'bowtie's can share
- **mp** (int, optional) – max penalty for mismatch; lower qual = lower penalty (default=6)
- **n-ceil** (str, optional) – func for max # non-A/C/G/Ts permitted in aln (default="L,0,0.15")
- **no-1mm-upfront** (bool, optional) – do not allow 1 mismatch alignments before attempting to scan for the optimal seeded alignments
- **no-contain** (bool, optional) – not concordant when one mate alignment contains other
- **no-discordant** (bool, optional) – suppress discordant alignments for paired reads
- **no-dovetail** (bool, optional) – not concordant when mates extend past each other
- **no-head** (bool, optional) – suppress header lines, i.e. lines starting with @
- **no-mixed** (bool, optional) – suppress unpaired alignments for paired reads
- **no-overlap** (bool, optional) – not concordant when mates overlap at all
- **no-sq** (bool, optional) – suppress @SQ header lines
- **no-unal** (bool, optional) – suppress SAM records for unaligned reads
- **nofw** (bool, optional) – do not align forward (original) version of read
- **non-deterministic** (bool, optional) – seed rand. gen. arbitrarily instead of using read attributes
- **norc** (bool, optional) – do not align reverse-complement version of read
- **np** (int, optional) – penalty for non-A/C/G/Ts in read/ref (default=1)
- **omit-sec-seq** (bool, optional) – put * in SEQ and QUAL fields for secondary alignments
- **phred33** (bool, optional) – Qualities are Phred+33 (default)
- **phred64** (bool, optional) – Qualities are Phred+64
- **pigz-blocksize** (str, optional) - default value: 2048
- **qc-filter** (bool, optional) – filter out reads that are bad according to QSEQ filter
- **quiet** (bool, optional) – print nothing to stderr except serious errors
- **rdg** (str, optional) – read gap open, extend penalties (default="5,3")
- **reorder** (bool, optional) – force SAM output order to match order of input reads
- **rf** (bool, optional) – -1, -2 mates align rev/fw
- **rfg** (str, optional) – reference gap open, extend penalties(default="5,3")
- **rg** (str, optional) – add <text> (lab:value) to @RG line of SAM header.Note: @RG line only printed when –rg-id is set.
- **rg-id** (str, optional) – set read group id, reflected in @RG line and RG:Z: opt field
- **score-min** (str, optional) – min acceptable alignment score w/r/t read length(G,20,8 for local, L,-0.6,-0.6 for end-to-end)(default="L,-0.6,-0.6")
- **seed** (int, optional) – seed for random number generator (default=0)
- **sensitive** (bool, optional) – Preset, same as: -D 15 -R 2 -N 0 -L 22 -i S,1,1.15 (default)
- **sensitive-local** (bool, optional) – Preset, same as: -D 15 -R 2 -N 0 -L 20 -i S,1,0.75 (default)
- **skip** (int, optional) – Skip the first <int> reads/pairs in the input. Default: none
- **threads** (int, optional) – number of alignment threads to launch (default=1)

- **time** (bool, optional) – print wall-clock time taken by search phases
- **trim3** (int, optional) – Trim <int> bases from 3'/right end of reads(default=0)
- **trim5** (int, optional) – Trim <int> bases from 5'/left end of reads (default=0)
- **upto** (int, optional) – Stop after the first <int> reads/pairs in the input. Default: no limit.
- **very-fast** (bool, optional) – Preset, same as: -D 5 -R 1 -N 0 -L 22 -i S,0,2.50
- **very-fast-local** (bool, optional) – Preset, same as: -D 5 -R 1 -N 0 -L 25 -i S,1,2.00
- **very-sensitive** (bool, optional) – Preset, same as: -D 20 -R 3 -N 0 -L 20 -i S,1,0.50
- **very-sensitive-local** (bool, optional) – Preset, same as: -D 20 -R 3 -N 0 -L 20 -i S,1,0.50

**Required tools:** bowtie2, dd, mkfifo, pigz

**CPU Cores:** 6

### bowtie2_generate_index

bowtie2-build builds a Bowtie index from a set of DNA sequences. bowtie2-build outputs a set of 6 files with suffixes .1.bt2, .2.bt2, .3.bt2, .4.bt2, .rev.1.bt2, and .rev.2.bt2. In the case of a large index these suffixes will have a bt2l termination. These files together constitute the index: they are all that is needed to align reads to that reference. The original sequence FASTA files are no longer used by Bowtie 2 once the index is built.

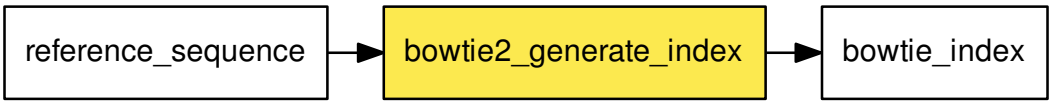
http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml#the-bowtie2-build-indexer

typical command line:

```
bowtie2-build [options]* <reference_in> <bt2_index_base>
```

**Connections:**

- Input Connection:
  - 'in/reference_sequence'
- Output Connection:
  - 'out/bowtie_index'




**Options:**

- **bmax** (int, optional) – The maximum number of suffixes allowed in a block. Allowing more suffixes per block makes indexing faster, but increases peak memory usage. Setting this option overrides any previous setting for –bmax, or –bmaxdivn. Default (in terms of the –bmaxdivn parameter) is –bmaxdivn 4. This is configured automatically by default; use -a/–noauto to configure manually.

- **bmaxdivn** (int, optional) – The maximum number of suffixes allowed in a block, expressed as a fraction of the length of the reference. Setting this option overrides any previous setting for –bmax, or –bmaxdivn. Default: –bmaxdivn 4. This is configured automatically by default; use -a/–noauto to configure manually.
- **cutoff** (int, optional) – Index only the first <int> bases of the reference sequences (cumulative across sequences) and ignore the rest.
- **dcv** (int, optional) – Use <int> as the period for the difference-cover sample. A larger period yields less memory overhead, but may make suffix sorting slower, especially if repeats are present. Must be a power of 2 no greater than 4096. Default: 1024. This is configured automatically by default; use -a/–noauto to configure manually.
- **dd-blocksize** (str, optional) - default value: 2M
- **ftabchars** (int, optional) – The ftab is the lookup table used to calculate an initial Burrows-Wheeler range with respect to the first <int> characters of the query. A larger <int> yields a larger lookup table but faster query times. The ftab has size 4^(<int>+1) bytes. The default setting is 10 (ftab is 4MB).
- **index-basename** (str, required) – Base name used for the bowtie2 index.
- **large-index** (bool, optional) – Force bowtie2-build to build a large index, even if the reference is less than 4 billion nucleotides long.
- **noauto** (bool, optional) – Disable the default behavior whereby bowtie2-build automatically selects values for the –bmax, –dcv and –packed parameters according to available memory. Instead, user may specify values for those parameters. If memory is exhausted during indexing, an error message will be printed; it is up to the user to try new parameters.
- **nodc** (bool, optional) – Disable use of the difference-cover sample. Suffix sorting becomes quadratic-time in the worst case (where the worst case is an extremely repetitive reference). Default: off.
- **offrate** (int, optional) – To map alignments back to positions on the reference sequences, it's necessary to annotate ('mark') some or all of the Burrows-Wheeler rows with their corresponding location on the genome. -o/–offrate governs how many rows get marked: the indexer will mark every 2^<int> rows. Marking more rows makes reference-position lookups faster, but requires more memory to hold the annotations at runtime. The default is 5 (every 32nd row is marked; for human genome, annotations occupy about 340 megabytes).
- **packed** (bool, optional) – Use a packed (2-bits-per-nucleotide) representation for DNA strings. This saves memory but makes indexing 2-3 times slower. Default: off. This is configured automatically by default; use -a/–noauto to configure manually.
- **pigz-blocksize** (str, optional) - default value: 2048
- **seed** (int, optional) – Use <int> as the seed for pseudo-random number generator.
- **threads** (int, optional) – By default bowtie2-build is using only one thread. Increasing the number of threads will speed up the index building considerably in most cases.

**Required tools:** bowtie2-build, dd, pigz

**CPU Cores:** 6

### bwa_backtrack

bwa-backtrack is the bwa algorithm designed for Illumina sequence reads up to 100bp. The computation of the alignments is done by running 'bwa aln' first, to align the reads, followed by running 'bwa samse' or 'bwa sampe' afterwards to generate the final SAM output.

http://bio-bwa.sourceforge.net/

typical command line for single-end data:

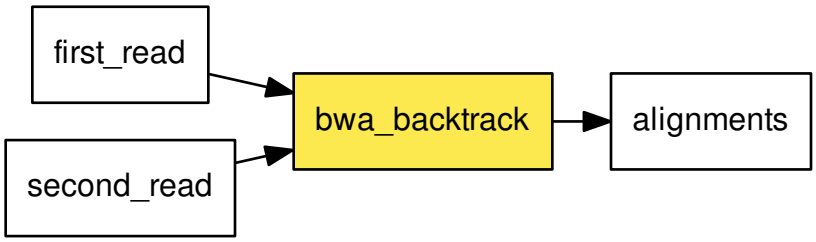
```
bwa aln <bwa-index> <first-read.fastq> > <first-read.sai>
bwa samse <bwa-index> <first-read.sai> <first-read.fastq> > <sam-output>
```

typical command line for paired-end data:

```
bwa aln <bwa-index> <first-read.fastq> > <first-read.sai>
bwa aln <bwa-index> <second-read.fastq> > <second-read.sai>
bwa sampe <bwa-index> <first-read.sai> <second-read.sai>
↪<first-read.fastq> <second-read.fastq> > <sam-output>
```

**Connections:**

- Input Connection:
  - 'in/first_read'
  - 'in/second_read'
- Output Connection:
  - 'out/alignments'




**Options:**

- **aln-0** (bool, optional) – When aln-b is specified, only use single-end reads in mapping.
- **aln-1** (bool, optional) – When aln-b is specified, only use the first read in a read pair in mapping (skip single-end reads and the second reads).
- **aln-2** (bool, optional) – When aln-b is specified, only use the second read in a read pair in mapping.
- **aln-B** (int, optional) – Length of barcode starting from the 5'-end. When INT is positive, the barcode of each read will be trimmed before mapping and will be written at the BC SAM tag. For paired-end reads, the barcode from both ends are concatenated. [0]
- **aln-E** (int, optional) – Gap extension penalty [4]
- **aln-I** (bool, optional) – The input is in the Illumina 1.3+ read format (quality equals ASCII-64).
- **aln-M** (int, optional) – Mismatch penalty. BWA will not search for suboptimal hits with a score lower than (bestScore-misMsc). [3]
- **aln-N** (bool, optional) – Disable iterative search. All hits with no more than maxDiff differences will be found. This mode is much slower than the default.
- **aln-O** (int, optional) – Gap open penalty [11]

- **aln-R** (int, optional) – Proceed with suboptimal alignments if there are no more than INT equally best hits. This option only affects paired-end mapping. Increasing this threshold helps to improve the pairing accuracy at the cost of speed, especially for short reads (~32bp).
- **aln-b** (bool, optional) – Specify the input read sequence file is the BAM format. For paired-end data, two ends in a pair must be grouped together and options aln-1 or aln-2 are usually applied to specify which end should be mapped. Typical command lines for mapping pair-end data in the BAM format are:

bwa aln ref.fa -b1 reads.bam > 1.sai bwa aln ref.fa -b2 reads.bam > 2.sai bwa sampe ref.fa 1.sai 2.sai reads.bam reads.bam > aln.sam

- **aln-c** (bool, optional) – Reverse query but not complement it, which is required for alignment in the color space. (Disabled since 0.6.x)
- **aln-d** (int, optional) – Disallow a long deletion within INT bp towards the 3'-end [16]
- **aln-e** (int, optional) – Maximum number of gap extensions, -1 for k-difference mode (disallowing long gaps) [-1]
- **aln-i** (int, optional) – Disallow an indel within INT bp towards the ends [5]
- **aln-k** (int, optional) – Maximum edit distance in the seed [2]
- **aln-l** (int, optional) – Take the first INT subsequence as seed. If INT is larger than the query sequence, seeding will be disabled. For long reads, this option is typically ranged from 25 to 35 for '-k 2'. [inf]
- **aln-n** (float, optional) – Maximum edit distance if the value is INT, or the fraction of missing alignments given 2% uniform base error rate if FLOAT. In the latter case, the maximum edit distance is automatically chosen for different read lengths. [0.04]
- **aln-o** (int, optional) – Maximum number of gap opens [1]
- **aln-q** (int, optional) – Parameter for read trimming. BWA trims a read down to argmax_x{sum_{i=x+1}^l(INT-q_i)} if q_l<INT where l is the original read length. [0]
- **aln-t** (int, optional) – Number of threads (multi-threading mode) [1]
  - default value: 1
- **dd-blocksize** (str, optional) - default value: 2M
- **index** (str, required) – Path to BWA index
- **pigz-blocksize** (str, optional) - default value: 2048
- **sampe-N** (int, optional) – Maximum number of alignments to output in the XA tag for disconcordant read pairs (excluding singletons). If a read has more than INT hits, the XA tag will not be written. [10]
- **sampe-P** (bool, optional) – Load the entire FM-index into memory to reduce disk operations (base-space reads only). With this option, at least 1.25N bytes of memory are required, where N is the length of the genome.
- **sampe-a** (int, optional) – Maximum insert size for a read pair to be considered being mapped properly. Since 0.4.5, this option is only used when there are not enough good alignment to infer the distribution of insert sizes. [500]
- **sampe-n** (int, optional) – Maximum number of alignments to output in the XA tag for reads paired properly. If a read has more than INT hits, the XA tag will not be written. [3]
- **sampe-o** (int, optional) – Maximum occurrences of a read for pairing. A read with more occurrneces will be treated as a single-end read. Reducing this parameter helps faster pairing. [100000]
- **sampe-r** (str, optional) – Specify the read group in a format like '@RG ID:foo SM:bar'. [null]

- **samse-n** (int, optional) – Maximum number of alignments to output in the XA tag for reads paired properly. If a read has more than INT hits, the XA tag will not be written. [3]
- **samse-r** (str, optional) – Specify the read group in a format like '@RG ID:foo SM:bar'. [null]

**Required tools:** bwa, dd, mkfifo, pigz

**CPU Cores:** 8

### bwa_generate_index

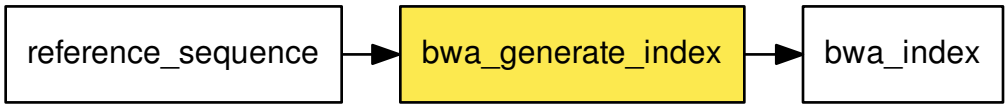
This step generates the index database from sequences in the FASTA format.

Typical command line:

```
bwa index -p <index-basename> <seqeunce.fasta>
```

**Connections:**

- Input Connection:
  - 'in/reference_sequence'
- Output Connection:
  - 'out/bwa_index'




**Options:**

- **index-basename** (str, required) – Prefix of the created index database

**Required tools:** bwa

**CPU Cores:** 6

### bwa_mem

Align 70bp-1Mbp query sequences with the BWA-MEM algorithm. Briefly, the algorithm works by seeding alignments with maximal exact matches (MEMs) and then extending seeds with the affine-gap Smith-Waterman algorithm (SW).

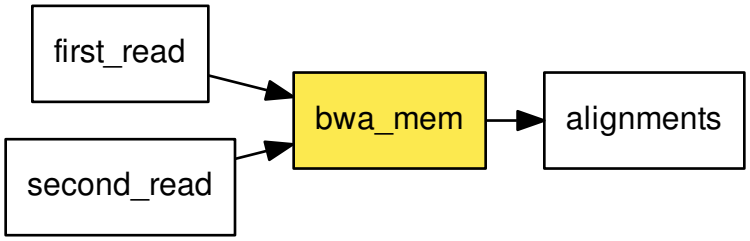
http://bio-bwa.sourceforge.net/bwa.shtml

Typical command line:

```
bwa mem [options] <bwa-index> <first-read.fastq> [<second-read.fastq>]
↪  > <sam-output>
```

**Connections:**

- Input Connection:
  - 'in/first_read'
  - 'in/second_read'
- Output Connection:
  - 'out/alignments'




**Options:**

- **A** (int, optional) – score for a sequence match, which scales options -TdBOELU unless overridden [1]
- **B** (int, optional) – penalty for a mismatch [4]
- **C** (bool, optional) – append FASTA/FASTQ comment to SAM output
- **D** (float, optional) – drop chains shorter than FLOAT fraction of the longest overlapping chain [0.50]
- **E** (str, optional) – gap extension penalty; a gap of size k cost '{-O} + {-E}*k' [1,1]
- **H** (str, optional) – insert STR to header if it starts with @; or insert lines in FILE [null]
- **L** (str, optional) – penalty for 5'- and 3'-end clipping [5,5]
- **M** (str, optional) – mark shorter split hits as secondary
- **O** (str, optional) – gap open penalties for deletions and insertions [6,6]
- **P** (bool, optional) – skip pairing; mate rescue performed unless -S also in use
- **R** (str, optional) – read group header line such as '@RG ID:foo SM:bar' [null]
- **S** (bool, optional) – skip mate rescue
- **T** (int, optional) – minimum score to output [30]
- **U** (int, optional) – penalty for an unpaired read pair [17]
- **V** (bool, optional) – output the reference FASTA header in the XR tag
- **W** (int, optional) – discard a chain if seeded bases shorter than INT [0]
- **Y** (str, optional) – use soft clipping for supplementary alignments
- **a** (bool, optional) – output all alignments for SE or unpaired PE
- **c** (int, optional) – skip seeds with more than INT occurrences [500]
- **d** (int, optional) – off-diagonal X-dropoff [100]

- **dd-blocksize** (str, optional) - default value: 256k
- **e** (bool, optional) – discard full-length exact matches
- **h** (str, optional) – if there are <INT hits with score >80% of the max score, output all in XA [5,200]
- **index** (str, required) – Path to BWA index
- **j** (bool, optional) – treat ALT contigs as part of the primary assembly (i.e. ignore <idxbase>.alt file)
- **k** (int, optional) – minimum seed length [19]
- **m** (int, optional) – perform at most INT rounds of mate rescues for each read [50]
- **p** (bool, optional) – smart pairing (ignoring in2.fq)
- **r** (float, optional) – look for internal seeds inside a seed longer than {-k} * FLOAT [1.5]
- **t** (int, optional) – number of threads [6]
  - default value: 6
- **v** (int, optional) – verbose level: 1=error, 2=warning, 3=message, 4+=debugging [3]
- **w** (int, optional) – band width for banded alignment [100]
- **x** (str, optional) – read type. Setting -x changes multiple parameters unless overriden [null]

pacbio: -k17 -W40 -r10 -A1 -B1 -O1 -E1 -L0 (PacBio reads to ref) ont2d: -k14 -W20 -r10 -A1 -B1 -O1 -E1 -L0 (Oxford Nanopore 2D-reads to ref) intractg: -B9 -O16 -L5 (intra-species contigs to ref)

- **y** (int, optional) – seed occurrence for the 3rd round seeding [20]

**Required tools:** bwa, dd, mkfifo, pigz

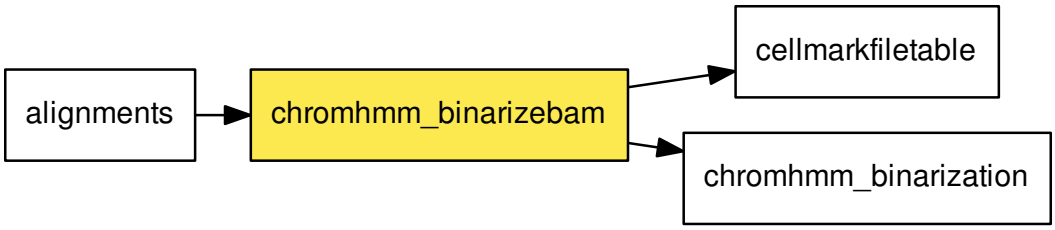
**CPU Cores:** 6

### chromhmm_binarizebam

This command converts coordinates of aligned reads into binarized data form from which a chromatin state model can be learned. The binarization is based on a poisson background model. If no control data is specified the parameter to the poisson distribution is the global average number of reads per bin. If control data is specified the global average number of reads is multiplied by the local enrichment for control reads as determined by the specified parameters. Optionally intermediate signal files can also be outputted and these signal files can later be directly converted into binary form using the BinarizeSignal command.

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/cellmarkfiletable'
  - 'out/chromhmm_binarization'




**Options:**

- **b** (int, optional) – The number of base pairs in a bin determining the resolution of the model learning and segmentation. By default this parameter value is set to 200 base pairs.
- **cell_mark_files** (dict, required) – A dictionary where the keys are the names of the run and the values are lists of lists. The lists of lists describe the content of a 'cellmarkfiletable' files as used by 'BinarizeBam'. But instead of file names use the run ID for the mark and control per line. That is a tab delimited file where each row contains the cell type or other identifier for a groups of marks, then the associated mark, then the name of a BAM file, and optionally a corresponding control BAM file. If a mark is missing in one cell type, but not others it will receive a 2 for all entries in the binarization file and -1 in the signal file. If the same cell and mark combination appears on multiple lines, then the union of all the reads across entries is taken except for control data where each unique file is only counted once.
- **center** (bool, optional) – If this flag is present then the center of the interval is used to determine the bin to assign a read. This can make sense to use if the coordinates are based on already extended reads. If this option is selected, then the strand information of a read and the shift parameter are ignored. By default reads are assigned to a bin based on the position of its 5' end as determined from the strand of the read after shifting an amount determined by the -n shift option.
- **chrom_sizes_file** (str, required) - **e** (int, optional) – Specifies the amount that should be subtracted from the end coordinate of a read so that both coordinates are inclusive and 0 based. The default value is 1 corresponding to standard bed convention of the end interval being 0-based but not inclusive.
- **f** (int, optional) – This indicates a threshold for the fold enrichment over expected that must be met or exceeded by the observed count in a bin for a present call. The expectation is determined in the same way as the mean parameter for the poission distribution in terms of being based on a uniform background unless control data is specified. This parameter can be useful when dealing with very deeply and/or unevenly sequenced data. By default this parameter value is 0 meaning effectively it is not used.
- **g** (int, optional) – This indicates a threshold for the signal that must be met or exceeded by the observed count in a bin for a present call. This parameter can be useful when desiring to directly place a threshold on the signal. By default this parameter value is 0 meaning effectively it is not used.
- **n** (int, optional) – The number of bases a read should be shifted to determine a bin assignment. Bin assignment is based on the 5' end of a read shifted this amount with respect to the strand orientation. By default this value is 100.
- **p** (float, optional) – This option specifies the tail probability of the poisson distribution that the binarization threshold should correspond to. The default value of this parameter is 0.0001.
- **s** (int, optional) – The amount that should be subtracted from the interval start coordinate so the interval is inclusive and 0 based. Default is 0 corresponding to the standard bed convention.

- **strictthresh** (bool, optional) – If this flag is present then the poisson threshold must be strictly greater than the tail probability, otherwise by default the largest integer count for which the tail includes the poisson threshold probability is used.
- **u** (int, optional) – An integer pseudocount that is uniformly added to every bin in the control data in order to smooth the control data from 0. The default value is 1.
- **w** (int, optional) – This determines the extent of the spatial smoothing in computing the local enrichment for control reads. The local enrichment for control signal in the x-th bin on the chromosome after adding pseudocountcontrol is computed based on the average control counts for all bins within x-w and x+w. If no controldir is specified, then this option is ignored. The default value is 5.

**Required tools:** ChromHMM, ln, ls, mkdir, printf, tar, xargs

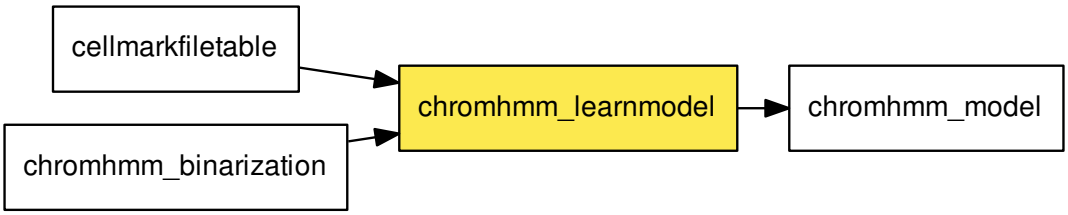
**CPU Cores:** 4

### chromhmm_learnmodel

This command takes a directory with a set of binarized data files and learns a chromatin state model. Binarized data files have "_binary" in the file name. The format for the binarized data files are that the first line contains the name of the cell separated by a tab with the name of the chromosome. The second line contains in tab delimited form the name of each mark. The remaining lines correspond to consecutive bins on the chromosome. The remaining lines in tab delimited form corresponding to each mark, with a "1" for a present call or "0" for an absent call and a "2" if the data is considered missing at that interval for the mark.

**Connections:**

- Input Connection:
  - 'in/cellmarkfiletable'
  - 'in/chromhmm_binarization'
- Output Connection:
  - 'out/chromhmm_model'




**Options:**

- **assembly** (str, required) – specifies the genome assembly. overlap and neighborhood enrichments will be called with default parameters using this genome assembly.Assembly names are e.g. hg18, hg19, GRCh38
- **b** (int, optional) – The number of base pairs in a bin determining the resolution of the model learning and segmentation. By default this parameter value is set to 200 base pairs.

- **color** (str, optional) – This specifies the color of the heat map. "r,g,b" are integer values between 0 and 255 separated by commas. By default this parameter value is 0,0,255 corresponding to blue.
- **d** (float, optional) – The threshold on the change on the estimated log likelihood that if it falls below this value, then parameter training will terminate. If this value is less than 0 then it is not used as part of the stopping criteria. The default value for this parameter is 0.001.
- **e** (float, optional) – This parameter is only applicable if the load option is selected for the init parameter. This parameter controls the smoothing away from 0 when loading a model. The emission value used in the model initialization is a weighted average of the value in the file and a uniform probability over the two possible emissions. The value in the file gets weight (1-loadsmoothemission) while uniform gets weight loadsmoothemission. The default value of this parameter is 0.02.
- **h** (float, optional) – A smoothing constant away from 0 for all parameters in the information based initialization. This option is ignored if random or load are selected for the initialization method. The default value of this parameter is 0.02.
- **holdcolumnorder** (bool, optional) – Including this flag suppresses the reordering of the mark columns in the emission parameter table display.
- **init** (str, optional) – This specifies the method for parameter initialization method. 'information' is the default method described in (Ernst and Kellis, Nature Methods 2012). 'random' - randomly initializes the parameters from a uniform distribution. 'load' loads the parameters specified in '-m modelinitialfile' and smooths them based on the value of the 'loadsmoothemission' and 'loadsmoothtransition' parameters. The default is information.
  - possible values: 'information', 'random', 'load'
- **l** (str, optional) – This file specifies the length of the chromosomes. It is a two column tab delimited file with the first column specifying the chromosome name and the second column the length. If this file is provided then no end coordinate will exceed what is specified in this file. By default BinarizeBed excludes the last partial bin along the chromosome, but if that is included in the binarized data input files then this file should be included to give a valid end coordinate for the last interval.
- **m** (str, optional) – This specifies the model file containing the initial parameters which can then be used with the load option
- **nobed** (bool, optional) – If this flag is present, then this suppresses the printing of segmentation information in the four column format. The default is to generate a four column segmentation file
- **nobrowser** (bool, optional) – If this flag is present, then browser files are not printed. If -nobed is requested then browserfile writing is also suppressed.
- **noenrich** (bool, optional) – If this flag is present, then enrichment files are not printed. If -nobed is requested then enrichment file writing is also suppressed.
- **numstates** (int, required) - **r** (int, optional) – This option specifies the maximum number of iterations over all the input data in the training. By default this is set to 200.
- **s** (int, optional) – This allows the specification of the random seed. Randomization is used to determine the visit order of chromosomes in the incremental expectation-maximization algorithm used to train the parameters and also used to generate the initial values of the parameters if random is specified for the init method.
- **stateordering** (str, optional) – This determines whether the states are ordered based on the emission or transition parameters. See (Ernst and Kellis, Nature Methods) for details. Default is 'emission'.
  - possible values: 'emission', 'transition'
- **t** (float, optional) – This parameter is only applicable if the load option is selected for the init parameter. This parameter controls the smoothing away from 0 when loading a model. The transition value used in the model initialization is a weighted average of the value in the file and a uniform probability

over the transitions. The value in the file gets weight (1-loadsmoothtransition) while uniform gets weight loadsmoothtransition. The default value is 0.5.

- **x** (int, optional) – This parameter specifies the maximum number of seconds that can be spent optimizing the model parameters. If it is less than 0, then there is no limit and termination is based on maximum number of iterations or a log likelihood change criteria. The default value of this parameter is -1.
- **z** (int, optional) – This parameter determines the threshold at which to set extremely low transition probabilities to 0 durining training. Setting extremely low transition probabilities makes model learning more efficient with essentially no impact on the final results. If a transition probability falls below 10^-zerotransitionpower during training it is set to 0. Making this parameter to low and thus the cutoff too high can potentially cause some numerical instability. By default this parameter is set to 8.

