MuG - FASTQ Pipelines Documentation

Release 0.1

Mark McDowall

Nov 19, 2018

Contents:

CHAPTER 1

Requirements and Installation

Source (github)

1.1 Requirements

1.1.1 Software

- Python 2.7.10+ or 3.6+
- bedtools (specifically bamtobed)
- bedToBigBed - http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/
- wigToBigWig - http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/
- BioBamBam2
- Bowtie2
- BWA
- FastQC
- GEMtools
- HDF5
- iNPS
- Kallisto
- libmaus2
- pyenv
- R 2.9.1+
- SAMtools

- MCL
- pigz

1.1.2 Python Modules

- numpy
- h5py
- pysam
- pyBigWig
- scipy
- matplotlib
- rpy2

Instructions for the followig modules are listed in the installation section. All other python modules should be installed with pip prior to the following libraries.

- BS-Seeker2
- TADbit

1.2 Installation

For a guide to the full installation procedure the *Full Installation*.

Directly from GitHub:

```
cd ${HOME}/code

git clone https://github.com/Multiscale-Genomics/mg-process-fastq.git

cd mg-process-fastq
```

Create the Python environment

```
pyenv-virtualenv 2.7.10 mg-process-fastq
pip install --editable .
```

Install the pyTADbit modules

```
cd ${HOME}/lib
wget https://github.com/3DGenomes/tadbit/archive/master.zip -O tadbit.zip
unzip tadbit.zip
cd tadbit-master

pyenv activate mg-process-fastq
pip install .
```

Check out the following software for use by the process_wgbs.py pipeline:

```
cd cd ${HOME}/lib
gti clone https://github.com/BSSeeker/BSseeker2.git

cd ${HOME}/code
cd mg-process-fastq
ln -s $code_root/bs_align bs_align
ln -s $code_root/bs_index bs_index
ln -s $code_root/bs_utils bs_utils

cd cd ${HOME}/code/mg-process-fastq/tool
ln -s $code_root/FilterReads.py FilterReads.py
```

1.3 Documentation

To build the documentation:

```
pip install Sphinx
pip install sphinx-autobuild
cd docs
make html
```

CHAPTER 2

Full Installation

The following document is for the full installation of all software required by the mg-process-fastq module and all programmes that it uses. The document has been written with Ubuntu Linux in mind, although many of the commands are cross platform (*nix) complient.

If you already have certain packages installed feel free to skip over certain steps. Likewise the bin, lib and code directories are relative to the home dir; if this is not the case for your system then make the required changes when running these commands.

2.1 Setup the System Environment

```
sudo apt-get install -y make build-essential libssl-dev zlib1g-dev       \\
libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm libncurses5-dev \\
libncursesw5-dev xz-utils tk-dev unzip mcl libgtk2.0-dev r-base-core     \\
libcurl4-gnutls-dev python-rpy2 git libtbb2 pigz liblzma-dev libhdf5-dev \\
texlive-latex-base tree libblas-dev liblapack-dev

cd ${HOME}
mkdir bin lib code
echo 'export PATH="${HOME}/bin:$PATH"' >> ~/.bash_profile
```

2.2 Setup pyenv and pyenv-virtualenv

This is required for managing the version of Python and the installation environment for the Python modules so that they can be installed in the user space.

```
git clone https://github.com/pyenv/pyenv.git ~/.pyenv
echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bash_profile
echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bash_profile
echo 'eval "$(pyenv init -)"' >> ~/.bash_profile
```

(continues on next page)

(continued from previous page)

```

# Add the .bash_profile to your .bashrc file
echo 'source ~/.bash_profile"' >> ~/.bashrc

git clone https://github.com/pyenv/pyenv-virtualenv.git ${PYENV_ROOT}/plugins/pyenv-
↪virtualenv

pyenv install 2.7.12
pyenv virtualenv 2.7.12 mg-process-fastq

# Python 3 environment required by iNPS
pyenv install 3.5.3
ln -s ${HOME}/.pyenv/versions/3.5.3/bin/python ${HOME}/bin/py3
```

