CUT&RUN-Flow

Release 0.11-dev

Daniel Stribling, Rolf Renne

Apr 05, 2022

DOCUMENTATION CONTENTS:

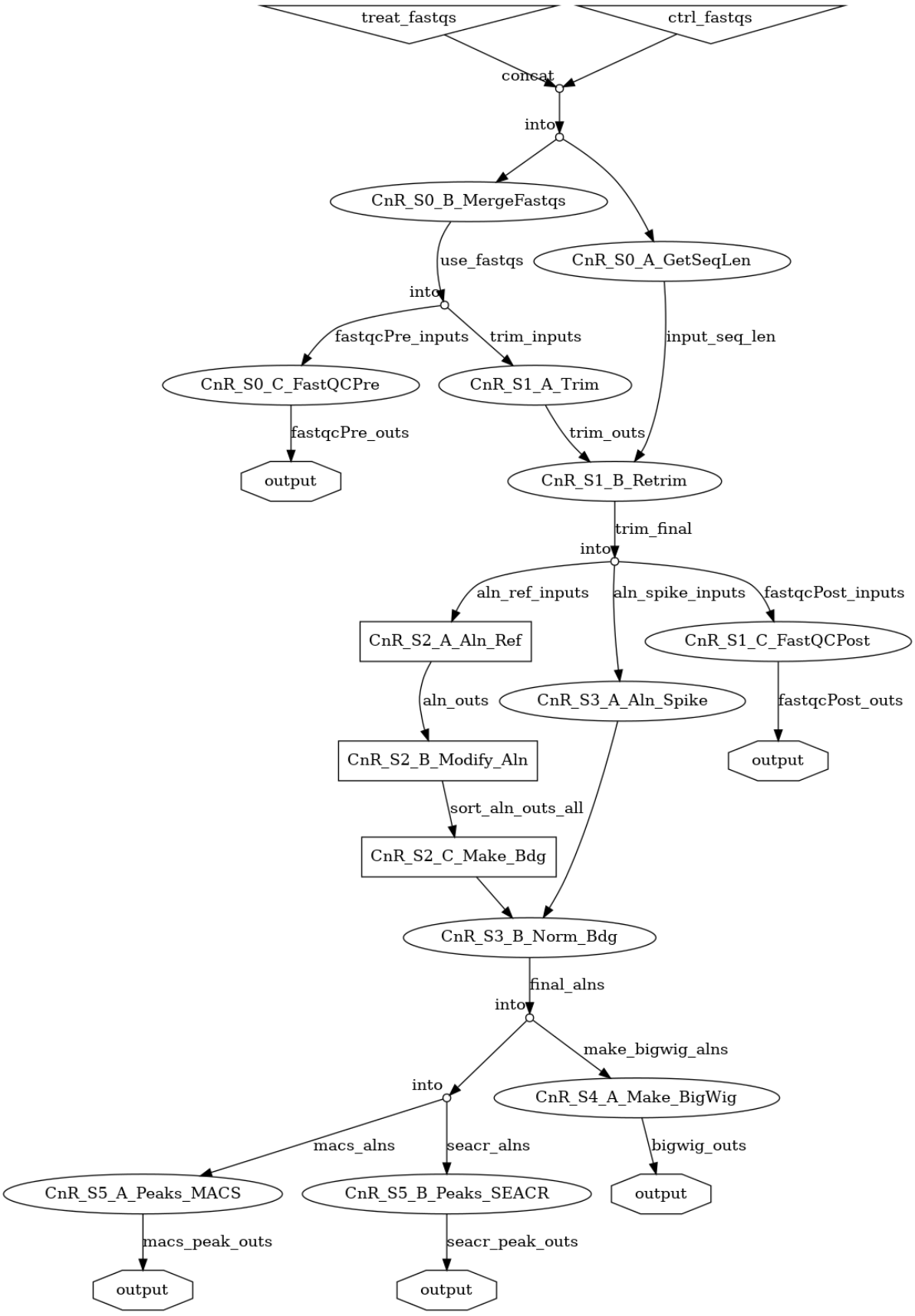
1 2 3

In addition to local configurations, Nextflow handles dependencies in separated working environments within the same pipeline using Conda[16] or Environment Modules[17] within your working environment, or using container-encapsulated execution with Docker[18] or Singularity[19]. **CnR-flow is pre-configured to auto-acquire dependencies with no additional setup, either using Conda recipes from the Bioconda project, or by using Docker or Singularity to execute Docker images hosted by the BioContainers project** (Bioconda[20]; BioContainers[21]).

CUT&RUN-Flow utilizes UCSC Genome Browser Tools[22] and Samtools[23] for reference library preparation, FastQC[24] for tag quality control, Trimmomatic[25] for tag trimming, Bowtie2[26] for tag alignment, Samtools[27], bedtools[28] and UCSC Genome Browser Tools[29] for alignment manipulation, and MACS2[30] and/or SEACR[31] for peak calling, as well as their associated language subdependencies of Java, Python2/3, R, and C++.

- One-step reference database prepration using a path (or URL) to a FASTA file.
- Ability to specify groups of samples containing both treatment (Ex: H3K4me3) and control (Ex: IgG) antibody groups, with automated association of each control sample with the respective treatment samples during the peak calling step
- Built-in normalization protocol to normalize to a sequence library of the user's choice when spike-in DNA is used in the CUT&RUN Protocol (Optional, includes an *E. coli* reference genome for utiliziation of *E. coli* as a spike-in control as described by *Meers et. al. (eLife 2019)* [see the *References* (page 39) section of this documentation[32]])
- OR: CPM-normalization to normalize total read counts between samples (beta).
- SLURM, PBS... and many other job scheduling environments enabled natively by Nextflow
- Output of memory-efficient CRAM (alignment), bedgraph (genome coverage), and bigWig (genome coverage) file formats

[1] https://github.com/rennelab/cnr-flow/releases
[2] https://app.circleci.com/pipelines/github/RenneLab/CnR-flow
[3] https://CnR-flow.readthedocs.io/en/latest/?badge=latest
[16] https://anaconda.org/
[17] http://modules.sourceforge.net/
[18] http://www.docker.com/
[19] https://sylabs.io/
[20] https://bioconda.github.io/
[21] https://biocontainers.pro/
[22] https://hgdownload.cse.ucsc.edu/admin/exe/
[23] http://www.htslib.org/
[24] https://www.bioinformatics.babraham.ac.uk/projects/fastqc/
[25] http://www.usadellab.org/cms/?page=trimmomatic
[26] http://bowtie-bio.sourceforge.net/bowtie2/index.shtml
[27] http://www.htslib.org/
[28] https://bedtools.readthedocs.io/en/latest/
[29] https://hgdownload.cse.ucsc.edu/admin/exe/
[30] https://github.com/macs3-project/MACS
[31] https://github.com/FredHutch/SEACR
[32] https://cnr-flow.readthedocs.io#


treat_fastqs
ctrl_fastqs
concat
into
CnR_S0_B_MergeFastqs
use_fastqs
CnR_S0_A_GetSeqLen
into
fastqcPre_inputs
trim_inputs
input_seq_len
CnR_S0_C_FastQCPre
CnR_S1_A_Trim
fastqcPre_outs
trim_outs
output
CnR_S1_B_Retrim
trim_final
into
aln_ref_inputs
aln_spike_inputs
fastqcPost_inputs
CnR_S2_A_Aln_Ref
CnR_S1_C_FastQCPost
aln_outs
CnR_S3_A_Aln_Spike
fastqcPost_outs
CnR_S2_B_Modify_Aln
output
sort_aln_outs_all
CnR_S2_C_Make_Bdg
CnR_S3_B_Norm_Bdg
final_alns
into
make_bigwig_alns
into
CnR_S4_A_Make_BigWig
macs_alns
seacr_alns
bigwig_outs
CnR_S5_A_Peaks_MACS
CnR_S5_B_Peaks_SEACR
output
macs_peak_outs
seacr_peak_outs
output
output

For a full list of required dependencies and tested versions, see the *Dependencies* (page 22) section of this documentation[33], and for dependency configuration options see the *Dependency Config* (page 23) section.

Quickstart: Here is a brief introduction on how to install and get started using the pipeline. For full details, see this documentation[34].

Prepare Task Directory:

Create a task directory, and navigate to it.

```
$ mkdir ./my_task  # (Example)
$ cd ./my_task     # (Example)
```

Install Nextflow (if necessary):

Download the nextflow executable to your current directory.
(You can move the nextflow executable and add to $PATH for future usage)

```
$ curl -s https://get.nextflow.io | bash

# For the following steps, use:
nextflow    # If nextflow executable on $PATH (assumed)
./nextflow  # If running nextflow executable from local directory
```

Download and Install CnR-flow:

Nextflow will download and store the pipeline in the user's Nextflow info directory (Default: `~/.nextflow/`)

```
$ nextflow run RenneLab/CnR-flow --mode initiate
```

Configure, Validate, and Test:

Conda:

- Install miniconda (if necessary). Installation instructions[35]
- The CnR-flow configuration with Conda should then work "out-of-the-box."

Docker:

- Add '-profile docker' to all nextflow commands

Singularity:

- Add '-profile singularity' to all nextflow commands

If using an alternative configuration, see the *Dependency Config* (page 23) section of this documentation[36] for dependency configuration options.

Once dependencies have been configured, validate all dependencies:

[33] https://cnr-flow.readthedocs.io#
[34] https://cnr-flow.readthedocs.io#
[35] https://docs.conda.io/en/latest/miniconda.html
[36] https://cnr-flow.readthedocs.io#

```
# Conda or other configs:
$ nextflow run CnR-flow --mode validate_all

# OR Docker Configuration:
$ nextflow run CnR-flow -profile docker --mode validate_all

# OR Singularity Configuration:
$ nextflow run CnR-flow -profile singularity --mode validate_all
```

Fill the required task input parameters in "nextflow.config" For detailed setup instructions, see the *Task Setup* (page 16) section of this documentation[37] *Additionally, for usage on a SLURM, PBS, or other cluster systems, configure your system executor, time, and memory settings.*

```
# Configure:
$ <vim/nano...> nextflow.config   # Task Input, Steps, etc. Configuration

#REQUIRED values to enter (all others *should* work as default):
# ref_fasta               (or some other ref-mode/location)
# treat_fastqs            (input paired-end fastq[.gz] file paths)
#   [OR fastq_groups]     (mutli-group input paired-end .fastq[.gz] file␣
↪paths)
```

Prepare and Execute Pipeline:

Prepare your reference databse (and normalization reference) from .fasta[.gz] file(s):

```
$ nextflow run CnR-flow --mode prep_fasta
```

Perform a test run to check inputs, paramater setup, and process execution:

```
$ nextflow run CnR-flow --mode dry_run
```

If satisifed with the pipeline setup, execute the pipeline:

```
$ nextflow run CnR-flow --mode run
```

Further documentation on CUT&RUN-Flow components, setup, and usage can be found in this documentation[38].