**Required tools:** ChromHMM, ls, mkdir, rm, tar, xargs

**CPU Cores:** 8

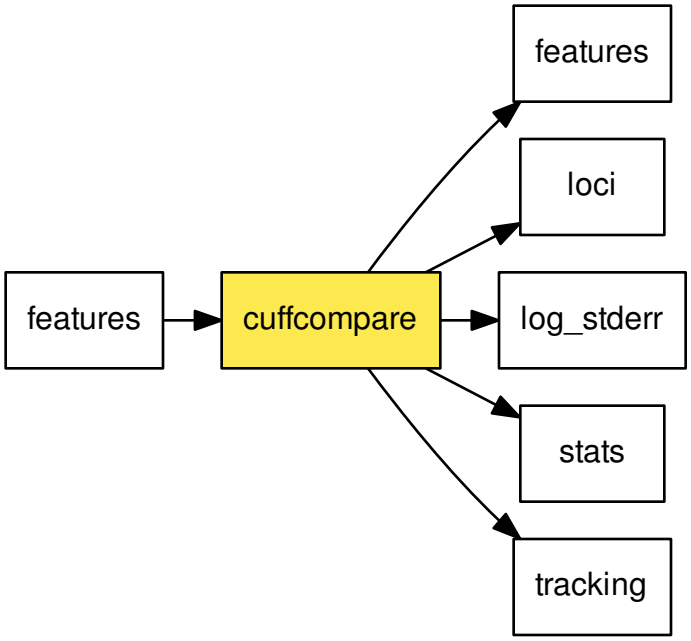
### cuffcompare

CuffCompare is part of the 'Cufflinks suite of tools' for differential expr. analysis of RNA-Seq data and their visualisation. This step compares a cufflinks assembly to known annotation. Cuffcompare provides classification, reference annotation mapping and various statistics for Cufflinks transfrags. For details about cuffcompare we refer to the author's webpage:

http://cole-trapnell-lab.github.io/cufflinks/cuffcompare/

**Connections:**

- Input Connection:
  - 'in/features'
- Output Connection:
  - 'out/features'
  - 'out/loci'
  - 'out/log_stderr'
  - 'out/stats'
  - 'out/tracking'




**Options:**

- **C** (bool, optional) – Enables the "contained" transcripts to be also written in the .combined.gtffile, with the attribute "contained_in" showing the first container transfrag found. By default, without this option, cuffcompare does not write in that file isoforms that were found to be fully contained/covered (with the same compatible intron structure) by other transfrags in the same locus.transcripts
- **F** (bool, optional) – Do not discard intron-redundant transfrags if they share the 5p end (if they differ only at the 3p end)
- **G** (bool, optional) – generic GFF input file(s): do not assume Cufflinks GTF, do not discard any intron-redundant transfrags
- **M** (bool, optional) – Discard (ignore) single-exon transfrags and reference transcripts
- **N** (bool, optional) – Discard (ignore) single-exon reference
- **Q** (bool, optional) – For "-r" option, consider only the input transcripts that overlap any of the reference transcripts (Sp-correction)
- **R** (bool, optional) – For "-r" option, consider only the reference transcripts that overlap any of the input transfrags (Sn-correction)
- **V** (bool, optional) – verbose processing mode (showing all GFF parsing warnings)
- **d** (int, optional) – Max. distance (range) for grouping transcript start sites (Default: 100)
- **e** (int, optional) – Max. distance (range) allowed from free ends of terminal exons of reference transcripts when assessing exon accuracy (Default: 100)
- **r** (str, optional) – An optional "reference" annotation GFF file containing a set of known mRNAs to use as a reference for assessing the accuracy of mRNAs or gene models given in <input.gtf>

- **run_id** (str, optional) – An arbitrary name of the new run (which is a merge of all samples).
  - default value: magic
- **s** (str, optional) – Can be a multi-fasta file with all the genomic sequences or a directory containing multiple single-fasta files (one file per contig); lower case bases will be used to classify input transcripts as repeats. NOTE that must contain one fasta file per reference chromosome, and each file must be named after the chromosome, and have a .fa or .fasta extension.

**Required tools:** cuffcompare

**CPU Cores:** 1

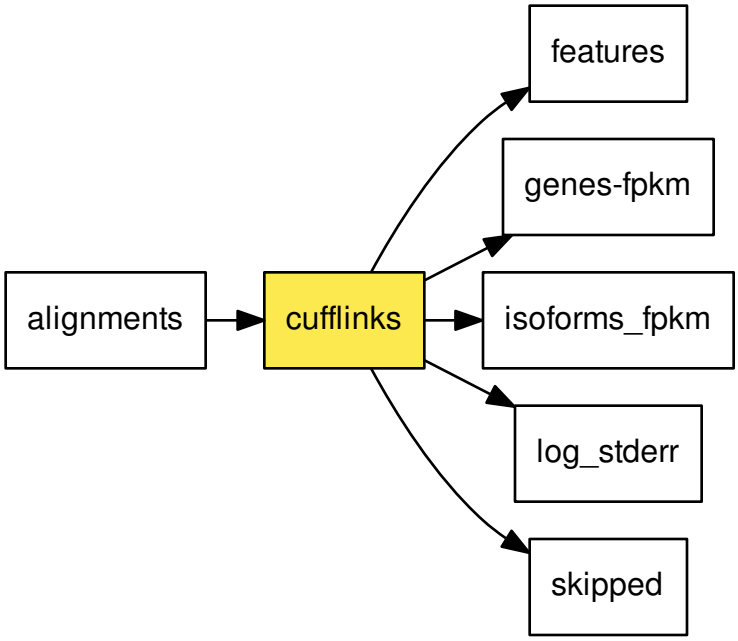
### cufflinks

CuffLinks is part of the 'Cufflinks suite of tools' for differential expr. analysis of RNA-Seq data and their visualisation. This step applies the cufflinks tool which assembles transcriptomes from RNA-Seq data and quantifies their expression and produces .gtf files with these annotations. For details on cufflinks we refer to the author's webpage:

http://cole-trapnell-lab.github.io/cufflinks/

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/features'
  - 'out/genes-fpkm'
  - 'out/isoforms_fpkm'
  - 'out/log_stderr'
  - 'out/skipped'




**Options:**

- **3-overhang-tolerance** (int, optional) – The number of bp allowed to overhang the 3 prime end of a reference transcript when determining if an assembled transcript should be merged with it (i.e., the assembled transcript is not novel). Default: 600
- **GTF** (bool, optional) – Quantitate against reference transcript annotations. Use with either RABT or ab initio assembly is not supported.
- **GTF-guide** (bool, optional) – Tells Cufflinks to use the supplied reference annotation a GFF file to guide RABT assembly. Reference transcripts will be tiled with faux-reads to provide additional information in assembly. Output will include all reference transcripts as well as any novel genes and isoforms that are assembled.
- **compatible-hits-norm** (bool, optional) – With this option, Cufflinks counts only those fragments compatible with some reference transcript towards the number of mapped hits used in the FPKM denominator. This option can be combined with -N/–upper-quartile-norm. Default: FALSE
- **frag-bias-correct** (str, optional) – Providing Cufflinks with a multifasta file via this option instructs it to run our new bias detection and correction algorithm which can significantly improve accuracy of transcript abundance estimates. Default: NULL
- **frag-len-mean** (int, optional) – This is the expected (mean) fragment length. The default is 200bp. Note: Cufflinks now learns the fragment length mean for each SAM file, so using this option is no longer recommended with paired-end reads. Default: 200
- **frag-len-std-dev** (int, optional) – The standard deviation for the distribution on fragment lengths. The default is 80bp. Note: Cufflinks now learns the fragment length standard deviation for each SAM file, so using this option is no longer recommended with paired-end reads. Default: 80
- **intron-overhang-tolerance** (int, optional) – The number of bp allowed to enter the intron of a reference

transcript when determining if an assembled transcript should be merged with it (i.e., the assembled transcript is not novel). Default: 50

- **junc-alpha** (float, optional) – The alpha value for the binomial test used during false positive spliced alignment filtration. Default: 0.001
- **label** (str, optional) – Cufflinks will report transfrags in GTF format, with a prefix given by this option. Default: CUFF
- **library-norm-method** (str, optional) – You can control how library sizes (i.e. sequencing depths) are normalized in Cufflinks and Cuffdiff. Cuffdiff has several methods that require multiple libraries in order to work. Library normalization methods supported by Cufflinks work on one library at a time. Normalization Method supported by Cufflinks: classic-fpkm (Library size factor is set to 1 - no scaling applied to FPKM values or fragment counts. Default: classic-fpkm
  - possible values: 'classic-fpkm'
- **library-type** (str, required) – In cases where Cufflinks cannot determine the platform and protocol used to generate input reads, you can supply this information manually, which will allow Cufflinks to infer source strand information with certain protocols. The available options are listed below. For paired-end data, we currently only support protocols where reads are point towards each other.Library type: fr-unstranded (default); examples: Standard Illumina; description: Reads from the left-most end of the fragment (in transcript coordinates) map to the transcript strand, and the right-most end maps to the opposite strand.Library type: fr-firststrand; examples: dUTP, NSR, NNSR; description: same as fr-unstranded except we enforce the rule that the right-most end of the fragment (in transcript coordinates) is the first sequenced (or only sequenced for single-end reads). Equivalently, it is assumed that only the strand generated during first strand synthesis is sequenced.Library type: fr-secondstrand; examples: Directional Illumina (Ligation), Standard SOLiD; same as fr-unstranded except we enforce the rule that the left-most end of the fragment (in transcript coordinates) is the first sequenced (or only sequenced for single-end reads). Equivalently, it is assumed that only the strand generated during second strand synthesis is sequenced. Default: fr-unstranded
  - possible values: 'ff-firststrand', 'fr-firststrand', 'ff-secondstrand', 'fr-secondstrand', 'ff-unstranded', 'fr-unstranded', 'transfrags'
- **mask-file** (str, optional) – Tells Cufflinks to ignore all reads that could have come from transcripts in this GTF file. We recommend including any annotated rRNA, mitochondrial transcripts other abundant transcripts you wish to ignore in your analysis in this file. Due to variable efficiency of mRNA enrichment methods and rRNA depletion kits, masking these transcripts often improves the overall robustness of transcript abundance estimates.
- **max-bundle-frags** (int, optional) – Sets the maximum number of fragments a locus may have before being skipped. Skipped loci are listed in skipped.gtf. Default: 500000
- **max-bundle-length** (int, optional) – Maximum genomic length allowed for a given bundle. Default: 3500000
- **max-frag-multihits** (str, optional) – Maximum number of alignments allowed per fragment. Default: unlim
- **max-intron-length** (int, optional) – The maximum intron length. Cufflinks will not report transcripts with introns longer than this, and will ignore SAM alignments with REF_SKIP CIGAR operations longer than this. Default: 300000
- **max-mle-iterations** (int, optional) – Sets the number of iterations allowed during maximum likelihood estimation of abundances. Default: 5000
- **max-multiread-fraction** (float, optional) – The fraction a transfrags supporting reads that may be multiply mapped to the genome. A transcript composed of more than this fraction will not be reported by the assembler. Default: 0.75 (75% multireads or more is suppressed).

- **min-frags-per-transfrag** (int, optional) – Assembled transfrags supported by fewer than this many aligned RNA-Seq fragments are not reported. Default: 10
- **min-intron-length** (int, optional) – Minimum intron size allowed in genome. Default: 50
- **min-isoform-fraction** (float, optional) – After calculating isoform abundance for a gene, Cufflinks filters out transcripts that it believes are very low abundance, because isoforms expressed at extremely low levels often cannot reliably be assembled, and may even be artifacts of incompletely spliced precursors of processed transcripts. This parameter is also used to filter out introns that have far fewer spliced alignments supporting them. The default is 0.1, or 10% of the most abundant isoform (the major isoform) of the gene. Range: 0.0-1.0. Default: 0.10
- **multi-read-correct** (bool, optional) – Tells Cufflinks to do an initial estimation procedure to more accurately weight reads mapping to multiple locations in the genome. Default: FALSE
- **no-effective-length-correction** (bool, optional) – Cufflinks will not employ its "effective" length normalization to transcript FPKM. Default: FALSE
- **no-faux-reads** (bool, optional) – This option disables tiling of the reference transcripts with faux reads. Use this if you only want to use sequencing reads in assembly but do not want to output assembled transcripts that lay within reference transcripts. All reference transcripts in the input annotation will also be included in the output. Default: FALSE
- **no-length-correction** (bool, optional) – Cufflinks will not normalize fragment counts by transcript length at all. Use this option when fragment count is independent of the size of the features being quantified (e.g. for small RNA libraries, where no fragmentation takes place, or 3 prime end sequencing, where sampled RNA fragments are all essentially the same length). Experimental option, use with caution. Default: FALSE
- **no-update-check** (bool, optional) – Turns off the automatic routine that contacts the Cufflinks server to check for a more recent version. Default: FALSE
- **num-frag-assign-draws** (int, optional) – Number of fragment assignment samples per generation. Default: 50
- **num-frag-count-draws** (int, optional) – Number of fragment generation samples. Default: 100
- **num-threads** (int, optional) – Number of threads used during analysis. Default: 1
- **overhang-tolerance** (int, optional) – The number of bp allowed to enter the intron of a transcript when determining if a read or another transcript is mappable to/compatible with it. The default is 8 bp based on the default bowtie/TopHat parameters. Default: 8
- **overlap-radius** (int, optional) – Transfrags that are separated by less than this distance (in bp) get merged together, and the gap is filled. Default: 50
- **pre-mrna-fraction** (float, optional) – Some RNA-Seq protocols produce a significant amount of reads that originate from incompletely spliced transcripts, and these reads can confound the assembly of fully spliced mRNAs. Cufflinks uses this parameter to filter out alignments that lie within the intronic intervals implied by the spliced alignments. The minimum depth of coverage in the intronic region covered by the alignment is divided by the number of spliced reads, and if the result is lower than this parameter value, the intronic alignments are ignored. The default is 15%. Range: 0.0-1.0. Default: 0.15
- **seed** (int, optional) – Value of random number generator seed. Default: 0
- **small-anchor-fraction** (float, optional) – Spliced reads with less than this percent of their length on each side of the junction are considered suspicious and are candidates for filtering prior to assembly. Default: 0.09
- **total-hits-norm** (bool, optional) – With this option, Cufflinks counts all fragments, including those not compatible with any reference transcript, towards the number of mapped hits used in the FPKM denominator. Default: TRUE

- **trim-3-avgcov-thresh** (int, optional) – Minimum average coverage required to attempt 3 prime trimming. Default: 10
- **trim-3-dropoff-frac** (float, optional) – The fraction of average coverage below which to trim the 3 prime end of an assembled transcript. Default: 0.1
- **upper-quartile-norm** (bool, optional) – DEPRECATED! Use –library-norm-method With this option, Cufflinks normalizes by the upper quartile of the number of fragments mapping to individual loci instead of the total number of sequenced fragments. This can improve robustness of differential expression calls for less abundant genes and transcripts.
- **verbose** (bool, optional) – Print lots of status updates and other diagnostic information. Default: FALSE

**Required tools:** cufflinks, mkdir, mv

**CPU Cores:** 6

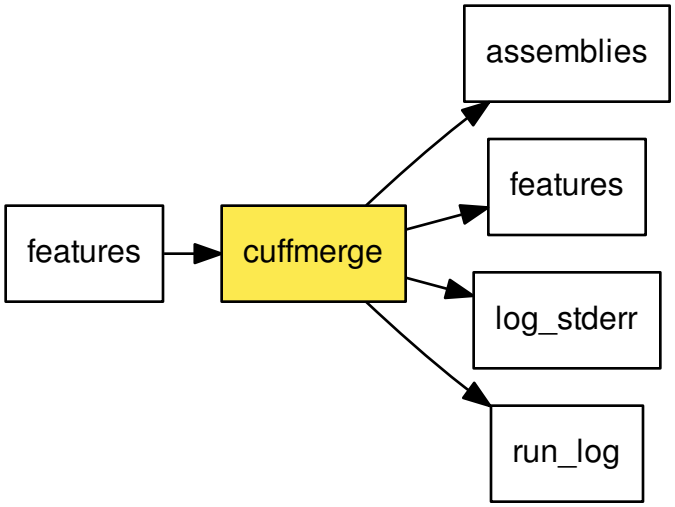
### cuffmerge

CuffMerge is part of the 'Cufflinks suite of tools' for differential expr. analysis of RNA-Seq data and their visualisation. This step applies the cuffmerge tool which merges several Cufflinks assemblies. For details on cuffmerge we refer to the author's webpage:

http://cole-trapnell-lab.github.io/cufflinks/cuffmerge/

**Connections:**

- Input Connection:
  - 'in/features'
- Output Connection:
  - 'out/assemblies'
  - 'out/features'
  - 'out/log_stderr'
  - 'out/run_log'




**Options:**

- **num-threads** (int, optional) – Use this many threads to merge assemblies.
  - default value: 6
- **ref-gtf** (str, optional) – A "reference" annotation GTF. The input assemblies are merged together with the reference GTF and included in the final output.
- **ref-sequence** (str, optional) – This argument should point to the genomic DNA sequences for the reference. If a directory, it should contain one fasta file per contig. If a multifasta file, all contigs should be present.
- **run_id** (str, optional) – An arbitrary name of the new run (which is a merge of all samples).
  - default value: magic

**Required tools:** cuffmerge, mkdir, mv, printf

**CPU Cores:** 6

### cutadapt

Cutadapt finds and removes adapter sequences, primers, poly-A tails and other types of unwanted sequence from your high-throughput sequencing reads.

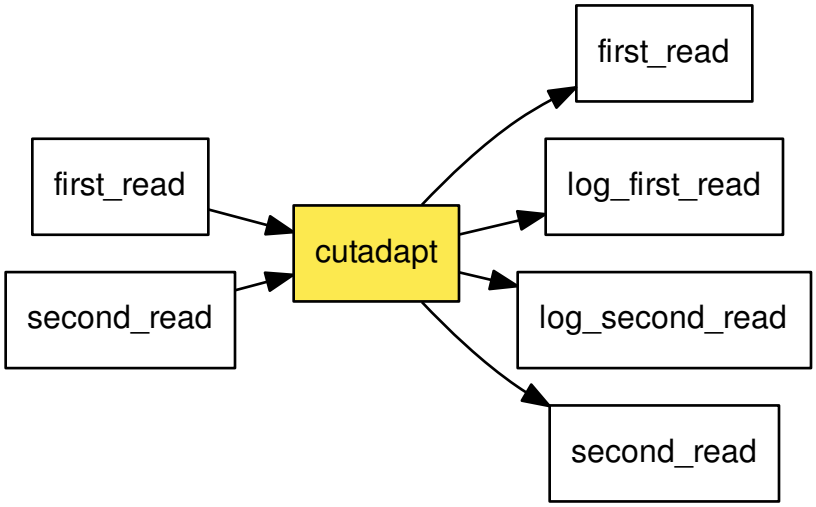
https://cutadapt.readthedocs.org/en/stable/

This step wraps release: cutadpat 1.5

**Connections:**

- Input Connection:
  - 'in/first_read'
  - 'in/second_read'
- Output Connection:

  - 'out/first_read'
  - 'out/log_first_read'
  - 'out/log_second_read'
  - 'out/second_read'




**Options:**

- **adapter-R1** (str, optional) – Adapter sequence to be clipped off of thefirst read.
- **adapter-R2** (str, optional) – Adapter sequence to be clipped off of thesecond read
- **adapter-file** (str, optional) – File containing adapter sequences to be clipped off of the reads.
- **adapter-type** (str, optional) – The type of the adapter that has been used for sequencing. a: adapter ligated to the 3' end; b: adapter ligated to the 3' or 5' end (If the adapter is found within the read or overlapping the 3' end of the read, the behavior is the same as for the -a value. If the adapter overlaps the 5' end (beginning of the read), the initial portion of the read matching the adapter is trimmed, but anything that follows is kept.); g: adapter ligated to the 5' end (If the adapter sequence starts with the character '^', the adapter is 'anchored'. An anchored adapter must appear in its entirety at the 5' end of the read (it is a prefix of the read). A non-anchored adapter may appear partially at the 5' end, or it may occur within the read. If it is found within a read, the sequence preceding the adapter is also trimmed. In all cases, the adapter itself is trimmed).
  - default value: -a
  - possible values: '-a', '-g', '-b'
- **bwa** (bool, optional) – BWA-compatible color space output. This enables colorspace, double-encode, trim-primer, strip-f3 and suffix:'/1'.
- **colospace** (bool, optional) – Colorspace mode: Also trim the color that is adjacent to the found adapter.
- **cut** (int, optional) – Remove bases from the beginning or end of each read. If LENGTH is positive, the bases are removed from the beginning of each read. If LENGTH is negative, the bases are removed from

the end of each read.

- **dd-blocksize** (str, optional) - default value: 2M
- **discard-trimmed** (bool, optional) – Discard reads that contain the adapter instead of trimming them. Also use -O in order to avoid throwing away too many randomly matching reads!
- **discard-untrimmed** (bool, optional) – Discard reads that do not contain the adapter.
- **double-encode** (bool, optional) – When in color space, double-encode colors (map 0,1,2,3,4 to A,C,G,T,N).
- **error-rate** (float, optional) – Maximum allowed error rate (no. of errors divided by the length of the matching region) (default: 0.1)
- **fix_qnames** (bool, required) – If set to true, only the leftmost string without spaces of the QNAME field of the FASTQ data is kept. This might be necessary for downstream analysis.
- **length-tag** (str, optional) – Search for TAG followed by a decimal number in the name of the read (description/comment field of the FASTA or FASTQ file). Replace the decimal number with the correct length of the trimmed read. For example, use –length-tag 'length=' to correct fields like 'length=123'.
- **maq** (bool, optional) – MAQ-compatible color space output. This enables colorspace, double-encode, trim-primer, strip-f3 and suffix:'/1'.
- **mask-adapter** (bool, optional) – Mask with 'N' adapter bases instead of trim (default: False)
- **match-read-wildcards** (bool, optional) – Allow 'N's in the read as matches to the adapter (default: False).
- **maximum-length** (int, optional) – Discard trimmed reads that are longer than LENGTH. Reads that are too long even before adapter removal are also discarded. In colorspace, an initial primer is not counted (default: no limit).
- **minimum-length** (int, optional) – Discard trimmed reads that are shorter than LENGTH. Reads that are too short even before adapter removal are also discarded. In colorspace, an initial primer is not counted (default: 0).
- **no-indels** (bool, optional) – Do not allow indels in the alignments, that is, allow only mismatches. This option is currently only supported for anchored 5' adapters (adapter-type: "-g" and adapter-R[1|2]: "^ADAPTER") (default: both mismatches and indels are allowed)
- **no-trim** (bool, optional) – Match and redirect reads to output/untrimmed-output as usual, but don't remove the adapters. (Default: False)
- **no-zero-cap** (bool, optional) – Do not change negative quality values to zero. Colorspace quality values of -1 would appear as spaces in the output FASTQ file. Since many tools have problems with that, negative qualities are converted to zero when trimming colorspace data. Use this option to keep negative qualities.
- **overlap** (int, optional) – Minimum overlap length. If the overlap between the read and the adapter is shorter than LENGTH, the read is not modified. This reduces the no. of bases trimmed purely due to short random adapter matches (default: 3).
- **pigz-blocksize** (str, optional) - default value: 2048
- **prefix** (str, optional) – Add this prefix to read names
- **quality-base** (int, optional) – Assume that quality values are encoded as ascii (quality + QUALITY_BASE). The default (33) is usually correct, except for reads produced by some versions of the Illumina pipeline, where this should be set to 64. (Default: 33)
  - possible values: '33', '64'

- **quality-cutoff** (int, optional) – Trim low-quality ends from reads before adapter removal. The algorithm is the same as the one used by BWA (Subtract CUTOFF from all qualities; compute partial sums from all indices to the end of the sequence; cut sequence at the index at which the sum is minimal) (default: 0)
- **strip-f3** (bool, optional) – For color space: Strip the _F3 suffix of read names
- **strip-suffix** (str, optional) – Remove this suffix from read names if present. Can be given multiple times.
- **suffix** (str, optional) – Add this suffix to read names
- **times** (int, optional) – Try to remove adapters at most COUNT times. Useful when an adapter gets appended multiple times (default: 1).
- **trim-primer** (bool, optional) – When in color space, trim primer base and the first color (which is the transition to the first nucleotide)
- **use_reverse_complement** (bool, required) – The reverse complement of adapter sequences 'adapter-R1' and 'adapter-R2' are used for adapter clipping.
- **zero-cap** (bool, optional) – Change negative quality values to zero. This is enabled by default when -c/–colorspace is also enabled. Use the above option to disable it.

**Required tools:** cat, cutadapt, dd, fix_qnames, mkfifo, pigz

**CPU Cores:** 4

### deepTools_bamCompare

This tool compares two BAM files based on the number of mapped reads. To compare the BAM files, the genome is partitioned into bins of equal size, then the number of reads found in each bin is counted per file, and finally a summary value is reported. This value can be the ratio of the number of reads per bin, the log2 of the ratio, or the difference. This tool can normalize the number of reads in each BAM file using the SES method proposed by Diaz et al. (2012) "Normalization, bias correction, and peak calling for ChIP-seq". Statistical Applications in Genetics and Molecular Biology, 11(3). Normalization based on read counts is also available. The output is either a bedgraph or bigWig file containing the bin location and the resulting comparison value. By default, if reads are paired, the fragment length reported in the BAM file is used. Each mate, however, is treated independently to avoid a bias when a mixture of concordant and discordant pairs is present. This means that each end will be extended to match the fragment length.

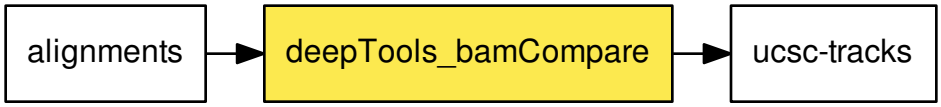
http://deeptools.readthedocs.io/en/latest/content/tools/bamCompare.html

Usage example:

```
bamCompare -b1 treatment.bam -b2 control.bam -o log2ratio.bw
```

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/ucsc-tracks'




**Options:**

- **binSize** (int, optional) – Size of the bins, in bases, for the output of the bigwig/bedgraph file. (default: 50)
- **blackListFileName** (str, optional) – A BED or GTF file containing regions that should be excluded from all analyses. Currently this works by rejecting genomic chunks that happen to overlap an entry. Consequently, for BAM files, if a read partially overlaps a blacklisted region or a fragment spans over it, then the read/fragment might still be considered. Please note that you should adjust the effective genome size, if relevant. (default: None)
- **centerReads** (bool, optional) – By adding this option, reads are centered with respect to the fragment length. For paired-end data, the read is centered at the fragment length defined by the two ends of the fragment. For single-end data, the given fragment length is used. This option is useful to get a sharper signal around enriched regions. (default: False)
- **extendReads** (int, optional) – This parameter allows the extension of reads to fragment size. If set, each read is extended, without exception. *NOTE*: This feature is generally NOT recommended for spliced-read data, such as RNA-seq, as it would extend reads over skipped regions. *Single-end*: Requires a user specified value for the final fragment length. Reads that already exceed this fragment length will not be extended. *Paired-end*: Reads with mates are always extended to match the fragment size defined by the two read mates. Unmated reads, mate reads that map too far apart (>4x fragment length) or even map to different chromosomes are treated like single-end reads. The input of a fragment length value is optional. If no value is specified, it is estimated from the data (mean of the fragment size of all mate reads). (default: False)
- **ignoreDuplicates** (bool, optional) – If set, reads that have the same orientation and start position will be considered only once. If reads are paired, the mate's position also has to coincide to ignore a read. (default: False)
- **ignoreForNormalization** (list, optional) – A list of space-delimited chromosome names containing those chromosomes that should be excluded for computing the normalization. This is useful when considering samples with unequal coverage across chromosomes, like male samples. An usage examples is –ignoreForNormalization chrX chrM. (default: None)
- **maxFragmentLength** (int, optional) – The maximum fragment length needed for read/pair inclusion. A value of 0 disables filtering and is needed for including single-end and orphan reads. (default: 0)
- **minFragmentLength** (int, optional) – The minimum fragment length needed for read/pair inclusion. Note that a value other than 0 will exclude all single-end reads. This option is primarily useful in ATACseq experiments, for filtering mono- or di-nucleosome fragments. (default: 0)
- **minMappingQuality** (int, optional) – If set, only reads that have a mapping quality score of at least this are considered. (default: None)
- **normalizeTo1x** (int, optional) – Report read coverage normalized to 1x sequencing depth (also known as Reads Per Genomic Content (RPGC)). Sequencing depth is defined as: (total number of mapped reads * fragment length) / effective genome size. The scaling factor used is the inverse of the sequencing depth computed for the sample to match the 1x coverage. To use this option, the effective genome

size has to be indicated after the option. The effective genome size is the portion of the genome that is mappable. Large fractions of the genome are stretches of NNNN that should be discarded. Also, if repetitive regions were not included in the mapping of reads, the effective genome size needs to be adjusted accordingly. Common values are: mm9:2,150,570,000; hg19:2,451,960,000; dm3:121,400,000 and ce10:93,260,000. See Table 2 of http://www.plosone.org/article/info:doi/10.1371/journal.pone.0030377 or http://www.nature.com/nbt/journal/v27/n1/fig_tab/nbt.1518_T1.html for several effective genome sizes. (default: None)

- **normalizeUsingRPKM** (bool, optional) – Use Reads Per Kilobase per Million reads to normalize the number of reads per bin. The formula is: RPKM (per bin) = number of reads per bin / ( number of mapped reads (in millions) * bin length (kb) ). Each read is considered independently,if you want to only count either of the mate pairs in paired-end data, use the –samFlag option. (default: False)
- **numberOfSamples** (int, optional) – *Only relevant when SES is chosen for the scaleFactorsMethod.* Number of samplings taken from the genome to compute the scaling factors. (default: 100000.0)
- **outFileFormat** (str, required) – Output file type. Either "bigwig" or "bedgraph". (default: "bigwig")
  - possible values: 'bigwig', 'bedgraph'
- **pseudocount** (float, optional) – small number to avoid x/0. Only useful together with –ratio log2 or –ratio ratio . (default: 1)
- **ratio** (str, optional) – The default is to output the log2ratio of the two samples. The reciprocal ratio returns the negative of the inverse of the ratio if the ratio is less than 0. The resulting values are interpreted as negative fold changes. *NOTE*: Only with –ratio subtract can –normalizeTo1x or –normalizeUsingRPKM be used. Instead of performing a computation using both files, the scaled signal can alternatively be output for the first or second file using the '–ratio first' or '–ratio second' (default: log2)
  - possible values: 'log2', 'ratio', 'subtract', 'add', 'mean', 'reciprocal_ratio', 'first,second'
- **region** (str, optional) – Region of the genome to limit the operation to - this is useful when testing parameters to reduce the computing time. The format is chr:start:end, for example –region chr10 or –region chr10:456700:891000. (default: None)
- **samFlagExclude** (int, optional) – Exclude reads based on the SAM flag. For example, to get only reads that map to the forward strand, use –samFlagExclude 16, where 16 is the SAM flag for reads that map to the reverse strand. (default: None)
- **samFlagInclude** (int, optional) – Include reads based on the SAM flag. For example, to get only reads that are the first mate, use a flag of 64. This is useful to count properly paired reads only once, as otherwise the second mate will be also considered for the coverage. (default: None)
- **sampleLength** (int, optional) – *Only relevant when SES is chosen for the scaleFactorsMethod.* To compute the SES, specify the length (in bases) of the regions (see –numberOfSamples) that will be randomly sampled to calculate the scaling factors. If you do not have a good sequencing depth for your samples consider increasing the sampling regions' size to minimize the probability that zero-coverage regions are used. (default: 1000)
- **samples** (list, required) – List of lists with two elements. Each element has to be the name of a run. Each run has to provide a SINGLE BAM file. Both BAM files are compared using deepTools bamCompare command.
- **scaleFactors** (str, optional) – Set this parameter manually to avoid the computation of scaleFactors. The format is scaleFactor1:scaleFactor2. For example, –scaleFactor 0.7:1 will cause the first BAM file tobe multiplied by 0.7, while not scaling the second BAM file (multiplication with 1). (default: None)
- **scaleFactorsMethod** (str, optional) – Method to use to scale the samples. (default: readCount)
  - possible values: 'readCount', 'SES'

- **skipNonCoveredRegions** (bool, optional) – This parameter determines if non-covered regions (regions without overlapping reads) in a BAM file should be skipped. The default is to treat those regions as having a value of zero. The decision to skip non-covered regions depends on the interpretation of the data. Non-covered regions may represent, for example, repetitive regions that should be skipped. (default: False)
- **smoothLength** (int, optional) – The smooth length defines a window, larger than the binSize, to average the number of reads. For example, if the –binSize is set to 20 and the –smoothLength is set to 60, then, for each bin, the average of the bin and its left and right neighbors is considered. Any value smaller than –binSize will be ignored and no smoothing will be applied. (default: None)

**Required tools:** bamCompare

**CPU Cores:** 10

### deepTools_bamPEFragmentSize

bamPEFragmentSize tool calculates the fragment sizes for read pairs given a BAM file from paired-end sequencing.Several regions are sampled depending on the size of the genome and number of processors to estimate thesummary statistics on the fragment lengths. Properly paired reads are preferred for computation, i.e., it will only use discordant pairs if no concordant alignments overlap with a given region. The default setting simply prints the summary statistics to the screen.

