Documentation for

ATAC-Seq pipeline

This document provides documentation and additional information for the the ATAC-Seq pipeline developed by Christian Arnold.

If you have questions or comments, feel free to contact us. We will be happy to answer any questions related to the software implementation. For questions, contact Christian Arnold (christian.arnold@embl.de).

If you use this software, please acknowledge the project.

This documentation may lack details for recently added features. Contact Christian if you are missing information.

Last update: 08/24/20

Table of Contents

1. Quick Start

I here describe in detail how to run our *ATAC-Seq* pipeline for proper ATAC-Seq preprocessing within the EMBL universe. If you want to run the pipeline in a different context, modifications are necessary. Contact me in this case. The pipeline is also available in the following repository: https://git.embl.de/carnold/ATACSeq_Snakemake

The following quick start briefly summarizes the necessary steps to use the pipeline:

1. Make sure you use at least Snakemake version 5.9.1 (type snakemake --version to find out). Either install it on your own via conda or use the module system on the cluster (module load snakemake). We strongly recommend running Snakemake with Singularity (i.e., a containerization approach), so you don't have to install the other required tools on your own. Instead, preconfigured containers are used throughout the pipeline with all the required tools already installed. To activate singularity, simply make sure singularity is activated in the params.sh file.
2. Create a new root folder for the analysis within `/scratch` (we are not supposed to run analysis via the `/g` filesystem), and create the two subfolders `input` and `output`.
3. Either copy your input files to the `input` directory or a subdirectory within `input` (like `data`) or at least have them ready on /scratch.
4. Copy the Snakemake-related files and templates that are needed to run the pipeline to the `input` directory. You can always find the newest versions in `example/input` or if you need another organism you may check also `example/templates`. Files in this folder are species-independent, species-specific ones with preset parameters are in the corresponding subfolders. You should have the following 5 files in the `input` directory: `config.json`, `params.sh`, `runSnakefile.sh`, `samples.csv` and `cluster.json`.[1]
5. Modify the following files and adjust them to your analysis:
   a) *cluster.json*: Modify the `output` and `error` path, the rest should be ok to leave as it is. They appear multiple times in the file, make sure to edit them all.
   b) *params.sh*:

1 Except for *`params.sh`* (the name of which can be adjusted in the *`runSnakefile.sh`* file), the exact names of these five files do not matter as long as they are correctly specified in the corresponding files in which they are defined. Feel free to modify them according to your needs. The reason for such a, at first sight, complicated setup is that it greatly facilitates managing and altering parameters and clearly separating analysis-associated and Snakemake-associated parameters. You will quickly realize that it becomes very easy to work with this setup, as is it identical across analyses and very flexible, thereby also increasing reproducibility and decreasing development time.

I. As a general note, do not (!) introduce spaces between the parameter name and the "=" and the assignment). All parameters not mentioned below have been set to reasonable values and there is usually no need to modify them unless you have special requirements such as rerunning (parts of) the pipeline, debugging, etc. For explanations of the other parameters, see the sections below.

II. The Snakefile you usually want to use it (the default): `/g/scb2/zaugg/carnold/Projects/AtacSeq/src/Snakemake/Snakefile` or `src/Snakemake/Snakefile` if you clone the Git repository.

III. Make sure and `dryRun` and `useSingularity` are set to `true` while `useConda` is set to `false`

IV. Adjust and verify the cluster settings

c) config.json

I. Adjust the output section and decide which output files you are interested in. You can save a lot of computing time by skipping various steps in the pipeline.

II. Adjust the *summaryFile* in the *samples* section

III. The parameters *knownSNPs* and *knownIndels* are only needed for human data and can be ignored otherwise (set them to NA to make it clear).

IV. Verify that the parameters corresponding to the genome version are correct (mostly in the section *additionalInput*).

d) samples.csv: Use the provided template and adjust accordingly. See the section below for details. Make sure to use ONLY absolute paths for file names, relative paths will result in errors in the beginning of the pipeline.

6. Quickly think about whether how to execute Snakemake (i.e. interactively or not). I recommend option 2.

   a) Execute Snakemake simply on the terminal. This is the easiest approach, but with one important disadvantage: If you set your PC in sleep mode or shutdown, the analysis is also interrupted.

   b) Use a method to allow Snakemake to run in the background independent of the state of your PC. The easiest approach is *screen* or something equivalent. It is easy to learn and preinstalled on most systems.