2.3 Installation Process

2.3.1 UCSC Tools

```
cd ${HOME}/lib
wget http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/bedToBigBed
wget http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/wigToBigWig

wget http://hgdownload.soe.ucsc.edu/admin/exe/linux.x86_64/faToTwoBit
wget http://hgdownload.soe.ucsc.edu/admin/exe/linux.x86_64/twoBitInfo

chmod +x bedToBigBed wigToBigWig faToTwoBit twoBitInfo
```

2.3.2 BioBamBam2

BioBamBam is used for the filtering of aligned reads as part of the ChIP-seq pipeline. It also requires the libmaus2 package to be installed.

```
cd ${HOME}/lib
git clone https://github.com/gt1/libmaus2.git
cd libmaus2
libtoolize
aclocal
autoheader
automake --force-missing --add-missing
autoconf
./configure --prefix=${HOME}/lib/libmaus2
make
make install

cd ${HOME}/lib
git clone https://github.com/gt1/biobambam2.git
cd biobambam2
autoreconf -i -f
./configure --with-libmaus2=${HOME}/lib/libmaus2 --prefix=${HOME}/lib/biobambam2
make install
```

2.3.3 Bowtie2 Aligner

```
cd ${HOME}/lib
wget --max-redirect 1 https://downloads.sourceforge.net/project/bowtie-bio/bowtie2/2.
↪3.4/bowtie2-2.3.4-linux-x86_64.zip
unzip bowtie2-2.3.4-linux-x86_64.zip
```

2.3.4 BWA Sequence Aligner

```
cd ${HOME}/lib
git clone https://github.com/lh3/bwa.git
cd bwa
make
```

2.3.5 FastQC

```
cd ${HOME}/lib
wget http://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.5.zip
unzip fastqc_v0.11.5.zip
cd FastQC/
chmod 755 fastqc
```

2.3.6 GEM Sequence Aligner

```
cd ${HOME}/lib
wget http://barnaserver.com/gemtools/releases/GEMTools-static-core2-1.7.1.tar.gz
tar -xzf GEMTools-static-core2-1.7.1.tar.gz
```

2.3.7 iNPS Peak Caller

```
cd ${HOME}/lib
mkdir iNPS
cd iNPS
wget http://www.picb.ac.cn/hanlab/files/iNPS_V1.2.2.zip
unzip iNPS_V1.2.2.zip

cd ${HOME}/bin
touch iNPS
cat iNPS <<EOL
#!/usr/bin/env bash
py3 ${HOME}/lib/iNPS/iNPS_V1.2.2.py "$@"
EOL

chmod 777 iNPS
```

2.3.8 Kallisto

```
cd ${HOME}/lib
wget https://github.com/pachterlab/kallisto/releases/download/v0.43.1/kallisto_linux-
↪v0.43.1.tar.gz
tar -xzf kallisto_linux-v0.43.1.tar.gz
```

2.3.9 SAMtools

```
cd ${HOME}/lib
git clone https://github.com/samtools/htslib.git
cd htslib
autoheader
autoconf
./configure --prefix=${HOME}/lib/htslib
make
make install

cd ${HOME}/lib
git clone https://github.com/samtools/samtools.git
cd samtools
autoheader
autoconf -Wno-syntax
./configure --prefix=${HOME}/lib/samtools
make
make install
```

2.3.10 bedTools

```
cd ${HOME}/lib
wget https://github.com/arq5x/bedtools2/releases/download/v2.26.0/bedtools-2.26.0.tar.
↪gz
tar -zxvf bedtools-2.26.0.tar.gz
cd bedtools2
make
```