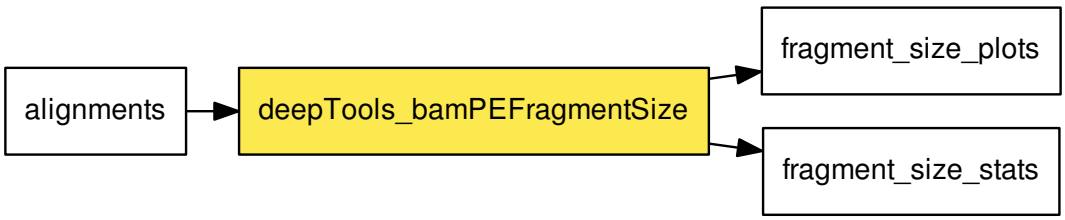
http://deeptools.readthedocs.io/en/latest/content/tools/bamPEFragmentSize.html

Usage example:

```
bamPEFragmentSize [-h] [--bamfiles bam files [bam files ...]]
                  [--histogram FILE] [--numberOfProcessors INT]
                  [--samplesLabel SAMPLESLABEL [SAMPLESLABEL ...]]
                  [--plotTitle PLOTTITLE]
                  [--maxFragmentLength MAXFRAGMENTLENGTH] [--logScale]
                  [--binSize INT] [--distanceBetweenBins INT]
                  [--blackListFileName BED file] [--verbose] [--version]
```

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/fragment_size_plots'
  - 'out/fragment_size_stats'



**Options:**

- **binSize** (int, optional) – Length in bases of the window used to sample the genome. (default 1000)
- **blackListFileName** (str, optional) – A BED file containing regions that should be excluded from all analyses. Currently this works by rejecting genomic chunks that happen to overlap an entry. Consequently, for BAM files, if a read partially overlaps a blacklisted region or a fragment spans over it, then the read/fragment might still be considered. Please note that you should adjust the effective genome size, if relevant. (default: None)
- **distanceBetweenBins** (int, optional) – To reduce the computation time, not every possible genomic bin is sampled. This option allows you to set the distance between bins actually sampled from. Larger numbers are sufficient for high coverage samples, while smaller values are useful for lower coverage samples. Note that if you specify a value that results in too few (<1000) reads sampled, the value will be decreased. (default 1000000)
- **histogram** (bool, optional) – If set saves a .png file with a histogram of fragment length distribution for each run.
- **logScale** (bool, optional) – Plot on the log scale
- **maxFragmentLength** (int, optional) – The maximum fragment length in the histogram. A value of 0 (the default) indicates to use twice the mean fragment length
- **samples** (dict, optional) – Dictionary with IDs of new runs as keys and lists of sample names as values. For each sample name a BAM file is expected to be the input from upstream steps. If not provided this step calculates summary statistics for each input file.

**Required tools:** bamPEFragmentSize

**CPU Cores:** 10

### deepTools_multiBamSummary

This step computes the read coverages for genomic regions for every BAM input file using multiBamSummary. For downstream analysis such as 'plotCorrelation' or 'plotPCA' you need to merge the output files.

The analysis can be performed for the entire genome by running the program in 'bins' mode. If you want to count the read coverage for specific regions only, use the BED-file mode instead. The standard output of multiBamSummary is a compressed numpy array (.npz). It can be directly used to calculate and visualize pairwise correlation values between the read coverages using the tool 'plotCorrelation'. Similarly, plotPCA can be used for principal component analysis of the read coverages using the .npz file. Note that using a single bigWig file is only recommended if you want to produce a bedGraph file (i.e., with the –outRawCounts option; the default output file cannot be used by ANY deepTools program if only a single file was supplied!).

http://deeptools.readthedocs.io/en/latest/content/tools/multiBamSummary.html

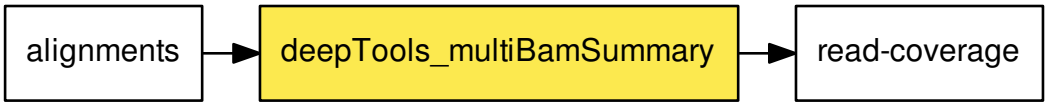
Usage example:

```
multiBamSummary [-h] [--version]  ...
```

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:

- 'out/read-coverage'




**Options:**

- **bed-file** (list, optional) – BED file that contains all regions that should be considered for the coverage analysis. If this option is set "multiBamSummary" is executed with "BED-file" subcommand, otherwise with "bins" subcommand.
- **binSize** (int, optional) – Length in bases of the window used to sample the genome. (default: 10000)
- **blackListFileName** (str, optional) – A BED or GTF file containing regions that should be excluded from all analyses. Currently this works by rejecting genomic chunks that happen to overlap an entry. Consequently, for BAM files, if a read partially overlaps a blacklisted region or a fragment spans over it, then the read/fragment might still be considered. Please note that you should adjust the effective genome size, if relevant. (default: None)
- **centerReads** (bool, optional) – By adding this option, reads are centered with respect to the fragment length. For paired-end data, the read is centered at the fragment length defined by the two ends of the fragment. For single-end data, the given fragment length is used. This option is useful to get a sharper signal around enriched regions. (default: False)
- **distanceBetweenBins** (int, optional) – By default, multiBamSummary considers consecutive bins of the specified –binSize. However, to reduce the computation time, a larger distance between bins can by given. Larger distances result in fewer bins considered. (default: 0)
- **exonID** (str, optional) – When a GTF file is used to provide regions, only entries with this value as their feature (column 2) will be processed as exons. CDS would be another common value for this. (default: exon)
- **extendReads** (bool, optional) – This parameter allows the extension of reads to fragment size. If set, each read is extended, without exception. *NOTE*: This feature is generally NOT recommended for spliced-read data, such as RNA-seq, as it would extend reads over skipped regions. *Single-end*: Requires a user specified value for the final fragment length. Reads that already exceed this fragment length will not be extended. *Paired-end*: Reads with mates are always extended to match the fragment size defined by the two read mates. Unmated reads, mate reads that map too far apart (>4x fragment length) or even map to different chromosomes are treated like single-end reads. The input of a fragment length value is optional. If no value is specified, it is estimated from the data (mean of the fragment size of all mate reads). (default: False)
- **ignoreDuplicates** (bool, optional) – If set, reads that have the same orientation and start position will be considered only once. If reads are paired, the mate's position also has to coincide to ignore a read. (default: False)
- **maxFragmentLength** (int, optional) – The maximum fragment length needed for read/pair inclusion. A value of 0 disables filtering and is needed for including single-end and orphan reads. (default: 0)
- **metagene** (bool, optional) – When either a BED12 or GTF file are used to provide regions, perform the computation on the merged exons, rather than using the genomic interval defined by the 5-prime and 3-

prime most transcript bound (i.e., columns 2 and 3 of a BED file). If a BED3 or BED6 file is used as input, then columns 2 and 3 are used as an exon. (default: False)

- **minFragmentLength** (int, optional) – The minimum fragment length needed for read/pair inclusion. Note that a value other than 0 will exclude all single-end reads. This option is primarily useful in ATACseq experiments, for filtering mono- or di-nucleosome fragments. (default: 0)
- **minMappingQuality** (int, optional) – If set, only reads that have a mapping quality score of at least this are considered. (default: None)
- **outRawCounts** (bool, optional) – Save the counts per region to a tab-delimited file. (default: False)
- **region** (str, optional) – Region of the genome to limit the operation to - this is useful when testing parameters to reduce the computing time. The format is chr:start:end, for example –region chr10 or –region chr10:456700:891000. (default: None)
- **samFlagExclude** (int, optional) – Exclude reads based on the SAM flag. For example, to get only reads that map to the forward strand, use –samFlagExclude 16, where 16 is the SAM flag for reads that map to the reverse strand. (default: None)
- **samFlagInclude** (int, optional) – Include reads based on the SAM flag. For example, to get only reads that are the first mate, use a flag of 64. This is useful to count properly paired reads only once, as otherwise the second mate will be also considered for the coverage. (default: None)
- **transcriptID** (str, optional) – When a GTF file is used to provide regions, only entries with this value as their feature (column 2) will be processed as transcripts. (default: transcript)
- **transcript_id_designator** (str, optional) – Each region has an ID (e.g., ACTB) assigned to it, which for BED files is either column 4 (if it exists) or the interval bounds. For GTF files this is instead stored in the last column as a key:value pair (e.g., as 'transcript_id "ACTB"', for a key of transcript_id and a value of ACTB). In some cases it can be convenient to use a different identifier. To do so, set this to the desired key. (default: transcript_id)

**Required tools:** multiBamSummary

**CPU Cores:** 10

### deepTools_plotFingerprint

This tool samples indexed BAM files and plots a profile of cumulative read coverages for each. All reads overlapping a window (bin) of the specified length are counted; these counts are sorted and the cumulative sum is finally plotted.

Usage example:

```
plotFingerprint -b treatment.bam control.bam -plot fingerprint.png
```

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/counts'
  - 'out/plots'


**Options:**

- **JSDsample** (str, optional) – Reference sample against which to compute the Jensen-Shannon distance and the CHANCE statistics. If this is not specified, then these will not be calculated. If –outQualityMetrics is not specified then this will be ignored. The Jensen-Shannon implementation is based on code from Sitanshu Gakkhar at BCGSC. The CHANCE implementation is based on code from Matthias Haimel. (default: None)
- **binSize** (int, optional) – Window size in base pairs to sample the genome. (default: 500)
- **blackListFileName** (str, optional) – A BED or GTF file containing regions that should be excluded from all analyses. Currently this works by rejecting genomic chunks that happen to overlap an entry. Consequently, for BAM files, if a read partially overlaps a blacklisted region or a fragment spans over it, then the read/fragment might still be considered. Please note that you should adjust the effective genome size, if relevant. (default: None)
- **centerReads** (bool, optional) – By adding this option, reads are centered with respect to the fragment length. For paired-end data, the read is centered at the fragment length defined by the two ends of the fragment. For single-end data, the given fragment length is used. This option is useful to get a sharper signal around enriched regions. (default: False)
- **extendReads** (int, optional) – This parameter allows the extension of reads to fragment size. If set, each read is extended, without exception. *NOTE*: This feature is generally NOT recommended for spliced-read data, such as RNA-seq, as it would extend reads over skipped regions. *Single-end*: Requires a user specified value for the final fragment length. Reads that already exceed this fragment length will not be extended. *Paired-end*: Reads with mates are always extended to match the fragment size defined by the two read mates. Unmated reads, mate reads that map too far apart (>4x fragment length) or even map to different chromosomes are treated like single-end reads. The input of a fragment length value is optional. If no value is specified, it is estimated from the data (mean of the fragment size of all mate reads). (default: False)
- **ignoreDuplicates** (bool, optional) – If set, reads that have the same orientation and start position will be considered only once. If reads are paired, the mate's position also has to coincide to ignore a read. (default: False)
- **maxFragmentLength** (int, optional) – The maximum fragment length needed for read/pair inclusion. A value of 0 disables filtering and is needed for including single-end and orphan reads. (default: 0)
- **minFragmentLength** (int, optional) – The minimum fragment length needed for read/pair inclusion. Note that a value other than 0 will exclude all single-end reads. This option is primarily useful in ATACseq experiments, for filtering mono- or di-nucleosome fragments. (default: 0)
- **minMappingQuality** (int, optional) – If set, only reads that have a mapping quality score of at least this are considered. (default: None)

- **numberOfSamples** (int, optional) – Number of bins that sampled from the genome, for which the overlapping number of reads is computed. (default: 500000.0)
- **outQualityMetrics** (str, optional) – Quality metrics can optionally be output to this file. The file will have one row per input BAM file and columns containing a number of metrics. Please see the online documentation for a longer explanation: http://deeptools.readthedocs.io/en/latest/content/feature/plotFingerprint_QC_metrics.html. (default: None)
- **plotFileFormat** (str, required) – File ending of the output figure. It will be used to determine the image format.
  - possible values: 'png', 'eps', 'pdf', 'svg'
- **region** (str, optional) – Region of the genome to limit the operation to - this is useful when testing parameters to reduce the computing time. The format is chr:start:end, for example –region chr10 or –region chr10:456700:891000. (default: None)
- **samFlagExclude** (int, optional) – Exclude reads based on the SAM flag. For example, to get only reads that map to the forward strand, use –samFlagExclude 16, where 16 is the SAM flag for reads that map to the reverse strand. (default: None)
- **samFlagInclude** (int, optional) – Include reads based on the SAM flag. For example, to get only reads that are the first mate, use a flag of 64. This is useful to count properly paired reads only once, as otherwise the second mate will be also considered for the coverage. (default: None)
- **samples** (list, required) – List of lists with run names. Each element has to be the name of a run. Each run has to provide a SINGLE BAM file. All BAM files are plotted and counted using deepTools plotFingerprint command.
- **skipZeros** (bool, optional) – If set, then regions with zero overlapping reads for *all* given BAM files are ignored. This will result in a reduced number of read counts than that specified in –numberOfSamples (default: False)

**Required tools:** plotFingerprint

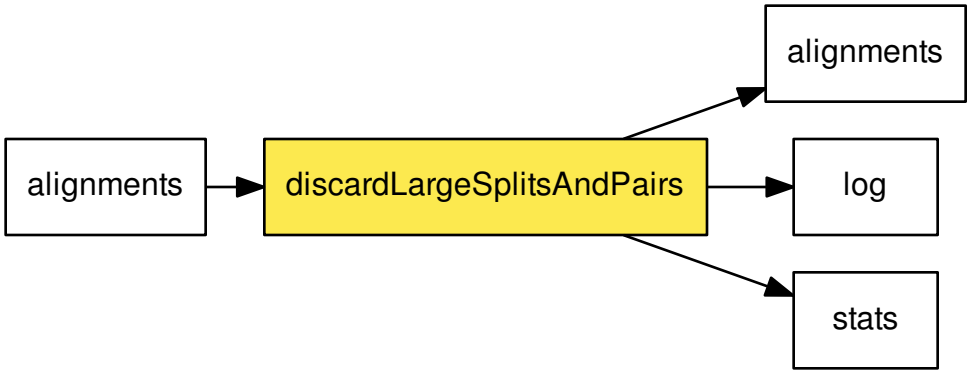
**CPU Cores:** 10

### discardLargeSplitsAndPairs

discardLargeSplitsAndPairs reads SAM formatted alignments of the mapped reads. It discards all split reads that skip more than splits_N nucleotides in their alignment to the ref genome. In addition, all read pairs that are mapped to distant region such that the final template will exceed N_mates nucleotides will also be discarded. All remaining reads are returned in SAM format. The discarded reads are also collected in a SAM formatted file and a statistic is returned.

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/alignments'
  - 'out/log'
  - 'out/stats'




**Options:**

- **M_mates** (str, required) – Size of template (in nucleotides) that would arise from a read pair. Read pairs that exceed this value are discarded.
- **N_splits** (str, required) – Size of the skipped region within a split read (in nucleotides). Split Reads that skip more nt than this value are discarded.

**Required tools:** dd, discardLargeSplitsAndPairs, pigz, samtools

**CPU Cores:** 4

### fastqc

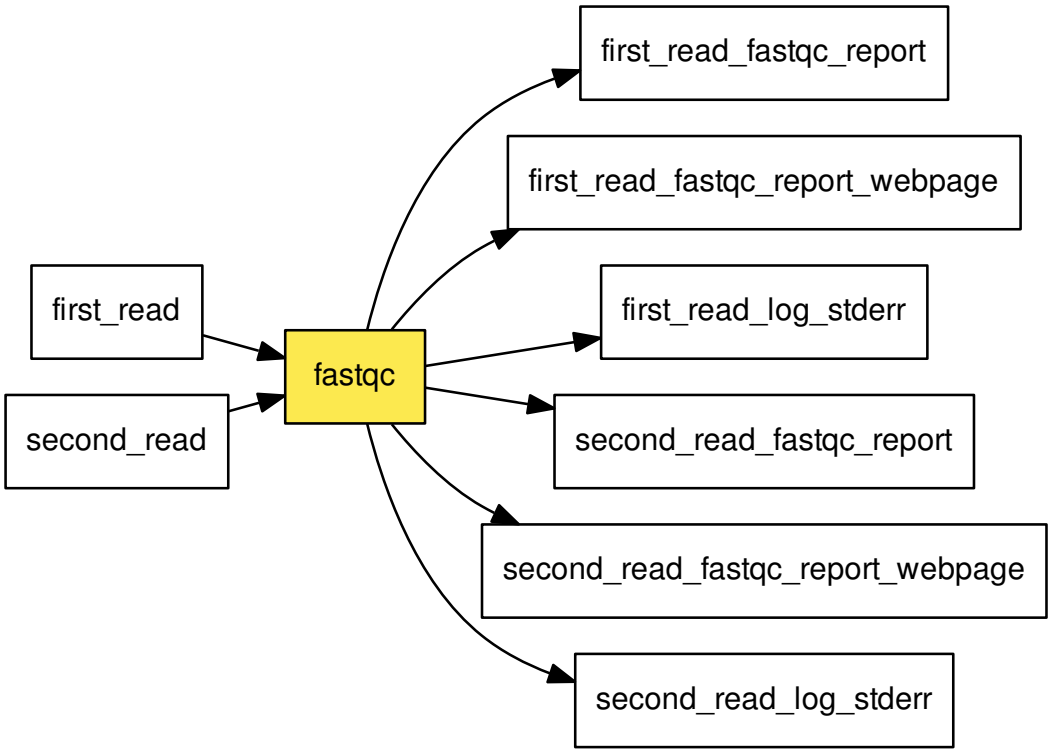
The fastqc step is a wrapper for the fastqc tool. It generates some quality metrics for fastq files. For this specific instance only the zip archive is preserved.

http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

Tested fastqc release: 0.11.2

**Connections:**

- Input Connection:
  - 'in/first_read'
  - 'in/second_read'
- Output Connection:
  - 'out/first_read_fastqc_report'
  - 'out/first_read_fastqc_report_webpage'
  - 'out/first_read_log_stderr'
  - 'out/second_read_fastqc_report'
  - 'out/second_read_fastqc_report_webpage'
  - 'out/second_read_log_stderr'




**Options:**

- **adapters** (str, optional) – Specifies a non-default file which contains the list of adapter sequences which will be explicity searched against the library. The file must contain sets of named adapters in the form name[tab]sequence. Lines prefixed with a hash will be ignored.
- **casava** (bool, optional) – Files come from raw casava output. Files in the same sample group (differing only by the group number) will be analysed as a set rather than individually. Sequences with the filter flag set in the header will be excluded from the analysis. Files must have the same names given to them by casava (including being gzipped and ending with .gz) otherwise they won't be grouped together correctly.
- **contaminants** (str, optional) – Specifies a non-default file which contains the list of contaminants to screen overrepresented sequences against. The file must contain sets of named contaminants in the form name[tab]sequence. Lines prefixed with a hash will be ignored.
- **dd-blocksize** (str, optional) - default value: 2M
- **dir** (str, optional) – Selects a directory to be used for temporary files written when generating report images. Defaults to system temp directory if not specified.
- **format** (str, optional) – Bypasses the normal sequence file format detection and forces the program to use the specified format. Valid formats are bam,sam, bam_mapped,sam_mapped and fastq
  - possible values: 'bam', 'sam', 'bam_mapped', 'sam_mapped', 'fastq'
- **java** (str, optional) – Provides the full path to the java binary you want to use to launch fastqc. If not supplied then java is assumed to be in your path.

- **kmers** (int, optional) – Specifies the length of Kmer to look for in the Kmer content module. Specified Kmer length must be between 2 and 10. Default length is 7 if not specified.
- **limits** (str, optional) – Specifies a non-default file which contains a set of criteria which will be used to determine the warn/error limits for the various modules. This file can also be used to selectively remove some modules from the output all together. The format needs to mirror the default limits.txt file found in the Configuration folder.
- **nofilter** (bool, optional) – If running with –casava then do not remove read flagged by casava as poor quality when performing the QC analysis.
- **nogroup** (bool, optional) – Disable grouping of bases for reads >50bp. All reports will show data for every base in the read. WARNING: Using this option will cause fastqc to crash and burn if you use it on really long reads, and your plots may end up a ridiculous size. You have been warned!
- **pigz-blocksize** (str, optional) - default value: 2048
- **threads** (int, optional) – Specifies the number of files which can be processed simultaneously. Each thread will be allocated 250MB of memory so you should not run more threads than your available memory will cope with, and not more than 6 threads on a 32 bit machine

**Required tools:** fastqc, mkdir, mv

**CPU Cores:** 4

### fastx_quality_stats

fastx_quality_stats generates a text file containing quality information of the input fastq data.

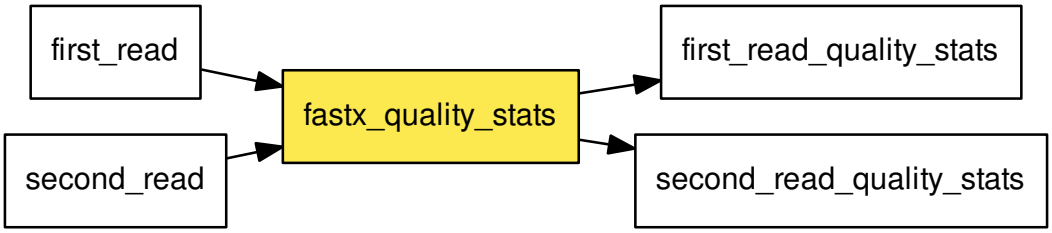
Documentation:

```
http://hannonlab.cshl.edu/fastx_toolkit/
```

Tested fastqc release: 0.0.13

**Connections:**

- Input Connection:
  - 'in/first_read'
  - 'in/second_read'
- Output Connection:
  - 'out/first_read_quality_stats'
  - 'out/second_read_quality_stats'




**Options:**

- **dd-blocksize** (str, optional) - default value: 2M
- **new_output_format** (bool, optional) – New output format (with more information per nucleotide/cycle).
- **pigz-blocksize** (str, optional) - default value: 2048
- **quality** (int, optional) - default value: 33

**Required tools:** cat, dd, fastx_quality_stats, mkfifo, pigz

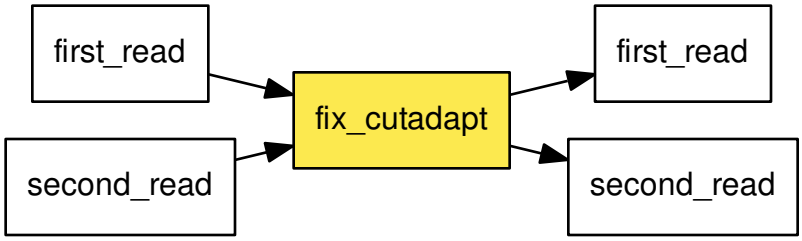
**CPU Cores:** 4

### fix_cutadapt

This step takes FASTQ data and removes both reads of a paired-end read, if one of them has been completely removed by cutadapt (or any other software).