[37] https://cnr-flow.readthedocs.io#
[38] https://cnr-flow.readthedocs.io#

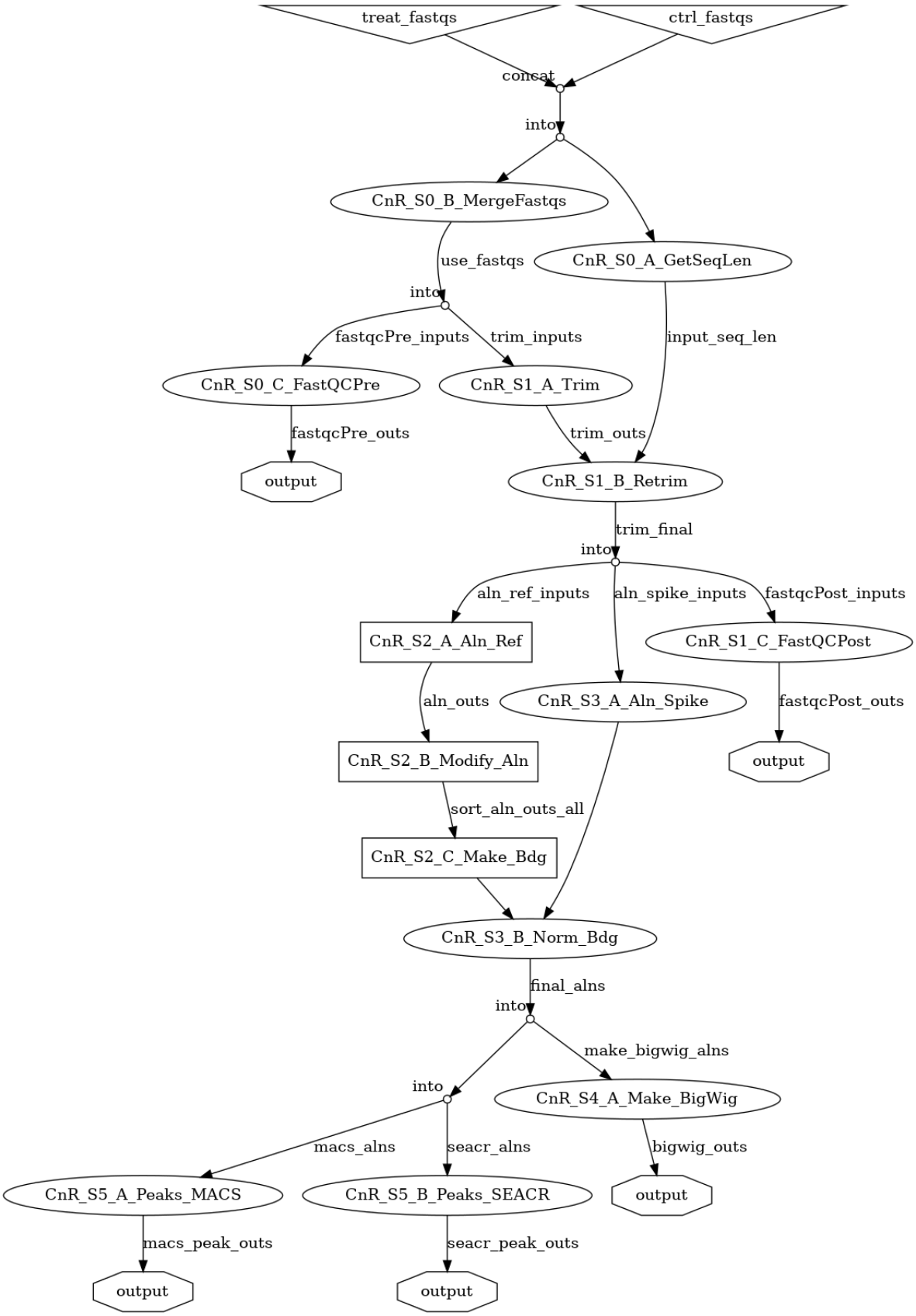
CHAPTER

ONE

WORKFLOW

The CUT&RUN-Flow (CnR-flow) workflow is designed to be simple to install and execute without sacrificing analysis capabilities. Nextflow[39] provides several features that facilitate this design, included automatic download of dependencies and i/o handling between workflow components. [Nextflow_Citation]

[39] http://www.nextflow.io



Full Workflow:

1.1 Download and Installation

(If necessary, begin by installing Nextflow[40] and Conda[41] as described in *Quickstart* (page 3))

Nextflow allows the pipeline to be downloaded simply by using the "nextflow run" command, and providing the user/repository path to the relevant github repository. CnR-flow has a one-step installation mode that can be performed simultaneously (`--mode initiate`).
Together, this gives the command:

```
$ nextflow run RenneLab/CnR-flow --mode initiate
```

This should download the pipeline, install the few required local files and dependencies, and prepare the pipeline for execution.

Note:

After the initial download, the pipeline can then be referred to by name, as with:
"nextflow run CnR-flow . . . "

1.2 Dependency Setup and Validation

CUT&RUN-Flow comes preconfigured to utilize the Conda[42] package management suite together with Bioconda[43] packages to handle pipeline dependencies (currently supported with Linux64 systems, with macOS support under development).
For even greater reproducibility, it is now recommended to utilize Docker[44] (`-profile docker`) or Singularity[45] (`-profile singularity`) to execute pipeline steps where these options are available.
for more details, or for an alternative configuration, see *Dependency Config* (page 23)

Once dependency setup has been completed, it can be tested using the `--mode validate` run mode.

```
# validate only steps enabled in nextflow.config
$ nextflow run CnR-flow --mode validate
```

[40] http://www.nextflow.io
[41] https://anaconda.org/
[42] https://anaconda.org/
[43] https://bioconda.github.io/
[44] http://www.docker.com/
[45] https://sylabs.io/

1.3 Reference Preparation

CnR-flow provides one-step preparation of alignment reference genome(s) for use by the pipeline. Either the local path or URL of a fasta file are provided to the `--ref_fasta (params.ref_fasta)` paramater and the execution is performed with:

```
$ nextflow run CnR-flow --mode prep_fasta
```

This copies the reference fasta to the directory specified by `--ref_dir (params.ref_dir)`, creates a bowtie2 alignment reference, creates a fasta index using Samtools, and creates a ".chrom.sizes" file using UCSC faCount[46]. The effective genome size is also calculated with faCount[47], using the (Total - N's) method. [faCount_Citation] Reference details are written to a ".refinfo.txt" in the same directory.

Note: If spike-in normalization is enabled, the same process will be repeated for the fasta file supplied to `--norm_ref_fasta (params.norm_ref_fasta)` for alignments to the spike-in control genome.

These referenes are then detected automatically, using the same parameter used for preparation setup. For more details, see *Reference Files Setup* (page 17). The list of all detectable prepared databases can be provided using the `--mode list_refs` run mode:

```
$ nextflow run CnR-flow --mode list_refs
```

1.4 Experimental Condition

CUT&RUN-Flow allows automated handling of treatment (Ex: H3K4me3) and and control (Ex: IgG) input files, performing the analysis steps on each condition in parallel, and then associating the treatment with the control for the final peak calling step. This can be performed either with a single treatment/control group, or with multiple groups in parallel. For more information, see *Task Setup* (page 16).

1.5 Preprocessing Steps

1.5.1 GetSeqLen

This step is enabled with paramater `--do_retrim (params.do_retrim)` (default: `true`). This step takes one example input fastq[.gz] file and determines the sequence length, for use in later steps.

[46] https://hgdownload.cse.ucsc.edu/admin/exe/
[47] https://hgdownload.cse.ucsc.edu/admin/exe/

1.5.2 MergeFastqs

This step is enabled with paramater `--do_merge_lanes (params.do_merge_lanes)` (default: `true`). If multiple sets of paired end files are provided that differ only by the "_L00#_" component of the name, these sequences are concatenated for further analysis.

For example, these files will be merged into the common name: 'my_sample_CTRL_R(1/2)_001.fastq.gz':

```
./my_sample_CTRL_L001_R1_001.fastq.gz ./my_sample_CTRL_L001_R2_001.fastq.gz
./my_sample_CTRL_L002_R1_001.fastq.gz ./my_sample_CTRL_L002_R2_001.fastq.gz
#... -->
./my_sample_CTRL_R1_001.fastq.gz ./my_sample_CTRL_R2_001.fastq.gz
```

1.5.3 FastQCPre

This step is enabled with paramater `--do_fastqc (params.do_fastqc)` (default: `true`). FastQC[48] is utilized to perform quality control checks on the input (presumably untrimmed) fastq[.gz] files. [FastQC_Citation]

1.5.4 Trim

This step is enabled with paramater `--do_trim (params.do_trim)` (default: `true`). Trimming of input fastq[.gz] files for read quality and adapter content is performed by Trimmomatic[49]. [Trimmomatic_Citation]

Default trimming parameters:

```
trimmomatic_adapter_mode   = "ILLUMINACLIP:"
trimmomatic_adapter_params = ":2:15:4:4:true"
trimmomatic_settings = "LEADING:20 TRAILING:20 SLIDINGWINDOW:4:15 MINLEN:25"
```

Default flags:

```
trimmomatic_flags = "-phred33"
```

1.5.5 FastQCPost

This step is enabled with paramater `--do_fastqc (params.do_fastqc)` (default: `true`). FastQC[50] is utilized to perform quality control checks on sequences after any/all trimming steps are performed. [FastQC_Citation]

[48] https://www.bioinformatics.babraham.ac.uk/projects/fastqc/
[49] http://www.usadellab.org/cms/?page=trimmomatic
[50] https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

1.6 Alignment Steps

1.6.1 Aln_Ref

Sequence reads are aligned to the reference genome using Bowtie2[51]. [Bowtie2_Citation]
Default alignment parameters were selected using concepts presented in work by the Henikoff Lab [Meers2019] and the Yuan Lab [CUTRUNTools_Citation].

Default flags:

```
aln_ref_flags = "--local --very-sensitive-local --phred33 -I 10 -X 700 --
↪dovetail --no-unal --no-mixed --no-discordant"
```

Warning: None of the output alignments (.sam/.bam/.cram) files produced in this step (or indeed, anywhere else in the pipeline) are normalized. The only normalized outputs are genome coverage tracks produced if normalization is enabled.

1.6.2 Modify_Aln

Output alignments are then subjected to several cleaning, filtering, and preprocessing steps utilizing Samtools[52]. [Samtools_Citation]
These are:

1. Removal of unmapped reads (samtools view)
2. Sorting by name (samtools sort [required for fixmate])
3. Adding/correcting mate pair information (samtools fixmate -m)
4. Sorting by genome coordinate (samtools sort)
5. Marking duplicates (samtools mkdup)
6. (Optional Processing Steps [see below])
7. Alignment compression BAM -> CRAM (samtools view)
8. Alignment indexing (samtools index)

Optional processing steps include:

- Removal of Duplicates
- Filtering to reads <= 120 bp in length

The desired category (or categories) of output are selected with `--use_aln_modes` (`params.use_aln_modes`). Multiple categores can be specifically selected using `params.use_aln_modes` as a list, and the resulting selections are analyzed and output in parallel.
(Example: `params.use_aln_modes = ['all', 'less_120_dedup'])`

[51] http://bowtie-bio.sourceforge.net/bowtie2/index.shtml
[52] http://www.htslib.org/

Option	Deduplicated	Length <= 120bp
all	false	false
all_dedup	true	false
less_120	false	true
less_120_dedup	true	true

Default mode:

```
use_aln_modes = ["all"]  // Options: ["all", "all_dedup", "less_120", "less_
↪120_dedup"]
```

1.6.3 Make_Bdg

Further cleaning steps are then performed on the outputs, to prepare the alignments for (optional) normalization and peak calling.

These modifications are performed as suggested by the Henikoff lab in the documentation for SEACR, https://github.com/FredHutch/SEACR/blob/master/README.md [SEACR_Citation] , and are performed utilizing Samtools[53] [Samtools_Citation] and bedtools[54] [bedtools_Citation].

These are:

1. Sorting by name and uncompress CRAM -> BAM (samtools sort)
2. Covert BAM to bedgraph (bedtools bamtobed)
3. Filter discordant tag pairs (awk)
4. Change bedtools bed format to BEDPE format (cut | sort)
5. Convert BEDPE to (non-normalized) bedgraph (bedtools genomecov)

Note: Genome coverage tracks output by this step are NOT normalized.

1.7 Normalization Steps

1.7.1 Aln_Spike

This step is enabled with paramater `--do_norm_spike (params.do_norm_spike)` (default: `true`).