2.4 Setup the symlinks

```
cd ${HOME}/bin

ln -s ${HOME}/lib/bedtools2/bin/bedtools bedtools

ln -s ${HOME}/lib/bedToBigBed bedToBigBed
ln -s ${HOME}/lib/wigToBigWig wigToBigWig
ln -s ${HOME}/lib/faToTwoBit faToTwoBit
ln -s ${HOME}/lib/twoBitInfo twoBitInfo

ln -s ${HOME}/lib/bwa/bwa bwa

ln -s ${HOME}/lib/bowtie2-2.3.4-linux-x86_64/bowtie2 bowtie2
```

(continues on next page)

(continued from previous page)

```
ln -s ${HOME}/lib/bowtie2-2.3.4-linux-x86_64/bowtie2-align-s bowtie2-align-s
ln -s ${HOME}/lib/bowtie2-2.3.4-linux-x86_64/bowtie2-align-l bowtie2-align-l
ln -s ${HOME}/lib/bowtie2-2.3.4-linux-x86_64/bowtie2-build bowtie2-build
ln -s ${HOME}/lib/bowtie2-2.3.4-linux-x86_64/bowtie2-build-s bowtie2-build-s
ln -s ${HOME}/lib/bowtie2-2.3.4-linux-x86_64/bowtie2-build-l bowtie2-build-l
ln -s ${HOME}/lib/bowtie2-2.3.4-linux-x86_64/bowtie2-inspect bowtie2-inspect
ln -s ${HOME}/lib/bowtie2-2.3.4-linux-x86_64/bowtie2-inspect-s bowtie2-inspect-s
ln -s ${HOME}/lib/bowtie2-2.3.4-linux-x86_64/bowtie2-inspect-l bowtie2-inspect-l

ln -s ${HOME}/lib/FastQC/fastqc

ln -s ${HOME}/lib/gemtools-1.7.1-core2/bin/gem-2-bed gem-2-bed
ln -s ${HOME}/lib/gemtools-1.7.1-core2/bin/gem-2-gem gem-2-gem
ln -s ${HOME}/lib/gemtools-1.7.1-core2/bin/gem-2-sam gem-2-sam
ln -s ${HOME}/lib/gemtools-1.7.1-core2/bin/gem-2-wig gem-2-wig
ln -s ${HOME}/lib/gemtools-1.7.1-core2/bin/gem-indexer gem-indexer
ln -s ${HOME}/lib/gemtools-1.7.1-core2/bin/gem-indexer_bwt-dna gem-indexer_bwt-dna
ln -s ${HOME}/lib/gemtools-1.7.1-core2/bin/gem-indexer_fasta2meta+cont gem-indexer_
↪fasta2meta+cont
ln -s ${HOME}/lib/gemtools-1.7.1-core2/bin/gem-indexer_generate gem-indexer_generate
ln -s ${HOME}/lib/gemtools-1.7.1-core2/bin/gem-info gem-info
ln -s ${HOME}/lib/gemtools-1.7.1-core2/bin/gem-mapper gem-mapper
ln -s ${HOME}/lib/gemtools-1.7.1-core2/bin/gemtools gemtools

ln -s ${HOME}/lib/iNPS/iNPS_V1.2.2.py iNPS_V1.2.2.py
ln -s ${HOME}/lib/kallisto_linux-v0.43.1/kallisto kallisto

ln -s ${HOME}/lib/htslib/bin/bgzip bgzip
ln -s ${HOME}/lib/htslib/bin/htsfile htsfile
ln -s ${HOME}/lib/htslib/bin/tabix tabix

ln -s ${HOME}/lib/samtools/bin/ace2sam ace2sam
ln -s ${HOME}/lib/samtools/bin/blast2sam.pl blast2sam.pl
ln -s ${HOME}/lib/samtools/bin/bowtie2sam.pl bowtie2sam.pl
ln -s ${HOME}/lib/samtools/bin/export2sam.pl export2sam.pl
ln -s ${HOME}/lib/samtools/bin/interpolate_sam.pl interpolate_sam.pl
ln -s ${HOME}/lib/samtools/bin/maq2sam-long maq2sam-long
ln -s ${HOME}/lib/samtools/bin/maq2sam-short maq2sam-short
ln -s ${HOME}/lib/samtools/bin/md5fa md5fa
ln -s ${HOME}/lib/samtools/bin/md5sum-lite md5sum-lite
ln -s ${HOME}/lib/samtools/bin/novo2sam.pl novo2sam.pl
ln -s ${HOME}/lib/samtools/bin/plot-bamstats plot-bamstats
ln -s ${HOME}/lib/samtools/bin/psl2sam.pl psl2sam.pl
ln -s ${HOME}/lib/samtools/bin/sam2vcf.pl sam2vcf.pl
ln -s ${HOME}/lib/samtools/bin/samtools samtools
ln -s ${HOME}/lib/samtools/bin/samtools.pl samtools.pl
ln -s ${HOME}/lib/samtools/bin/seq_cache_populate.pl seq_cache_populate.pl
ln -s ${HOME}/lib/samtools/bin/soap2sam.pl soap2sam.pl
ln -s ${HOME}/lib/samtools/bin/varfilter.py varfilter.py
ln -s ${HOME}/lib/samtools/bin/wgsim wgsim
ln -s ${HOME}/lib/samtools/bin/wgsim_eval.pl wgsim_eval.pl
ln -s ${HOME}/lib/samtools/bin/zoom2sam.pl zoom2sam.pl

ln -s ${HOME}/lib/biobambam2/bin/bam12auxmerge bam12auxmerge
ln -s ${HOME}/lib/biobambam2/bin/bam12split bam12split
ln -s ${HOME}/lib/biobambam2/bin/bam12strip bam12strip
ln -s ${HOME}/lib/biobambam2/bin/bamadapterclip bamadapterclip
```