**Connections:**

- Input Connection:
  - 'in/first_read'
  - 'in/second_read'
- Output Connection:
  - 'out/first_read'
  - 'out/second_read'




**Options:**

- **dd-blocksize** (str, optional) - default value: 2M
- **pigz-blocksize** (str, optional) - default value: 2048

**Required tools:** cat, dd, fix_cutadapt, mkfifo, pigz

**CPU Cores:** 4

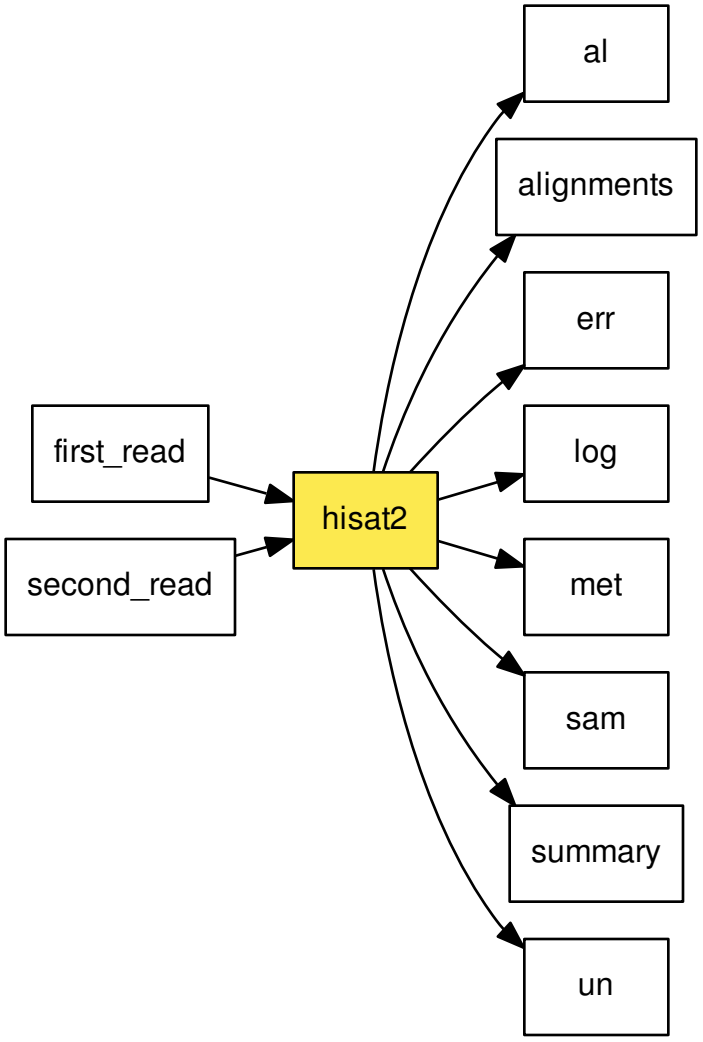
### hisat2

HISAT2 is a fast and sensitive alignment program for mapping next-generation sequencing reads (both DNA and RNA) to a population of human genomes (as well as to a single reference genome).

https://ccb.jhu.edu/software/hisat2/index.shtml

**Connections:**

- Input Connection:
  - 'in/first_read'
  - 'in/second_read'
- Output Connection:
  - 'out/al'
  - 'out/alignments'
  - 'out/err'
  - 'out/log'
  - 'out/met'
  - 'out/sam'
  - 'out/summary'
  - 'out/un'




**Options:**

- **add-chrname** (bool, optional) – add 'chr' to reference names in alignment
- **avoid-pseudogene** (bool, optional) – tries to avoid aligning reads to pseudogenes (experimentaloption).
- **c** (bool, optional) – <m1>, <m2>, <r> are sequences themselves, not files
- **dd-blocksize** (str, optional) - default value: 2M
- **dta** (bool, optional) – reports alignments tailored for transcript assemblers
- **dta-cufflinks** (bool, optional) – reports alignments tailored specifically for cufflinks
- **f** (bool, optional) – query input files are (multi-)FASTA .fa/.mfa
- **ht2-idx** (str, required) – Index filename prefix (minus trailing .X.ht2).
- **ignore-quals** (bool, optional) – treat all quality values as 30 on Phred scale (off)

- **int-quals** (bool, optional) – qualities encoded as space-delimited integers
- **k** (int, optional) – (default: 5) report up to <int> alns per read
- **known-splicesite-infile** (str, optional) – provide a list of known splice sites <path>
- **library_type** (str, optional) – -1, -2 mates align fw/rev, rev/fw, fw/fw (–fr)
  - default value: fr
  - possible values: 'fr', 'rf', 'ff'
- **max-intronlen** (int, optional) – maximum intron length (500000)
- **maxins** (int, optional) – maximum fragment length (500), only valid with –no-spliced-alignment
- **min-intronlen** (int, optional) – minimum intron length (20)
- **minins** (int, optional) – minimum fragment length (0), only valid with –no-spliced-alignment
- **mm** (bool, optional) – use memory-mapped I/O for index; many 'hisat2's can share
- **mp** (str, optional) – max and min penalties for mismatch; lower qual = lower penalty <6,2>
- **n-ceil** (str, optional) – func for max # non-A/C/G/Ts permitted in aln (L,0,0.15)
- **no-discordant** (bool, optional) – suppress discordant alignments for paired reads
- **no-mixed** (bool, optional) – suppress unpaired alignments for paired reads
- **no-softclip** (bool, optional) – no soft-clipping
- **no-spliced-alignment** (bool, optional) – disable spliced alignment
- **no-temp-splicesite** (bool, optional) – disable the use of splice sites found
- **no-templatelen-adjustment** (bool, optional) – disables template length adjustment for RNA-seq reads
- **nofw** (bool, optional) – do not align forward (original) version of read (off)
- **non-deterministic** (bool, optional) – seed rand. gen. arbitrarily instead of using read attributes
- **norc** (bool, optional) – do not align reverse-complement version of read (off)
- **novel-splicesite-infile** (str, optional) – provide a list of novel splice sites <path>
- **novel-splicesite-outfile** (str, optional) – report a list of splice sites <path>
- **np** (int, optional) – penalty for non-A/C/G/Ts in read/ref (1)
- **offrate** (int, optional) – override offrate of index; must be >= index's offrate
- **pen-canintronlen** (str, optional) – penalty for long introns (G,-8,1) with canonical splice sites
- **pen-cansplice** (int, optional) – penalty for a canonical splice site (0)
- **pen-noncanintronlen** (str, optional) – penalty for long introns (G,-8,1) with noncanonical splice sites
- **pen-noncansplice** (int, optional) – penalty for a non-canonical splice site (12)
- **phred33** (bool, optional) – qualities are Phred+33 (default)
- **phred64** (bool, optional) – qualities are Phred+64
- **pigz-blocksize** (str, optional) - default value: 2048
- **q** (bool, optional) – query input files are FASTQ .fq/.fastq (default)
- **qc-filter** (bool, optional) – filter out reads that are bad according to QSEQ filter
- **qseq** (bool, optional) – query input files are in Illumina's qseq format

- **r** (bool, optional) – query input files are raw one-sequence-per-line
- **rdg** (str, optional) – read gap open, extend penalties (5,3)
- **remove-chrname** (bool, optional) – remove 'chr' from reference names in alignment
- **reorder** (bool, optional) – force SAM output order to match order of input reads
- **rfg** (str, optional) – –rfgreference gap open, extend penalties (5,3)
- **rna-strandness** (str, optional) – specify strand-specific information (unstranded)
  - possible values: 'unstranded', 'FR', 'RF'
- **score-min** (str, optional) – min acceptable alignment score w/r/t read length
- **seed** (int, optional) – seed for random number generator (0)
- **skip** (int, optional) – skip the first <int> reads/pairs in the input (none)
- **sp** (str, optional) – max and min penalties for soft-clipping; lower qual = lower penalty <2,1>
- **threads** (int, optional) – number of alignment threads to launch (1)
- **tmo** (bool, optional) – reports only those alignments within known transcriptome
- **trim3** (int, optional) – trim <int> bases from 3'/right end of reads (0)
- **trim5** (int, optional) – trim <int> bases from 5'/left end of reads (0)
- **upto** (int, optional) – stop after first <int> reads/pairs (no limit)

**Required tools:** hisat2

**CPU Cores:** 6

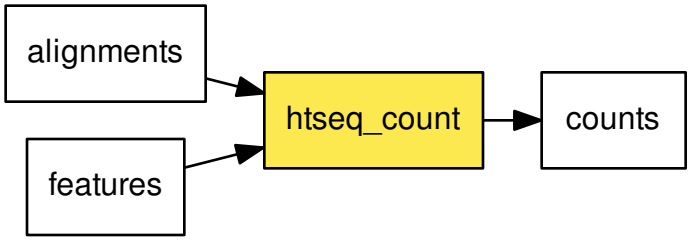
### htseq_count

The htseq-count script counts the number of reads overlapping a feature. Input needs to be a file with aligned sequencing reads and a list of genomic features. For more information see:

http://www-huber.embl.de/users/anders/HTSeq/doc/count.html

**Connections:**

- Input Connection:
  - 'in/alignments'
  - 'in/features'
- Output Connection:
  - 'out/counts'




**Options:**

- **a** (int, optional) - **dd-blocksize** (str, optional) - default value: 2M
- **feature-file** (str, optional) - **idattr** (str, optional) - default value: gene_id
- **merge_id** (str, optional) – The name of the run from assembly mergingsteps (stringtie_merge, or cuffmerge)
  - default value: magic
- **mode** (str, optional) - default value: union - possible values: 'union', 'intersection-strict', 'intersection-nonempty'
- **order** (str, required) - possible values: 'name', 'pos'
- **pigz-blocksize** (str, optional) - default value: 2048
- **stranded** (str, required) - possible values: 'yes', 'no', 'reverse'
- **threads** (int, optional) – start <n> threads (default:2)
  - default value: 2
- **type** (str, optional) - default value: exon

**Required tools:** dd, htseq-count, pigz, samtools

**CPU Cores:** 2

### macs2

Model-based Analysis of ChIP-Seq (MACS) is a algorithm, for the identifcation of transcript factor binding sites. MACS captures the influence of genome complexity to evaluate the significance of enriched ChIP regions, and MACS improves the spatial resolution of binding sites through combining the information of both sequencing tag position and orientation. MACS can be easily used for ChIP-Seq data alone, or with control sample data to increase the specificity.

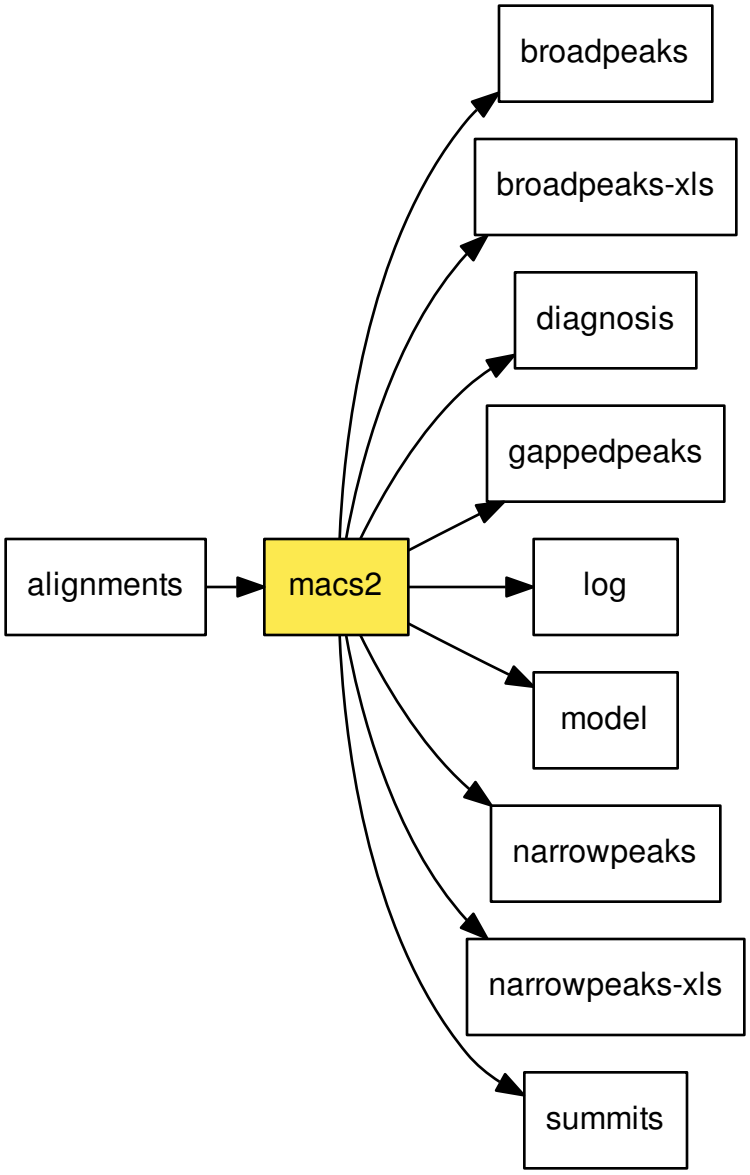
https://github.com/taoliu/MACS

typical command line for single-end data:

```
macs2 callpeak --treatment <aligned-reads> [--control <aligned-reads>]
               --name <run-id> --gsize 2.7e9
```

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/broadpeaks'
  - 'out/broadpeaks-xls'
  - 'out/diagnosis'
  - 'out/gappedpeaks'
  - 'out/log'
  - 'out/model'
  - 'out/narrowpeaks'
  - 'out/narrowpeaks-xls'
  - 'out/summits'




**Options:**

- **bdg** (bool, optional) – If this flag is on, MACS will store the fragment pileup, control lambda, -log10pvalue and -log10qvalue scores in bedGraph files. The bedGraph files will be stored in current directory named NAME+'_treat_pileup.bdg' for treatment data, NAME+'_control_lambda.bdg' for local lambda values from control, NAME+'_treat_pvalue.bdg' for Poisson pvalue scores (in -log10(pvalue) form), and NAME+'_treat_qvalue.bdg' for q-value scores from Benjamini-Hochberg-Yekutieli procedure <http://en.wikipedia.org/wiki/False_discovery_rate#Dependent_tests>
- **broad** (bool, optional) – When this flag is on, MACS will try to composite broad regions in BED12 ( a gene-model-like format ) by putting nearby highly enriched regions into a broad region with loose cutoff.

The broad region is controlled by another cutoff through –broad-cutoff. The maximum length of broad region length is 4 times of d from MACS. DEFAULT: False

- **broad-cutoff** (float, optional) – Cutoff for broad region. This option is not available unless –broad is set. If -p is set, this is a pvalue cutoff, otherwise, it's a qvalue cutoff. DEFAULT: 0.1
- **buffer-size** (int, optional) – LEGACY option.
- **bw** (int, optional) – The band width which is used to scan the genome ONLY for model building. You can set this parameter as the sonication fragment size expected from wet experiment. The previous side effect on the peak detection process has been removed. So this parameter only affects the model building.
- **call-summits** (bool, optional) – MACS will now reanalyze the shape of signal profile (p or q-score depending on cutoff setting) to deconvolve subpeaks within each peak called from general procedure. It's highly recommended to detect adjacent binding events. While used, the output subpeaks of a big peak region will have the same peak boundaries, and different scores and peak summit positions.
- **control** (dict, required) – Defines the controls and correspondent treatments in a YAML hash. Hash keys are the run IDs of the control datasets and hash values are the run IDs of the treatment datasets.
- **down-sample** (bool, optional) – When set, random sampling method will scale down the bigger sample. By default, MACS uses linear scaling. This option will make the results unstable and irreproducible since each time, random reads would be selected, especially the numbers (pileup, pvalue, qvalue) would change. Consider to use 'randsample' script before MACS2 runs instead.
- **extsize** (int, optional) – While '–nomodel' is set, MACS uses this parameter to extend reads in 5'->3' direction to fix-sized fragments. For example, if the size of binding region for your transcription factor is 200 bp, and you want to bypass the model building by MACS, this parameter can be set as 200. This option is only valid when –nomodel is set or when MACS fails to build model and –fix-bimodal is on.
- **fix-bimodal** (bool, optional) – Whether turn on the auto paired-peak model process. If it's set, when MACS failed to build paired model, it will use the nomodel settings, the '–extsize' parameter to extend each tags. If set, MACS will be terminated if paried-peak model is failed.
- **format** (str, required) – Format of tag file, can be 'ELAND', 'BED', 'ELANDMULTI', 'ELANDEXPORT', 'ELANDMULTIPET' (for pair-end tags), 'SAM', 'BAM', 'BOWTIE', 'BAMPE' or 'BEDPE'. Default is 'AUTO' which will allow MACS to decide the format automatically. 'AUTO' is also useful when you combine different formats of files. Note that MACS can't detect 'BAMPE' or 'BEDPE' format with 'AUTO', and you have to implicitly specify the format for 'BAMPE' and 'BEDPE'. For more information about the formats see https://github.com/taoliu/MACS/
  - default value: AUTO
  - possible values: 'AUTO', 'ELAND', 'ELANDMULTI', 'ELANDMULTIPET', 'ELANDEXPORT', 'BED', 'BEDPE', 'SAM', 'BAM', 'BAMPE', 'BOWTIE'
- **gsize** (str, required) – PLEASE assign this parameter to fit your needs! It's the mappable genome size or effective genome size which is defined as the genome size which can be sequenced. Because of the repetitive features on the chromsomes, the actual mappable genome size will be smaller than the original size, about 90% or 70% of the genome size. The default hs – 2.7e9 is recommended for UCSC human hg18 assembly. Here are all precompiled parameters for effective genome size: hs:2.7e9; mm:1.87e9; ce:9e7; dm:1.2e8
  - default value: 2.7e9
- **keep-dup** (int, optional) – It controls the MACS behavior towards duplicate tags at the exact same location – the same coordination and the same strand. The default 'auto' option makes MACS calculate the maximum tags at the exact same location based on binomal distribution using 1e-5 as pvalue cutoff; and the 'all' option keeps every tags. If an integer is given, at most this number of tags will be kept at the same location. The default is to keep one tag at the same location. Default: 1

- **llocal** (str, optional) – 'slocal' and 'llocal' control which two levels of regions will be checked around the peak regions to calculate the maximum lambda as local lambda. By default, MACS considers 1000bp for small local region(–slocal), and 10000bps for large local region(–llocal) which captures the bias from a long range effect like an open chromatin domain. You can tweak these according to your project. Remember that if the region is set too small, a sharp spike in the input data may kill the significant peak.
- **mfold** (str, optional) – This parameter is used to select the regions within MFOLD range of high-confidence enrichment ratio against background to build model. The regions must be lower than upper limit, and higher than the lower limit of fold enrichment. DEFAULT:5,50 means using all regions not too low (>5) and not too high (<50) to build paired-peaks model. If MACS can not find more than 100 regions to build model, it will use the –extsize parameter to continue the peak detection ONLY if –fix-bimodal is set.
- **nolambda** (bool, optional) – With this flag on, MACS will use the background lambda as local lambda. This means MACS will not consider the local bias at peak candidate regions.
- **nomodel** (bool, optional) – While on, MACS will bypass building the shifting model.
- **pvalue** (float, optional) – The pvalue cutoff. If 'pvalue' is specified, MACS2 will use pvalue instead of qvalue.
- **qvalue** (float, optional) – The qvalue (minimum FDR) cutoff to call significant regions. Default is 0.05. For broad marks, you can try 0.05 as cutoff. Q-values are calculated from p-values using Benjamini-Hochberg procedure.
- **read-length** (int, optional) – LEGACY option.
- **shift** (int, optional) - **slocal** (str, optional) – 'slocal' and 'llocal' control which two levels of regions will be checked around the peak regions to calculate the maximum lambda as local lambda. By default, MACS considers 1000bp for small local region(–slocal), and 10000bps for large local region(–llocal) which captures the bias from a long range effect like an open chromatin domain. You can tweak these according to your project. Remember that if the region is set too small, a sharp spike in the input data may kill the significant peak.
- **to-large** (bool, optional) – When set, linearly scale the smaller dataset to the same depth as larger dataset, by default, the larger dataset will be scaled towards the smaller dataset. Beware, to scale up small data would cause more false positives.
- **tsize** (int, optional) – The size of sequencing tags. If you don't specify it, MACS will try to use the first 10 sequences from your input treatment file to determine the tag size. Specifying it will override the automatically determined tag size.
- **verbose** (int, optional) – If you don't want to see any message during the running of MACS, set it to 0. But the CRITICAL messages will never be hidden. If you want to see rich information like how many peaks are called for every chromosome, you can set it to 3 or larger than 3.
  - possible values: '0', '1', '2', '3'

**Required tools:** macs2, mkdir, mv

**CPU Cores:** 4

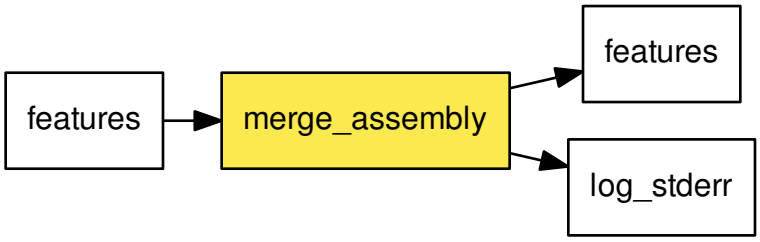
### merge_assembly

This step merges a single gtf file that has been produced by a previous step (e.g. cuffmerge or cuffcompare) with a reference annotation. No lines are discarded. The two files are simply concatenated and subsequently sorted by position.

**Connections:**

- Input Connection:

  - 'in/features'
- Output Connection:
  - 'out/features'
  - 'out/log_stderr'




**Options:**

- **reference** (str, required) – The reference annotation file that should be merged.
- **temp-sort-dir** (str, required) – Intermediate sort files are stored into this directory. Note that this directory needs to be present before running this step.

**Required tools:** cat, sort

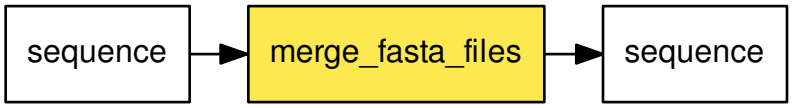
**CPU Cores:** 1

### merge_fasta_files

This step concatenates all .fasta(.gz) files that belong to a certain sample.

**Connections:**

- Input Connection:
  - 'in/sequence'
- Output Connection:
  - 'out/sequence'




**Options:**

- **compress-output** (bool, optional) – Produce gzipped output.

  - default value: True
- **dd-blocksize** (str, optional) - default value: 2M
- **merge-all-runs** (bool, optional) – Merge sequences from all runs.
- **output-fasta-basename** (str, optional) – Name used as prefix for FASTA output.
- **pigz-blocksize** (str, optional) - default value: 2048

**Required tools:** cat, dd, mkfifo, pigz

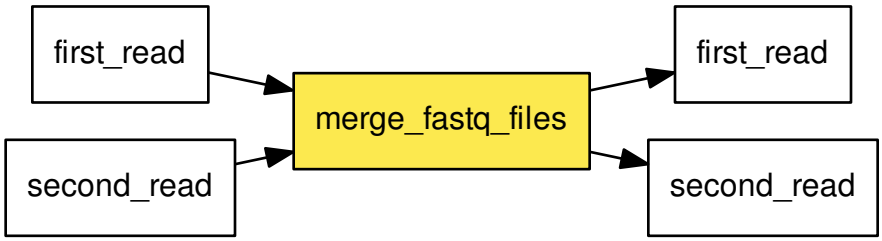
**CPU Cores:** 4

### merge_fastq_files

This step concatenates all .fastq(.gz) files belonging to a certain sample. First and second read files are merged separately. The output files are gzipped.

**Connections:**

- Input Connection:
  - 'in/first_read'
  - 'in/second_read'
- Output Connection:
  - 'out/first_read'
  - 'out/second_read'




**Options:**

- **dd-blocksize** (str, optional) - default value: 2M
- **pigz-blocksize** (str, optional) - default value: 2048

**Required tools:** cat, dd, mkfifo, pigz

**CPU Cores:** 4

### merge_numpy_zip_arrays

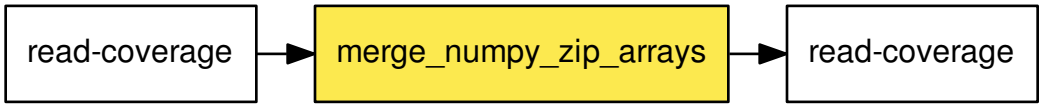
This step can be used to concatenate multiple zipped Numpy arrays which are the output of deepTools multiBamSummary subcommand.

Usage example:

```
merge_numpy_arrays.py [-h] [file-1.npz file-2.npz ... file-n.npz]  files
```

**Connections:**

- Input Connection:
  - 'in/read-coverage'
- Output Connection:
  - 'out/read-coverage'



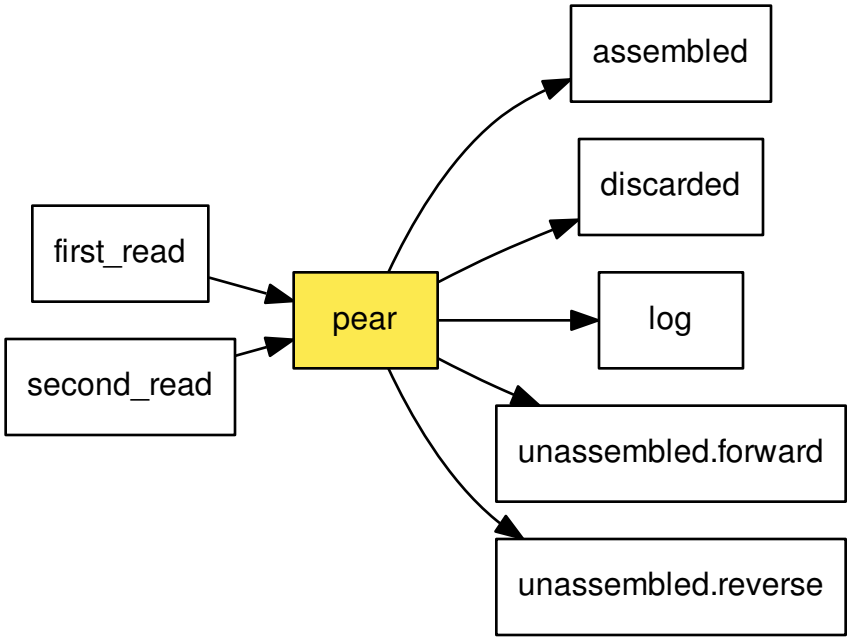

**Required tools:** merge_numpy_arrays.py

**CPU Cores:** 10

### pear

**Connections:**

- Input Connection:
  - 'in/first_read'
  - 'in/second_read'
- Output Connection:
  - 'out/assembled'
  - 'out/discarded'
  - 'out/log'
  - 'out/unassembled.forward'
  - 'out/unassembled.reverse'




**Options:**

- **cap** (int, optional) – Specify the upper bound for the resulting quality score.
  - default value: 40
  - possible values: '0', '33', '34', '35', '36', '37', '38', '39', '40', '41', '42', '43', '44', '45', '46', '47', '48', '49', '50', '51', '52', '53', '54', '55', '56', '57', '58', '59', '60', '61', '62', '63', '64', '65', '66', '67', '68', '69', '70', '71', '72', '73', '74', '75', '76', '77', '78', '79', '80', '81', '82', '83', '84', '85', '86', '87', '88', '89', '90', '91', '92', '93', '94', '95', '96', '97', '98', '99', '100', '101', '102', '103', '104', '105', '106', '107', '108', '109', '110', '111', '112', '113', '114', '115', '116', '117', '118', '119', '120', '121', '122', '123', '124', '125', '126', '127', '128', '129', '130', '131', '132', '133', '134', '135', '136', '137', '138', '139', '140', '141', '142', '143', '144', '145', '146', '147', '148', '149', '150', '151', '152', '153', '154', '155', '156', '157'
- **empirical-freqs** (bool, optional) – Disable empirical basefrequencies.
- **m** (int, optional) – Specify the maximum possible length of the assembled sequences.
  - default value: 300
- **max-uncalled-base** (float, optional) – Specify the maximal proportion of uncalled bases in a read.
  - default value: 1.0
  - possible values: '[0.0..1.0]'
- **memory** (str, optional) – Specify the amount of memory to be used.
- **min-overlap** (int, optional) – Specify the minimum overlap size.
  - default value: 10

- **min-trim-length** (int, optional) – Specify the minimum length of reads after trimming the low quality part.
  - default value: 1
- **n** (int, optional) – Specify the minimum possible length of the assembled sequences.
  - default value: 50
- **nbase** (bool, optional) – When resolving a mismatching base-pair out of which non is degenerate, set the merged base to N and use the highest quality score of the two bases.
- **p-value** (float, optional) – Specify a p-value for the statistical test.
  - default value: 0.01
  - possible values: '0.0001', '0.001', '0.01', '0.05', '1.0'
- **phred-base** (int, optional) – Base PHRED quality score.
  - default value: 33
  - possible values: '33', '64'
- **q** (int, optional) – Specify the quality score threshold for trimming the low quality part of a read.
  - possible values: '0', '33', '34', '35', '36', '37', '38', '39', '40', '41', '42', '43', '44', '45', '46', '47', '48', '49', '50', '51', '52', '53', '54', '55', '56', '57', '58', '59', '60', '61', '62', '63', '64', '65', '66', '67', '68', '69', '70', '71', '72', '73', '74', '75', '76', '77', '78', '79', '80', '81', '82', '83', '84', '85', '86', '87', '88', '89', '90', '91', '92', '93', '94', '95', '96', '97', '98', '99', '100', '101', '102', '103', '104', '105', '106', '107', '108', '109', '110', '111', '112', '113', '114', '115', '116', '117', '118', '119', '120', '121', '122', '123', '124', '125', '126', '127', '128', '129', '130', '131', '132', '133', '134', '135', '136', '137', '138', '139', '140', '141', '142', '143', '144', '145', '146', '147', '148', '149', '150', '151', '152', '153', '154', '155', '156', '157'
- **score-method** (int, optional) – Specify the scoring method.
  - default value: 2
  - possible values: '1', '2', '3'
- **test-method** (int, optional) – Specify the type of statistical tesit.
  - default value: 1
  - possible values: '1', '2'
- **threads** (int, optional) – Number of threads to use.

**Required tools:** pear

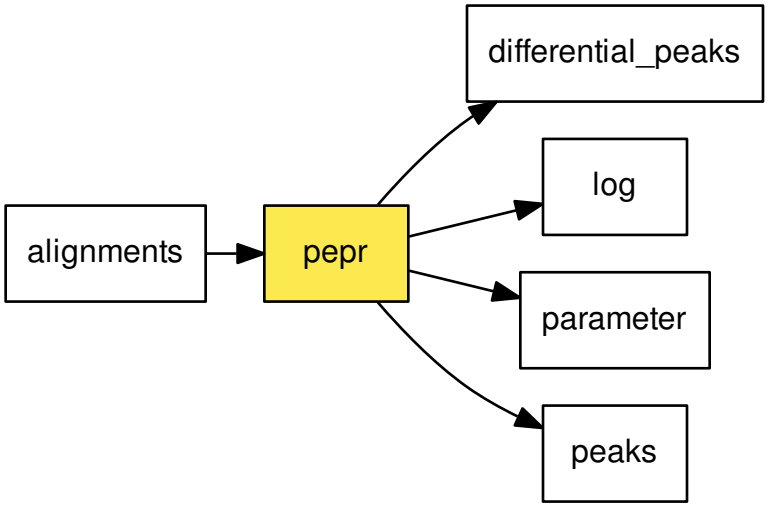
**CPU Cores:** 1

### pepr

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/differential_peaks'
  - 'out/log'
  - 'out/parameter'

- 'out/peaks'




**Options:**

- **chip_vs_input** (dict, required) – A YAML dictionary that contains: runID: rep1: [<List of runIDs>] input1: [<List of runIDs>]

**[ rep2: [<List of runIDs>]**

input2: [<List of runIDs>] ]rep2 and input2 are optional and will only be used for differential peak calling

- **diff** (bool, required) – Tell PePr to perform differential binding analysis or not.
- **file-format** (str, required) – Read file format. Currently support bed, sam, bam, sampe (sam paired-end), bampe (bam paired-end)
- **normalization** (str, optional) – inter-group, intra-group, scale, or no. Default is intra-group for peak-calling and inter-group for differential binding analysis. PePr is using a modified TMM method to normalize for the difference in IP efficiencies between samples (see the supplementary methods of the paper). It is making an implicit assumption that there is substantial overlap of peaks in every sample. However, it is sometimes not true between groups (for example, between TF ChIP-seq and TF knockout). So for differential binding analysis, switch to intra-group normalization. scale is simply scaling the reads so the total library sizes are the same. no normalization will not do normalization.
  - possible values: 'inter-group', 'intra-group', 'scale', 'no'
- **peaktype** (str, optional) – sharp or broad. Default is broad. PePr treats broad peaks (like H3K27me3) and sharp peaks (like most transcriptions factors) slightly different. Specify this option if you know the feature of the peaks.
  - possible values: 'sharp', 'broad'
- **shiftsize** (str, optional) – Half the fragment size. The number of bases to shift forward and reverse strand reads toward each other. If not specified by user, PePr will empirically estimate this number from the data for each ChIP sample.

- **threshold** (float, optional) – p-value cutoff. Default:1e-5.
- **windowsize** (int, optional) – Sliding window size. If not specified by user, PePr will estimate this by calculating the average width of potential peaks. The lower and upper bound for PePr estimate is 100bp and 1000bp. User provided window size is not constrained, but we recommend to stay in this range (100-1000bp).