7. Start or rerun the analysis.

a) Login to login-gui02.cluster.embl.de, navigate to the input folder, and type `sh runSnakefile.sh`. Make sure to do a dry-run first. Snakemake should now start and proceed with the pipeline after a minute or so of building the DAG.

b) Adjust the settings in the file `params.sh`, set dryrun to false and submitToCluster=true. Restart with `sh runSnakefile.sh`.

c) If you see red messages popping up, errors occurred and you should check what went wrong. See the `LOGS_AND_BENCHMARKS` directory within the specified output directory (`config.json`) for details.

2. Detailed explanations

We now describe the various files that need to be present to start any of the analyses pipeline we describe in detail.

2.1. Cluster configuration file for the analysis (cluster.json)

This file in JSON format is only needed if you run the analysis in a cluster environment. If you run an analysis locally, you can ignore this file.

Currently, our wrapper script supports only SLURM (see the `params.sh` section for details). We cannot explain all technical details here, please check the Snakemake documentation and our examples for orientation. The provided `cluster.json` templates for each pipeline work flawlessly for us and should also work for you. If your cluster system is not SLURM, the cluster config has to be modified, which should however be quick and easy because the same principle applies for other systems as well and Snakemake makes it already easy to abstract from the specific architecture. The values in the `__default__` section apply to all rules unless overwritten explicitly below the default section.

For SLURM, the file has to contain the following parameters in the *`__default__`* section:

- `queueSLURM`: the queue name where jobs should be transmitted to. Default: htc.
- `group`: the name of the group you are in (optional)
- `name`: an arbitrary name for each job. Wildcards are supported. The default name we use us `analysisName.{rule}.{wildcards}`. Replace `analysisName` by any name of your choice
- `memory`: requested default memory. We suggest a default of at least 30000. Memory requirements for the various rules have been set generously so that it should also work for big datasets, but this can never be insured of course. If you receive out of memory errors, adjust the values accordingly.
- `maxTime`: The maximum allowed running time. Depending on the complexity of the analyses, individual steps may need multiple hours to finish.
- `qos`: quality of service. Default “normal”. Only change if you know what you are doing.

- `nodes`: Only consider nodes with a particular architecture. We recommend leaving this at the default `avx2` to avoid “Illegal operation” errors.
- `nCPUs`: Leave untouched. Default "{threads}"
- `output` and `error`: path to the output and error file, respectively. Adjust the output directory accordingly as specified in the file `config.json` and **leave the trailing `LOGS_AND_BENCHMARKS/{rule}.out/err` part unchanged**

The wrapper script transforms these values into the following syntax that is passed on to Snakemake: *sbatch -p {cluster.queueSLURM} -J {cluster.name} -A {cluster.group} --cpus-per-task {cluster.nCPUs} -C {cluster.nodes} --time {cluster.maxTime} –qos={cluster.qos} --mem {cluster.memory} -o \"{cluster.output}\" -e \"{cluster.error}\" --mail-type NONE \" --parsable"*

Feel free to extend this to suit your needs by adding parameters to the cluster config and editing the wrapper script. See also https://slurm.schedmd.com/sbatch.html for details.

2.2. Parameters for calling Snakemake (params.sh)

This file defines Snakemake-specific parameters only, all of which are explained in the following. Parameters in **bold** are the most important ones and the ones that usually have to be modified, while non-bold indicates parameters that only have to be touched in particular cases. Parameters in gray are either obselete or do not have to be touched unless there is a very specific reason.

| configFile |

String. Default “config.json”. Corresponds to --*configfile* in Snakemake.
Path to the config file. Since the config file is usually located in the same directory, this can be left untouched unless it has been renamed or moved.

| snakefile |

String. Default “Snakefile”. Path to the Snakefile. Corresponds to --*snakefile* in Snakemake.
The Snakefile to run.

| nCores |

Integer > 0. Default 5. Corresponds to --*cores* in Snakemake.