(continues on next page)

(continued from previous page)

```
ln -s ${HOME}/lib/biobambam2/bin/bamadapterfind bamadapterfind
ln -s ${HOME}/lib/biobambam2/bin/bamalignfrac bamalignfrac
ln -s ${HOME}/lib/biobambam2/bin/bamauxmerge bamauxmerge
ln -s ${HOME}/lib/biobambam2/bin/bamauxsort bamauxsort
ln -s ${HOME}/lib/biobambam2/bin/bamcat bamcat
ln -s ${HOME}/lib/biobambam2/bin/bamchecksort bamchecksort
ln -s ${HOME}/lib/biobambam2/bin/bamclipreinsert bamclipreinsert
ln -s ${HOME}/lib/biobambam2/bin/bamcollate bamcollate
ln -s ${HOME}/lib/biobambam2/bin/bamcollate2 bamcollate2
ln -s ${HOME}/lib/biobambam2/bin/bamdownsamplerandom bamdownsamplerandom
ln -s ${HOME}/lib/biobambam2/bin/bamexplode bamexplode
ln -s ${HOME}/lib/biobambam2/bin/bamfilteraux bamfilteraux
ln -s ${HOME}/lib/biobambam2/bin/bamfilterflags bamfilterflags
ln -s ${HOME}/lib/biobambam2/bin/bamfilterheader bamfilterheader
ln -s ${HOME}/lib/biobambam2/bin/bamfilterheader2 bamfilterheader2
ln -s ${HOME}/lib/biobambam2/bin/bamfilterlength bamfilterlength
ln -s ${HOME}/lib/biobambam2/bin/bamfiltermc bamfiltermc
ln -s ${HOME}/lib/biobambam2/bin/bamfilternames bamfilternames
ln -s ${HOME}/lib/biobambam2/bin/bamfilterrg bamfilterrg
ln -s ${HOME}/lib/biobambam2/bin/bamfixmateinformation bamfixmateinformation
ln -s ${HOME}/lib/biobambam2/bin/bamflagsplit bamflagsplit
ln -s ${HOME}/lib/biobambam2/bin/bamheap2 bamheap2
ln -s ${HOME}/lib/biobambam2/bin/bamindex bamindex
ln -s ${HOME}/lib/biobambam2/bin/bamintervalcomment bamintervalcomment
ln -s ${HOME}/lib/biobambam2/bin/bamintervalcommenthist bamintervalcommenthist
ln -s ${HOME}/lib/biobambam2/bin/bamlastfilter bamlastfilter
ln -s ${HOME}/lib/biobambam2/bin/bammapdist bammapdist
ln -s ${HOME}/lib/biobambam2/bin/bammarkduplicates bammarkduplicates
ln -s ${HOME}/lib/biobambam2/bin/bammarkduplicates2 bammarkduplicates2
ln -s ${HOME}/lib/biobambam2/bin/bammarkduplicatesopt bammarkduplicatesopt
ln -s ${HOME}/lib/biobambam2/bin/bammaskflags bammaskflags
ln -s ${HOME}/lib/biobambam2/bin/bammdnm bammdnm
ln -s ${HOME}/lib/biobambam2/bin/bammerge bammerge