This step calculates a normalization factor for scaling output genome coverage tracks.

Strategy: A dual-alignment strategy is used to filter out any reads that cross-map to both the primary reference and the normalization references. Sequence pairs that align to the normalization reference are then re-aligned to the primary reference. The number of read pairs that align to both references

[53] http://www.htslib.org/
[54] https://bedtools.readthedocs.io/en/latest/

is then subtracted from the normalization factor output by this step, depending on the value of `--norm_mode (params.norm_mode)` (default: `true`).

Details:

Sequence reads are first aligned to the normalization reference genome using Bowtie2[55]. [Bowtie2_Citation] Default alignment parameters are the same as with alignment to the primary reference genome.

Default flags:

```
aln_norm_flags = params.aln_ref_flags
```

All reads that aligned to the normalization reference are then again aligned to the primary reference using Bowtie2[56]. [Bowtie2_Citation]

Counts are then performed of **pairs** of sequence reads that align (and re-align, respectively) to each reference using Samtools[57] (via `samtools view`). [Samtools_Citation] The count of aligned pairs to the spike-in genome reference is then returned, with the number of cross-mapped pairs subtracted depending on the value of `--norm_mode (params.norm_mode)`.

norm_mode	Normalization Factor Used
all	norm_ref_aligned (pairs)
adj	norm_ref_aligned - cross_map_aligned (pairs)

```
norm_mode = 'adj' // Options: ['adj', 'all']
seacr_norm_mode = "auto" // Options: "auto", "norm", "non"
```

1.7.2 Norm_Bdg

This step is enabled with paramater `--do_norm_spike (params.do_norm_spike)` (default: `true`).
This step uses a normalization factor to create scaled genome coverage tracks. The calculation as provided by the Henikoff Lab:
https://github.com/Henikoff/Cut-and-Run/blob/master/spike_in_calibration.csh [Meers2019] is:

$$count_{norm} = (count_{site} * norm\_scale)/norm\_factor$$

Thus, the scaling factor used is calucated as:

$$scale\_factor = norm\_scale/norm\_factor$$

Where `norm_factor` is calculated in the previous step, and the arbitrary `norm_scale` is provided by the parameter: `--norm_scale (params.norm_scale)`.

[55] http://bowtie-bio.sourceforge.net/bowtie2/index.shtml
[56] http://bowtie-bio.sourceforge.net/bowtie2/index.shtml
[57] http://www.htslib.org/

Default `norm_scale` value:

```
norm_scale = 1000  // Arbitrary value for scaling of normalized counts.
```

The normalized genome coverage track is then created by bedtools[58] using the `-scale` option. [bedtools_Citation]

1.7.3 Aln_CPM

(Beta Implementation)
This step is enabled with paramater `--do_norm_cpm (params.do_norm_cpm)` (default: `false`).
As an alternative option to spike-in normalization, this "counts-per-million" normalization method calculates a normalization factor for scaling output genome coverage tracks by dividng by the total number of reference-aligned tags per sample, multiplying by 1M, and then by an arbitrary scaling factor.

Counts of reference-aligned reads are made using using Samtools[59] (via `samtools view`). [Samtools_Citation]

The per-site scaled count is then calculated:

$$count_{norm} = (count_{site} * norm\_scale)1,000,000/total\_ref\_aln\_pairs$$

Thus, the scaling factor used is calucated as:

$$scale\_factor = norm\_scale * 1,000,000/total\_ref\_aln\_pairs$$

The arbitrary `norm_cpm_scale` is provided by the parameter: `--norm_cpm_scale (params.norm_cpm_scale)`.

Default `norm_cpm_scale` value:

```
norm_cpm_scale = 1000 // Arbitrary value for scaling of normalized counts.
```

The normalized genome coverage track is then created by bedtools[60] using the `-scale` option. [bedtools_Citation]

[58] https://bedtools.readthedocs.io/en/latest/
[59] http://www.htslib.org/
[60] https://bedtools.readthedocs.io/en/latest/

1.8 Conversion Steps

1.8.1 Make_BigWig

This step is enabled with paramater `--do_make_bigwig (params.do_make_bigwig)` (default: `true`).
This step converts the output genome coverage file from the previous steps as in the UCSC bigWig file format using UCSC bedGraphToBigWig[61], a genome coverage format with significantly decreased file size. [bedGraphToBigWig_Citation]

> **Warning:** The bigWig file format is a "lossy" file format that cannot be reconverted to bedGraph with all information intact.

1.9 Peak Calling Steps

One or more peak callers can be used for peak calling. The peak caller used is determined by `--peak_callers (params.peak_callers)`. This can either be provided a single argument, as with:

```
--peak_callers seacr (params.peak_callers = "seacr")
```

. . . or can be configured using a list:

```
params.peak-callers = ['macs', 'seacr']
```

Default `peak_callers` value:

```
peak_callers = ['macs', 'seacr']  // Options: ['macs', 'seacr']
```

1.9.1 Peaks_MACS2

This step is enabled if `macs` is included in `params.peak-callers`.
This step calls peaks using the **non-normalized** alignment data produced in previous steps, using the MACS2[62] peak_caller. [MACS2_Citation]

Default MACS2 Settings:

```
// Macs2 Settings
macs_qval = '0.01'
macs_flags = ''
```

[61] https://hgdownload.cse.ucsc.edu/admin/exe/
[62] https://github.com/macs3-project/MACS

1.9.2 Peaks_SEACR

This step is enabled if `seacr` is included in `params.peak_callers`.
This step calls peaks using the **normalized** alignment data produced in previous steps (if normalization is enabled, using the SEACR[63] peak caller. [SEACR_Citation]

Special thanks to Michael Meers[64] *and the* Henikoff Group[65] *for their permission to distribute SEACR bundled with this pipeline.*

Parameters: `--seacr_norm_mode (params.seacr_norm_mode)` passes either `norm` or `non` to SEACR. Options:

- **`'auto'`:**
  - if `do_norm = true` - Passes `'non'` to SEACR
  - if `do_norm = false` - passes `'norm'` to SEACR
- `'norm'`
- `'non'`

`--seacr_fdr (params.seacr_fdr)` is passed directly to SEACR.

`--seacr_call_stringent (params.seacr_call_stringent)` - SEACR is called in "stringent" mode.
`--seacr_call_relaxed (params.seacr_call_relaxed)` - SEACR is called in "relaxed" mode.
(If both of these are true, both outputs will be produced)

Default SEACR Settings:

```
// SEACR Settings
seacr_fdr_threshhold = "0.01"
seacr_norm_mode = "auto" // Options: "auto", "norm", "non"
seacr_call_stringent = true
seacr_call_relaxed = true
```

[63] https://github.com/FredHutch/SEACR
[64] https://github.com/mpmeers
[65] http://research.fhcrc.org/henikoff

CHAPTER

TWO

TASK SETUP

2.1 Task Setup Overview

After running using `--mode initiate`, *CUT&RUN-Flow* will copy the task configuration template into your current working directory. For a full example of this file, see *Task nextflow.config* (page 27).

```
# Configure:
$ <vim/nano...> ./my_task/nextflow.config    # Task Input, Steps, etc. Configuration
```

Task-level inputs such as input files and reference fasta files must be configured here (see *Input File Setup* (page 18)). Additional task-specific settings are also configured here, such as output read naming rules and output file locations (see *Output Setup* (page 20)).

Note: These settings are provided for user customizability, but in the majority of cases the default settings should work fine.

Many pipeline settings can justifiably be configured either on a task-specifc basis (in *Task nextflow.config* (page 27)) or as defaults for the pipeline (in *Pipe nextflow.config* (page 30)). These include nextflow "executor" settings for use of SLURM[66] and Environment Modules[67] and associated settings such as memory and cpu usage. These settings are described here, in *Executor Setup* (page 19), but can also be set in the *Pipe nextflow.config* (page 30).

Likewise, settings for individual pipeline components such as Trimmomatic[68] tag trimming paramaters, or the `qval` used for MACS2[69] peak calling can be provided in either config file, or both (for a description of these parameters, see *Workflow* (page 5)).

Note: If any settings are provided in both the above *Task nextflow.config* (page 27) file and the *Pipe nextflow.config* (page 30) file located in the pipe directory, the task-directory settings will take precedence.

---

[66] https://slurm.schedmd.com/
[67] http://modules.sourceforge.net/
[68] http://www.usadellab.org/cms/?page=trimmomatic
[69] https://github.com/macs3-project/MACS

For more information on Nextflow configuration precedence, see *config*.

2.2 Reference Files Setup

CUT&RUN-Flow handles reference database preparation with a series of steps utilizing `--mode prep_fasta`. The location of the fasta used for preparation is provided to the `--ref_fasta (params.ref_fasta)` paramater as either a file path or URL.

Reference preparation is then performed using:

```
$ nextflow CnR-flow --mode prep_fasta
```

This will place the prepared reference files in the directory specified by `--refs_dir (params.refs_dir)` (see *Output Setup* (page 20)). Once prepared, the this parameter can be dynamically used during pipeline execution to detect the reference name and location, depending on the value of the `--ref_mode (params.ref_mode)` parameter.

Ref Modes:

- `'fasta'` : Get reference name from `--ref_fasta (params.ref_fasta)` (which must then be set)
- `'name'` : Get reference name from `--ref_name (params.ref_name)` (which must then be set)
- `'manual'` : Set required paramaters manually:

Ref Required Manual Paramaters:

- `--ref_name (params.ref_name)` : Reference Name
- `--ref_bt2db_path (params.ref_bt2db_path)` : Reference Bowtie2 Alignment Reference Path
- `--ref_chrom_sizes_path (params.ref_chrom_sizes_path)` : Path to <reference>.chrom_sizes file
- `--ref_eff_genome_size (params.ref_eff_genome_size)` : Effective genome size for reference.

The `--ref_mode (params.ref_mode)` parameter also applies to the preparation and location of the fasta used for the normalization reference if `--do_norm (params.do_norm)`. These paramaters are named in parallel using a `norm_[ref...]` prefix and are autodetected from the value of `--norm_ref_fasta (params.norm_ref_fasta)` or `--norm_ref_name (params.norm_ref_name)` depending on the value of `--ref_mode (params.ref_mode)`. For details on normalization steps, see *Normalization Steps* (page 11).

2.3 Input File Setup

Two (mutually-exclusive) options are provided for supplying input sample fastq[.gz] files to the workflow.

Single Sample Group:

A single group of samples with zero or one (post-combination) control sample(s) for all treatment samples.

- `--treat_fastqs (params.treat_fastqs)`
- `--ctrl_fastqs (params.ctrl_fastqs)`

```
params {
// CnR-flow Input Files:
//   Provided fastqs must be in glob pattern matching pairs.
//     Example: ['./relpath/to/base*R{1,2}*.fastq']
//     Example: ['/abs/path/to/other*R{1,2}*.fastq']

treat_fastqs   = []    // REQUIRED, Single-group Treatment fastq Pattern
ctrl_fastqs    = []    //           Single-group Control   fastq pattern

}
```

Note: Note, for convenience, if the same file is found both as a treatment and control, the copy passed to treatment will be ignored (facilitates easy pattern matching).

Warning: Input files must be paired-end, and in fastq[.gz] format. Nextflow requires the use of this (strange-looking) `R{1,2}` naming construct, (matches either R1 or R2) which ensures that files are fed into the pipeline as pairs.

Multiple Sample Group:

A multi-group layout, with groups of samples provided where each group has a control sample. (All groups are required to have a control sample in this mode.)