Maximum number of CPUs per rule (if the rule supports parallel execution). Make to set this to reasonable values. If Snakemake is executed in a cluster environment, this value is currently force set to 16. Only relevant for rules with threads directive values of > 1.

| dryRun |

Logical. TRUE or FALSE. Default TRUE. Corresponds to --*dryrun* in Snakemake.
Run in dry mode without actually computing something? See the Snakemake help for --*dryrun for* more details.

| submitToCluster |

Logical. TRUE or FALSE. Default FALSE. Corresponds to --*cluster* in Snakemake.
Execute Snakemake rules with the given submit command e.g. bsub or SLURM? Set to TRUE to run the analysis on the cluster. If set to TRUE, a cluster config file (cluster.json) has to be specified. See the Snakemake help for --*cluster* and the corresponding section for cluster.json above for more details.

| clusterConfig |

String. Path to the cluster config. Default “cluster.json”. Only relevant if *submitToCluster* is set to TRUE.
Path to the cluster config file (relative path suffices usually) that defines the wildcards used in 'cluster' for specific rules, instead of having them specified in the Snakefile. See the cluster config section for more details.

| useConda |

Logical. TRUE or FALSE. Default FALSE. Corresponds to --*use-conda* in Snakemake.
Should Snakemake use conda and run each rule with a conda directive in its own isolated environment? See the Snakemake help for --*use-conda for* more details.

| condaDir |

String. Path to any directory. Default "`/g/scb2/zaugg/zaugg_shared/Programs/Snakemake/conda`". Corresponds to --*conda-prefix* in Snakemake.

Specify a directory in which the 'conda' and 'conda-archive' directories are created. These are used to store conda environments and their archives, respectively. If not supplied, the value is set to the '.snakemake' directory relative to the invocation directory. See the Snakemake help for --*conda-prefix for* more details.

| forceRerunAll |

Logical. TRUE or FALSE. Default FALSE. Corresponds to --*forceall* in Snakemake.
Forces to rerun the whole pipeline even if the output files are already present. See the Snakemake help for --*forceall for* more details.

| ignoreTemp |

Logical. TRUE or FALSE. Default TRUE. Corresponds to --*notemp* in Snakemake.
If set to TRUE, temporary files (as declared by the temp directive) are not deleted. This might be a reasonably good idea in the beginning to check if the pipeline runs through. Also, this is useful when running only a part of the workflow since temp() may lead to deletion of files required by other parts of the workflow. To save disk space, set to FALSE. Advanced option. See the Snakemake help for --*notemp for* more details.

| touchOutputFiles |

Logical. TRUE or FALSE. Default FALSE.
Don't run Snakemake, just update the time stamps for output files. This is sometimes useful when you manually mess with the timestamps of the output files outside of Snakemake. Advanced option.

| allowedRules |

String. Name of rule(s), separated by space. Default "" (empty string): option disabled. Corresponds to --*allowed-rules* in Snakemake.
Which rules are allowed to run? If you want to restrict which rules are allowed to run, name the rules as they appear in the Snakefile, separated by spaces. Advanced option. See the Snakemake help for –*allowed-rules for* more details.

| runSpecificRule |

String. Name of rule. Default "" (empty string): option disabled. Corresponds to --*forcerun* in Snakemake.
Force the re-execution or creation of the given rules or files. Name the rule as it appears in the Snakefile. Advanced option. See the Snakemake help for *–forcerun for* more details.

| rerunIncomplete |

Logical. TRUE or FALSE. Default TRUE. Corresponds to *–rerun-incomplete* in Snakemake.
Re-run all jobs the output of which is recognized as incomplete. See the Snakemake help for *–rerun-incomplete for* more details.

| runAlsoDownstreamRules |

Logical. TRUE or FALSE. Default TRUE. Only relevant if *runSpecificRule* is set to a non-default value.
If set to FALSE, ONLY the rule as specified in *runSpecificRule* will be run (internally, the --*until* option is appended after --*forcerun* with the same rule); otherwise, all downstream rules are also triggered. Advanced option. See the Snakemake help for --*until* for more details.

| useSLURM |

Logical. TRUE or FALSE. Default FALSE.
If set to TRUE, preconfigures Snakemake for SLURM systems using *sbatch*. If set to FALSE, preconfigures Snakemake for LSF systems using *bsub*. See the Snakemake help for --*cluster* for more details. If SLURM is used, make sure to adjust the cluster.json accordingly.

| maxJobsCluster |

Integer > 0 & < 500. Default 200. Only relevant if *submitToCluster* is set to TRUE. Corresponds to --*jobs* in Snakemake.
The maximum number of simultaneous jobs that are allowed to run. Be careful with this option and do not set it to higher values unless you know the IO and general performance of the underlying system is not noticeably influenced. See the Snakemake help for -- *jobs* for more details.

| maxRestartsPerJob |

Integer >= 0. Default 0. Corresponds to --*restart-times* in Snakemake.
The number of times a job that fails should be re-executed. Increase to higher values for unstable cluster systems (we recommend 0, the cluster is stable enough that this is usually not necessary) .
See the Snakemake help for -- *restart-times* for more details.