ln -s ${HOME}/lib/biobambam2/bin/bamnumericalindex bamnumericalindex
ln -s ${HOME}/lib/biobambam2/bin/bamrank bamrank
ln -s ${HOME}/lib/biobambam2/bin/bamranksort bamranksort
ln -s ${HOME}/lib/biobambam2/bin/bamrecalculatecigar bamrecalculatecigar
ln -s ${HOME}/lib/biobambam2/bin/bamrecompress bamrecompress
ln -s ${HOME}/lib/biobambam2/bin/bamreset bamreset
ln -s ${HOME}/lib/biobambam2/bin/bamscrapcount bamscrapcount
ln -s ${HOME}/lib/biobambam2/bin/bamseqchksum bamseqchksum
ln -s ${HOME}/lib/biobambam2/bin/bamsormadup bamsormadup
ln -s ${HOME}/lib/biobambam2/bin/bamsort bamsort
ln -s ${HOME}/lib/biobambam2/bin/bamsplit bamsplit
ln -s ${HOME}/lib/biobambam2/bin/bamsplitdiv bamsplitdiv
ln -s ${HOME}/lib/biobambam2/bin/bamstreamingmarkduplicates bamstreamingmarkduplicates
ln -s ${HOME}/lib/biobambam2/bin/bamtagconversion bamtagconversion
ln -s ${HOME}/lib/biobambam2/bin/bamtofastq bamtofastq
ln -s ${HOME}/lib/biobambam2/bin/bamvalidate bamvalidate
ln -s ${HOME}/lib/biobambam2/bin/bamzztoname bamzztoname
ln -s ${HOME}/lib/biobambam2/bin/fastaexplode fastaexplode
ln -s ${HOME}/lib/biobambam2/bin/fastqtobam fastqtobam
ln -s ${HOME}/lib/biobambam2/bin/fastqtobampar fastqtobampar
ln -s ${HOME}/lib/biobambam2/bin/filtersam filtersam
ln -s ${HOME}/lib/biobambam2/bin/kmerprob kmerprob
ln -s ${HOME}/lib/biobambam2/bin/lasToBAM lasToBAM
ln -s ${HOME}/lib/biobambam2/bin/normalisefasta normalisefasta
```

2.5 Prepare the Python Environment

2.5.1 Install APIs and Pipelines

Checkout the code for the DM API and the mg-process-fastq pipelines:

```
cd ${HOME}/code
pyenv activate mg-process-fastq
pip install git+https://github.com/Multiscale-Genomics/mg-dm-api.git
pip install git+https://github.com/Multiscale-Genomics/mg-tool-api.git

git clone https://github.com/Multiscale-Genomics/mg-process-fastq.git
cd mg-process-fastq
pip install -e .
pip install -r requirements.txt
```

2.5.2 Install MACS2

This should get installed as part of the installation in the mg-process-fastq package, if not then it will need to be installed separately.