**Required tools:** mkdir, mv, pepr, tar

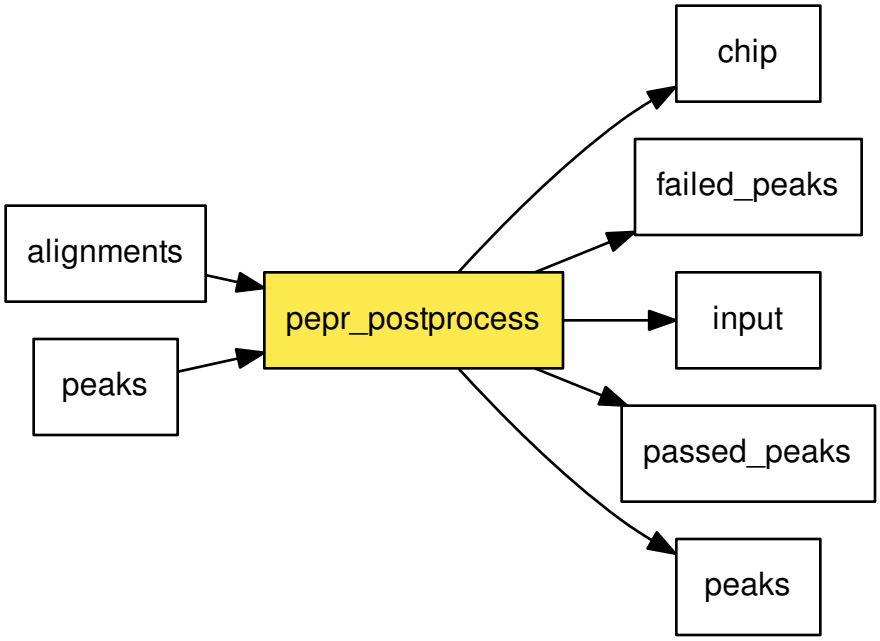
**CPU Cores:** 4

### pepr_postprocess

Post processing of peaks called by PePr. Attention: Filter criteria are hard coded!

**Connections:**

- Input Connection:
  - 'in/alignments'
  - 'in/peaks'
- Output Connection:
  - 'out/chip'
  - 'out/failed_peaks'
  - 'out/input'
  - 'out/passed_peaks'
  - 'out/peaks'




**Options:**

- **chip_vs_input** (dict, required) – A YAML dictionary that contains: runID: rep1: [<List of runIDs>] input1: [<List of runIDs>]
- **file-type** (str, required) – Read file format. Currently support bed, sam, bam
  - possible values: 'bed', 'sam', 'bam'
- **narrow-peak-boundary** (bool, optional) - **remove-artefacts** (bool, optional) - default value: True

**Required tools:** ln, pepr-postprocess

**CPU Cores:** 4

### picard_add_replace_read_groups

Replace read groups in a BAM file. This tool enables the user to replace all read groups in the INPUT file with a single new read group and assign all reads to this read group in the OUTPUT BAM file.

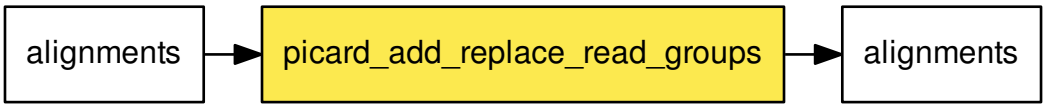
Documentation:

```
https://broadinstitute.github.io/picard/command-line-overview.html
↪#AddOrReplaceReadGroups
```

**Connections:**

- Input Connection:
  - 'in/alignments'

- Output Connection:
  - 'out/alignments'




**Options:**

- **COMPRESSION_LEVEL** (int, optional) – Compression level for all compressed files created (e.g. BAM and GELI). Default value: 5. This option can be set to "null" to clear the default value.
- **CREATE_INDEX** (bool, optional) – Whether to create a BAM index when writing a coordinate-sorted BAM file. Default value: false. This option can be set to "null" to clear the default value.
- **CREATE_MD5_FILE** (bool, optional) – Whether to create an MD5 digest for any BAM or FASTQ files created. Default value: false. This option can be set to "null" to clear the default value.
- **GA4GH_CLIENT_SECRETS** (str, optional) – Google Genomics API client_secrets.json file path. Default value: client_secrets.json. This option can be set to "null" to clear the default value.
- **MAX_RECORDS_IN_RAM** (int, optional) – When writing SAM files that need to be sorted, this will specify the number of records stored in RAM before spilling to disk. Increasing this number reduces the number of file handles needed to sort a SAM file, and increases the amount of RAM needed. Default value: 500000. This option can be set to "null" to clear the default value.
- **QUIET** (bool, optional) – Whether to suppress job-summary info on System.err. Default value: false. This option can be set to "null" to clear the default value.
- **REFERENCE_SEQUENCE** (str, optional) – Reference sequence file. Default value: null.
- **RGCN** (str, optional) – Read Group sequencing center name. Default value: null.
- **RGDS** (str, optional) – Read Group description. Default value: null.
- **RGDT** (str, optional) – Read Group run date. Default value: null.
- **RGID** (str, optional) – Read Group ID Default value: 1. This option can be set to 'null' to clear the default value.
- **RGLB** (str, required) – Read Group library
- **RGPG** (str, optional) – Read Group program group. Default value: null.
- **RGPI** (int, optional) – Read Group predicted insert size. Default value: null.
- **RGPL** (str, required) – Read Group platform (e.g. illumina, solid)
- **RGPM** (str, optional) – Read Group platform model. Default value: null.
- **RGPU** (str, required) – Read Group platform unit (eg. run barcode)
- **SORT_ORDER** (str, optional) – Optional sort order to output in. If not supplied OUTPUT is in the same order as INPUT. Default value: null. Possible values: {unsorted, queryname, coordinate, duplicate}
  - possible values: 'unsorted', 'queryname', 'coordinate', 'duplicate'

- **TMP_DIR** (str, optional) – A file. Default value: null. This option may be specified 0 or more times.
- **VALIDATION_STRINGENCY** (str, optional) – Validation stringency for all SAM files read by this program. Setting stringency to SILENT can improve performance when processing a BAM file in which variable-length data (read, qualities, tags) do not otherwise need to be decoded. Default value: STRICT. This option can be set to "null" to clear the default value.
  - possible values: 'STRICT', 'LENIENT', 'SILENT'
- **VERBOSITY** (str, optional) – Control verbosity of logging. Default value: INFO. This option can be set to "null" to clear the default value.
  - possible values: 'ERROR', 'WARNING', 'INFO', 'DEBUG'

**Required tools:** picard-tools

**CPU Cores:** 6

### picard_markduplicates

Identifies duplicate reads.

This tool locates and tags duplicate reads in a BAM or SAM file, where duplicate reads are defined as originating from a single fragment of DNA. Duplicates can arise during sample preparation e.g. library construction using PCR. See also EstimateLibraryComplexity for additional notes on PCR duplication artifacts. Duplicate reads can also result from a single amplification cluster, incorrectly detected as multiple clusters by the optical sensor of the sequencing instrument. These duplication artifacts are referred to as optical duplicates.

The MarkDuplicates tool works by comparing sequences in the 5 prime positions of both reads and read-pairs in a SAM/BAM file. An BARCODE_TAG option is available to facilitate duplicate marking using molecular barcodes. After duplicate reads are collected, the tool differentiates the primary and duplicate reads using an algorithm that ranks reads by the sums of their base-quality scores (default method).

The tool's main output is a new SAM or BAM file, in which duplicates have been identified in the SAM flags field for each read. Duplicates are marked with the hexadecimal value of 0x0400, which corresponds to a decimal value of 1024. If you are not familiar with this type of annotation, please see the following blog post for additional information.

Although the bitwise flag annotation indicates whether a read was marked as a duplicate, it does not identify the type of duplicate. To do this, a new tag called the duplicate type (DT) tag was recently added as an optional output in the 'optional field' section of a SAM/BAM file. Invoking the TAGGING_POLICY option, you can instruct the program to mark all the duplicates (All), only the optical duplicates (OpticalOnly), or no duplicates (DontTag). The records within the output of a SAM/BAM file will have values for the 'DT' tag (depending on the invoked TAGGING_POLICY), as either library/PCR-generated duplicates (LB), or sequencing-platform artifact duplicates (SQ). This tool uses the READ_NAME_REGEX and the OPTICAL_DUPLICATE_PIXEL_DISTANCE options as the primary methods to identify and differentiate duplicate types. Set READ_NAME_REGEX to null to skip optical duplicate detection, e.g. for RNA-seq or other data where duplicate sets are extremely large and estimating library complexity is not an aim. Note that without optical duplicate counts, library size estimation will be inaccurate.

MarkDuplicates also produces a metrics file indicating the numbers of duplicates for both single- and paired-end reads.

The program can take either coordinate-sorted or query-sorted inputs, however the behavior is slightly different. When the input is coordinate-sorted, unmapped mates of mapped records and supplementary/secondary alignments are not marked as duplicates. However, when the input is query-sorted (actually query-grouped), then unmapped mates and secondary/supplementary reads are not excluded from the duplication test and can be marked as duplicate reads.

If desired, duplicates can be removed using the REMOVE_DUPLICATE and REMOVE_SEQUENCING_DUPLICATES options.

Usage example:

```
java -jar picard.jar MarkDuplicates                I=input.bam                 ␣
↪O=marked_duplicates.bam               M=marked_dup_metrics.txt
```

Documentation:

```
https://broadinstitute.github.io/picard/command-line-overview.html
↪#MarkDuplicates
```

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/alignments'
  - 'out/metrics'


**Options:**

- **ASSUME_SORTED** (str, optional) – If true, assume that the input file is coordinate sorted even if the header says otherwise. Default value: false. This option can be set to 'null' to clear the default value. Possible values: {true, false}
  - possible values: 'true', 'null', 'false'
- **ASSUME_SORT_ORDER** (str, optional) – If not null, assume that the input file has this order even if the header says otherwise. Default value: null. Possible values: {unsorted, queryname, coordinate, duplicate, unknown}
  - possible values: 'unsorted', 'queryname', 'coordinate', 'duplicate', 'unknown'
- **BARCODE_TAG** (str, optional) – Barcode SAM tag (ex. BC for 10X Genomics) Default value: null.
- **COMMENT** (str, optional) – Comment(s) to include in the output file's header. Default value: null. This option may be specified 0 or more times.
- **COMPRESSION_LEVEL** (int, optional) – Compression level for all compressed files created (e.g. BAM and GELI). Default value: 5.

- **CREATE_INDEX** (bool, optional) – Whether to create a BAM index when writing a coordinate-sorted BAM file. Default value: false. This option can be set to "null" to clear the default value.
- **CREATE_MD5_FILE** (bool, optional) – Whether to create an MD5 digest for any BAM or FASTQ files created. Default value: false. This option can be set to "null" to clear the default value.
- **DUPLICATE_SCORING_STRATEGY** (str, optional) – The scoring strategy for choosing the non-duplicate among candidates. This option can be set to 'null' to clear the default value. Default value: SUM_OF_BASE_QUALITIES. Possible values: {SUM_OF_BASE_QUALITIES, TOTAL_MAPPED_REFERENCE_LENGTH, RANDOM}
  - possible values: 'SUM_OF_BASE_QUALITIES', 'TOTAL_MAPPED_REFERENCE_LENGTH', 'RANDOM'
- **GA4GH_CLIENT_SECRETS** (str, optional) – Google Genomics API client_secrets.json file path. Default value: client_secrets.json. This option can be set to "null" to clear the default value.
- **MAX_FILE_HANDLES** (int, optional) - **MAX_FILE_HANDLES_FOR_READ_ENDS_MAP** (int, optional) – Maximum number of file handles to keep open when spilling read ends to disk. Set this number a little lower than the per-process maximum number of file that may be open. This number can be found by executing the 'ulimit -n' command on a Unix system. Default value: 8000.
- **MAX_RECORDS_IN_RAM** (int, optional) – When writing SAM files that need to be sorted, this will specify the number of records stored in RAM before spilling to disk. Increasing this number reduces the number of file handles needed to sort a SAM file, and increases the amount of RAM needed. Default value: 500000.
- **OPTICAL_DUPLICATE_PIXEL_DISTANCE** (int, optional) – The maximum offset between two duplicate clusters in order to consider them optical duplicates. The default is appropriate for unpatterned versions of the Illumina platform. For the patterned flowcell models, 2500 is more appropriate. For other platforms and models, users should experiment to find what works best. Default value: 100. This option can be set to 'null' to clear the default value.
- **PROGRAM_GROUP_COMMAND_LINE** (str, optional) – Value of CL tag of PG record to be created. If not supplied the command line will be detected automatically. Default value: null.
- **PROGRAM_GROUP_NAME** (str, optional) – Value of PN tag of PG record to be created. Default value: MarkDuplicates. This option can be set to 'null' to clear the default value.
- **PROGRAM_GROUP_VERSION** (str, optional) – Value of VN tag of PG record to be created. If not specified, the version will be detected automatically. Default value: null.
- **PROGRAM_RECORD_ID** (str, optional) – The program record ID for the @PG record(s) created by this program. Set to null to disable PG record creation. This string may have a suffix appended to avoid collision with other program record IDs. Default value: MarkDuplicates. This option can be set to 'null' to clear the default value.
- **QUIET** (bool, optional) – Whether to suppress job-summary info on System.err. Default value: false. This option can be set to "null" to clear the default value.
- **READ_NAME_REGEX** (str, optional) – Regular expression that can be used to parse read names in the incoming SAM file. Read names are parsed to extract three variables: tile/region, x coordinate and y coordinate. These values are used to estimate the rate of optical duplication in order to give a more accurate estimated library size. Set this option to null to disable optical duplicate detection, e.g. for RNA-seq or other data where duplicate sets are extremely large and estimating library complexity is not an aim. Note that without optical duplicate counts, library size estimation will be inaccurate. The regular expression should contain three capture groups for the three variables, in order. It must match the entire read name. Note that if the default regex is specified, a regex match is not actually done, but instead the read name is split on colon character. For 5 element names, the 3rd, 4th and 5th elements are assumed to be tile, x and

y values. For 7 element names (CASAVA 1.8), the 5th, 6th, and 7th elements are assumed to be tile, x and y values. Default value: . This option can be set to 'null' to clear the default value.

- **READ_ONE_BARCODE_TAG** (str, optional) – Read one barcode SAM tag (ex. BX for 10X Genomics) Default value: null.
- **READ_TWO_BARCODE_TAG** (str, optional) – Read two barcode SAM tag (ex. BX for 10X Genomics) Default value: null.
- **REFERENCE_SEQUENCE** (str, optional) – Reference sequence file. Default value: null.
- **REMOVE_DUPLICATES** (str, optional) – If true do not write duplicates to the output file instead of writing them with appropriate flags set. Default value: false. This option can be set to 'null' to clear the default value. Possible values: {true, false}
  - default value: true
  - possible values: 'true', 'null', 'false'
- **REMOVE_SEQUENCING_DUPLICATES** (str, optional) – If true remove 'optical' duplicates and other duplicates that appear to have arisen from the sequencing process instead of the library preparation process, even if REMOVE_DUPLICATES is false. If REMOVE_DUPLICATES is true, all duplicates are removed and this option is ignored. Default value: false. This option can be set to 'null' to clear the default value. Possible values: {true, false}
  - possible values: 'true', 'false', 'null'
- **SORTING_COLLECTION_SIZE_RATIO** (float, optional) – This number, plus the maximum RAM available to the JVM, determine the memory footprint used by some of the sorting collections. If you are running out of memory, try reducing this number. Default value: 0.25.
- **TAGGING_POLICY** (str, optional) – Determines how duplicate types are recorded in the DT optional attribute. Default value: DontTag. This option can be set to 'null' to clear the default value. Possible values: {DontTag, OpticalOnly, All}
  - possible values: 'DontTag', 'OpticalOnly', 'All'
- **TAG_DUPLICATE_SET_MEMBERS** (str, optional) – If a read appears in a duplicate set, add two tags. The first tag, DUPLICATE_SET_SIZE_TAG (DS), indicates the size of the duplicate set. The smallest possible DS value is 2 which occurs when two reads map to the same portion of the reference only one of which is marked as duplicate. The second tag, DUPLICATE_SET_INDEX_TAG (DI), represents a unique identifier for the duplicate set to which the record belongs. This identifier is the index-in-file of the representative read that was selected out of the duplicate set. Default value: false. This option can be set to 'null' to clear the default value. Possible values: {true, false}
  - possible values: 'true', 'false', 'null'
- **TMP_DIR** (str, optional) – A file. Default value: null. This option may be specified 0 or more times.
- **VALIDATION_STRINGENCY** (str, optional) – Validation stringency for all SAM files read by this program. Setting stringency to SILENT can improve performance when processing a BAM file in which variable-length data (read, qualities, tags) do not otherwise need to be decoded. Default value: STRICT. This option can be set to "null" to clear the default value.
  - possible values: 'STRICT', 'LENIENT', 'SILENT', 'null'
- **VERBOSITY** (str, optional) – Control verbosity of logging. Default value: INFO. This option can be set to "null" to clear the default value.
  - possible values: 'ERROR', 'WARNING', 'INFO', 'DEBUG'

**Required tools:** picard-tools

**CPU Cores:** 12

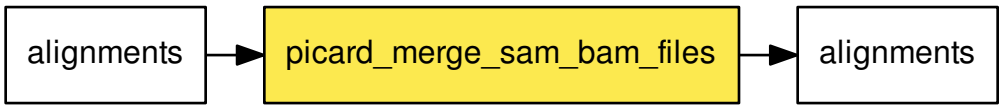
### picard_merge_sam_bam_files

Documentation:

```
https://broadinstitute.github.io/picard/command-line-overview.html
↪#MergeSamFiles
```

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/alignments'




**Options:**

- **ASSUME_SORTED** (bool, optional) – If true, assume that the input files are in the same sort order as the requested output sort order, even if their headers say otherwise. Default value: false. This option can be set to 'null' to clear the default value. Possible values: {true, false}
- **COMMENT** (str, optional) – Comment(s) to include in the merged output file's header. Default value: null.
- **COMPRESSION_LEVEL** (int, optional) – Compression level for all compressed files created (e.g. BAM and GELI). Default value: 5. This option can be set to "null" to clear the default value.
- **CREATE_INDEX** (bool, optional) – Whether to create a BAM index when writing a coordinate-sorted BAM file. Default value: false. This option can be set to "null" to clear the default value.
- **CREATE_MD5_FILE** (bool, optional) – Whether to create an MD5 digest for any BAM or FASTQ files created. Default value: false. This option can be set to "null" to clear the default value.
- **GA4GH_CLIENT_SECRETS** (str, optional) – Google Genomics API client_secrets.json file path. Default value: client_secrets.json. This option can be set to "null" to clear the default value.
- **INTERVALS** (str, optional) – An interval list file that contains the locations of the positions to merge. Assume bam are sorted and indexed. The resulting file will contain alignments that may overlap with genomic regions outside the requested region. Unmapped reads are discarded. Default value: null.
- **MAX_RECORDS_IN_RAM** (int, optional) – When writing SAM files that need to be sorted, this will specify the number of records stored in RAM before spilling to disk. Increasing this number reduces the number of file handles needed to sort a SAM file, and increases the amount of RAM needed. Default value: 500000. This option can be set to "null" to clear the default value.
- **MERGE_SEQUENCE_DICTIONARIES** (bool, optional) – Merge the sequence dictionaries. Default value: false. This option can be set to 'null' to clear the default value. Possible values: {true, false}

- **QUIET** (bool, optional) – Whether to suppress job-summary info on System.err. Default value: false. This option can be set to "null" to clear the default value.
- **REFERENCE_SEQUENCE** (str, optional) – Reference sequence file. Default value: null.
- **SORT_ORDER** (str, optional) – Sort order of output file. Default value: coordinate. This option can be set to 'null' to clear the default value. Possible values: {unsorted, queryname, coordinate, duplicate}
  - possible values: 'unsorted', 'queryname', 'coordinate', 'duplicate'
- **TMP_DIR** (str, optional) – A file. Default value: null. This option may be specified 0 or more times.
- **USE_THREADING** (bool, optional) – Option to create a background thread to encode, compress and write to disk the output file. The threaded version uses about 20% more CPU and decreases runtime by ~20% when writing out a compressed BAM file. Default value: false. This option can be set to 'null' to clear the default value. Possible values: {true, false}
- **VALIDATION_STRINGENCY** (str, optional) – Validation stringency for all SAM files read by this program. Setting stringency to SILENT can improve performance when processing a BAM file in which variable-length data (read, qualities, tags) do not otherwise need to be decoded. Default value: STRICT. This option can be set to "null" to clear the default value.
  - possible values: 'STRICT', 'LENIENT', 'SILENT'
- **VERBOSITY** (str, optional) – Control verbosity of logging. Default value: INFO. This option can be set to "null" to clear the default value.
  - possible values: 'ERROR', 'WARNING', 'INFO', 'DEBUG'

**Required tools:** ln, picard-tools

**CPU Cores:** 12

### post_cufflinksSuite

**The cufflinks suite can be used to assembly new transcripts and** merge those with known annotations. However, the output .gtf files need to be reformatted in several aspects afterwards. This step can be used to reformat and filter the cufflinksSuite .gtf file.

**Connections:**

- Input Connection:
  - 'in/features'
- Output Connection:
  - 'out/features'
  - 'out/log_stderr'


**Options:**

- **class-list** (str, optional) – Class codes to be removed; possible '=,c,j,e,i,o,p,r,u,x,s,.'
- **filter-by-class** (bool, required) – Remove gtf if any class is found in class_code field, requieres class_list
- **filter-by-class-and-gene-name** (bool, required) – Combines remove-by-class and remove-by-gene-name
- **remove-by-gene-name** (bool, required) – Remove gtf if matches 'string' in gene_name field
- **remove-gencode** (bool, required) – Hard removal of gtf line which match 'ENS' in gene_name field
- **remove-unstranded** (bool, required) – Removes transcripts without strand specifity
- **run_id** (str, optional) – An arbitrary name of the new run (which is a merge of all samples).
  - default value: magic
- **string** (str, optional) – String to match in gtf field gene_name for discarding

**Required tools:** cat, post_cufflinks_merge

**CPU Cores:** 6

### preseq_complexity_curve

The preseq package is aimed at predicting the yield of distinct reads from a genomic library from an initial sequencing experiment. The estimates can then be used to examine the utility of further sequencing, optimize the sequencing depth, or to screen multiple libraries to avoid low complexity samples.

c_curve computes the expected yield of distinct reads for experiments smaller than the input experiment in a .bed or .bam file through resampling. The full set of parameters can be outputed by simply typing the program name. If output.txt is the desired output file name and input.bed is the input .bed file, then simply type:

```
preseq c_curve -o output.txt input.sort.bed
```

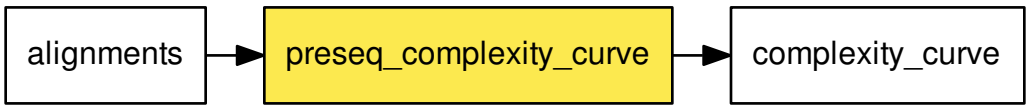
Documentation:

```
http://smithlabresearch.org/software/preseq/
```

**Connections:**

- Input Connection:
  - 'in/alignments'

- Output Connection:
  - 'out/complexity_curve'




**Options:**

- **hist** (bool, optional) – input is a text file containing the observed histogram
- **pe** (bool, required) – input is paired end read file
- **seg_len** (int, optional) – maximum segment length when merging paired end bam reads (default: 5000)
- **step** (int, optional) – step size in extrapolations (default: 1e+06)
- **vals** (bool, optional) – input is a text file containing only the observed counts
- **verbose** (bool, optional) – print more information

**Required tools:** preseq

**CPU Cores:** 4

### preseq_future_genome_coverage

The preseq package is aimed at predicting the yield of distinct reads from a genomic library from an initial sequencing experiment. The estimates can then be used to examine the utility of further sequencing, optimize the sequencing depth, or to screen multiple libraries to avoid low complexity samples.

gc_extrap computes the expected genomic coverage for deeper sequencing for single cell sequencing experiments. The input should be a mr or bed file. The tool bam2mr is provided to convert sorted bam or sam files to mapped read format.

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/future_genome_coverage'

**Options:**

- **bin_size** (int, optional) – bin size (default: 10)
- **bootstraps** (int, optional) – number of bootstraps (default: 100)
- **cval** (float, optional) – level for confidence intervals (default: 0.95)
- **extrap** (int, optional) – maximum extrapolation in base pairs (default: 1e+12)
- **max_width** (int, optional) – max fragment length, set equal to read length for single end reads
- **quick** (bool, optional) – quick mode: run gc_extrap without bootstrapping for confidence intervals
- **step** (int, optional) – step size in bases between extrapolations (default: 1e+08)
- **terms** (int, optional) – maximum number of terms

**Required tools:** preseq

**CPU Cores:** 4

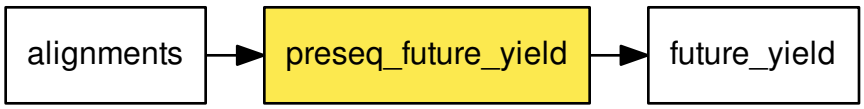
### preseq_future_yield

The preseq package is aimed at predicting the yield of distinct reads from a genomic library from an initial sequencing experiment. The estimates can then be used to examine the utility of further sequencing, optimize the sequencing depth, or to screen multiple libraries to avoid low complexity samples.

lc_extrap computes the expected future yield of distinct reads and bounds on the number of total distinct reads in the library and the associated confidence intervals.

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/future_yield'




**Options:**

- **bootstraps** (int, optional) – number of bootstraps (default: 100)
- **cval** (float, optional) – level for confidence intervals (default: 0.95)
- **dupl_level** (float, optional) – fraction of duplicate to predict (default: 0.5)
- **extrap** (int, optional) – maximum extrapolation (default: 1e+10)
- **hist** (bool, optional) – input is a text file containing the observed histogram
- **pe** (bool, required) – input is paired end read file

- **quick** (bool, optional) – quick mode, estimate yield without bootstrapping for confidence intervals
- **seg_len** (int, optional) – maximum segment length when merging paired end bam reads (default: 5000)
- **step** (int, optional) – step size in extrapolations (default: 1e+06)
- **terms** (int, optional) – maximum number of terms
- **vals** (bool, optional) – input is a text file containing only the observed counts

**Required tools:** preseq

**CPU Cores:** 4

### reformatCigar

The segemehl release from April 2017 (segemehl/export.13) uses the expanded version of Cigar strings: It clearly separates between match (=) and mismatch (X) while in older Cigar string definitions both are encoded with an M character. However, subsequent tools like htseq-count are not able to handle these newer versions of Cigar strings.

This step transforms the new Cigar string encoding back to the old one.

This step depends on the segemehl_2017 step. You still may need to add the subsequenct step s2c before calling cufflinks and/or htseq-count.

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/alignments'
  - 'out/log'


**Options:**

- **blocksize** (int, optional) – Blocksize to read the input file, in Megabytes.Default: 2 (2,000,000 bytes)
- **threads** (int, optional) – Number of threads 2B started. (Default: 1). Beware that this is only for (un-)compressing, the reformating is using a single CPU only.

**Required tools:** cat, pigz, segemehl_2017_reformatCigar

**CPU Cores:** 1

### remove_duplicate_reads_runs

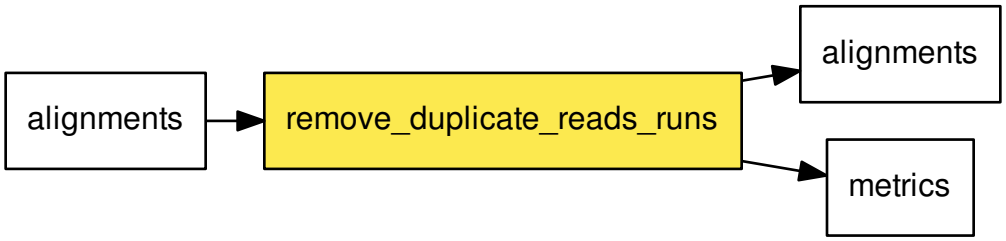
Duplicates are removed by Picard tools 'MarkDuplicates'.

typical command line:

```
MarkDuplicates INPUT=<SAM/BAM> OUTPUT=<SAM/BAM>
               METRICS_FILE=<metrics-out> REMOVE_DUPLICATES=true
```

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/alignments'
  - 'out/metrics'




**Required tools:** MarkDuplicates

**CPU Cores:** 12

### rgt_thor

THOR is an HMM-based approach to detect and analyze differential peaks in two sets of ChIP-seq data from distinct biological conditions with replicates. THOR performs genomic signal processing, peak calling and p-value calculation in an integrated framework. For differential peak calling without replicates use ODIN.

For more information please refer to:

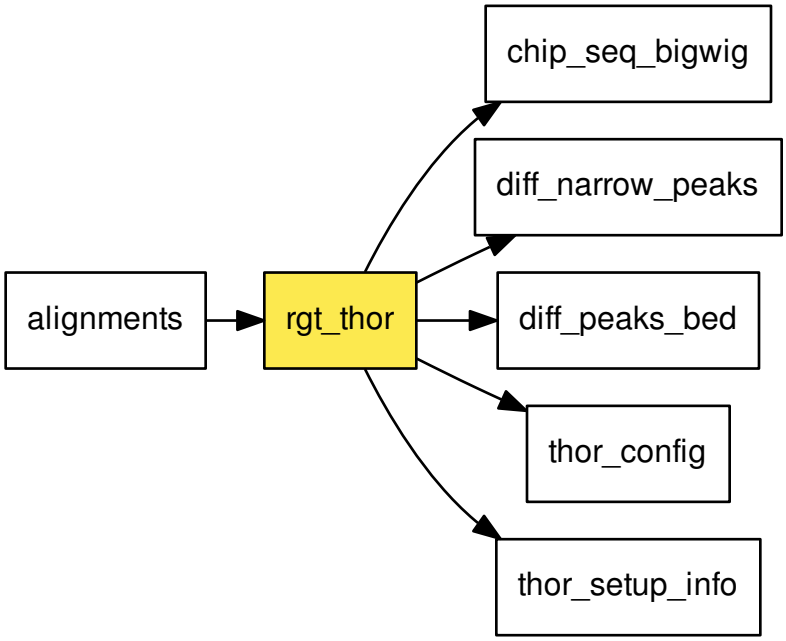
Allhoff, M., Sere K., Freitas, J., Zenke, M., Costa, I.G. (2016), Differential Peak Calling of ChIP-seq Signals with Replicates with THOR, Nucleic Acids Research, epub gkw680 [paper][supp].