| useVerbose |
Logical. TRUE or FALSE. Default FALSE. Corresponds to *–verbose and --printshellcmds* in Snakemake.
Print debugging output. If set to TRUE, also --*printshellcmds* is added.

| workflowGraphFileType |
String. Either “pdf” or “svg” or ”png”. Default “pdf”.
Ignored if skipSummaryAndDAG is set to TRUE (the default). You can usually leave this untouched unless you also suffer from the "bug" that the program dot has a problem producing PDF files (then change to "svg" or "png")

2.3. General configuration file for the analysis (config.json)

Section "output"

| outdir |

STRING. Absolute path to the output directory. Will be created if not yet present.

| doBaseRecalibration |

BOOLEAN. “true” or “false”. Default “false”. Should base qualities be recalibrated using *GATK* to detect and correct systematic errors in base quality scores? This step can be time-consuming and needs the following other parameters: *GATK_jar, knownSNPs*, *knownIndels*.

| doPeakCalling |

BOOLEAN. “true” or “false”. Default “true”. Should the pipeline use MACS2 to call peaks? If yes, peaks will be called in 3 different flavors (ENCODE, stringent, non-stringent). See the section “par_peakCalling” in the configuration file for details.

| alsoMergeReplicates |

BOOLEAN. “true” or “false”. Default “true”. Should, in addition to treating all input files separately, the pipeline merge all replicate files and do all downstream analysis in addition? If set to true, for each individual as specified in the sample table, all samples belonging to this individual ID will be merged to one file after the post-processing.

| correctGCBias |

BOOLEAN. “true” or “false”. Default “true”. Should GC bias be assessed and corrected for the final BAM files?

If set to true, the GC bias will be assessed and corrected using deepTools. Additionally, all downstream steps of the pipeline will be done for both GC-corrected and original files (including peak calling, PCA, coverage, etc).

| generateCoverageFiles |

BOOLEAN. “true” or “false”. Default “true”. Should the pipeline produce coverage files and diagnostic plots?

If set to true, coverage files for the final BAM files in bigwig and bedgraph.gz format will be produced as well as a coverage plot using *deepTools*.

| doIDR |

BOOLEAN. “true” or “false”. Default “true”.

Section "general"

| maxCoresPerRule |

INTEGER. Default 12. Maximum number of cores per rule. For local computation, the minimum of this value and the --cores parameter will define the number of CPUs per rule, while in a cluster setting, the minimum of this value and the number of cores on the node the jobs runs is used.

Section "samples"

| summaryFile |

STRING. No default. Absolute path to the sample summary file. See section 2.4 for details.

| pairedEnd |

BOOLEAN. “true” or “false”. Default “true”. Paired-end data?

<u>Single-end ATAC-Seq data is not yet supported with this pipeline. If set to "false", the Snakemake pipeline will abort in the beginning.</u>

Section "additionalInput"

| scriptsDir |

STRING. Absolute path to the scripts directory. Default "/g/scb2/zaugg/carnold/Projects/AtacSeq/src/Snakemake/src"

| trimmomatic_adapters |

STRING. Absolute path to the adapters file for Trimmomatic in fasta format. Default “/g/scb2/zaugg/zaugg_shared/Programs/Trimmomatic-0.33/adapters/NexteraPE-PE.fa”.

There is usually no need to change this unless for your experiment, this adapter file is not suited.

| blacklistRegions |

STRING. Absolute path to a BED file that contains the genomic regions that should be filtered from the peaks. The default depends on the genome assembly, see the templates for details.

<u>Only needed if *doPeakCalling* is set to true.</u>

| knownSNPs | and | knownIndels |

STRING. The default depends on the genome assembly, see the templates for details. Absolute path to a database of known polymorphic sites (SNPs and Indels, respectively). Supported formats from GATK: BCF2, BEAGLE, BED, BEDTABLE, EXAMPLEBINARY, GELITEXT, RAWHAPMAP, REFSEQ, SAMPILEUP, SAMREAD, TABLE, VCF, VCF3.

<u>This is only needed if (1) *doBaseRecalibration* is set to true and (2) the genome is either hg19 or hg38 and ignored otherwise.</u>

| refGenome_fasta | and | refGenome_2bit |

STRING. The default depends on the genome assembly, see the templates for details. Absolute path to the reference genome in fasta and 2bit format, respectively, both of which have to correspond to the same genome assembly version as used for the alignment as well as the database of polymorphic sites (knownSNPs and knownIndels, if applicable).

| annotationGTF |

STRING. The default depends on the genome assembly, see the templates for details. Absolute path to an genome annotation file in GTF format.