- `params.fastq_groups`

```
params {
// Can specify multiple treat/control groups as Groovy mapping.
//   Specified INSTEAD of treat_fasts/ctrl_fastqs parameters.
//   Note: There should be only one control sample per group
//     (after optional lane combination)
// Example:
// fastq_groups = [
//   'group_1_name': ['treat': 'relpath/to/treat1*R{1,2}*',
//                    'ctrl':  'relpath/to/ctrl1*R{1,2}*'
//                   ],
//   'group_2_name': ['treat': ['relpath/to/g2_treat1*R{1,2}*',
//                              '/abs/path/to/g2_treat2*R{1,2}*'
//                             ],
//                    'ctrl':  'relpath/to/g2_ctrl1*R{1,2}*'
//                   ]
// ]
//fastq_groups = []
```

(continues on next page)

(continued from previous page)

```
}
```

Multiple pairs of files representing the same sample/replicate that were sequenced on different lanes can be automatically recognized and combined (default: `true`). For more information see: *MergeFastqs* (page 9).

2.4 Executor Setup

Nextflow provides extensive options for using cluster-based job scheduling, such as SLURM[70], PBS[71], etc. These options are worth reviewing in the nextflow docs: *executor*. The specific executor is selected with the configuration setting: `process.executor = 'option'`. The default value of `process.executor = 'local'` runs the execution on the local filesystem.

Specific settings of note:

Option	Example
`process.executor`	`'slurm'`
`process.memory`	`'4 GB'`
`process.cpus`	`4`
`process.time`	`'1h'`
`process.clusterOptions`	`'--qos=low'`

To facilitate process efficiency (and for adequate capacity) for different parts of the process, memory-related process labels have been applied to the processes: `'small_mem'`, `'norm_mem'`, and `'big_mem'`. These are specified using `process.withLabel: my_label { key = value }` Example: `process.withLabel: big_mem { memory = '16 GB' }`.
A `1n/2n/4n` or `1n/2n/8n` strategy is recommended for the respective `small_mem/norm_mem/big_mem` options. (for details on nextflow process labels, see process[72]). Additionally, mutliple cpu usage is disabled for processes that do not support (or aren't significanlly more effective) with multiple processes, and so the `process.cpus` setting only applies to processes within the pipeline with multiple CPUS enabled.

```
// Process Settings (For use of PBS, SLURM, etc)
process {
    // --Executor, see: https://www.nextflow.io/docs/latest/executor.html
    //executor = 'slurm'  // for running processes using SLURM (Default:
↪'local')
    // Process Walltime, See https://www.nextflow.io/docs/latest/process.html
↪#process-time
    //time = '12h'
    // Process CPUs, See https://www.nextflow.io/docs/latest/process.html
↪#cpus
    //cpus = 8
    //
```

(continues on next page)

[70] https://slurm.schedmd.com/
[71] https://www.openpbs.org/
[72] https://www.nextflow.io/docs/latest/process.html#label

(continued from previous page)

```
    // Memory: See https://www.nextflow.io/docs/latest/process.html#process-
↪memory
    // Set Memory for specific task sizes (1n/2n/4n scheme recommended)
    //withLabel: big_mem   { memory = '16 GB' }
    //withLabel: norm_mem  { memory = '4 GB'  }
    //withLabel: small_mem { memory = '2 GB'  }
    // -*OR*- Set Memory for all processes
    //memory = "16 GB"

    ext.ph = null //Placeholder to prevent errors.
}
```

2.5 Output Setup

Output options can control the quantity, naming, and location of output files from the pipeline.

publish_files: Three modes are available for selecting the number of output files from the pipeline:

- `minimal` : Only the final alignments are output. (Trimmed Fastqs are Excluded)
- `default` : Multiple types of alignments are output. (Trimmed Fastqs are included)
- `all` : All files produced by the pipline (excluding deleted intermediates) are output.

This option is selected with `--publish_files (params.publish_files)`.

publish_mode: This mode selects the value for the Nextflow `process.publishDir` mode used to output files (for details, see: publishDir[73]). Available options are:

- `'copy'` : Copy output files (from the nextflow working directory) to the output folder.
- `'symlink'` : Link to the output files located in the nextflow working directory.

trim_name_prefix & trim_name_suffix:

`params.trim_name_prefix` & `params.trim_name_suffix`
These options allow trimming of a prefix or suffix from sample names (after any merging steps).

out_dir: `--out_dir (params.out_dir)` : Location for output of the files.

refs_dir: `--refs_dir (params.refs_dir)` : Location for placing and searching for reference directories.

```
params {
    // ------- General Pipeline Output Paramaters --------
    publish_files    = 'default' // Options: ["minimal", "default", "all"]
    publish_mode     = 'copy'    // Options: ["symlink", "copy"]

    // Name trim guide: ( regex-based )
    //    ~/groovy-slashy-string/  ;  "~" denotes groovy pattern type.
    //    ~/^/ matches beginning   ;   ~/$/ matches end
    trim_name_prefix = ''        // Example: ~/^myprefix./ removes "myprefix.
↪" prefix.
    trim_name_suffix = ''        // Example: ~/_mysuffix$/ removes "_mysuffix
↪" suffix.
```

(continues on next page)

[73] https://www.nextflow.io/docs/latest/process.html#publishdir

(continued from previous page)

```
    // Workflow Output Default Naming Scheme:
    //   Absolute paths for output:
    out_dir          = "${launchDir}/cnr_output"
    refs_dir         = "${launchDir}/cnr_references"
}
```

CHAPTER

THREE

PIPE SETUP

3.1 Pipe Settings Location

The Default *CUT&RUN-Flow* settings used by the system are located in *CUT&RUN-Flow's* installation directory. This is by default located in the user's `$HOME` directory as such:

```
# Pipe Defaults Configuration:
$NXF_HOME/assets/RenneLab/CnR-flow/nextflow.config  # Pipe Executor, Dependency,␣
↪Resource, etc. Configuration
#Default: $HOME/.nextflow/assets/RenneLab/CnR-flow/nextflow.config
```

It is recommended that dependency configuration and other pipe-level settings be configured here. An example of this file is provided in *Pipe nextflow.config* (page 30).

Note: If any settings are provided in both the above *Pipe nextflow.config* (page 30) file and the *Task nextflow.config* (page 27) file located in the task directory, the task-directory settings will take precedence. For more information on Nextflow configuration precedence, see *config*.

3.2 Dependencies

The following external dependencies are utilized by *CUT&RUN-Flow*:

Note: Versions utilized in containers may differ slightly from those listed based on availabilty.

Dependency	Min-Tested Ver.	Citation	Default Conda
Nextflow	20.10.4	[Nextflow_Citation]	N/A
UCSC-faCount	366	[faCount_Citation]	bioconda::ucsc-facount=366'
Bowtie2	2.4.4	[Bowtie2_Citation]	bioconda::bowtie2=2.4.1
Samtools	1.9	[Samtools_Citation]	bioconda::samtools=1.9
FastQC	0.11.9	[FastQC_Citation]	bioconda::fastqc=0.11.9
Trimmomatic	0.39	[Trimmomatic_Citation]	bioconda::trimmomatic=0.39
bedtools	2.29.2	[bedtools_Citation]	bioconda::bedtools=2.29.2
UCSC-bedGraphToBigWig	377	[bedGraphToBigWig_Citation]	conda-forge::libgcc-ng conda-forge::libpng con
MACS2	2.2.6	[MACS2_Citation]	bioconda::macs2=2.2.6

Dependency	Min-Tested Ver.	Citation	Default Conda
R	3.6.0	[R_Citation]	r=3.6.0
SEACR	1.3	[SEACR_Citation]	r=3.6.0 bioconda::seacr=1.3-2 bioconda::bedtoo

3.3 Dependency Config

3.3.1 Conda/Bioconda

CUT&RUN-Flow comes preconfigured to use the Conda[74] package manager, along with tools from the Bioconda[75] package suite for automated dependency handling. [Bioconda_Citation]
Nextflow[76] automatically acquires, stores, and activates each conda environment as it is required in the pipeline. For more information on Nextflow's, usage of conda, see `conda`.

```
// Dependency Configuration Using Anaconda
params.facount_conda          = 'bioconda::ucsc-facount=366'
params.bowtie2_conda          = 'bioconda::bowtie2=2.4.4 conda-forge::libgcc-
↪ng=9.3'
params.fastqc_conda           = 'bioconda::fastqc=0.11.9'
params.trimmomatic_conda      = 'bioconda::trimmomatic=0.39'
params.bedtools_conda         = 'bioconda::bedtools=2.29.2'
params.macs2_conda            = 'bioconda::macs2=2.2.6'
params.seacr_conda            = "r=3.6.0 bioconda::seacr=1.3 ${params.
↪bedtools_conda}"
params.samtools_conda         = 'bioconda::samtools=1.9'
params.bedgraphtobigwig_conda = 'conda-forge::libpng conda-forge::libuuid␣
↪conda-forge::mysql-connector-c conda-forge::openssl conda-forge::zlib␣
↪bioconda::ucsc-bedgraphtobigwig=377'

params.trimmomatic_adapterpath = "${projectDir}/ref_dbs/trimmomatic_adapters/
↪TruSeq3-PE-2.fa"
```

3.3.2 Singularity Containers

In addition to a default setup with Conda, *CUT&RUN-Flow* also comes preconfigured to use Docker[77] containers from the BioContainers[78] project using Singularity[79]. [BioContainers_Citation]
Nextflow[80] automatically pulls and runs singularity containers as required during the pipeline. This feature is enabled with `-profile singularity`.

```
// Dependency Configuration Using Containers with Singularity:
singularity.enabled = true
params.facount_container          = "docker://quay.io/biocontainers/ucsc-
↪facount:366--h5eb252a_0"
```

(continues on next page)

[74] https://anaconda.org/
[75] https://bioconda.github.io/
[76] http://www.nextflow.io
[77] http://www.docker.com/
[78] https://biocontainers.pro/
[79] https://sylabs.io/
[80] http://www.nextflow.io

(continued from previous page)

```
params.bowtie2_container           = "docker://quay.io/biocontainers/
↪bowtie2:2.4.4--py38he5f0661_1"
params.bowtie2_samtools_container  = "docker://quay.io/biocontainers/mulled-
↪v2-
↪c742dccc9d8fabfcff2af0d8d6799dbc711366cf:b6524911af823c7c52518f6c886b86916d062940-
↪0"
// Mulled: bowtie2=2.4.4,samtools=1.13
params.fastqc_container            = "docker://quay.io/biocontainers/
↪fastqc:0.11.9--hdfd78af_1"
params.trimmomatic_container       = "docker://quay.io/biocontainers/
↪trimmomatic:0.39--1"
params.macs2_container             = "docker://quay.io/biocontainers/macs2:2.
↪2.6--py37h516909a_0"
params.seacr_container             = "docker://quay.io/biocontainers/seacr:1.
↪3--hdfd78af_2"
params.samtools_container          = "docker://quay.io/biocontainers/
↪samtools:1.13--h8c37831_0"
params.bedgraphtobigwig_container  = "docker://quay.io/biocontainers/ucsc-
↪bedgraphtobigwig:377--h0b8a92a_2"
params.samtools_bedtools_container = "docker://quay.io/biocontainers/mulled-
↪v2-
↪fc325951871d402a00bdf9d0e712a5b81b8e0cb3:38034b9703d6561a40bcaf2f1ec16f8b158fde97-
↪0"
// Mulled: samtools=1.14.0,bedtools=2.30.0,ucsc-facount=377,python=3.8
params.samtools_facount_container  = "${params.samtools_bedtools_container}"
params.bedtools_container          = "${params.samtools_bedtools_container}"

params.trimmomatic_adapterpath     = "/usr/local/share/trimmomatic/adapters/
↪TruSeq3-PE-2.fa"
```