Feel free to post your question in our googleGroup or write an e-mail: rgtusers@googlegroups.com

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:

- 'out/chip_seq_bigwig'
- 'out/diff_narrow_peaks'
- 'out/diff_peaks_bed'
- 'out/thor_config'
- 'out/thor_setup_info'




**Options:**

- **binsize** (int, optional) – Size of bins for creating the signal.
- **chrom_sizes_file** (str, required) - **config_file** (dict, required) – A dictionary with
- **deadzones** (str, optional) – Define blacklisted genomic regions to be ignored by the peak caller.
- **exts** (str, optional) – Read's extension size for BAM files (comma separated list for each BAM file in config file). If option is not chosen, estimate extension sizes from reads.
- **factors-inputs** (str, optional) – Normalization factors for input-DNA (comma separated list for each BAM file in config file). If option is not chosen, estimate factors.
- **genome** (str, required) – FASTA file containing the complete genome sequence
- **housekeeping-genes** (str, optional) – Define housekeeping genes (BED format) used for normalizing.
- **merge** (bool, optional) – Merge peaks which have a distance less than the estimated mean fragment size (recommended for histone data).
- **no-correction** (bool, optional) – Do not use multiple test correction for p-values (Benjamini/Hochberg).
- **no-gc-content** (bool, optional) – Do not normalize towards GC content.

- **pvalue** (float, optional) – P-value cutoff for peak detection. Call only peaks with p-value lower than cutoff.
- **report** (bool, optional) – Generate HTML report about experiment.
- **save-input** (bool, optional) – Save input DNA bigwig (if input was provided).
- **scaling-factors** (str, optional) – Scaling factor for each BAM file (not control input-DNA) as comma separated list for each BAM file in config file. If option is not chosen, follow normalization strategy (TMM or HK approach)
- **step** (int, optional) – Stepsize with which the window consecutively slides across the genome to create the signal.

**Required tools:** printf, rgt-THOR

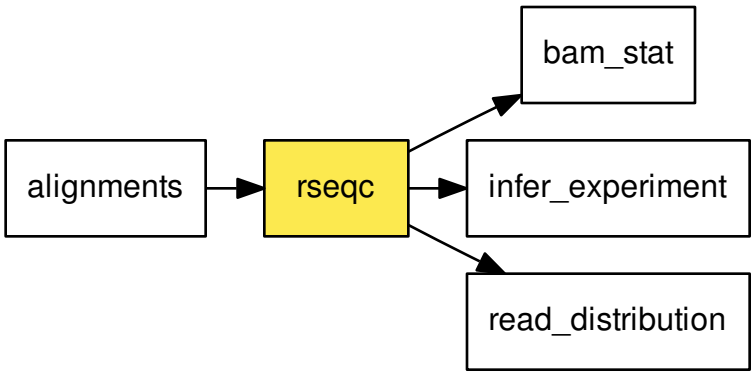
**CPU Cores:** 4

### rseqc

The RSeQC step can be used to evaluate aligned reads in a BAM file. RSeQC does not only report raw sequence-based metrics, but also quality control metrics like read distribution, gene coverage, and sequencing depth.

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/bam_stat'
  - 'out/infer_experiment'
  - 'out/read_distribution'




**Options:**

- **reference** (str, required) – Reference gene model in bed fomat. [required]

**Required tools:** bam_stat.py, cat, infer_experiment.py, read_distribution.py

**CPU Cores:** 1

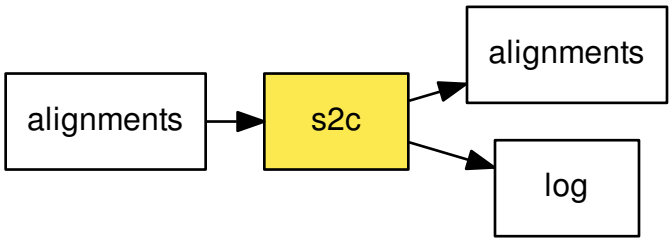
### s2c

s2c formats the output of segemehl mapping to be compatible with the cufflinks suite of tools for differential expr. analysis of RNA-Seq data and their visualisation. For details on cufflinks we refer to the author's webpage:

http://cole-trapnell-lab.github.io/cufflinks/

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/alignments'
  - 'out/log'




**Options:**

- **maxDist** (int, optional) – specifies the maximal distance of a splice junction. junctions with disctance higher than this value are classified as fusions (default is 200.000nt)
- **tmp_dir** (str, required) – Temp directory for 's2c.py'. This can be in the /work/username/ path, since it is only temporary.

**Required tools:** cat, dd, fix_s2c, pigz, s2c, samtools

**CPU Cores:** 6

### sam_to_sorted_bam

The step sam_to_sorted_bam builds on 'samtools sort' to sort SAM files and output BAM files.

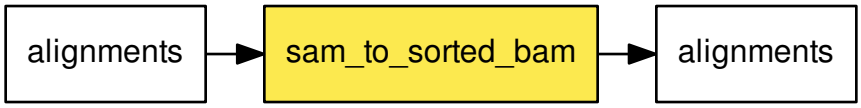
Sort alignments by leftmost coordinates, or by read name when -n is used. An appropriate @HD-SO sort order header tag will be added or an existing one updated if necessary.

Documentation:

```
http://www.htslib.org/doc/samtools.html
```

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/alignments'




**Options:**

- **dd-blocksize** (str, optional) - default value: 256k
- **genome-faidx** (str, required) - **sort-by-name** (bool, required) - **temp-sort-dir** (str, required) – Intermediate sort files are stored intothis directory.

**Required tools:** dd, pigz, samtools

**CPU Cores:** 8

### samtools

The step samtools wraps parts of the 'samtools' packages. It is intended for reformatting SAM/BAM files and not completely implemented.

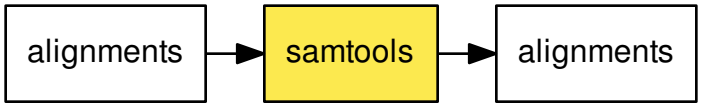
Feel free to add the samtools options you need!

The options listed below are implemented.

For a description/explanation about SAM flags, we refer to - the samtools manual page http://www.htslib.org/doc/samtools.html - the Picard page to explain SAM flags https://broadinstitute.github.io/picard/explain-flags.html

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/alignments'




**Options:**

- **F_skip** (int, optional) – Do not output alignments with any bits set in INT present in the FLAG field.
- **dd-blocksize** (str, optional) – Blocksize for dd tool.
  - default value: 2048
- **f_keep** (int, optional) – Only output alignments with all bits set in INT present in the FLAG field.
- **genome-faidx** (str, required) - **keep_header** (bool, optional) – Include the header in the output.
  - default value: True
- **output_bam** (bool, optional) – Output in the BAM format.
- **q_mapq** (int, optional) – Skip alignments with MAPQ smaller than this value.
- **sort-by-name** (bool, required) – Sort by read names (i.e., the QNAME field) rather than by chromosomal coordinates.
- **temp-sort-dir** (str, required) – Intermediate sort files are stored intothis directory.

**Required tools:** dd, pigz, samtools

**CPU Cores:** 8

### samtools_faidx

Index reference sequence in the FASTA format or extract subsequence from indexed reference sequence. If no region is specified, faidx will index the file and create <ref.fasta>.fai on the disk. If regions are specified, the subsequences will be retrieved and printed to stdout in the FASTA format.

The sequences in the input file should all have different names. If they do not, indexing will emit a warning about duplicate sequences and retrieval will only produce subsequences from the first sequence with the duplicated name.

**Connections:**

- Input Connection:
  - 'in/sequence'
- Output Connection:
  - 'out/indices'


**Required tools:** mv, samtools

**CPU Cores:** 4

### samtools_index

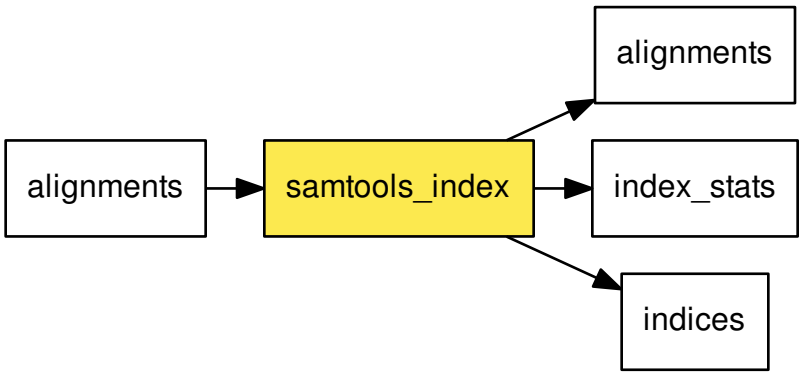
Index a coordinate-sorted BAM or CRAM file for fast random access. (Note that this does not work with SAM files even if they are bgzip compressed to index such files, use tabix(1) instead.)

Documentation:

```
http://www.htslib.org/doc/samtools.html
```

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/alignments'
  - 'out/index_stats'
  - 'out/indices'




**Options:**

- **index_type** (str, required) - possible values: 'bai', 'csi'

**Required tools:** ln, samtools

**CPU Cores:** 4

### samtools_merge

**The step samtools_merge builds on 'samtools merge' to merge sorted SAM/BAM files and output**
SAM/BAM/CRAM files.

Merge adds readgroup tag to preserve the information on the orginial sample.

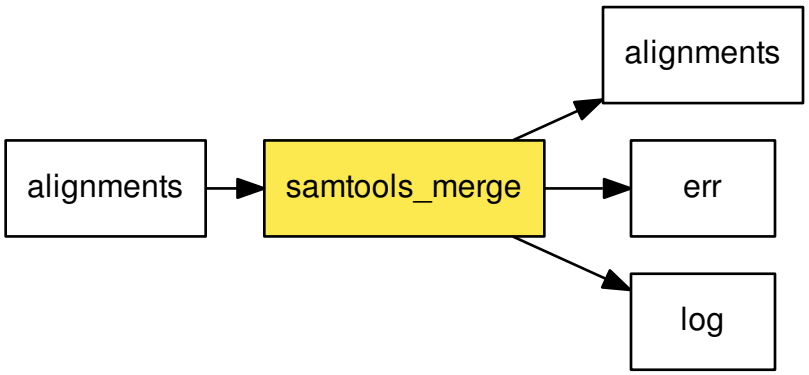
This step wraps samtools merge from release 1.7.

Documentation:

http://www.htslib.org/doc/samtools.html

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/alignments'
  - 'out/err'
  - 'out/log'




**Options:**

- **1** (bool, optional) – compress level 1
- **R** (str, optional) – merge file in the specified region STR [all]
- **c** (bool, optional) – Combine @RG headers with colliding IDs [alter IDs to be distinct]
- **dd-blocksize** (str, optional) - default value: 4M
- **f** (bool, optional) – overwrite the output BAM if exist

- **l** (int, optional) – compression level, from 0 to 9 [-1]
- **n** (bool, optional) – sort by read names
- **p** (bool, optional) – Combine @PG headers with colliding IDs [alter IDs to be distinct]
- **pigz-blocksize** (str, optional) - default value: 4096
- **r** (bool, optional) – attach RG tag (inferred from file names)
- **run_id** (str, optional) – A name for the run. Since this step merges multiple samples into a single one, the run_id cannot be the sample name anymore.
  - default value: mergeSBam
- **s** (str, optional) – override random seed
- **t** (str, optional) – Input files are sorted by TAG value
- **threads** (int, optional) – Number of additional threads to use [0]
- **u** (bool, optional) – uncompressed BAM output

**Required tools:** dd, pigz, samtools

**CPU Cores:** 6

### samtools_sort

The step samtools_sort builds on 'samtools sort' to sort SAM files and output SAM/BAM/CRAM files.

Sort alignments by leftmost coordinates, or by read name when -n is used.

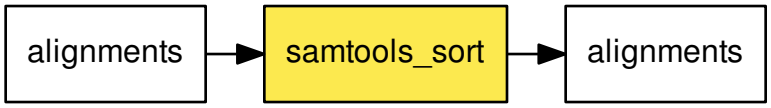
An appropriate @HD-SO sort order header tag will be added or an existing one updated if necessary.

Documentation:

```
http://www.htslib.org/doc/samtools.html
```

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/alignments'




**Options:**

- **compression-level** (int, optional) – Set compression level, from 0 (uncompressed) to 9 (best)
- **dd-blocksize** (str, optional) - default value: 4096k

- **genome-faidx** (str, required) – samtools index of the corresponding genome
- **max-mem-per-thread** (str, optional) – Set maximum memory per thread; suffix K/M/G recognized [768M]
  - default value: 768M
- **output-format** (str, required) – Write output as FORMAT (sam/bam/cram)
  - default value: sam
- **sort-by-name** (bool, optional) - **temp-sort-dir** (str, required) – Intermediate sort files are stored intothis directory.

**Required tools:** dd, pigz, samtools

**CPU Cores:** 6

### samtools_stats

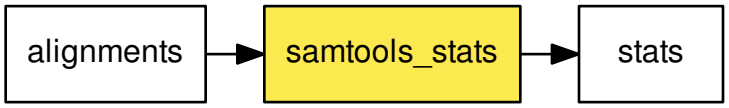
samtools stats collects statistics from BAM files and outputs in a text format. The output can be visualized graphically using plot-bamstats.

Documentation:

```
http://www.htslib.org/doc/samtools.html
```

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/stats'




**Options:**

- **dd-blocksize** (str, optional) - default value: 256k

**Required tools:** dd, pigz, samtools

**CPU Cores:** 1

### segemehl

segemehl is a software to map short sequencer reads to reference genomes. Unlike other methods, segemehl is able to detect not only mismatches but also insertions and deletions. Furthermore, segemehl is not limited to a specific read length and is able to mapprimer- or polyadenylation contaminated reads correctly.

This step creates at first two FIFOs. The first is used to provide the genome data for segemehl and the second is used for the output of the unmapped reads:

```
mkfifo genome_fifo unmapped_fifo
cat <genome-fasta> -o genome_fifo
```

The executed segemehl command is this:

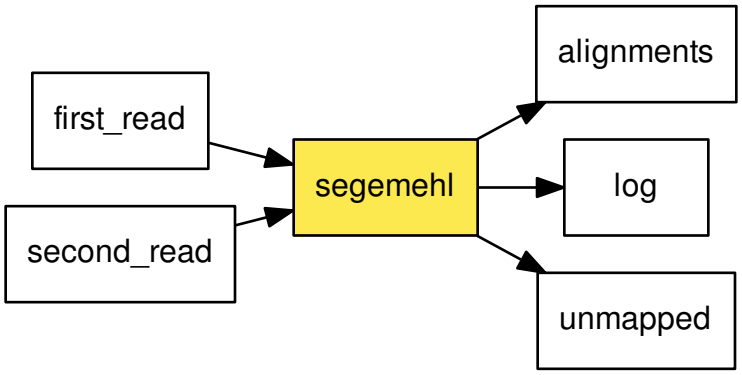
```
segemehl -d genome_fifo -i <genome-index-file> -q <read1-fastq>
         [-p <read2-fastq>] -u unmapped_fifo -H 1 -t 11 -s -S -D 0
         -o /dev/stdout |  pigz --blocksize 4096 --processes 2 -c
```

The unmapped reads are saved via these commands:

```
cat unmapped_fifo | pigz --blocksize 4096 --processes 2 -c >
<unmapped-fastq>
```

**Connections:**

- Input Connection:
  - 'in/first_read'
  - 'in/second_read'
- Output Connection:
  - 'out/alignments'
  - 'out/log'
  - 'out/unmapped'




**Options:**

- **MEOP** (bool, optional) – output MEOP field for easier variance calling in SAM (XE:Z:)
- **SEGEMEHL** (bool, optional) – output SEGEMEHL format (needs to be selected for brief)
- **accuracy** (int, optional) – min percentage of matches per read in semi-global alignment (default:90)
- **autoclip** (bool, optional) – autoclip unknown 3prime adapter

- **bisulfite** (int, optional) – bisulfite mapping with methylC-seq/Lister et al. (=1) or bs-seq/Cokus et al. protocol (=2) (default:0)
  - possible values: '0', '1', '2'
- **brief** (bool, optional) – brief output
- **clipacc** (int, optional) – clipping accuracy (default:70)
- **dd-blocksize** (str, optional) - default value: 2M
- **differences** (int, optional) – search seeds initially with <n> differences (default:1)
  - default value: 1
- **dropoff** (int, optional) – dropoff parameter for extension (default:8)
- **evalue** (float, optional) – max evalue (default:5.000000)
- **extensionpenalty** (int, optional) – penalty for a mismatch during extension (default:4)
- **extensionscore** (int, optional) – score of a match during extension (default:2)
- **filebins** (str, optional) – file bins with basename <string> for easier data handling (default:none)
  - default value: none
- **fix-qnames** (bool, optional) – The QNAMES field of the input will be purged from spaces and everything thereafter.
- **genome** (str, required) – Path to genome file
- **hardclip** (bool, optional) – enable hard clipping
- **hitstrategy** (int, optional) – report only best scoring hits (=1) or all (=0) (default:1)
  - default value: 1
  - possible values: '0', '1'
- **index** (str, required) – path/filename of db index (default:none)
- **index2** (str, optional) – path/filename of second db index (default:none)
- **jump** (int, optional) – search seeds with jump size <n> (0=automatic) (default:0)
- **maxinsertsize** (int, optional) – maximum size of the inserts (paired end) (default:5000)
- **maxinterval** (int, optional) – maximum width of a suffix array interval, i.e. a query seed will be omitted if it matches more than <n> times (default:100)
- **maxsplitevalue** (float, optional) – max evalue for splits (default:50.000000)
- **minfraglen** (int, optional) – min length of a spliced fragment (default:20)
- **minfragscore** (int, optional) – min score of a spliced fragment (default:18)
- **minsize** (int, optional) – minimum size of queries (default:12)
- **minsplicecover** (int, optional) – min coverage for spliced transcripts (default:80)
- **nohead** (bool, optional) – do not output header
- **order** (bool, optional) – sorts the output by chromsome and position (might take a while!)
- **pigz-blocksize** (str, optional) - default value: 2048
- **polyA** (bool, optional) – clip polyA tail
- **prime3** (str, optional) – add 3' adapter (default:none)

- **prime5** (str, optional) – add 5' adapter (default:none)
- **showalign** (bool, optional) – show alignments
- **silent** (bool, optional) – shut up!
  - default value: True
- **splicescorescale** (float, optional) – report spliced alignment with score s only if <f>*s is larger than next best spliced alignment (default:1.000000)
- **splits** (bool, optional) – detect split/spliced reads (default:none)
- **threads** (int, optional) – start <n> threads (default:10)
  - default value: 10

**Required tools:** cat, dd, fix_qnames, mkfifo, pigz, segemehl

**CPU Cores:** 10

### segemehl_2017

segemehl_2017 is a software to map short sequencer reads to reference genomes. Unlike other methods, segemehl is able to detect not only mismatches but also insertions and deletions. Furthermore, segemehl is not limited to a specific read length and is able to mapprimer- or polyadenylation contaminated reads correctly.

This step is a wrapper for an unpublished version of segemehl that we re- ceived from Steve Hoffmann on April, 24th 2017. It automatically generates the bed file containing the realigned split reads (split junctions). No need to run testrealign.x after segemehl anymore.

This step creates at first two FIFOs. The first is used to provide the genome data for segemehl and the second is used for the output of the unmapped reads:

```
mkfifo genome_fifo unmapped_fifo
cat <genome-fasta> -o genome_fifo
```

The executed segemehl command is this:

```
segemehl -d genome_fifo -i <genome-index-file> -q <read1-fastq>
         [-p <read2-fastq>] -u unmapped_fifo -H 1 -t 11 -s -S -D 0
         -o /dev/stdout |  pigz --blocksize 4096 --processes 2 -c
```

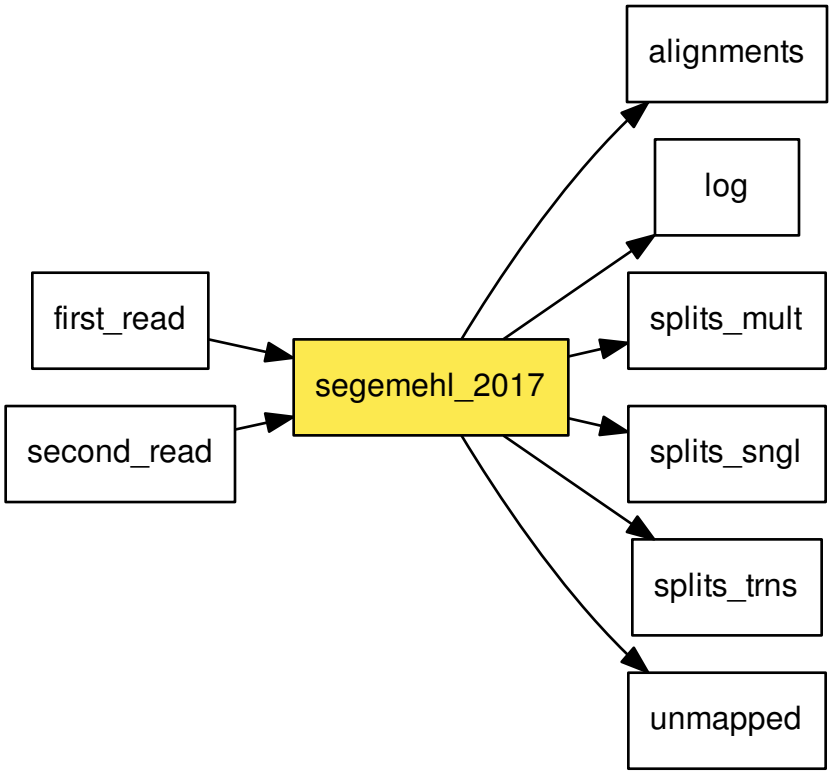
The unmapped reads are saved via these commands:

```
cat unmapped_fifo | pigz --blocksize 4096 --processes 2 -c >
<unmapped-fastq>
```

**Connections:**

- Input Connection:
  - 'in/first_read'
  - 'in/second_read'
- Output Connection:
  - 'out/alignments'
  - 'out/log'

- 'out/splits_mult'
- 'out/splits_sngl'
- 'out/splits_trns'
- 'out/unmapped'




**Options:**

- **MEOP** (bool, optional) – output MEOP field for easier variance calling in SAM (XE:Z:)
- **accuracy** (int, optional) – min percentage of matches per read in semi-global alignment (default:90)
- **bisulfite** (int, optional) – bisulfite mapping with methylC-seq/Lister et al. (=1) or bs-seq/Cokus et al. protocol (=2) (default:0)
  - possible values: '0', '1', '2'
- **brief** (bool, optional) – brief output
- **briefcigar** (bool, optional) – brief cigar string (M vs X and =)
- **checkidx** (bool, optional) – check index
- **clipacc** (int, optional) – clipping accuracy (default:70)
- **database** (str, required) – (Space separated list of ) filename(s) of database (e.g. genome) sequene(s)

- **dd-blocksize** (str, optional) - default value: 1M
- **differences** (int, optional) – search seeds initially with <n> differences (default:1)
- **dropoff** (int, optional) – dropoff parameter for extension (default:8)
- **evalue** (float, optional) – max evalue (default:5.000000)
- **extensionpenalty** (int, optional) – penalty for a mismatch during extension (default:4)
- **filebins** (str, optional) – file bins with basename <string> for easier data handling (default:none)
- **fix-qnames** (bool, optional) – The QNAMES field of the input will be purged from spaces and everything thereafter.
- **hitstrategy** (int, optional) – report only best scoring hits (=1) or all (=0) (default:1)
  - possible values: '0', '1'
- **index** (str, optional) – Path to database index for segemehl (default:none)
- **index2** (str, optional) – Path to second database index for segemehl (default:none)
- **jump** (int, optional) – search seeds with jump size <n> (0=automatic) (default:0)
- **maxinsertsize** (int, optional) – maximum size of the inserts (paired end) (default:5000)
- **maxinterval** (int, optional) – maximum width of a suffix array interval, i.e. a query seed will be omitted if it matches more than <n> times (default:100)
- **maxout** (int, optional) – maximum number of alignments that will be reported. If set to zero, all alignments will be reported (default:0)
- **maxsplitevalue** (float, optional) – max evalue for splits (default:50.000000)
- **minfraglen** (int, optional) – min length of a spliced fragment (default:20)
- **minfragscore** (int, optional) – min score of a spliced fragment (default:18)
- **minsize** (int, optional) – minimum size of queries (default:12)
- **minsplicecover** (int, optional) – min coverage for spliced transcripts (default:80)
- **nohead** (bool, optional) – do not output header
- **nosuflinks** (bool, optional) – dont use suflinks (does not affect index construction, for short reads only, increases runtime!)
- **prime3** (str, optional) – add 3' adapter (default:none)
- **prime5** (str, optional) – add 5' adapter (default:none)
- **readgroupfile** (str, optional) – filename to read @RG header (default:none)
- **readgroupid** (str, optional) – read group id (default:none)
- **showalign** (bool, optional) – show alignments
- **splicescorescale** (float, optional) – report spliced alignment with score s only if <f>*s is larger than next best spliced alignment (default:1.000000)
- **splits** (bool, optional) – detect split/spliced reads (default:none)
- **threads** (int, optional) – start <n> threads (default:1)
  - default value: 1

**Required tools:** cat, dd, fix_qnames, mkfifo, pigz, segemehl

**CPU Cores:** 10

### segemehl_generate_index

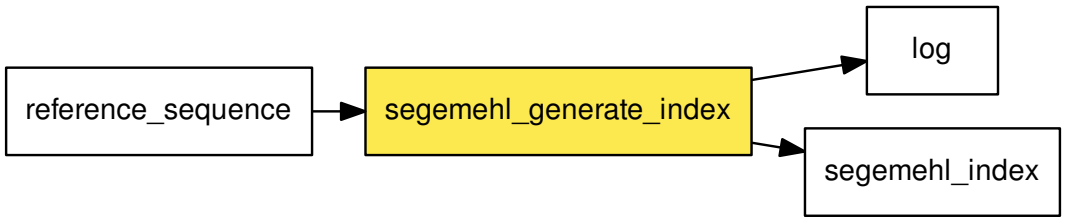
The step segemehl_generate_index generates a index for given reference sequences.

Documentation:

```
http://www.bioinf.uni-leipzig.de/Software/segemehl/
```

**Connections:**

- Input Connection:
  - 'in/reference_sequence'
- Output Connection:
  - 'out/log'
  - 'out/segemehl_index'




**Options:**

- **dd-blocksize** (str, optional) - default value: 2M
- **index-basename** (str, required) – Basename for created segemehl index.
- **pigz-blocksize** (str, optional) - default value: 2048
- **threads** (int, optional) – start <n> threads (default:4)

**Required tools:** dd, mkfifo, pigz, segemehl

**CPU Cores:** 4

### segemehl_generate_index_bisulfite

The step segemehl_generate_index_bisulfite generates a pair of indexes for given reference for segemehl bisulfite sequencing mapping.

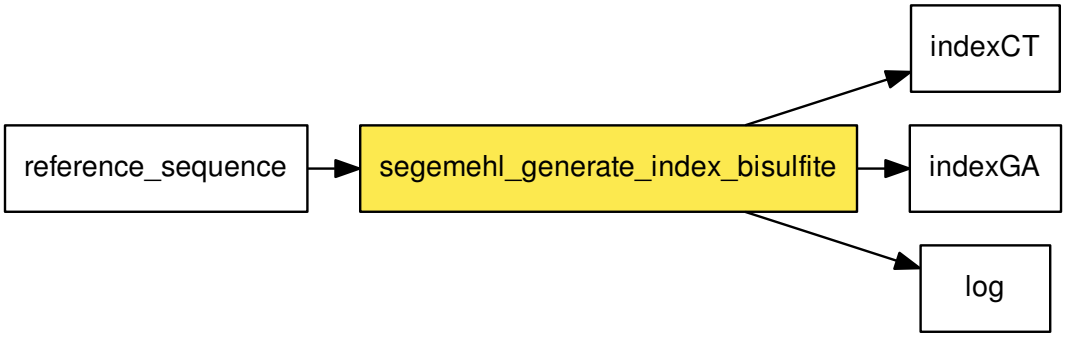
Documentation:

```
http://www.bioinf.uni-leipzig.de/Software/segemehl/
```

**Connections:**

- Input Connection:
  - 'in/reference_sequence'

- Output Connection:
  - 'out/indexCT'
  - 'out/indexGA'
  - 'out/log'




**Options:**

- **bisulfite** (int, optional) – bisulfite mapping with methylC-seq/Lister et al. (=1) or bs-seq/Cokus et al. protocol (=2) (default:0)
  - default value: 1
  - possible values: '1', '2'
- **dd-blocksize** (str, optional) - default value: 2M
- **generate** (str, optional) – Filename of first (CT) db index that is generated and store to disk (efault: index-basename.ctidx).
- **generate2** (str, optional) – Filename of 2nd (GA) db index that is generated and store to disk (efault: index-basename.ctidx).
- **index-basename** (str, required) – Basename for created segemehl index.
- **pigz-blocksize** (str, optional) - default value: 2048
- **threads** (int, optional) – start <n> threads (default:4)

**Required tools:** dd, mkfifo, pigz, segemehl

**CPU Cores:** 4

### sra_fastq_dump

sra tools is a suite from NCBI to handle sra (short read archive) files. fastq-dump is an sra tool that dumps the content of an sra file in fastq format

The following options cannot be set, as they would interefere with the pipeline implemented in this step

**-O|–outdir <path> Output directory, default is working** directory '.' )

**-Z|–stdout Output to stdout, all split data become** joined into single stream

**--gzip** Compress output using gzip

**--bzip2** Compress output using bzip2

**Multiple File Options Setting these options will produce more**

than 1 file, each of which will be suffixed according to splitting criteria.