Section "executables"

| java_exec |

STRING. Default “java”. Name of the executable for Java. Java version must be at least 1.8!

| GATK_jar |

STRING. Default "/g/scb2/zaugg/carnold/Projects/AtacSeq/src/Snakemake/tools/GenomeAnalysisTK.jar". Absolute path to a JAR file for the GATK suite.

| PICARD_jar |

STRING. Default "/g/scb2/zaugg/zaugg_shared/Programs/Picardtools/picardOld.jar". Absolute path to a JAR file for the Picard suite.

Section "par_trimming"

| trimmomatic_ILLUMINACLIP |

STRING. Default "1:30:4:5:true". ILLUMINACLIP value. See trimmomatic manual

| trimmomatic_trailing |

INTEGER. Default 3. TRAILING value. See trimmomatic manual

| trimmomatic_minlen |

INTEGER. Default 20. MINLEN value. See trimmomatic manual

| trimmomatic_phredType |

STRING. Default "phred33". Phred type. See trimmomatic manual. The "-" is added automatically by the Snakemake pipeline.

Section "par_align"

| bowtie2_sensitivity |

STRING. Default "--very-sensitive". Sensitivity. Leave empty for the default sensitivity. See bowtie2 manual.

| bowtie2_X |

INTEGER. Default 2000. Value for parameter X. See bowtie2 manual.

| bowtie2_refGenome |

STRING. Default "/g/scb/zaugg/zaugg_shared/annotations/hg19/referenceGenome/Bowtie2/hg19". Absolute path to the reference genome. See bowtie2 manual for more details.

| assemblyVersion |

STRING. Default “hg19”. Reference genome assembly version. Must match the one used by the alignment program.

Section "par_postalign"

| minMAPQscore |

INTEGER. Default 10. Minimum MAPQ score. Reads with a lower MAPQ quality will be removed during the post-alignment processing.

| ValidationStringencySortSam |

STRING. Default “LENIENT”. Value of the VALIDATION_STRINGENCY from SortSam (Picard tools). See the manual for details.

| ValidationStringencyMarkDuplicates |

STRING. Default “SILENT”. Value of the VALIDATION_STRINGENCY from MarkDuplicates (Picard tools). See the manual for details.

| CIGAR |

STRING. Defauled “ID”. Used for filtering reads. Relates to the one letter abbreviations for CIGAR strings such as I for insertion and D for deletion.

Specify all the one letter abbreviations in the CIGAR string of a read here that should be filtered. "ID" keeps a read only if the CIGAR string does not contain the letters "I" and "D" (e.g., only M for example)

| adjustRSS_forward|

INTEGER. Default 4. Adjustment of the read start positions on the forward strand. Should be a positive number. See the Buenrostro paper for details.

| adjustRSS_reverse|

INTEGER. Default -5. Adjustment of the read start positions on the reverse strand. Should be a negative number. See the Buenrostro paper for details.

Section "par_scripts"

| STATS_script_withinThr|

INTEGER. Default 4000. The region size in bp that specifies what is considered within a TSS.

The STATS script does a TSS enrichment test to test whether or not ATAC-Seq reads are primarily located within annotated TSS as opposed to outside of TSS regions. A value of 4000 means the region from -2kb up to +2kb of annotated TSS.

| STATS_script_outsideThr|

INTEGER. Default 1000. The size of the region adjacent to the within TSS region that is considered outside of a TSS.

A value of 1000 therefore denotes the 1kb region up- and downstream of the within TSS region (from -3 to -2kb upstream and from +2 to +3 kb downstream of annotated TSS.)

| STATS_script_geneTypesToKeep|

STRING. Default "protein_coding". Gene type to keep / do the analayses for. Allowed are gene types as specified by GENCODE. The default is "protein_coding".

| FL_distr_script_cutoff|

INTEGER. Default 600. Fragment length cutoff. All reads with a fragment length less than this value will be filtered for the purpose of this script.