3.3.3 Docker Containers

In addition to a default setup with Conda, *CUT&RUN-Flow* also comes preconfigured to use Docker[81] containers. Nextflow[82] automatically pulls and runs singularity containers as required during the pipeline. This feature is enabled with `-profile docker`.

```
// Dependency Configuration Using Containers with Docker:
docker.enabled = true
docker.runOptions = "-v ${launchDir}:${launchDir}"
//docker.runOptions = "-v ${params.refs_dir}:${params.refs_dir}"

params.facount_container           = "quay.io/biocontainers/ucsc-facount:366-
↪-h5eb252a_0"
params.bowtie2_container           = "quay.io/biocontainers/bowtie2:2.4.4--
↪py38he5f0661_1"
params.bowtie2_samtools_container  = "quay.io/biocontainers/mulled-v2-
↪c742dccc9d8fabfcff2af0d8d6799dbc711366cf:b6524911af823c7c52518f6c886b86916d062940-
↪0"
// Mulled: bowtie2=2.4.4,samtools=1.13
params.fastqc_container            = "quay.io/biocontainers/fastqc:0.11.9--
↪hdfd78af_1"
```

(continues on next page)

[81] http://www.docker.com/
[82] http://www.nextflow.io

(continued from previous page)

```
params.trimmomatic_container       = "quay.io/biocontainers/trimmomatic:0.39-
↪-1"
params.macs2_container             = "quay.io/biocontainers/macs2:2.2.6--
↪py37h516909a_0"
params.seacr_container             = "quay.io/biocontainers/seacr:1.3--
↪hdfd78af_2"
params.samtools_container          = "quay.io/biocontainers/samtools:1.13--
↪h8c37831_0"
params.bedgraphtobigwig_container  = "quay.io/biocontainers/ucsc-
↪bedgraphtobigwig:377--h0b8a92a_2"
params.samtools_bedtools_container = "quay.io/biocontainers/mulled-v2-
↪fc325951871d402a00bdf9d0e712a5b81b8e0cb3:38034b9703d6561a40bcaf2f1ec16f8b158fde97-
↪0"
// Mulled: samtools=1.14.0,bedtools=2.30.0,ucsc-facount=377,python=3.8
params.samtools_facount_container  = "${params.samtools_bedtools_container}"
params.bedtools_container          = "${params.samtools_bedtools_container}"

params.trimmomatic_adapterpath     = "/usr/local/share/trimmomatic/adapters/
↪TruSeq3-PE-2.fa"
```

3.3.4 Modules

CUT&RUN-FLow comes with an alternative "easy" configuration option utilizing Environment Modules[83]. Each conda dependency paramater has an equivalent "module" paramater. Each module will then be loaded at runtime for the appropriate steps of the pipeline. For more information on Nextflow's use of Environment Modules, see `process` ("modules section").

```
// Dependency Configuration Using Environment Modules
//   (values will vary depending on system)
params.facount_module          = ""  // Ex: "ucsc/20200320"
params.bowtie2_module          = ""  // Ex: "bowtie2/2.3.5.1"
params.fastqc_module           = ""  // Ex: "fastqc/0.11.7"
params.trimmomatic_module      = ""  // Ex: "trimmomatic/0.39"
params.bedtools_module         = ""  // Ex: "bedtools/2.29.2"
params.macs2_module            = ""  // Ex: "macs/2.2.7.1"
params.seacr_module            = ""  // Ex: "R/4.0 seacr/1.3 ${params.
↪bedtools_module}"
params.bedgraphtobigwig_module = ""  // Ex: "ucsc/20200320"

params.trimmomatic_adapterpath = "${projectDir}/ref_dbs/trimmomatic_adapters/
↪TruSeq3-PE-2.fa"
```

[83] http://modules.sourceforge.net/

3.3.5 Executable System Calls

To accommodate custom module or local setups, each required dependency system call can be customized:

```
// System Call Settings
//   Call executed on the system, defaults assume that each tool is available
//     on the system PATH
//   Can replace with direct path as desired:
//   Ex:
//        samtools_call = "samtools"
//     or samtools_call = "/path/to/samtools/dir/samtools"
java_call             = "java"
bowtie2_build_call    = "bowtie2-build"
facount_call          = "faCount"
samtools_call         = "samtools"
fastqc_call           = "fastqc"
trimmomatic_call      = "trimmomatic"
bowtie2_call          = "bowtie2"
bedtools_call         = "bedtools"
macs2_call            = "macs2"
bedgraphtobigwig_call = "bedGraphToBigWig"
seacr_call            = "SEACR_1.3.sh"
seacr_R_script        = "SEACR_1.3.R"
```

CHAPTER

FOUR

EXAMPLE FILES

4.1 Task nextflow.config

This nextflow.config file is copied to the task directory by the command:

```
#mkdir ./my_task_dir
#cd ./my_task_dir
nextflow run RenneLab/CnR-flow --mode initiate
```

This configuration file contains paramaters for input file and task workflow setup.

Listing 1: .../my_task_dir/nextflow.config (Task Settings)

```
//Daniel Stribling
//Renne Lab, University of Florida
//CnR-flow Configuration File
//
// Config syntax:
// This config file uses the syntax described here:
//   https://www.nextflow.io/docs/latest/config.html
// Primarily, "//" acts as a comment and allows removal/deactivation of lines.
//
// Settings within scopes, Ex:
//   params {
//     setting = value
//     ...
//   }
// are equivalent to:
//   params.setting = value"
//
// Configuration Hierarchy:
//   Different Nextflow configuration files have the precedence order:
//     task-dir  >  pipe-dir  >  user-default  >  pipe-default
```

(continues on next page)

(continued from previous page)

```
//
//   This configruation is (presumably) in the task directory:
//   (after running "nextflow CnR-flow --mode initiate" )
//   and will therefore superceed pipe-setup, user, and pipe hardcoded defaults.
//   For more detail visit: https://www.nextflow.io/docs/latest/config.html
//
//   This configuration only includes task-specific parameters and parameters
//   designed to be modified at runtime. Other parameters are provided in the
//   pipe configuration, at CnR-flow/nextflow.config. Parameters
//   provided here will override system defaults.
//

// Process Settings (For use of PBS, SLURM, etc)
process {
    // --Executor, see: https://www.nextflow.io/docs/latest/executor.html
    //executor = 'slurm'  // for running processes using SLURM (Default: 'local')
    // Process Walltime, See https://www.nextflow.io/docs/latest/process.html#process-
↪time
    //time = '12h'
    // Process CPUs, See https://www.nextflow.io/docs/latest/process.html#cpus
    //cpus = 8
    //
    // Memory: See https://www.nextflow.io/docs/latest/process.html#process-memory
    // Set Memory for specific task sizes (1n/2n/4n scheme recommended)
    //withLabel: big_mem   { memory = '16 GB' }
    //withLabel: norm_mem  { memory = '4 GB'  }
    //withLabel: small_mem { memory = '2 GB'  }
    // -*OR*- Set Memory for all processes
    //memory = "16 GB"

    ext.ph = null //Placeholder to prevent errors.
}

params {
    // REQUIRED values to enter (all others should work as default):
    // ref_fasta               (or some other ref-mode/location)
    // treat_fastqs            (input paired-end fastq[.gz] file paths)
    //   [OR fastq_groups]     (mutli-group input paired-end .fastq[.gz] file paths)

    // Automatic alignment reference preparation/usage settings:
    ref_mode       = 'fasta'  // Options: ['name', 'fasta', 'manual']
    ref_fasta      = ''       // REQUIRED: Ex: '/path/to/my/reference.fasta[.gz]'
    // Default Pre-Supplied Normalization library is Ecoli:
    norm_ref_fasta = "${projectDir}/ref_dbs/GCF_000005845.2_ASM584v2_genomic.fna.gz"

    // CnR-flow Input Files:
    //   Provided fastqs must be in glob pattern matching pairs.
    //     Example: ['./relpath/to/base*R{1,2}*.fastq']
    //     Example: ['/abs/path/to/other*R{1,2}*.fastq']

    treat_fastqs   = []    // REQUIRED, Single-group Treatment fastq Pattern
    ctrl_fastqs    = []    //           Single-group Control   fastq pattern

    // Can specify multiple treat/control groups as Groovy mapping.
    //   Specified INSTEAD of treat_fasts/ctrl_fastqs parameters.
    //   Note: There should be only one control sample per group
    //     (after optional lane combination)
```

(continues on next page)

(continued from previous page)

```
    // Example:
    // fastq_groups = [
    //   'group_1_name': ['treat': 'relpath/to/treat1*R{1,2}*',
    //                    'ctrl':  'relpath/to/ctrl1*R{1,2}*'
    //                   ],
    //   'group_2_name': ['treat': ['relpath/to/g2_treat1*R{1,2}*',
    //                              '/abs/path/to/g2_treat2*R{1,2}*'
    //                             ],
    //                    'ctrl':  'relpath/to/g2_ctrl1*R{1,2}*'
    //                   ]
    // ]
    //fastq_groups = []

    // ------- Step-Specific Pipeline Paramaters --------
    // Step Settings:
    do_merge_lanes = true  // Merge sample names differing only by lane-ID (Ex: L001,␣
↪L002)
    do_fastqc      = true  // Perform FastQC Analysis
    do_trim        = true  // Trim tags using Trimmomatic
    do_norm_spike  = true  // Normalize using aligment count to a spike-in reference
    do_norm_cpm    = false // Normalize using millions of reads per sample
    do_make_bigwig = true  // Create UCSC bigWig files from final alignments

    // FastQC Settings:
    fastqc_flags   = ''

    // Alignment Mode Options (params.use_aln_modes) :
    //   "all"            : Use all reads
    //   "all_dedup"      : Use all reads, and remove duplicates.
    //   "less_120"       : Use trimmed reads <= 120 bp
    //   "less_120_dedup" : Use trimmed reads <= 120 bp and remove duplicates.
    //   (Multiple modes can be performed in parallel. Ex: ['all', 'less_120'])

    use_aln_modes  = ["all"]  // Options: ["all", "all_dedup", "less_120", "less_120_
↪dedup"]

    // Peak Caller Options (params.peak_callers):
    //   "macs2" : Call Peaks with Macs2
    //   "seacr" : Call Peaks with SEACR
    //   (Multiple callers can be used in parallel. Ex: ['macs2', 'seacr'])

    peak_callers   = ['macs', 'seacr']  // Options: ['macs', 'seacr']

    // Trimmomatic Trim Settings

    // --Trimmomatic Adapter Path, Downloaded after "nextflow CnR-flow --mode initiate
↪"
    //trimmomatic_adapterpath = "${projectDir}/ref_dbs/trimmomatic_adapters/TruSeq3-
↪PE-2.fa"
    trimmomatic_adapter_mode   = "ILLUMINACLIP:"
    trimmomatic_adapter_params = ":2:15:4:4:true"
    trimmomatic_settings    = "LEADING:20 TRAILING:20 SLIDINGWINDOW:4:15 MINLEN:25"
    trimmomatic_flags       = "-phred33"

    // Bowtie2 Alignment Settings
    aln_ref_flags  = "--local --very-sensitive-local --phred33 -I 10 -X 700 --
↪dovetail --no-unal --no-mixed --no-discordant"
```

(continues on next page)

(continued from previous page)

```
    // Normalization Settings
    aln_norm_flags = params.aln_ref_flags
    norm_scale     = 1000  // Arbitrary value for scaling of normalized counts.
    norm_mode      = 'adj' // Options: ['adj', 'all']

    // CPM Normalization Settings
    norm_cpm_scale = 1000 // Arbitrary value for scaling of normalized counts.

    // Macs2 Settings
    macs_qval      = '0.01'
    macs_flags     = ''

    // SEACR Settings
    seacr_fdr_threshhold = "0.01"
    seacr_norm_mode      = "auto" // Options: "auto", "norm", "non"
    seacr_call_stringent = true
    seacr_call_relaxed   = true

    // ------- General Pipeline Output Paramaters --------
    publish_files    = 'default' // Options: ["minimal", "default", "all"]
    publish_mode     = 'copy'    // Options: ["symlink", "copy"]

    // Name trim guide: ( regex-based )
    //    ~/groovy-slashy-string/  ;  "~" denotes groovy pattern type.
    //    ~/^/ matches beginning   ;   ~/$/ matches end
    trim_name_prefix = ''        // Example: ~/^myprefix./ removes "myprefix." prefix.
    trim_name_suffix = ''        // Example: ~/_mysuffix$/ removes "_mysuffix" suffix.

    // Workflow Output Default Naming Scheme:
    //   Absolute paths for output:
    out_dir          = "${launchDir}/cnr_output"
    refs_dir         = "${launchDir}/cnr_references"
}
```

4.2 Pipe nextflow.config

This nextflow.config file is contains configuration paramaters for pipeline setup.