**--split-files** Dump each read into separate file.Files will receive suffix corresponding to read number

**--split-3** Legacy 3-file splitting for mate-pairs: First biological reads satisfying dumping conditions are placed in files *_1.fastq and *_2.fastq If only one biological read is present it is placed in *.fastq Biological reads and above are ignored.

-G|–spot-group Split into files by SPOT_GROUP (member name) -R|–read-filter <[filter]> Split into files by READ_FILTER value

optionally filter by value: pass|reject|criteria|redacted

-T|–group-in-dirs Split into subdirectories instead of files -K|–keep-empty-files Do not delete empty files

Details on fastq-dump can be found at https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=toolkit_doc&f=fastq-dump

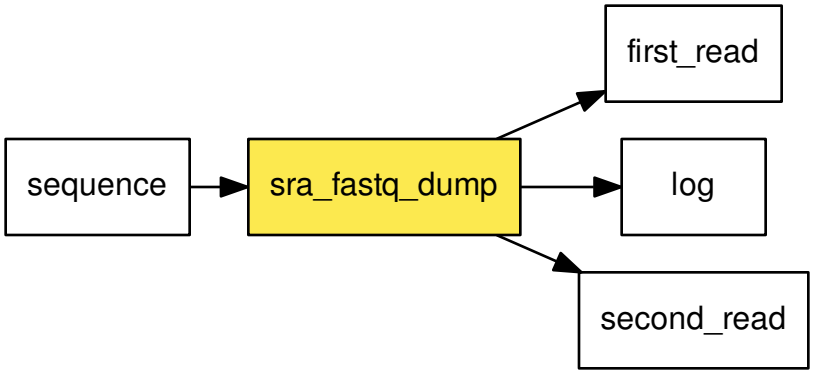
To make IO cluster friendly, fastq-dump is not reading th sra file directly. Rather, dd with configurable blocksize is used to provide the sra file via a fifo to fastq-dump.

The executed calls lools like this

mkfifo sra_fifo dd bs=4M if=<sra-file> of=sra_fifo fastq-dump -Z sra_fifo | pigz –blocksize 4096 –processes 2 > file.fastq

**Connections:**

- Input Connection:
  - 'in/sequence'
- Output Connection:
  - 'out/first_read'
  - 'out/log'
  - 'out/second_read'




**Options:**

- **accession** (str, optional) – Replaces accession derived from <path> in filename(s) and deflines (only for single table dump)
- **aligned** (bool, optional) – Dump only aligned sequences
- **aligned-region** (str, optional) – Filter by position on genome. Name can either be accession.version (ex:NC_000001.10) or file specific name (ex:"chr1" or "1"). "from" and "to" are 1-based coordinates. <name[:from-to]>
- **clip** (bool, optional) – Apply left and right clips
- **dd-blocksize** (str, optional) - default value: 256k
- **defline-qual** (str, optional) – Defline format specification for quality.
- **defline-seq** (str, optional) – Defline format specification for sequence.
- **disable-multithreading** (bool, optional) – disable multithreading
- **dumpbase** (bool, optional) – Formats sequence using base space (default for other than SOLiD).
- **dumpcs** (bool, optional) – Formats sequence using color space (default for SOLiD),"cskey" may be specified for translation.
- **fasta** (int, optional) – FASTA only, no qualities, optional line wrap width (set to zero for no wrapping). <[line width]>
- **helicos** (bool, optional) – Helicos style defline
- **legacy-report** (bool, optional) – use legacy style "Written spots" for tool
- **log-level** (str, optional) – Logging level as number or enum string One of (fatal|sys|int|err|warn|info) or (0-5). Current/default is warn. <level>
- **matepair-distance** (str, optional) – Filter by distance beiween matepairs. Use "unknown" to find matepairs split between the references. Use from-to to limit matepair distance on the same reference. <from-to|unknown>
- **maxSpotId** (int, optional) – Maximum spot id to be dumped. Use with "minSpotId" to dump a range.
- **max_cores** (int, optional) – Maximum number of cores available on the cluster

  - default value: 10
- **minReadLen** (int, optional) – Filter by sequence length >= <len>
- **minSpotId** (int, optional) – Minimum spot id to be dumped. Use with "maxSpotId" to dump a range.
- **ncbi_error_report** (str, optional) – Control program execution environment report generation (if implemented). One of (never|error|always). Default is error. <error>
- **offset** (int, optional) – Offset to use for quality conversion, default is 33
- **origfmt** (bool, optional) – Defline contains only original sequence name
- **qual-filter** (bool, optional) – Filter used in early 1000 Genomes data: no sequences starting or ending with >= 10N
- **qual-filter-1** (bool, optional) – Filter used in current 1000 Genomes data
- **read-filter** (str, optional) – Split into files by READ_FILTER value optionally filter by value: pass|reject|criteria|redacted
- **readids** (bool, optional) – Append read id after spot id as "accession.spot.readid" on defline.
- **skip-technical** (bool, optional) – Dump only biological reads
- **split-spot** (str, optional) – Split spots into individual reads
- **spot-groups** (str, optional) – Filter by SPOT_GROUP (member): name[,...]
- **suppress-qual-for-cskey** (bool, optional) – supress quality-value for cskey
- **table** (str, optional) – Table name within cSRA object, default is "SEQUENCE"
- **unaligned** (bool, optional) – Dump only unaligned sequences
- **verbose** (bool, optional) – Increase the verbosity level of the program. Use multiple times for more verbosity.

**Required tools:** dd, fastq-dump, mkfifo, pigz

**CPU Cores:** 10

### stringtie

**StringTie is a fast and highly efficient assembler of RNA-Seq alignments into potential** transcripts. It uses a novel network flow algorithm as well as an optional de novo assembly step to assemble and quantitate full-length transcripts representing multiple splice variants for each gene locus. Its input can include not only the alignments of raw reads used by other transcript assemblers, but also alignments longer sequences that have been assembled from those reads.In order to identify differentially expressed genes between experiments, StringTie's output can be processed by specialized software like Ballgown, Cuffdiff or other programs (DESeq2, edgeR, etc.).

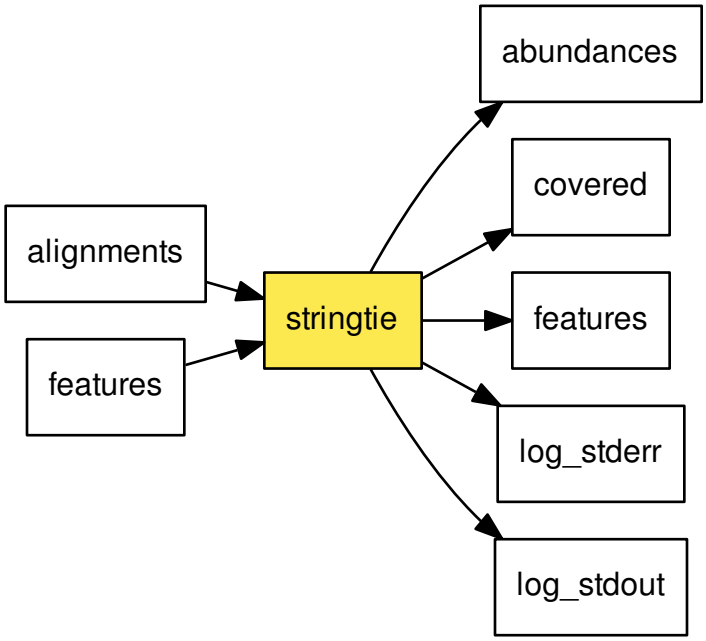
NOTE: This step implements that part of stringtie that assembles new transcripts. If you want stringtie to assemble transcripts from multiple input files please use step stringtie_merge!

https://ccb.jhu.edu/software/stringtie/

**Connections:**

- Input Connection:
  - 'in/alignments'
  - 'in/features'
- Output Connection:

  - 'out/abundances'
  - 'out/covered'
  - 'out/features'
  - 'out/log_stderr'
  - 'out/log_stdout'




**Options:**

- **B** (bool, optional) – Enable the ouput of Ballgown input table files (.ctab). containing coverage data for the reference transcripts given with the -G option. (See the Ballgown documentation for a description of these files.) With this option StringTie can be used as a direct replacement of the tablemaker program included with the Ballgown distribution.
- **G** (str, optional) – reference annotation to use for guiding the assembly process (GTF/GFF3)
- **M** (float, optional) – fraction of bundle allowed to be covered by multi-hit reads (default:0.95)
- **a** (int, optional) – minimum anchor length for junctions (default: 10)
- **c** (float, optional) – minimum reads per bp coverage to consider for transcript assembly (default: 2.5)
- **dd-blocksize** (str, optional) – Provide the blocksize for dd tool.
  - default value: 2M
- **e** (bool, optional) – only estimate the abundance of given reference transcripts (requires -G)
- **f** (float, optional) – minimum isoform fraction (default: 0.1)
- **fifo** (bool, optional) – Enable the FIFO functionality for splitting large input files.

- **g** (int, optional) – gap between read mappings triggering a new bundle (default: 50)
- **j** (float, optional) – minimum junction coverage (default: 1)
- **l** (str, optional) – name prefix for output transcripts (default: STRG)
- **library_type** (str, required) – Assume stranded library fr-firststrand (–rf) or fr-secondstrand (–fr).
  - possible values: 'rf', 'fr'
- **m** (int, optional) – minimum assembled transcript length (default: 200)
- **merged-id** (str, optional) – The run-id that has been used in the samtools_merge step before
  - default value: mergeSBam
- **p** (int, optional) – number of threads (CPUs) to use (default: 1)
- **t** (bool, optional) – disable trimming of predicted transcripts based on coverage (default: coverage trimming is enabled)
- **v** (bool, optional) – verbose (log bundle processing details)
- **x** (str, optional) – Ignore all read alignments (and thus do not attempt to perform transcript assembly) on the specified reference sequences. Parameter <seqid_list> can be a single reference sequence name (e.g. -x chrM) or a comma-delimited list of sequence names (e.g. -x "chrM,chrX,chrY"). This can speed up StringTie especially in the case of excluding the mitochondrial genome, whose genes may have very high coverage in some cases, even though they may be of no interest for a particular RNA-Seq analysis. The reference sequence names are case sensitive, they must match identically the names of chromosomes/contigs of the target genome against which the RNA-Seq reads were aligned in the first place.

**Required tools:** dd, mkfifo, stringtie

**CPU Cores:** 6

### stringtie_merge

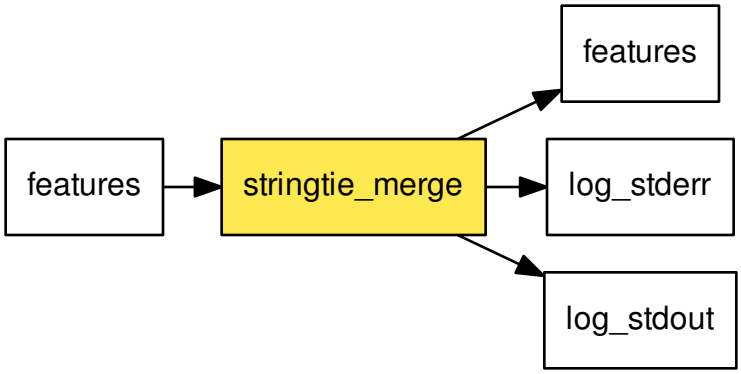
**StringTie is a fast and highly efficient assembler of RNA-Seq alignments into potential** transcripts. merge is a mode of the StringTie tool that is used to assemble transcripts from multiple input files (assemblies). It generates a unified non-redundant set of isoforms.

NOTE: This step implements the merging part of stringtie. If you want stringtie to assemble transcripts from multiple BAM files please use step stringtie!

https://ccb.jhu.edu/software/stringtie/

**Connections:**

- Input Connection:
  - 'in/features'
- Output Connection:
  - 'out/features'
  - 'out/log_stderr'
  - 'out/log_stdout'




**Options:**

- **F** (float, optional) – minimum input transcript FPKM to include in the merge (default: 1.0)
- **G** (str, optional) – reference annotation to use for guiding the assembly process (GTF/GFF3)
- **T** (float, optional) – minimum input transcript TPM to include in the merge (default: 1.0)
- **c** (float, optional) – minimum input transcript coverage to include in the merge (default: 0)
- **f** (float, optional) – minimum isoform fraction (default: 0.01)
- **g** (int, optional) – gap between transcripts to merge together (default: 250)
- **i** (bool, optional) – keep merged transcripts with retained introns; by default
- **l** (str, optional) – name prefix for output transcripts (default: MSTRG)
- **m** (int, optional) – minimum input transcript length to include in the merge (default: 50)
- **run_id** (str, optional) – A name for the run. Since this step merges multiple assemblies (gtf files) into a single one, the run_id cannot be the sample name anymore.
  - default value: mergeGTF

**Required tools:** stringtie

**CPU Cores:** 6

### stringtie_prepDE

**StringTie is a fast and highly efficient assembler of RNA-Seq alignments into potential** transcripts. prepDE.py is Python script to extract this read count information directly from the files generated by StringTie (run with the -e parameter). It generates two CSV files containing the count matrices for genes and transcripts, using the coverage values found in the output of stringtie -e

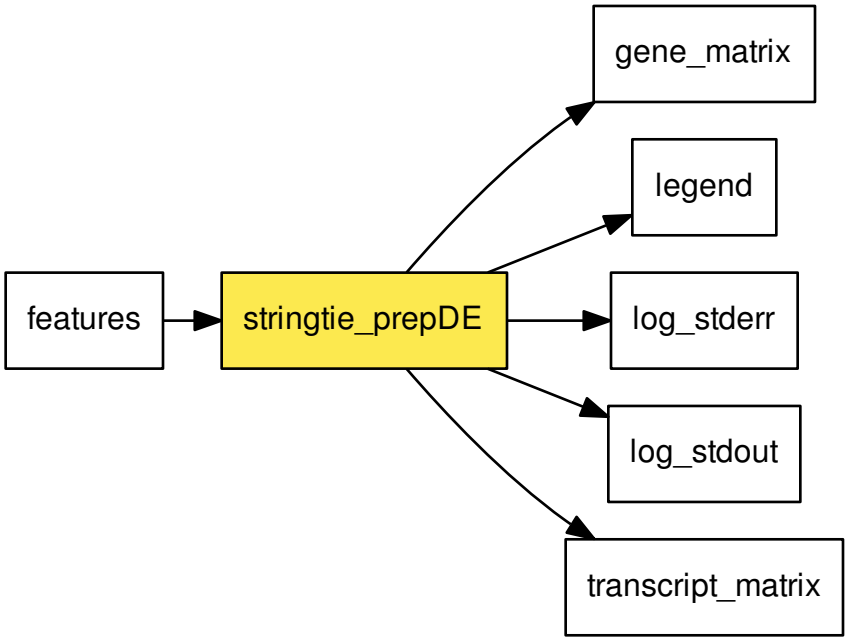
Here, all the transcripts.gtf files from a previous stringtie call are collected, written to a file and provided to the prepDE.py script for conversion (all at once).

NOTE: This step implements the prepDE.py part of stringtie. If you want stringtie to assemble transcripts from multiple BAM files please or merge assemblies use step stringtie or stringtie_merge, resp.!

https://ccb.jhu.edu/software/stringtie/

**Connections:**

- Input Connection:
  - 'in/features'
- Output Connection:
  - 'out/gene_matrix'
  - 'out/legend'
  - 'out/log_stderr'
  - 'out/log_stdout'
  - 'out/transcript_matrix'




**Options:**

- **cluster** (bool, optional) – whether to cluster genes that overlap with different gene IDs, ignoring ones with geneID pattern (see below)
- **key** (str, optional) – if clustering, what prefix to use for geneIDs assigned by this script [default: prepG]
- **length** (int, optional) – the average read length [default: 75]
- **pattern** (str, optional) – a regular expression that selects the sample subdirectories
- **run_id** (str, optional) – A name for the run. Since this step merges multiple samples into a single one, the run_id cannot be the sample name anymore.
  - default value: prepDEall

- **string** (str, optional) – if a different prefix is used for geneIDs assigned by StringTie [default: MSTRG

**Required tools:** prepDE, printf

**CPU Cores:** 6

### subsetMappedReads

subsetMappedReads selects a provided number of mapped reads from a file in .sam or .bam format. Depending on the set options the first N mapped reads and their mates (for paired end sequencing) are returned in .sam format. If the number of requested reads exceeds the number of available mapped reads, all mapped reads are returned.

**Connections:**

- Input Connection:
  - 'in/alignments'
- Output Connection:
  - 'out/alignments'
  - 'out/log'


**Options:**

- **Nreads** (str, required) – Number of reads to extract from input file.
- **genome-faidx** (str, required) - **paired_end** (bool, required) – The reads are expected to have a mate, due to paired end sequencing.

**Required tools:** cat, dd, head, pigz, samtools

**CPU Cores:** 1

### tophat2

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.

http://tophat.cbcb.umd.edu/

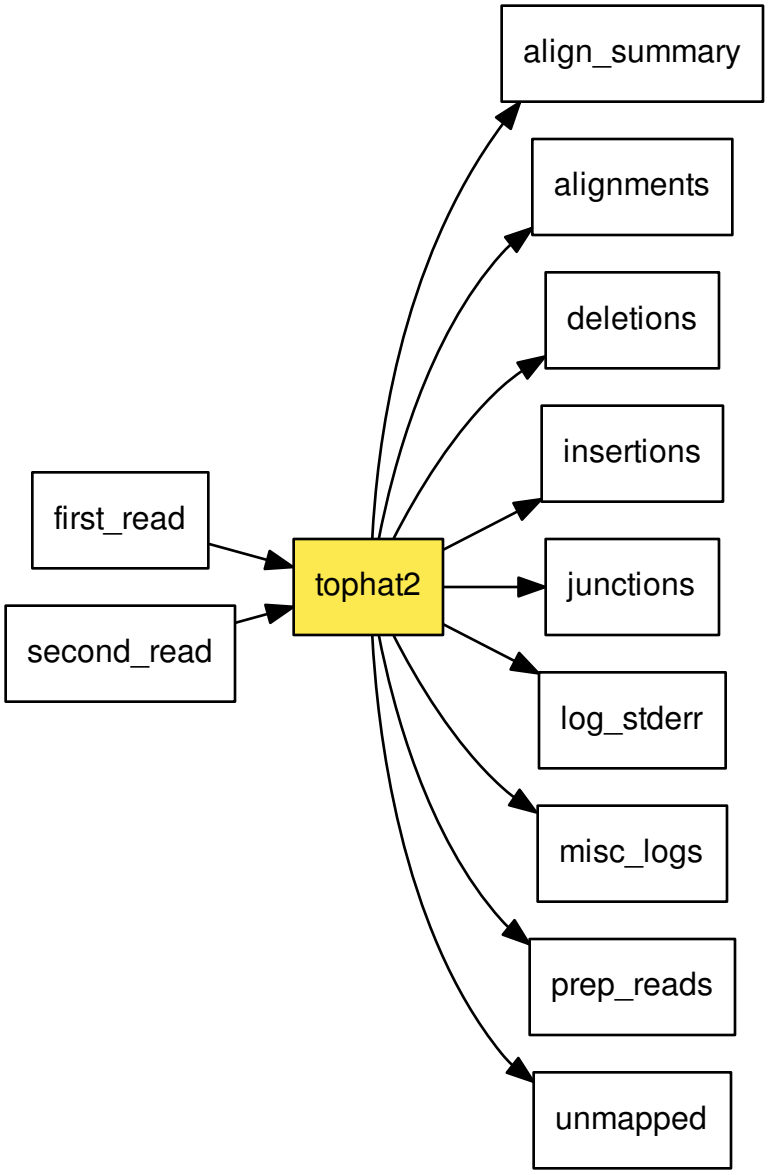
typical command line:

```
tophat [options]* <index_base> <reads1_1[,...,readsN_1]>          [reads1_2,..
↪.readsN_2]
```

Tested on release: TopHat v2.0.13

**Connections:**

- Input Connection:
  - 'in/first_read'
  - 'in/second_read'
- Output Connection:
  - 'out/align_summary'
  - 'out/alignments'
  - 'out/deletions'
  - 'out/insertions'
  - 'out/junctions'
  - 'out/log_stderr'
  - 'out/misc_logs'
  - 'out/prep_reads'
  - 'out/unmapped'




**Options:**

- **bowtie1** (bool, optional) – Use bowtie1. Default: bowtie2
- **color** (bool, optional) – Solid - color space
- **color-out** (bool, optional) – Colored output
- **index** (str, required) – Path to genome index for tophat2
- **integer-quals** (bool, optional) – Provide/Use (?) integer qualities.
- **library_type** (str, required) – The default is unstranded (fr-unstranded). If either fr-firststrand or fr-

secondstrand is specified, every read alignment will have an XS attribute tag as explained below. Consider supplying library type options below to select the correct RNA-seq protocol.(https://ccb.jhu.edu/software/tophat/manual.shtml)

  - possible values: 'fr-unstranded', 'fr-firststrand', 'fr-secondstrand'

- **max-deletion-length** (int, optional) – Max size of deletion
- **max-insertion-length** (int, optional) – Max size of insertion
- **max-intron-length** (int, optional) – maximal intron length
- **max-multihits** (int, optional) – Maximal number of multiple hits
- **min-anchor** (int, optional) – Size of minimal anchor.
- **min-intron-length** (int, optional) – Minimal intron length
- **phred64-quals** (bool, optional) – Qualities are phred64 qualities (same as solexa1.3-quals).
- **prefilter-multihits** (bool, optional) – for -G/–GTF option, enable an initial bowtie search against the genome
- **quals** (bool, optional) – Provide/Use (?) qualities.
- **read-edit-dist** (int, optional) – Read edit distance
- **read-gap-length** (int, optional) – Size of gap length
- **read-mismatches** (int, optional) - **read-realign-edit-dist** (int, optional) – Read alignment distance. Default: read-edit-dist + 1.
- **solexa-quals** (bool, optional) – Qualities are solexa qualities.
- **solexa1.3-quals** (bool, optional) – Qualities are solexa1.3 qualities (same as phred64-quals).
- **splice-mismatches** (int, optional) – Number of splice mismatches
  - possible values: '0', '1', '2'
- **supress-hits** (bool, optional) – Supress hits
- **transcriptome-max-hits** (int, optional) – Max hits in transcriptome

**Required tools:** mkdir, mv, tar, tophat2

**CPU Cores:** 6

### trim_galore

A wrapper tool around Cutadapt and FastQC to consistently apply quality and adapter trimming to FastQ files, with some extra functionality for MspI-digested RRBS-type (Reduced Representation Bisufite-Seq) libraries.

https://www.bioinformatics.babraham.ac.uk/projects/trim_galore/

Note for RRBS using the NuGEN Ovation RRBS System 1-16 kit:

Owing to the fact that the NuGEN Ovation kit attaches a varying number of nucleotides (0-3) after each MspI site Trim Galore should be run WITHOUT the option –rrbs. This trimming is accomplished in a subsequent diversity trimming step afterwards (see their manual).

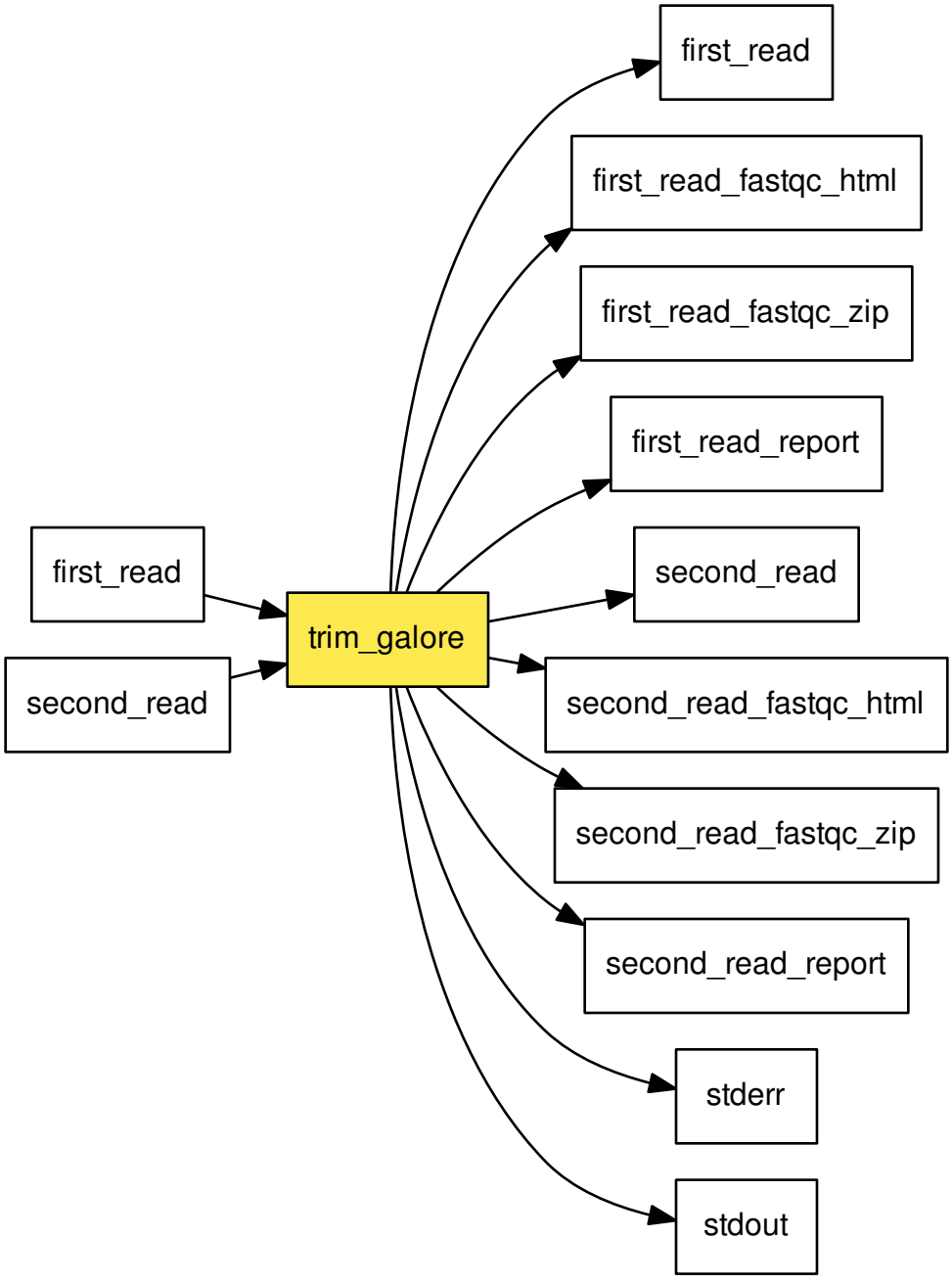
Note for RRBS using MseI:

If your DNA material was digested with MseI (recognition motif: TTAA) instead of MspI it is NOT necessary to specify –rrbs or –non_directional since virtually all reads should start with the sequence

'TAA', and this holds true for both directional and non-directional libraries. As the end-repair of 'TAA' restricted sites does not involve any cytosines it does not need to be treated especially. Instead, simply run Trim Galore! in the standard (i.e. non-RRBS) mode.

**Connections:**

- Input Connection:
  - 'in/first_read'
  - 'in/second_read'
- Output Connection:
  - 'out/first_read'
  - 'out/first_read_fastqc_html'
  - 'out/first_read_fastqc_zip'
  - 'out/first_read_report'
  - 'out/second_read'
  - 'out/second_read_fastqc_html'
  - 'out/second_read_fastqc_zip'
  - 'out/second_read_report'
  - 'out/stderr'
  - 'out/stdout'




**Options:**

- **adapter** (str, optional) – Adapter sequence to be trimmed. If not specified explicitly, Trim Galore willtry to auto-detectwhether the Illumina universal, Nextera transposase or Illuminasmall RNA adapter sequence was used.Also see '–illumina', '–nextera' and '–small_rna'. If no adapter can be detected within thefirst 1 million sequencesof the first file specified Trim Galore defaults to '–illumina'.