Section "par_peakCalling"

| modelNonStringent |

STRING. Default "--nolambda –nomodel". Peak calling model for non-stringent peak calling.

| modelStringent |

STRING. Default "—nomodel”. Peak calling model for stringent peak calling.

| modelStringent_minQValue |

NUMERIC [0, 1]. Default 0.01. Minimum q-value threshold for stringent peak calling.

| modelNonStringent_minQValue |

NUMERIC [0, 1]. Default 0.1. Minimum q-value threshold for non-stringent peak calling.

| modelNonStringent_slocal |

INTEGER. Default 10000. Value for slocal parameter for non-stringent peak calling.

| Encode_pValThreshold |

NUMERIC [0, 1]. Default 0.1. p-value threshold for ENCODE peak calling.

| Encode_modelBroadAndGapped |

STRING. Default "--nomodel --shift -75 --extsize 150 --broad --keep-dup all". Model for Encode peak calling (broad and gapped mode)

| Encode_modelNarrow |

STRING. Default "--nomodel --shift -75 --extsize 150 -B --SPMR --keep-dup all –call-summits". Model for Encode peak calling (narrow mode)

Section "par_deepTools"

| effectiveGenomeSize |

INTEGER. The default depends on the genome assembly, see the templates for details. Length of the “mappable” genome in bp as defined by *deepTools* (see *https://deeptools.readthedocs.io/en/develop/content/feature/effectiveGenomeSize.html*).

| bamCoverage_normalizationCoverage |

STRING. Default “normalizeTo1x”. Either "normalizeTo1x NUMBER" OR "normalizeUsingRPKM" (note the missing (!) leading "--"), where NUMBER denotes the number of base pairs that are mappable. Beware of the mapping dependence on the read length for this number: The reported numbers on the websites are for 30bp reads, and we now have much longer reads usually. See Koehler et al. (2011) for numbers[2].

| bamCoverage_binSize |

INTEGER. Default 10. Size of the bins, in bases

| bamCoverage_otherOptions |

STRING. Default "--extendReads". Additional options that are supported by bamCoverage. Note that the "--" or "-" has to be present here.

2 Koehler, R., Issac, H., Cloonan, N., & Grimmond, S. M. (2011). The uniqueome: a mappability resource for short-tag sequencing. Bioinformatics, 27(2), 272-274.

2.4. Input metadata (samples.csv)

This file summarizes the data and corresponding metadata that are available and that should be used for the analysis. The format is flexible and may contain additional columns that are ignored by the pipeline, so it should be used to capture all the available information in a single place. Importantly, the file must be saved as tab-separated, the exact name does not matter as long as it is correctly specified in the configuration file. It must contain at least contain the following columns (the exact names do matter):

- *'individual':* The name of the individual the sample belongs to. Importantly, if multiple replicates exist for a particular sample, then their ID for individual should be identical! See the template for examples. This column is used to merge replicates, if this is set to TRUE, then all samples *with the same individual ID* will additionally be merged.
- *'sampleName':* An arbitrary unique name of the sample. We recommend a format like this: {individual}_rep{X}, with {individual} denoting the individual name and {X} the replicate number.
- *'path_inputForward'* and *'path_inputReverse'*: **The absolute path** to the forward and reverse reads of the sample. **Relative paths will result in errors in the beginning of the pipeline**. Note that currently, only paired-end reads are supported. The file must be in gzipped fastq format (ending in .gz).
- *'Flowcell_ID':* unique identifier for a particular flow cell. Can be set to NA if unknown.
- *'lane_ID':* lane of the flow cell. If all samples used the same lane, set to “lane1” or any other name.
- *'Technology': S*equencing technology used to generate the sequencing data. Valid values: ILLUMINA, SOLID, LS454, HELICOS and PACBIO. We only tested the pipeline for Illumina data so far.
- *'Library_ID':*library-specific identifier. If all samples used the same library, set to “default” or any other name.

3. Important Notes

3.1. Wrapper script

In addition to the options in the Snakemake `params.sh` file, the following Snakemake-related options are currently always invoked by the wrapper script: `--keep-going, --reason, --latency-wait 30, --stats, --local-cores 1`.

In cluster mode, if `useSLURM=true, submitToCluster=true` and `customJobStatusScript=true`, the following custom job script is used: `/g/scb/zaugg/zaugg_shared/scripts/Christian/src/Snakemake/SLURM_jobStatus.py`.

Although the wrapper script supports a variety of Snakemake-related options, it has also limitations. Options not mentioned in the `params.sh` file above are currently not supported by the wrapper script. This includes, for example, the usage of particular advanced Snakemake options or cluster-related options such as using DRMAA. Feel free to extend the wrapper script to suit your needs!

Since the --nolock parameter is enabled by default, **do not attempt to run multiple Snakemake analyses in the same folder**! Unless you changed the –nolock option, this will not immediately result in an error, but it will corrupt the metadata files that Snakemake produces.