For Python 2.7:

```
cd ${HOME}/code
pyenv activate mg-process-fastq
pip install MACS2

ln -s ${HOME}/.pyenv/versions/mg-process-fastq/bin/macs2 ${HOME}/bin/macs2
```

For Python 3.6:

```
cd ${HOME}/code
pyenv activate mg-process-fastq
git clone https://github.com/taoliu/MACS.git
cd MACS
git checkout MACS2p3

cython MACS2/*.pyx
cython MACS2/IO/*.pyx
python setup_w_cython.py install

pip install .
alias macs2="macs2p3"
```

2.5.3 Install iDEAR

```
cd ${HOME}/lib
source("https://bioconductor.org/biocLite.R")
biocLite("BSgenome")
biocLite("DESeq2")
if(!require("devtools")) install.packages("devtools")
devtools::install_bitbucket("juanlmateo/idear")
```

2.5.4 Install TADbit

```
cd ${HOME}/lib
wget https://github.com/3DGenomes/TADbit/archive/dev.zip -O tadbit.zip
unzip tadbit.zip
cd TADbit-dev

# If the pyenv env is not called mg-process-fastq then change this to match,
# the same is true for the version of python
python setup.py install --install-lib=${HOME}/.pyenv/versions/mg-process-fastq/lib/
↪python2.7/site-packages/ --install-scripts=${HOME}/bin
```

2.5.5 Install BSseeker

```
cd ${HOME}/lib
git clone https://github.com/BSSeeker/BSseeker2.git

cd ${HOME}/code/mg-process-fastq
ln -s ${HOME}/lib/BSseeker2/bs_align bs_align
ln -s ${HOME}/lib/BSseeker2/bs_index bs_index
ln -s ${HOME}/lib/BSseeker2/bs_utils bs_utils
```

2.5.6 Trim Galore

```
cd ${HOME}/lib
pip install cutadapt
wget -O trim_galore.tar.gz https://github.com/FelixKrueger/TrimGalore/archive/0.5.0.
↪tar.gz
tar -xzf trim_galore.tar.gz

cd ${HOME}/bin
ln -s ${HOME}/lib/TrimGalore-0.5.0/trim_galore trim_galore
```

Running on a COMPSs VM the symlink will need to be created in a system accessible area:

```
sudo ln -s ${HOME}/lib/TrimGalore-0.4.3/trim_galore /usr/local/bin/trim_galore
pip install cutadapt
```

2.6 Post Installation Tidyup

```
cd ${HOME}/lib
rm *.zip *.tar.gz
```

CHAPTER 3

Pipelines

3.1 Download and index genome files

This pipeline is for the indexing of genomes once they have been loaded into the VRE. It indexes each new genome with Bowtie2, BWA and GEM. These indexes can then be used by the other pipelines.

3.1.1 Running from the command line

Parameters

taxon_id [int] Species taxonomic ID

assembly [str] Genomic assembly ID

genome [str] Location of the genomes FASTA file

Returns

Bowtie2 index files BWA index files GEM index file

Example

When running the pipeline on a local machine without COMPSs:

```
python process_genome.py                                \
   --config tests/json/config_genome_indexer.json \
   --in_metadata tests/json/input_genome_indexer.json \
   --out_metadata tests/json/output_genome_indexer.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                                  \
   --lang=python                           \
   --library_path=${HOME}/bin              \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug \
   process_genome.py \
      --config tests/json/config_genome_indexer.json \
      --in_metadata tests/json/input_genome_indexer.json \
      --out_metadata tests/json/output_genome_indexer.json
```

3.1.2 Methods

class process_genome.**process_genome**(*configuration=None*)

Workflow to download and pre-index a given genome

run(*input_files*, *metadata*, *output_files*)

Main run function for the indexing of genome assembly FASTA files. The pipeline uses Bowtie2, BWA and GEM ready for use in pipelines that rely on alignment.