- **adapter2** (str, optional) – Optional adapter sequence to be trimmed off read 2 of paired-end files. Thisoption requires'–paired' to be specified as well. If the libraries to be trimmedare smallRNA then a2 will be setto the Illumina small RNA 5' adapter automatically(GATCGTCGGACT).
- **clip_R1** (int, optional) – Instructs Trim Galore to remove <int> bp from the 5' end of read 1 (or single-endreads). This maybe useful if the qualities were very poor, or if there is somesort of unwanted bias at the 5' end.Default: OFF.
- **clip_R2** (int, optional) – Instructs Trim Galore to remove <int> bp from the 5' end of read 2 (paired-end readsonly). Thismay be useful if the qualities were very poor, or if there is some sortof unwanted bias at the 5'end. For paired-end BS-Seq, it is recommended to removethe first few bp because the end-repairreaction may introduce a bias towards lowmethylation. Please refer to the M-bias plot section in theBismark User Guide forsome examples. Default: OFF.
- **dd-blocksize** (str, optional) - default value: 2M
- **dont_gzip** (bool, optional) – Output files won't be compressed with GZIP. This option overrides –gzip.
- **e** (float, optional) – Maximum allowed error rate (no. of errors divided by the length of the matchingregion) (default:0.1)
- **fastqc** (bool, optional) – Run FastQC in the default mode on the FastQ file once trimming is complete.
- **fastqc_args** (str, optional) – Passes extra arguments to FastQC. If more than one argument is to be passedto FastQC they must bein the form "arg1 arg2 etc.". An example would be:–fastqc_args "–nogroup –outdir /home/". Passingextra arguments willautomatically invoke FastQC, so –fastqc does not have to bespecifiedseparately.
- **first_read** (str, optional) – Part of the file name that marks all files containing sequencing data of the first read. Example: '_R1' or '_1'. Default: 'R1'
  - default value: R1
- **gzip** (bool, optional) – Compress the output file with GZIP. If the input files are GZIP-compressedthe output files willautomatically be GZIP compressed as well. As of v0.2.8 the compression will take place on the fly.
- **illumina** (bool, optional) – Adapter sequence to be trimmed is the first 13bp of the Illumina universal adapter'AGATCGGAAGAGC' instead of the default auto-detection of adapter sequence.
- **keep** (bool, optional) – Keep the quality trimmed intermediate file. Default: off, which meansthe temporary file is beingdeleted after adapter trimming. Only hasan effect for RRBS samples since other FastQ files are nottrimmedfor poor qualities separately.
- **length** (int, optional) – Discard reads that became shorter than length INT because of eitherquality or adapter trimming. Avalue of '0' effectively disablesthis behaviour. Default: 20 bp.For paired-end files, both readsof a read-pair need to be longer than<INT> bp to be printed out to validated paired-end files (seeoption –paired).If only one read became too short there is the possibility of keeping suchunpairedsingle-end reads (see –retain_unpaired). Default pair-cutoff: 20 bp.
- **length_1** (int, optional) – Unpaired single-end read length cutoff needed for read 1 to be written to '.unpaired_1.fq 'output file. These reads may be mapped in single-end mode.Default: 35 bp.
- **length_2** (int, optional) – Unpaired single-end read length cutoff needed for read 2 to be written to '.unpaired_2.fq 'output file. These reads may be mapped in single-end mode.Default: 35 bp.
- **max_length** (int, optional) – Discard reads that are longer than <INT> bp after trimming. This is only advised for smallRNAsequencing to remove non-small RNA sequences.
- **max_n** (int, optional) – The total number of Ns (as integer) a read may contain before it will be removed altogether.In apaired-end setting, either read exceeding this limit will result in the entirepair being removedfrom the trimmed output files.

- **nextera** (bool, optional) – Adapter sequence to be trimmed is the first 12bp of the Nextera adapter 'CTGTCTCTTATA' instead ofthe default auto-detection of adapter sequence.
- **non_directional** (bool, optional) – Selecting this option for non-directional RRBS libraries will screenquality-trimmed sequences for'CAA' or 'CGA' at the start of the readand, if found, removes the first two basepairs. Like withthe option '–rrbs' this avoids using cytosine positions that were filled-induring the end-repairstep. '–non_directional' requires '–rrbs' tobe specified as well. Note that this option doesnot set '–clip_r2 2' inpaired-end mode.
- **paired** (bool, optional) – This option performs length trimming of quality/adapter/RRBS trimmed reads forpaired-end files. Topass the validation test, both sequences of a sequence pairare required to have a certain minimumlength which is governed by the option–length (see above). If only one read passes this lengththreshold theother read can be rescued (see option –retain_unpaired). Using this option letsyoudiscard too short read pairs without disturbing the sequence-by-sequence orderof FastQ files whichis required by many aligners.Trim Galore! expects paired-end files to be supplied in a pairwisefashion, e.g.file1_1.fq file1_2.fq SRR2_1.fq.gz SRR2_2.fq.gz ... .
- **phred33** (bool, optional) – Instructs Cutadapt to use ASCII+33 quality scores as Phred scores (Sanger/Illumina 1.9+ encoding)for quality trimming. Default: ON.
- **phred64** (bool, optional) – Instructs Cutadapt to use ASCII+64 quality scores as Phred scores(Illumina 1.5 encoding) forquality trimming.
- **pigz-blocksize** (str, optional) - default value: 2048
- **quality** (int, optional) – Trim low-quality ends from reads in addition to adapter removal. For RRBS samples, quality trimmingwill be performed first, and adaptertrimming is carried in a second round. Other files are qualityand adaptertrimmed in a single pass. The algorithm is the same as the one used by BWA(Subtract INTfrom all qualities; compute partial sums from all indicesto the end of the sequence; cut sequence atthe index at which the sum isminimal). Default Phred score: 20.
- **retain_unpaired** (bool, optional) – If only one of the two paired-end reads became too short, the longer-read will be written to either'.unpaired_1.fq' or '.unpaired_2.fq' output files. The length cutoff for unpaired single-endreads isgoverned by the parameters -r1/–length_1 and -r2/–length_2. Default: OFF.
- **rrbs** (bool, optional) – Specifies that the input file was an MspI digested RRBS sample (recognitionsite: CCGG). Single-endor Read 1 sequences (paired-end) which were adapter-trimmedwill have a further 2 bp removed fromtheir 3' end. Sequences which were merelytrimmed because of poor quality will not be shortenedfurther. Read 2 of paired-endlibraries will in addition have the first 2 bp removed from the 5' end(by setting '–clip_r2 2'). This is to avoid using artificial methylation calls from thefilled-incytosine positions close to the 3' MspI site in sequenced fragments. This option is notrecommended for users of the NuGEN ovation RRBS System 1-16kit (see below).
- **second_read** (str, optional) – Part of the file name that marks all files containing sequencing data of the second read. Example: '_R2' or '_2'. Default: 'R2'
  - default value: R2
- **small_rna** (bool, optional) – Adapter sequence to be trimmed is the first 12bp of the Illumina Small RNA 3' Adapter'TGGAATTCTCGG' instead of the default auto-detection of adapter sequence. Selectingto trimsmallRNA adapters will also lower the –length value to 18bp. If the smallRNAlibraries arepaired-end then a2 will be set to the Illumina small RNA 5' adapterautomatically (GATCGTCGGACT)unless -a 2 had been defined explicitly.
- **stringency** (int, optional) – Overlap with adapter sequence required to trim a sequence. Defaults to avery stringent setting of1, i.e. even a single bp of overlapping sequencewill be trimmed off from the 3' end of any read.
- **three_prime_clip_R1** (int, optional) – Instructs Trim Galore to remove <int> bp from the 3' end of read

1 (or single-endreads) AFTERadapter/quality trimming has been performed. This may remove some unwantedbias from the 3' end thatis not directly related to adapter sequence or basecall quality.Default: OFF.

- **three_prime_clip_R2** (int, optional) – Instructs Trim Galore to remove <int> bp from the 3' end of read 2 AFTERadapter/quality trimminghas been performed. This may remove some unwanted bias fromthe 3' end that is not directly relatedto adapter sequence or basecall quality.Default: OFF.
- **trim-n** (bool, optional) – Removes Ns from either side of the read. This option does currently not work in RRBS mode.
- **trim1** (bool, optional) – Trims 1 bp off every read from its 3' end. This may be needed for FastQ files thatare to bealigned as paired-end data with Bowtie. This is because Bowtie (1) regardsalignments like this: R1————————————> or this: ————————————> R1 R2 <————————————<———————— R2as invalid (whenever a start/end coordinate is contained within the otherread).NOTE: If you are planning to use Bowtie2, BWA etc. you don't need to specify this option.

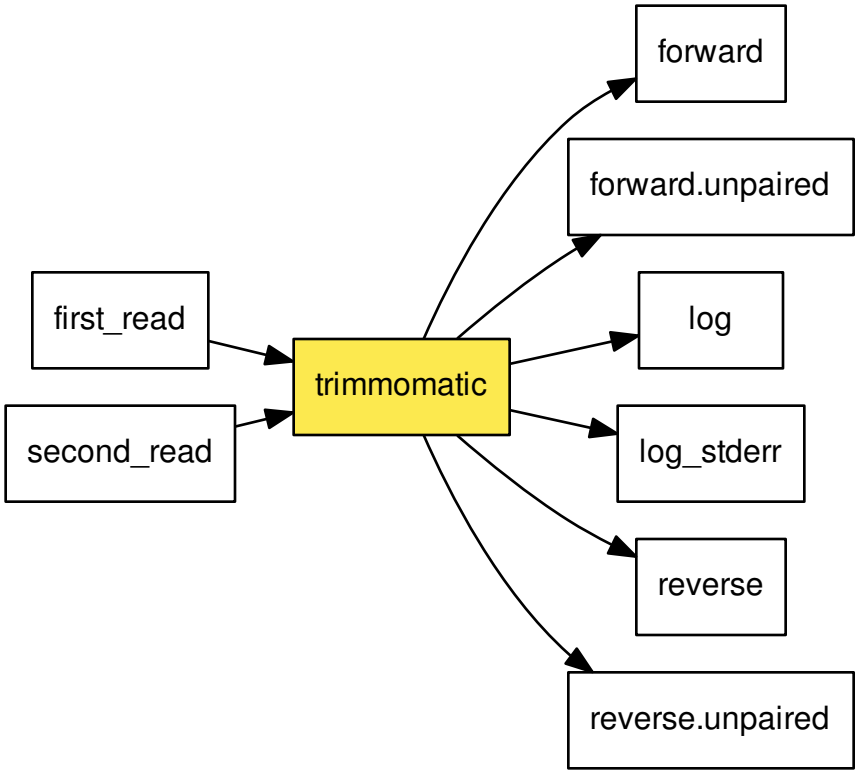
**Required tools:** cutadapt, trim_galore

**CPU Cores:** 4

### trimmomatic

**Connections:**

- Input Connection:
  - 'in/first_read'
  - 'in/second_read'
- Output Connection:
  - 'out/forward'
  - 'out/forward.unpaired'
  - 'out/log'
  - 'out/log_stderr'
  - 'out/reverse'
  - 'out/reverse.unpaired'




**Options:**

- **base_file_name** (int, optional) – The prefix commoon to all output files, replacing the run-ID
- **jar_path** (str, optional) – Path to trimmomatic.jar
  - default value: trimmomatic
- **phred-base** (int, optional) – PHRED-base
  - possible values: '33', '64'
- **steps** (list, required) – List defining the analysis, in terms of trimmomatic-commands
- **threads** (int, optional) – Number of threads to use
  - default value: 1

**Required tools:** java

**CPU Cores:** 1

## 1.13 API documentation

### 1.13.1 Pipeline-specific modules

#### abstract_step

Classes AbstractStep and AbstractSourceStep are defined here.

The class AbstractStep has to be inherited by all processing step classes. The class AbstractSourceStep has to be inherited by all source step classes.

Processing steps generate output files from input files whereas source steps only provide output files. Both step types may generates tasks, but only source steps can introduce files from outside the destination path into the pipeline.

**class** abstract_step.**AbstractStep**(*pipeline*)

**add_connection**(*connection*, *constraints=None*)
Add a connection, which must start with 'in/' or 'out/'.

**add_dependency**(*parent*)
Add a parent step to this steps dependencies.

parent – parent step this step depends on

**add_input_connection**(*connection*, *constraints=None*)
Add an input connection to this step

**add_option**(*key*, **option_types*, ***kwargs*)
Add an option. Multiple types may be specified.

**add_output_connection**(*connection*, *constraints=None*)
Add an output connection to this step

**declare_run**(*run_id*)
Declare a run. Use it like this:

```
with self.declare_run(run_id) as run:
    # add output files and information to the run here
```

**dependencies = None**
All steps this step depends on.

**finalize**()
Finalizes the step.

The intention is to make further changes to the step impossible, but apparently, it's checked nowhere at the moment.

**find_upstream_info_for_input_paths**(*input_paths*, *key*)
Find a piece of public information in all upstream steps. If the information is not found or defined in more than one upstream step, this will crash.

**generate_one_report**()
Gathers the output files for each outgoing connection and calls self.reports() to do the job of creating a report.

**generate_report**(*run_id*)
Gathers the output files for each outgoing connection and calls self.reports() to do the job of creating a report.

**get_cores**()
Returns the number of cores used in this step.

**get_in_connections**()
Return all in-connections for this step

**get_input_runs**()
Return a dict which contains all runs per parent steps.

**get_module_loads**()
Return dictionary with module load commands to execute before starting any other command of this step

**get_module_unloads**()
Return dictionary with module unload commands to execute before starting any other command of this step

**get_option**(*key*)
Query an option.

**get_options**()
Returns a dictionary of all given options

**get_out_connections**()
Return all out-connections for this step

**get_post_commands**()
Return dictionary with commands to execute after finishing any other command of this step

**get_pre_commands**()
Return dictionary with commands to execute before starting any other command of this step

**get_run**(*run_id*)
Returns a single run object for run_id or None.

**get_run_ids**()
Returns sorted list of runs generated by step.

**get_run_ids_in_connections_input_files**()
Return a dictionary with all run IDs from parent steps, the in connections they provide data for, and the names of the files:

```
run_id_1:
    in_connection_1: [input_path_1, input_path_2, ...]
    in_connection_2: ...
run_id_2: ...
```

Format of `in_connection`: `in/<connection>`. Input paths are absolute.

**get_run_ids_out_connections_output_files**()
Return a dictionary with all run IDs of the current step, their out connections, and the files that belong to them:

```
run_id_1:
    in_connection_1: [input_path_1, input_path_2, ...]
    in_connection_2: ...
run_id_2: ...
```

Format of `in_connection`: `in/<connection>`. Input paths are absolute.

**get_run_state**(*run_id*)
Returns run state of a run.

Determine the run state (that is, not *basic* but *extended* run state) of a run, building on the value returned by *get_run_state_basic()*.

**If a run is ready, this will:**

- return **executing** if an up-to-date *executing ping file* is found
- otherwise return **queued** if a *queued ping file* is found

**If a run is waiting, this will:**

- return **queued** if a *queued ping file* is found

Otherwise, it will just return the value obtained from *get_run_state_basic()*.

Attention: The status indicators **executing** and **queued** may be temporarily wrong due to the possiblity of having out-of-date ping files lying around.

**get_run_state_basic**(*run_id*)
Determines basic run state of a run.

Determine the basic run state of a run, which is, at any time, one of **waiting**, **ready**, or **finished**.

These states are determined from the current configuration and the timestamps of result files present in the file system. In addition to these three basic states, there are two additional states which are less reliable (see *get_run_state()*).

**get_runs**()
Getter method for runs of this step.

If there are no runs as this method is called, they are created here.

**classmethod get_step_class_for_key**(*key*)
Returns a step (or source step) class for a given key which corresponds to the name of the module the class is defined in. Pass 'cutadapt' and you will get the cutadapt.Cutadapt class which you may then instantiate.

**get_step_name**()
Returns this steps name.

Returns the step name which is initially equal to the step type (== module name) but can be changed via set_step_name() or via the YAML configuration.

**get_step_type**()
Returns the original step name (== module name).

**get_tool**(*key*)
Return full path to a configured tool.

**is_option_set_in_config**(*key*)
Determine whether an optional option (that is, a non-required option) has been set in the configuration.

**reports**(*run_id*, *out_connection_output_files*)
Abstract method this must be implemented by actual step.

Raise NotImplementedError if subclass does not override this method.

**require_tool**(*tool*)
Declare that this step requires an external tool. Query it later with *get_tool()*.

**run**(*run_id*)
Create a temporary output directory and execute a run. After the run has finished, it is checked that all output files are in place and the output files are moved to the final output location. Finally, YAML annotations are written.

**runs**(*run_ids_connections_files*)
Abstract method this must be implemented by actual step.

Raise NotImplementedError if subclass does not override this method.

**set_cores**(*cores*)
Specify the number of CPU cores this step will use.

**set_options**(*options*)
Checks and stores step options.

The options are either set to values given in YAML config or the default values set in self.add_option().

**set_step_name**(*step_name*)
Change the step name.

The step name is initially set to the module name. This method is used in case we need multiple steps of the same kind.

**class** abstract_step.**AbstractSourceStep**(*pipeline*)
A subclass all source steps inherit from and which distinguishes source steps from all real processing steps because they do not yield any tasks, because their "output files" are in fact files which are already there.

Note that the name might be a bit misleading because this class only applies to source steps which 'serve' existing files. A step which has no input but produces input data for other steps and actually has to do something for it, on the other hand, would be a normal AbstractStep subclass because it produces tasks.

### pipeline

**exception** pipeline.**ConfigurationException**(*value*)

**class** pipeline.**Pipeline**(***kwargs*)
The Pipeline class represents the entire processing pipeline which is defined and configured via the configuration file config.yaml.

Individual steps may be defined in a tree, and their combination with samples as generated by one or more source leads to an array of tasks.

**all_tasks_topologically_sorted = None**
List of all tasks in topological order.

**check_tools**()
checks whether all tools references by the configuration are available and records their versions as determined by `[tool] --version` etc.

**cluster_type = None**
The cluster type to be used (must be one of the keys specified in cluster_config).

**config = None**
Dictionary representation of configuration YAML file.

**file_dependencies = None**
This dict stores file dependencies within this pipeline, but regardless of step, output file tag or run ID. This dict has, for all output files generated by the pipeline, a set of input files that output file depends on.

**file_dependencies_reverse = None**
This dict stores file dependencies within this pipeline, but regardless of step, output file tag or run ID. This dict has, for all input files required by the pipeline, a set of output files which are generated using this input file.

**input_files_for_task_id = None**
This dict stores a set of input files for every task id in the pipeline.

**output_files_for_task_id = None**
This dict stores a set of output files for every task id in the pipeline.

**states = Enum(['READY', 'EXECUTING', 'WAITING', 'QUEUED', 'FINISHED'])**
Possible states a task can be in.

**steps = None**
This dict stores step objects by their name. Each step knows his dependencies.

**task_for_task_id = None**
This dict stores task objects by task IDs.

**task_id_for_output_file = None**
This dict stores a task ID for every output file created by the pipeline.

**task_ids_for_input_file = None**
This dict stores a set of task IDs for every input file used in the pipeline.

**topological_step_order = None**
List with topologically ordered steps.

### run

**class** run.**Run**(*step*, *run_id*)
The Run class is a helper class which represents a run in a step. Declare runs inside AbstractStep.runs() via:

```
with self.new_run(run_id) as run:
    # declare output files, private and public info here
```

After that, use the available methods to configure the run. The run has typically no information about input connections only about input files.

**add_empty_output_connection**(*tag*)
An empty output connection has 'None' as output file and 'None' as input file.

**add_output_file**(*tag*, *out_path*, *in_paths*)
Add an output file to this run. Output file names must be unique across all runs defined by a step, so it may be a good idea to include the run_id into the output filename.

- ***tag*: You must specify the connection annotation which must have been** previously declared via *AbstractStep.add_connection("out/...")*, but this doesn't have to be done in the step constructor, it's also possible in *declare_runs()* right before this method is called.
- ***out_path*: The output file path, without a directory. The pipeline** assigns directories for you (this parameter must not contain a slash).
- ***in_paths*: A list of input files this output file depends on. It is crucial** to get this right, so that the pipeline can determine which steps are up-to-date at any given time. You have to specify absolute paths here, including a directory, and you can obtain them via *AbstractStep.run_ids_and_input_files_for_connection* and related functions.

**add_private_info**(*key*, *value*)
Add private information to a run. Use this to store data which you will need when the run is executed. As opposed to public information, private information is not visible to subsequent steps.

You can store paths to input files here, but not paths to output files as their expected location is not defined until we're in *AbstractStep.execute* (hint: they get written to a temporary directory inside *execute()*).

**add_public_info**(*key*, *value*)
Add public information to a run. For example, a FASTQ reader may store the index barcode here for subsequent steps to query via `AbstractStep.find_upstream_info()`.

**add_temporary_directory**(*prefix=''*, *suffix=''*, *designation=None*)
Convenience method for creation of temporary directories. Basically, just calls self.add_temporary_file(). The magic happens in ProcessPool.__exit__()

**get_basic_state**()
Determines basic run state of a run.

Determine the basic run state of a run, which is, at any time, one of **waiting**, **ready**, or **finished**.

These states are determined from the current configuration and the timestamps of result files present in the file system. In addition to these three basic states, there are two additional states which are less reliable (see *get_run_state()*).

**get_execution_hashtag**()
Creates a hash tag based on the commands to be executed.

This causes runs to be marked for rerunning if the commands to be executed change.

**get_input_files_for_output_file**(*out_path*)
Return all input files a given output file depends on.

**get_output_directory**()
Returns the final output directory.

**get_output_directory_du_jour**()
Returns the state-dependent output directory of the step.

Returns this steps output directory according to its current state: - if we are currently calling a step's declare_runs() method, this will return None - if we are currently calling a step's execute() method, this will return the temporary directory - otherwise, it will return the real output directory

**get_output_directory_du_jour_placeholder**()
Returns a placeholder for the temporary output directory, which needs to be replaced by the actual temp directory inside the abstract_step.execute() method

**get_output_files_abspath**()
Return a dictionary of all defined output files, grouped by connection annotation:

```
annotation_1:
    out_path_1: [in_path_1, in_path_2, ...]
    out_path_2: ...
annotation_2: ...
```

The `out_path` consists of the output directory du jour and the output file name.

**get_output_files_for_annotation_and_tags**(*annotation*, *tags*)
Retrieve a set of output files of the given annotation, assigned to the same number of specified tags. If you have two 'alignment' output files and they are called *out-a.txt* and *out-b.txt*, you can use this function like this:

- *tags*: ['a', 'b']
- result: {'a': 'out-a.txt', 'b': 'out-b.txt'}

**get_private_info**(*key*)
Query private information which must have been previously stored via " "*add_private_info()*.

**get_public_info**(*key*)
Query public information which must have been previously stored via " "*add_public_info()*.

**get_single_output_file_for_annotation**(*annotation*)
Retrieve exactly one output file of the given annotation, and crash if there isn't exactly one.

**get_temp_output_directory**()
Returns the temporary output directory of a run.

**has_private_info**(*key*)
Query whether a piece of public information has been defined.

**has_public_info**(*key*)
Query whether a piece of public information has been defined.

**remove_temporary_paths**()
Everything stored in self._temp_paths is examined and deleted if possible. The list elements are removed in LIFO order. Also, self._known_paths 'type' info is updated here. NOTE: Included additional stat checks to detect FIFOs as well as other special files.

**update_public_info**(*key*, *value*)
Update public information already existing in a run. For example, all steps which handle FASTQ files want to know how to distinguish between files of read 1 and files of read 2. So each step that provides FASTQ should update this information if the file names are altered. The stored information can be acquired via: `AbstractStep.find_upstream_info()`.

**write_annotation_file**(*path*)
Write the YAML annotation after a successful or failed run. The annotation can later be used to render the process graph.

### task

**class** task.**Task**(*pipeline*, *step*, *run_id*, *run_index*)
A task represents a certain run of a certain step.

**get_parent_tasks**()
Returns a list of parent tasks which this task depends on.

**get_pipeline**()
Returns the pipeline this task belongs to.

**get_run**()
Returns the run object for this task.

**get_step**()
Returns the step of this task.

**get_task_state**()
Proxy method for step.get_run_state().

**get_task_state_basic**()
Proxy method for step.get_run_state().

**input_files**()
Return a list of input files required by this task.

**output_files**()
Return a list of output files produced by this task.

**run**()
Run the task. Skip if it's already finished. Raise Exception if it's not ready.

### 1.13.2 Miscellaneous modules

#### process_pool

This module can be used to launch child processes and wait for them. Processes may either run on their own or pipelines can be built with them.

**class** process_pool.**ProcessPool**(*run*)

The process pool provides an environment for launching and monitoring processes. You can launch any number of unrelated processes plus any number of pipelines in which several processes are chained together.

Use it like this:

```
with process_pool.ProcessPool(self) as pool:
    # launch processes or create pipelines here
```

When the scope opened by the *with* statement is left, all processes are launched and being watched. The process pool then waits until all processes have finished. You cannot launch a process pool within another process pool, but you can launch multiple pipeline and independent processes within a single process pool. Also, you can launch several process pools sequentially.

**COPY_BLOCK_SIZE = 4194304**

When *stdout* or *stderr* streams should be written to output files, this is the buffer size which is used for writing.

**class Pipeline**(*pool*)

This class can be used to chain multiple processes together.

Use it like this:

```
with pool.Pipeline(pool) as pipeline:
    # append processes to the pipeline here
```

**append**(*args*, *stdout_path=None*, *stderr_path=None*, *hints={}*)

Append a process to the pipeline. Parameters get stored and are passed to *ProcessPool.launch()* later, so the same behaviour applies.

**SIGTERM_TIMEOUT = 10**

After a SIGTERM signal is issued, wait this many seconds before going postal.

**TAIL_LENGTH = 1024**

Size of the tail which gets recorded from both *stdout* and *stderr* streams of every process launched with this class, in bytes.

**get_log**()

Return the log as a dictionary.

**classmethod kill**()

Kills all user-launched processes. After that, the remaining process will end and a report will be written.

**classmethod kill_all_child_processes**()

Kill all child processes of this process by sending a SIGTERM to each of them. This includes all children which were not launched by this module, and their children etc.

**launch**(*args*, *stdout_path=None*, *stderr_path=None*, *hints={}*)

Launch a process. Arguments, including the program itself, are passed in *args*. If the program is not a binary but a script which cannot be invoked directly from the command line, the first element of *args* must be a list like this: *['python', 'script.py']*.

Use *stdout_path* and *stderr_path* to redirect *stdout* and *stderr* streams to files. In any case, the output of both streams gets watched, the process pool calculates SHA1 checksums automatically and also keeps the

last 1024 bytes of every stream. This may be useful if a process crashes and writes error messages to *stderr* in which case you can see them even if you didn't redirect *stderr* to a log file.

Hints can be specified but are not essential. They help to determine the direction of arrows for the run annotation graphs rendered by GraphViz (sometimes, it's not clear from the command line whether a certain file is an input or output file to a given process).

**log**(*message*)
Append a message to the pipeline log.

**exception** process_pool.**TimeoutException**

### fscache

**class** fscache.**FSCache**
Use this class if you expect to make the same os.path.* calls many times during a short time. The first time you call a method with certain arguments, the call is made, but all subsequent calls are served from a cache.

Usage example:

```
# Instantiate a new file system cache.
fsc = FSCache()

# This call will stat the file system.
print(fsc.exists('/home'))

# This call will leave the file system alone, the cached result will be returned.
print(fsc.exists('/home'))
```

You may call any method which is available in os.path.

### misc

**class** misc.**Enum**(*_list*)

misc.**append_suffix_to_path**(*path*, *suffix*)
Append a suffix to a path, for example:

- path: /home/michael/chocolate-cookies.txt.gz
- suffix: done right
- result: /home/michael/chocolate-cookies-done-right.txt.gz

misc.**assign_strings**(*paths*, *tags*)
Assign N strings (path names, for example) to N tags. Example:

- paths = ['RIB0000794-cutadapt-R1.fastq.gz', 'RIB0000794-cutadapt-R2.fastq.gz']
- tags = ['R1', 'R2']
- result = { 'R1': 'RIB0000794-cutadapt-R1.fastq.gz', 'R2': 'RIB0000794-cutadapt-R2.fastq.gz' }

If this is not possible without ambiguities, a StandardError is thrown. Attention: The number of paths must be equal to the number of tags, a 1:1 relation is returned, if possible.

misc.**bytes_to_str**(*num*)
Convert a number representing a number of bytes into a human-readable string such as "4.7 GB"

misc.**duration_to_str**(*duration*, *long=False*)
Minor adjustment for Python's duration to string conversion, removed microsecond accuracy and replaces 'days' with 'd'

misc.**natsorted**(*l*)
Return a 'naturally sorted' permutation of l.

Credits: http://www.codinghorror.com/blog/2007/12/sorting-for-humans-natural-sort-order.html

## 1.14 Information for uap Developers

### 1.14.1 Documentation

The official documentation is hosted on readthedocs (add link). But as developer you need to be able to create the documentation locally. So let's focus at first on:

#### Create Documentation Locally

For testing purposes its necessary to be able to build the documentation on your local machine. Follow the instructions below.

#### Prerequisites

Before the documentation can be build, we need to install some packages and libraries.

The documentation is build with Sphinx, so install it (for Ubuntu):

```
$ sudo aptitude install python-sphinx
```

The uap documentation requires the `psutil` python library to be available in your global python environment:

```
$ sudo pip install psutil
```

It also requires the readthedocs theme installed (for Ubuntu):

```
$ sudo aptitude install python-sphinx-rtd-theme sphinx-rtd-theme-common
```

#### Build Documentation

The documentation can be build by:

```
$ cd doc
$ make html
```

This should do the trick.

#### Create Documentation on RTD

Commit your changes to one of the branches that are configured for automatic builds via RTD. Push your commits to the yigbt-repository. This should do the trick.

If you are not allowed to directly push your commits. Please provide a pull request via github. This should do the trick as well.

### 1.14.2 Continuous Integration

At the moment we use Travis CI as continuous integration platform. The goal is to test as many step use cases as possible. Travis installs uap in a clean Docker image. Installation instructions for software that needs to be installed additionally, needs to be provided you want to install

#### .travis.yml

This file contains the test matrix, the commands executed before installing uap, the commands executed to install uap and the commands which are executed to perform the tests you are interested in.

At the moment most of the subcommands are tested on Travis CI using a dummy workflow.

#### travis_uap_config.yaml

The dummy workflow we use to test steps is defined in the file `example-configurations/travis-example/travis_uap_config.yaml`. So if you create a new step, please add it to this configuration file. Provide as many step configurations as you think are appropriate.

CHAPTER 2

# Remarks

This documentation has been created using and .

CHAPTER 3

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

# Index

## G



## H



## I



## K



## L



## M



## N



## O



## P

## R



## S



## T



## U



## W