Listing 2: . . . /CnR-flow/nextflow.config (Pipe Settings)

```
//Daniel Stribling
//Renne Lab, University of Florida
//CnR-flow Configuration File
//
// Config syntax:
// This config file uses the syntax described here:
//   https://www.nextflow.io/docs/latest/config.html
// Primarily, "//" acts as a comment and allows removal/deactivation of lines.
//
// Settings within scopes, Ex:
//   params {
//     setting = value
```

(continues on next page)

(continued from previous page)

```
//     ...
//   }
// are equivalent to:
//   params.setting = value"
//
// Configuration Hierarchy:
//   Different Nextflow configuration files have the precedence order:
//     task-dir  >  pipe-dir  >  user-default  >  pipe-default
//
//   (presumably) This configruation is in the CnR-flow/nextflow.config,
//   and will therefore superceed user and pipe defaults,
//   but be overridden by the task-specific configuration file
//   in the case of conflicting settings.
//   For more detail visit: https://www.nextflow.io/docs/latest/config.html
//
//   Users wishing to modify dependency configuration should specifically
//   direct attention to the conda/module section.
//   Other default parameters are provided below.

// Pipeline Details
manifest {
    author = 'Daniel Stribling, Rolf Renne'
    defaultBranch = 'master'
    description = """\
    CUT&RUN-Flow, A Nextflow pipeline for QC, tag trimming, normalization, and peak
    calling for paired-end sequence data from CUT&RUN experiments.
    """.stripIndent()
    //doi
    homePage = 'http://www.RenneLab.com'
    mainScript = 'CnR-flow.nf'
    name = 'CUT&RUN-Flow'
    nextflowVersion = '>=20.10.6'
    version = '0.11-dev'
}

// Process Settings (For use of PBS, SLURM, etc)
process {
    // --Executor, see: https://www.nextflow.io/docs/latest/executor.html
    //executor = 'slurm'  // for running processes using SLURM (Default: 'local')
    // Process Walltime, See https://www.nextflow.io/docs/latest/process.html#process-
↪time
    //time = '12h'
    // Process CPUs, See https://www.nextflow.io/docs/latest/process.html#cpus
    //cpus = 8
    //
    // Memory: See https://www.nextflow.io/docs/latest/process.html#process-memory
    // Set Memory for specific task sizes (1n/2n/4n scheme recommended)
    //withLabel: big_mem   { memory = '16 GB' }
    //withLabel: norm_mem  { memory = '4 GB'  }
    //withLabel: small_mem { memory = '2 GB'  }
    // -*OR*- Set Memory for all processes
    //memory = "16 GB"

    ext.ph = null //Placeholder to prevent errors.
}
```

(continues on next page)

(continued from previous page)

```
// ------- Dependency Configuration --------
// Standard configuration is setup using conda.
// Docker / Singularity execution is preferred where available.
// Environment Modules also supported.
// Where defined, dependencies are used in order [dep]_container > [dep]_module > [dep]_conda
profiles {
    standard {
        // Dependency Configuration Using Anaconda
        params.facount_conda          = 'bioconda::ucsc-facount=366'
        params.bowtie2_conda          = 'bioconda::bowtie2=2.4.4 conda-forge::libgcc-ng=9.3'
        params.fastqc_conda           = 'bioconda::fastqc=0.11.9'
        params.trimmomatic_conda      = 'bioconda::trimmomatic=0.39'
        params.bedtools_conda         = 'bioconda::bedtools=2.29.2'
        params.macs2_conda            = 'bioconda::macs2=2.2.6'
        params.seacr_conda            = "r=3.6.0 bioconda::seacr=1.3 ${params.bedtools_conda}"
        params.samtools_conda         = 'bioconda::samtools=1.9'
        params.bedgraphtobigwig_conda = 'conda-forge::libpng conda-forge::libuuid conda-forge::mysql-connector-c conda-forge::openssl conda-forge::zlib bioconda::ucsc-bedgraphtobigwig=377'

        params.trimmomatic_adapterpath = "${projectDir}/ref_dbs/trimmomatic_adapters/TruSeq3-PE-2.fa"
    }

    singularity {
        // Dependency Configuration Using Containers with Singularity:
        singularity.enabled = true
        params.facount_container           = "docker://quay.io/biocontainers/ucsc-facount:366--h5eb252a_0"
        params.bowtie2_container           = "docker://quay.io/biocontainers/bowtie2:2.4.4--py38he5f0661_1"
        params.bowtie2_samtools_container  = "docker://quay.io/biocontainers/mulled-v2-c742dccc9d8fabfcff2af0d8d6799dbc711366cf:b6524911af823c7c52518f6c886b86916d062940-0"
        // Mulled: bowtie2=2.4.4,samtools=1.13
        params.fastqc_container            = "docker://quay.io/biocontainers/fastqc:0.11.9--hdfd78af_1"
        params.trimmomatic_container       = "docker://quay.io/biocontainers/trimmomatic:0.39--1"
        params.macs2_container             = "docker://quay.io/biocontainers/macs2:2.2.6--py37h516909a_0"
        params.seacr_container             = "docker://quay.io/biocontainers/seacr:1.3--hdfd78af_2"
        params.samtools_container          = "docker://quay.io/biocontainers/samtools:1.13--h8c37831_0"
        params.bedgraphtobigwig_container  = "docker://quay.io/biocontainers/ucsc-bedgraphtobigwig:377--h0b8a92a_2"
        params.samtools_bedtools_container = "docker://quay.io/biocontainers/mulled-v2-fc325951871d402a00bdf9d0e712a5b81b8e0cb3:38034b9703d6561a40bcaf2f1ec16f8b158fde97-0"
        // Mulled: samtools=1.14.0,bedtools=2.30.0,ucsc-facount=377,python=3.8
        params.samtools_facount_container  = "${params.samtools_bedtools_container}"
        params.bedtools_container          = "${params.samtools_bedtools_container}"
```

(continues on next page)

(continued from previous page)

```
        params.trimmomatic_adapterpath     = "/usr/local/share/trimmomatic/adapters/
↪TruSeq3-PE-2.fa"
    }

    docker {
        // Dependency Configuration Using Containers with Docker:
        docker.enabled = true
        docker.runOptions = "-v ${launchDir}:${launchDir}"
        //docker.runOptions = "-v ${params.refs_dir}:${params.refs_dir}"

        params.facount_container           = "quay.io/biocontainers/ucsc-facount:366--
↪h5eb252a_0"
        params.bowtie2_container           = "quay.io/biocontainers/bowtie2:2.4.4--
↪py38he5f0661_1"
        params.bowtie2_samtools_container  = "quay.io/biocontainers/mulled-v2-
↪c742dccc9d8fabfcff2af0d8d6799dbc711366cf:b6524911af823c7c52518f6c886b86916d062940-0"
        // Mulled: bowtie2=2.4.4,samtools=1.13
        params.fastqc_container            = "quay.io/biocontainers/fastqc:0.11.9--
↪hdfd78af_1"
        params.trimmomatic_container       = "quay.io/biocontainers/trimmomatic:0.39--
↪1"
        params.macs2_container             = "quay.io/biocontainers/macs2:2.2.6--
↪py37h516909a_0"
        params.seacr_container             = "quay.io/biocontainers/seacr:1.3--
↪hdfd78af_2"
        params.samtools_container          = "quay.io/biocontainers/samtools:1.13--
↪h8c37831_0"
        params.bedgraphtobigwig_container  = "quay.io/biocontainers/ucsc-
↪bedgraphtobigwig:377--h0b8a92a_2"
        params.samtools_bedtools_container = "quay.io/biocontainers/mulled-v2-
↪fc325951871d402a00bdf9d0e712a5b81b8e0cb3:38034b9703d6561a40bcaf2f1ec16f8b158fde97-0"
        // Mulled: samtools=1.14.0,bedtools=2.30.0,ucsc-facount=377,python=3.8
        params.samtools_facount_container  = "${params.samtools_bedtools_container}"
        params.bedtools_container          = "${params.samtools_bedtools_container}"

        params.trimmomatic_adapterpath     = "/usr/local/share/trimmomatic/adapters/
↪TruSeq3-PE-2.fa"
    }

    module {
        // Dependency Configuration Using Environment Modules
        //   (values will vary depending on system)
        params.facount_module          = ""  // Ex: "ucsc/20200320"
        params.bowtie2_module          = ""  // Ex: "bowtie2/2.3.5.1"
        params.fastqc_module           = ""  // Ex: "fastqc/0.11.7"
        params.trimmomatic_module      = ""  // Ex: "trimmomatic/0.39"
        params.bedtools_module         = ""  // Ex: "bedtools/2.29.2"
        params.macs2_module            = ""  // Ex: "macs/2.2.7.1"
        params.seacr_module            = ""  // Ex: "R/4.0 seacr/1.3 ${params.
↪bedtools_module}"
        params.bedgraphtobigwig_module = ""  // Ex: "ucsc/20200320"

        params.trimmomatic_adapterpath = "${projectDir}/ref_dbs/trimmomatic_adapters/
↪TruSeq3-PE-2.fa"
    }
}
```

(continues on next page)

(continued from previous page)

```
params {
    // System Call Settings
    //   Call executed on the system, defaults assume that each tool is available
    //     on the system PATH
    //   Can replace with direct path as desired:
    //   Ex:
    //        samtools_call = "samtools"
    //     or samtools_call = "/path/to/samtools/dir/samtools"
    java_call             = "java"
    bowtie2_build_call    = "bowtie2-build"
    facount_call          = "faCount"
    samtools_call         = "samtools"
    fastqc_call           = "fastqc"
    trimmomatic_call      = "trimmomatic"
    bowtie2_call          = "bowtie2"
    bedtools_call         = "bedtools"
    macs2_call            = "macs2"
    bedgraphtobigwig_call = "bedGraphToBigWig"
    seacr_call            = "SEACR_1.3.sh"
    seacr_R_script        = "SEACR_1.3.R"


    // ------- Step-Specific Pipeline Paramaters --------
    // Step Settings:
    do_merge_lanes = true  // Merge sample names differing only by lane-ID (Ex: L001,␣
↪L002)
    do_fastqc      = true  // Perform FastQC Analysis
    do_trim        = true  // Trim tags using Trimmomatic
    do_norm_spike  = true  // Normalize using aligment count to a spike-in reference
    do_norm_cpm    = false // Normalize using millions of reads per sample
    do_make_bigwig = true  // Create UCSC bigWig files from final alignments

    // FastQC Settings:
    fastqc_flags   = ''

    // Alignment Mode Options (params.use_aln_modes) :
    //   "all"            : Use all reads
    //   "all_dedup"      : Use all reads, and remove duplicates.
    //   "less_120"       : Use trimmed reads <= 120 bp
    //   "less_120_dedup" : Use trimmed reads <= 120 bp and remove duplicates.
    //   (Multiple modes can be performed in parallel. Ex: ['all', 'less_120'])

    use_aln_modes  = ["all"]  // Options: ["all", "all_dedup", "less_120", "less_120_
↪dedup"]

    // Peak Caller Options (params.peak_callers):
    //   "macs2" : Call Peaks with Macs2
    //   "seacr" : Call Peaks with SEACR
    //   (Multiple callers can be used in parallel. Ex: ['macs2', 'seacr'])

    peak_callers   = ['macs', 'seacr']  // Options: ['macs', 'seacr']

    // Trimmomatic Trim Settings

    // --Trimmomatic Adapter Path, Downloaded after "nextflow CnR-flow --mode initiate
↪"
    //trimmomatic_adapterpath = "${projectDir}/ref_dbs/trimmomatic_adapters/TruSeq3-
↪PE-2.fa"
```