Parameters

- **input_files** (*dict*) –

  genome [str] List of file locations

- **metadata** (*dict*) –

  genome [dict] Required meta data

- **output_files** (*dict*) –

  bwa_index [str] Location of the BWA index archive files

  bwt_index [str] Location of the Bowtie2 index archive file

  gem_index [str] Location of the GEM index file

  genome_gem [str] Location of a the FASTA file generated for the GEM indexing step

Returns

- **outputfiles** (*dict*) – List of locations for the output index files
- **output_metadata** (*dict*) – Metadata about each of the files

3.2 BioBamBam Alignment Filtering

This pipeline to filter sequencing artifacts from aligned reads.

3.2.1 Running from the command line

Parameters

config [str] Configuration JSON file

in_metadata [str] Location of input JSON metadata for files

out_metadata [str] Location of output JSON metadata for files

Returns

filtered [file] Filtered bam file

Example

REQUIREMENT - Needs an aligned bam file

When running the pipeline on a local machine without COMPSs:

```
python process_biobambam.py \
   --config tests/json/config_biobambam.json \
   --in_metadata tests/json/input_biobambam.json \
   --out_metadata tests/json/output_biobambam.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                       \
   --lang=python                \
   --library_path=${HOME}/bin \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug            \
   process_biobambam.py           \
      --config tests/json/config_biobambam.json \
      --in_metadata tests/json/input_biobambam.json \
      --out_metadata tests/json/output_biobambam.json
```

3.2.2 Methods

class process_biobambam.**process_biobambam**(*configuration=None*)

Functions for filtering FastQ alignments with BioBamBam.

run(*input_files*, *metadata*, *output_files*)

Main run function for filtering FastQ aligned reads using BioBamBam.

Parameters

- **input_files** (*dict*) – Location of the initial input files required by the workflow

  bam [str] Location of BAM file

- **metadata** (*dict*) – Input file meta data associated with their roles

  bam : str

- **output_files** (*dict*) – Output file locations

  filtered : str

Returns

- **output_files** (*dict*) – Output file locations associated with their roles, for the output

  filtered [str] Filtered version of the bam file

- **output_metadata** (*dict*) – Output metadata for the associated files in output_files

  filtered : Metadata

3.3 Bowtie2 Alignment

This pipeline aligns FASTQ reads to a given indexed genome. The pipeline can handle single-end and paired-end reads.

3.3.1 Running from the command line

Parameters

config [str] Configuration JSON file

in_metadata [str] Location of input JSON metadata for files

out_metadata [str] Location of output JSON metadata for files

Returns

bam [file] Aligned reads in bam file

Example

REQUIREMENT - Needs the indexing step to be run first

When running the pipeline on a local machine without COMPSs:

```
python process_align_bowtie.py \
   --config tests/json/config_bowtie2.json \
   --in_metadata tests/json/input_bowtie2.json \
   --out_metadata tests/json/output_bowtie2.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                     \
   --lang=python              \
   --library_path=${HOME}/bin \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug          \
   process_align_bowtie.py         \
      --config tests/json/config_bowtie2_single.json \
      --in_metadata tests/json/input_bowtie2_single_metadata.json \
      --out_metadata tests/json/output_bowtie2_single.json
```

```
runcompss                     \
   --lang=python              \
   --library_path=${HOME}/bin \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug          \
   process_align_bowtie.py         \
      --config tests/json/config_bowtie2_paired.json \
      --in_metadata tests/json/input_bowtie2_paired_metadata.json \
      --out_metadata tests/json/output_bowtie2_paired.json
```

3.3.2 Methods

class process_align_bowtie.**process_bowtie**(*configuration=None*)
Functions for aligning FastQ files with Bowtie2

run(*input_files*, *metadata*, *output_files*)
Main run function for aligning FastQ reads with Bowtie2.