(continues on next page)

(continued from previous page)

```
    trimmomatic_adapter_mode   = "ILLUMINACLIP:"
    trimmomatic_adapter_params = ":2:15:4:4:true"
    trimmomatic_settings    = "LEADING:20 TRAILING:20 SLIDINGWINDOW:4:15 MINLEN:25"
    trimmomatic_flags       = "-phred33"

    // Bowtie2 Alignment Settings
    aln_ref_flags  = "--local --very-sensitive-local --phred33 -I 10 -X 700 --
↪dovetail --no-unal --no-mixed --no-discordant"

    // Normalization Settings
    aln_norm_flags = params.aln_ref_flags
    norm_scale     = 1000  // Arbitrary value for scaling of normalized counts.
    norm_mode      = 'adj' // Options: ['adj', 'all']

    // CPM Normalization Settings
    norm_cpm_scale = 1000 // Arbitrary value for scaling of normalized counts.

    // Macs2 Settings
    macs_qval      = '0.01'
    macs_flags     = ''

    // SEACR Settings
    seacr_fdr_threshhold = "0.01"
    seacr_norm_mode      = "auto" // Options: "auto", "norm", "non"
    seacr_call_stringent = true
    seacr_call_relaxed   = true

    // ------- General Pipeline Output Paramaters --------
    publish_files    = 'default' // Options: ["minimal", "default", "all"]
    publish_mode     = 'copy'    // Options: ["symlink", "copy"]

    // Name trim guide: ( regex-based )
    //    ~/groovy-slashy-string/  ;  "~" denotes groovy pattern type.
    //    ~/^/ matches beginning   ;   ~/$/ matches end
    trim_name_prefix = ''        // Example: ~/^myprefix./ removes "myprefix." prefix.
    trim_name_suffix = ''        // Example: ~/_mysuffix$/ removes "_mysuffix" suffix.

    // Workflow Output Default Naming Scheme:
    //   Absolute paths for output:
    out_dir          = "${launchDir}/cnr_output"
    refs_dir         = "${launchDir}/cnr_references"
    //   Subdirectory Settigns:
    log_dir          = 'logs'
    merge_fastqs_dir = 'S0_B_merged_reads'
    fastqc_pre_dir   = 'S0_C_FastQC_pre'
    trim_dir         = 'S1_A_fastq_trimomatic'
    fastqc_post_dir  = 'S1_C_FastQC_post'
    aln_dir_ref      = 'S2_A_aln_ref'
    aln_dir_mod      = 'S2_B_aln_mod'
    aln_dir_bdg      = 'S2_C_aln_bdg'
    aln_dir_spike    = 'S3_A_aln_spikein'
    aln_dir_norm     = 'S3_B_aln_norm'
    aln_dir_norm_cpm = 'S3_X_aln_normCPM'
    aln_bigwig_dir   = 'S4_A_aln_bigWig'
    peaks_dir_macs   = 'S5_A_peaks_macs'
    peaks_dir_seacr  = 'S5_B_peaks_seacr'
    prep_bt2db_suf   = 'bt2_db'
```

(continues on next page)

(continued from previous page)

```
}

// Locations to store conda and/or singularity environments for reuse.
conda.cacheDir       = "${projectDir}/envs_conda/"
singularity.cacheDir = "${projectDir}/envs_singularity/"
singularity.enabled  = false

// ------- Individual Task Settings, Included for Completeness --------

    //params {
    // REQUIRED values to enter (all others should work as default):
    // ref_fasta               (or some other ref-mode/location)
    // treat_fastqs            (input paired-end fastq[.gz] file paths)
    //   [OR fastq_groups]     (mutli-group input paired-end .fastq[.gz] file paths)

    // Automatic alignment reference preparation/usage settings:
    //ref_mode       = 'fasta'  // Options: ['name', 'fasta', 'manual']
    //ref_fasta      = ''       // REQUIRED: Ex: '/path/to/my/reference.fasta[.gz]'
    // Default Pre-Supplied Normalization library is Ecoli:
    //norm_ref_fasta = "${projectDir}/ref_dbs/GCF_000005845.2_ASM584v2_genomic.fna.gz"

    // CnR-flow Input Files:
    //   Provided fastqs must be in glob pattern matching pairs.
    //     Example: ['./relpath/to/base*R{1,2}*.fastq']
    //     Example: ['/abs/path/to/other*R{1,2}*.fastq']

    //treat_fastqs   = []    // REQUIRED, Single-group Treatment fastq Pattern
    //ctrl_fastqs    = []    //           Single-group Control   fastq pattern

    // Can specify multiple treat/control groups as Groovy mapping.
    //   Specified INSTEAD of treat_fasts/ctrl_fastqs parameters.
    //   Note: There should be only one control sample per group
    //     (after optional lane combination)
    // Example:
    // fastq_groups = [
    //   'group_1_name': ['treat': 'relpath/to/treat1*R{1,2}*',
    //                    'ctrl':  'relpath/to/ctrl1*R{1,2}*'
    //                   ],
    //   'group_2_name': ['treat': ['relpath/to/g2_treat1*R{1,2}*',
    //                              '/abs/path/to/g2_treat2*R{1,2}*'
    //                             ],
    //                    'ctrl':  'relpath/to/g2_ctrl1*R{1,2}*'
    //                   ]
    // ]
    //fastq_groups = []
```

4.3 Task nextflow.config (no comments)

This nextflow.config file is equivalent to *Task nextflow.config* (page 27), but with comments removed.

Listing 3: .../my_task_dir/nextflow.config.nodoc (Task Settings, Without Comments)

```
process {
    //executor = 'slurm'  // for running processes using SLURM (Default: 'local')
    //time = '12h'
    //cpus = 8
    //withLabel: big_mem   { memory = '16 GB' }
    //withLabel: norm_mem  { memory = '4 GB'  }
    //withLabel: small_mem { memory = '2 GB'  }
    //memory = "16 GB"
    ext.ph = null //Placeholder to prevent errors.
}
params {
    ref_mode       = 'fasta'  // Options: ['name', 'fasta', 'manual']
    ref_fasta      = ''       // REQUIRED: Ex: '/path/to/my/reference.fasta[.gz]'
    norm_ref_fasta = "${projectDir}/ref_dbs/GCF_000005845.2_ASM584v2_genomic.fna.gz"
    treat_fastqs   = []    // REQUIRED, Single-group Treatment fastq Pattern
    ctrl_fastqs    = []    //           Single-group Control   fastq pattern
    //fastq_groups = []
    do_merge_lanes = true  // Merge sample names differing only by lane-ID (Ex: L001,␣
↪L002)
    do_fastqc      = true  // Perform FastQC Analysis
    do_trim        = true  // Trim tags using Trimmomatic
    do_norm_spike  = true  // Normalize using aligment count to a spike-in reference
    do_norm_cpm    = false // Normalize using millions of reads per sample
    do_make_bigwig = true  // Create UCSC bigWig files from final alignments
    fastqc_flags   = ''
    use_aln_modes  = ["all"]  // Options: ["all", "all_dedup", "less_120", "less_120_
↪dedup"]
    peak_callers   = ['macs', 'seacr']  // Options: ['macs', 'seacr']
    //trimmomatic_adapterpath = "${projectDir}/ref_dbs/trimmomatic_adapters/TruSeq3-
↪PE-2.fa"
    trimmomatic_adapter_mode   = "ILLUMINACLIP:"
    trimmomatic_adapter_params = ":2:15:4:4:true"
    trimmomatic_settings    = "LEADING:20 TRAILING:20 SLIDINGWINDOW:4:15 MINLEN:25"
    trimmomatic_flags       = "-phred33"
    aln_ref_flags  = "--local --very-sensitive-local --phred33 -I 10 -X 700 --
↪dovetail --no-unal --no-mixed --no-discordant"
    aln_norm_flags = params.aln_ref_flags
    norm_scale     = 1000  // Arbitrary value for scaling of normalized counts.
    norm_mode      = 'adj' // Options: ['adj', 'all']
    norm_cpm_scale = 1000 // Arbitrary value for scaling of normalized counts.
    macs_qval      = '0.01'
    macs_flags     = ''
    seacr_fdr_threshhold = "0.01"
    seacr_norm_mode      = "auto" // Options: "auto", "norm", "non"
    seacr_call_stringent = true
    seacr_call_relaxed   = true
    publish_files    = 'default' // Options: ["minimal", "default", "all"]
    publish_mode     = 'copy'    // Options: ["symlink", "copy"]
    trim_name_prefix = ''        // Example: ~/^myprefix./ removes "myprefix." prefix.
    trim_name_suffix = ''        // Example: ~/_mysuffix$/ removes "_mysuffix" suffix.
```

(continues on next page)

(continued from previous page)

```
    out_dir          = "${launchDir}/cnr_output"
    refs_dir         = "${launchDir}/cnr_references"
}
```

4.4 Task nextflow.config (minimal)

This nextflow.config file is equivalent to the task and executor settings only from *Task nextflow.config* (page 27), with comments removed.

Listing 4: .../my_task_dir/nextflow.config.minimal (Task Settings, Minimal Without Comments)

```
process {
    //executor = 'slurm'  // for running processes using SLURM (Default: 'local')
    //time = '12h'
    //cpus = 8
    //withLabel: big_mem   { memory = '16 GB' }
    //withLabel: norm_mem  { memory = '4 GB'  }
    //withLabel: small_mem { memory = '2 GB'  }
    //memory = "16 GB"
    ext.ph = null //Placeholder to prevent errors.
}
params {
    ref_mode       = 'fasta'  // Options: ['name', 'fasta', 'manual']
    ref_fasta      = ''       // REQUIRED: Ex: '/path/to/my/reference.fasta[.gz]'
    norm_ref_fasta = "${projectDir}/ref_dbs/GCF_000005845.2_ASM584v2_genomic.fna.gz"
    treat_fastqs   = []    // REQUIRED, Single-group Treatment fastq Pattern
    ctrl_fastqs    = []    //           Single-group Control   fastq pattern
    //fastq_groups = []
}
```

CHAPTER

FIVE

REFERENCES

5.1 References (BibTeX)

Download Citations.bib

```
@article{di2017nextflow,
  title={Nextflow enables reproducible computational workflows},
  author={Di Tommaso, Paolo and Chatzou, Maria and Floden, Evan W and Barja,␣
↪Pablo Prieto and Palumbo, Emilio and Notredame, Cedric},
  journal={Nature biotechnology},
  volume={35},
  number={4},
  pages={316--319},
  year={2017},
  publisher={Nature Publishing Group}
}

@article{gruning2018bioconda,
  title={Bioconda: sustainable and comprehensive software distribution for␣
↪the life sciences},
  author={Gr{\"u}ning, Bj{\"o}rn and Dale, Ryan and Sj{\"o}din, Andreas and␣
↪Chapman, Brad A and Rowe, Jillian and Tomkins-Tinch, Christopher H and␣
↪Valieris, Renan and K{\"o}ster, Johannes},
  journal={Nature methods},
  volume={15},
  number={7},
  pages={475--476},
  year={2018},
  publisher={Nature Publishing Group}
}

@article{zhu2019cut,
  title={CUT\&RUNTools: a flexible pipeline for CUT\&RUN processing and␣
↪footprint analysis},
  author={Zhu, Qian and Liu, Nan and Orkin, Stuart H and Yuan, Guo-Cheng},
  journal={Genome biology},
  volume={20},
  number={1},
  pages={192},
  year={2019},
  publisher={Springer}
```

(continues on next page)

(continued from previous page)

```
}

@article{meers2019peak,
  title={Peak calling by Sparse Enrichment Analysis for CUT\&RUN chromatin␣
↪profiling},
  author={Meers, Michael P and Tenenbaum, Dan and Henikoff, Steven},
  journal={Epigenetics \& chromatin},
  volume={12},
  number={1},
  pages={42},
  year={2019},
  publisher={Springer}
}

@Manual{,
  title = {R: A Language and Environment for Statistical Computing},
  author = {{R Core Team}},
  organization = {R Foundation for Statistical Computing},
  address = {Vienna, Austria},
  year = {2017},
  url = {https://www.R-project.org/},
}

@article{10.1093/bioinformatics/btx192,
    author = {da Veiga Leprevost, Felipe and Grüning, Björn A and Alves␣
↪Aflitos, Saulo and Röst, Hannes L and Uszkoreit, Julian and Barsnes,␣
↪Harald and Vaudel, Marc and Moreno, Pablo and Gatto, Laurent and Weber,␣
↪Jonas and Bai, Mingze and Jimenez, Rafael C and Sachsenberg, Timo and␣
↪Pfeuffer, Julianus and Vera Alvarez, Roberto and Griss, Johannes and␣
↪Nesvizhskii, Alexey I and Perez-Riverol, Yasset},
    title = "{BioContainers: an open-source and community-driven framework␣
↪for software standardization}",
    journal = {Bioinformatics},
    volume = {33},
    number = {16},
    pages = {2580-2582},
    year = {2017},
    month = {03},
    abstract = "{BioContainers (biocontainers.pro) is an open-source and␣
↪community-driven framework which provides platform independent executable␣
↪environments for bioinformatics software. BioContainers allows labs of all␣
↪sizes to easily install bioinformatics software, maintain multiple␣
↪versions of the same software and combine tools into powerful analysis␣
↪pipelines. BioContainers is based on popular open-source projects Docker␣
↪and rkt frameworks, that allow software to be installed and executed under␣
↪an isolated and controlled environment. Also, it provides infrastructure␣
↪and basic guidelines to create, manage and distribute bioinformatics␣
↪containers with a special focus on omics technologies. These containers␣
↪can be integrated into more comprehensive bioinformatics pipelines and␣
↪different architectures (local desktop, cloud environments or HPC␣
↪clusters).The software is freely available at github.com/BioContainers/.}",
    issn = {1367-4803},
    doi = {10.1093/bioinformatics/btx192},
    url = {https://doi.org/10.1093/bioinformatics/btx192},
    eprint = {https://academic.oup.com/bioinformatics/article-pdf/33/16/2580/
↪25163480/btx192.pdf},
}
```

(continues on next page)

(continued from previous page)

```
@article{langmead2012fast,
  title={Fast gapped-read alignment with Bowtie 2},
  author={Langmead, Ben and Salzberg, Steven L},
  journal={Nature methods},
  volume={9},
  number={4},
  pages={357},
  year={2012},
  publisher={Nature Publishing Group}
}

@article{kent2002human,
  title={The human genome browser at UCSC},
  author={Kent, W James and Sugnet, Charles W and Furey, Terrence S and
↪Roskin, Krishna M and Pringle, Tom H and Zahler, Alan M and Haussler,
↪David},
  journal={Genome research},
  volume={12},
  number={6},
  pages={996--1006},
  year={2002},
  publisher={Cold Spring Harbor Lab}
}

@article{li2009sequence,
  title={The sequence alignment/map format and SAMtools},
  author={Li, Heng and Handsaker, Bob and Wysoker, Alec and Fennell, Tim and
↪Ruan, Jue and Homer, Nils and Marth, Gabor and Abecasis, Goncalo and
↪Durbin, Richard},
  journal={Bioinformatics},
  volume={25},
  number={16},
  pages={2078--2079},
  year={2009},
  publisher={Oxford University Press}
}

@misc{andrews2015quality,
  title={A quality control tool for high throughput sequence data. 2010},
  author={Andrews, Simon and FastQC, A},
  year={2015}
}

@article{bolger2014trimmomatic,
  title={Trimmomatic: a flexible trimmer for Illumina sequence data},
  author={Bolger, Anthony M and Lohse, Marc and Usadel, Bjoern},
  journal={Bioinformatics},
  volume={30},
  number={15},
  pages={2114--2120},
  year={2014},
  publisher={Oxford University Press}
}

@article{quinlan2010bedtools,
  title={BEDTools: a flexible suite of utilities for comparing genomic
↪features},
```

(continues on next page)

(continued from previous page)

```
  author={Quinlan, Aaron R and Hall, Ira M},
  journal={Bioinformatics},
  volume={26},
  number={6},
  pages={841--842},
  year={2010},
  publisher={Oxford University Press}
}

@article{kent2010bigwig,
  title={BigWig and BigBed: enabling browsing of large distributed datasets},
  author={Kent, W James and Zweig, Ann S and Barber, G and Hinrichs, Angie S␣
↪and Karolchik, Donna},
  journal={Bioinformatics},
  volume={26},
  number={17},
  pages={2204--2207},
  year={2010},
  publisher={Oxford University Press}
}

@article{zhang2008model,
  title={Model-based analysis of ChIP-Seq (MACS)},
  author={Zhang, Yong and Liu, Tao and Meyer, Clifford A and Eeckhoute, J{\
↪'e}r{\^o}me and Johnson, David S and Bernstein, Bradley E and Nusbaum,␣
↪Chad and Myers, Richard M and Brown, Myles and Li, Wei and others},
  journal={Genome biology},
  volume={9},
  number={9},
  pages={1--9},
  year={2008},
  publisher={BioMed Central}
}
```

CHAPTER

SIX

ABOUT

6.1 Renne Lab

Principal Investigator: Rolf Renne
Henry E. Innes Professor of Cancer Research
University of Florida
UF Health Cancer Center
UF Department of Molecular Genetics and Microbiology
UF Genetics Institute
http://www.RenneLab.com

6.2 Lead Developer

Dan Stribling <ds@ufl.edu>
https://github.com/dstrib
https://orcid.org/0000-0002-0649-9506
University of Florida, Renne Lab

6.3 Changelog

v0.11-dev - ongoing:

- Refinements
- Bugfixes
- Added additional template nextflow.config files for task
- Added macOS support for Conda machines (for all dependencies)
- Added input data file integrity checks to Merge process
- Added internal output checks for early error catching
- Updated Conda Bowtie2 Version to 2.4.4 and added component dependency
- Merge “validate” and “validate-all” modes into “validate” mode
- Remove “dry-run” mode as low utility per complexity
- Switch to nextflow-implementation of memory handling

- Add support for task execution via Docker / Singularity containers
- Corrected error in handling of normalized alignment files
- Moved SEACR to external dependency config
- Removed CUTRUNTools:kseqtest from pipeline
- Migrate aumotated testing from TravisCI to CircleCI
- Add Beta support for Singularity/Docker dependency configurations
- Add CircleCI Docker dependency config testing
- Add default Singularity and Docker support configs via profiles

v0.10 - 2020-09-04:

- Refinements
- Changed verbose task logging to implementation with "beforeScript"
- Complete Initial Documentation
- Moved macs2 peak-calling out of alpha testing
- "Reordered" output step directories
- Tuned resource usage defaults
- Added process memory usage categories
- Move UCSC/Kent tools to external dependency setup
- Added bigWig track format creation step
- Overhauled alignment modification step
- Removed Picard dependency
- Changed (non-track) alignment output files to CRAM (compressed BAM)

v0.09:

- Refinements
- Added one-step database preparation
- Implemented 'list_refs' mode
- Implemented automatic reference paramater finding
- Shifted paramaters to config files
- Implemented initiate mode
- Added minimal documentation
- Added UCSC (faCount) automated installation
- Added automated acquisition of Trimmomatic adapters
- Implemented MACS2 peak calling
- Added autodetection of tag sequence length

v0.08:

- Initial Github Upload

BIBLIOGRAPHY

[Meers2019] Meers, Michael P., et al. "Improved CUT&RUN chromatin profiling tools." Elife 8 (2019): e46314.

[Nextflow_Citation] Di Tommaso, Paolo, et al. "Nextflow enables reproducible computational workflows." Nature biotechnology 35.4 (2017): 316-319.

[Bioconda_Citation] Grüning, Björn, et al. "Bioconda: sustainable and comprehensive software distribution for the life sciences." Nature methods 15.7 (2018): 475-476.

[CUTRUNTools_Citation] Zhu, Qian, et al. "CUT&RUNTools: a flexible pipeline for CUT&RUN processing and footprint analysis." Genome biology 20.1 (2019): 192.

[SEACR_Citation] Meers, Michael P., Dan Tenenbaum, and Steven Henikoff. "Peak calling by Sparse Enrichment Analysis for CUT&RUN chromatin profiling." Epigenetics & chromatin 12.1 (2019): 42.

[R_Citation] R Core Team (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/

[BioContainers_Citation] Felipe da Veiga Leprevost, Björn A Grüning, Saulo Alves Aflitos, Hannes L Röst, Julian Uszkoreit, Harald Barsnes, Marc Vaudel, Pablo Moreno, Laurent Gatto, Jonas Weber, Mingze Bai, Rafael C Jimenez, Timo Sachsenberg, Julianus Pfeuffer, Roberto Vera Alvarez, Johannes Griss, Alexey I Nesvizhskii, Yasset Perez-Riverol, BioContainers: an open-source and community-driven framework for software standardization, Bioinformatics, Volume 33, Issue 16, 15 August 2017, Pages 2580–2582, https://doi.org/10.1093/bioinformatics/btx192

[Bowtie2_Citation] Langmead, Ben, and Steven L. Salzberg. "Fast gapped-read alignment with Bowtie 2." Nature methods 9.4 (2012): 357.

[faCount_Citation] Kent WJ, Sugnet CW, Furey TS, Roskin KM, Pringle TH, Zahler AM, Haussler D. The human genome browser at UCSC. Genome Res. 2002 Jun;12(6):996-1006.

[Samtools_Citation] Li H.*, Handsaker B.*, Wysoker A., Fennell T., Ruan J., Homer N., Marth G., Abecasis G., Durbin R. and 1000 Genome Project Data Processing Subgroup (2009) The Sequence alignment/map (SAM) format and SAMtools. Bioinformatics, 25, 2078-9. [PMID: 19505943]

[FastQC_Citation] Andrews, S. (2010). FastQC: A Quality Control Tool for High Throughput Sequence Data [Online]. Available online at: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

[Trimmomatic_Citation] Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: A flexible trimmer for Illumina Sequence Data. Bioinformatics, btu170.

[bedtools_Citation] Quinlan AR and Hall IM, 2010. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics. 26, 6, pp. 841–842.

[bedGraphToBigWig_Citation] BigWig and BigBed tools: Kent WJ, Zweig AS, Barber G, Hinrichs AS, Karolchik D. BigWig and BigBed: enabling browsing of large distributed data sets. Bioinformatics. 2010 Sep 1;26(17):2204-7.

[MACS2_Citation] Zhang et al. Model-based Analysis of ChIP-Seq (MACS). Genome Biol (2008) vol. 9 (9) pp. R137