Currently this can only handle a single data file and a single background file.

Parameters

- **input_files** (*dict*) – Location of the initial input files required by the workflow

  genome [str] Genome FASTA file

  index [str] Location of the BWA archived index files

  loc [str] Location of the FASTQ reads files

  fastq2 [str] [OPTIONAL] Location of the FASTQ reads file for paired end data

- **metadata** (*dict*) – Input file meta data associated with their roles

  genome : str index : str loc : str fastq2 : str

- **output_files** (*dict*) – Output file locations

  bam [str] Output bam file location

Returns

- **output_files** (*dict*) – Output file locations associated with their roles, for the output

  bam [str] Aligned FASTQ short read file locations

- **output_metadata** (*dict*) – Output metadata for the associated files in output_files

  bam : Metadata

3.4 BSgenome Builder

This pipeline can process FASTQ to identify protein-DNA binding sites.

3.4.1 Running from the command line

Parameters

config [str] Configuration JSON file

in_metadata [str] Location of input JSON metadata for files

out_metadata [str] Location of output JSON metadata for files

Returns

bsgenome [file] BSgenome index

genome_2bit [file] Compressed representation of the genome required for generating the index

chrom_size [file] Location of the chrom.size file

seed_file [file] Configuaration file for generating the BSgenome R package

Example

When running the pipeline on a local machine without COMPSs:

```
python process_bsgenome.py                                \
   --config tests/json/config_bsgenome.json \
   --in_metadata tests/json/input_bsgenome.json \
   --out_metadata tests/json/output_bsgenome.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                     \
   --lang=python              \
   --library_path=${HOME}/bin \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug          \
   process_bsgenome.py         \
      --config tests/json/config_bsgenome.json \
      --in_metadata tests/json/input_bsgenome.json \
      --out_metadata tests/json/output_bsgenome.json
```

3.4.2 Methods

class process_bsgenome.**process_bsgenome**(*configuration=None*)

Workflow to download and pre-index a given genome

run(*input_files*, *metadata*, *output_files*)

Main run function for the indexing of genome assembly FASTA files. The pipeline uses Bowtie2, BWA and GEM ready for use in pipelines that rely on alignment.

Parameters

- **input_files** (*dict*) –

  genome [str] Location of the FASTA input file

- **metadata** (*dict*) –

  genome [dict] Required meta data

- **output_files** (*dict*) –

  BSgenome [str] Location of a the BSgenome R package

Returns

- **outputfiles** (*dict*) – List of locations for the output index files
- **output_metadata** (*dict*) – Metadata about each of the files

3.5 BS Seeker2 Indexer

This pipeline can process FASTQ to identify protein-DNA binding sites.

3.5.1 Running from the command line

Parameters

config [str] Configuration JSON file

in_metadata [str] Location of input JSON metadata for files

out_metadata [str] Location of output JSON metadata for files

Returns

index [file] BS Seeker2 index

Example

When running the pipeline on a local machine without COMPSs:

```
python process_bs_seeker_index.py                              \
   --config tests/json/config_wgbs_index.json \
   --in_metadata tests/json/input_wgbs_index_metadata.json \
   --out_metadata tests/json/output_wgbs_index.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                     \
   --lang=python              \
   --library_path=${HOME}/bin \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug          \
   process_bs_seeker_index.py         \
      --config tests/json/config_wgbs_index.json \
      --in_metadata tests/json/input_wgbs_index_metadata.json \
      --out_metadata tests/json/output_wgbs_index.json
```

3.5.2 Methods

class process_bs_seeker_index.**process_bs_seeker_index**(*configuration=None*)

Functions for aligning FastQ files with BWA

run(*input_files*, *metadata*, *output_files*)

Main run function for generatigng the index files required by BS Seeker2.

Parameters

- **input_files** (*dict*) – List of strings for the locations of files. These should include:

  genome_fa [str] Genome assembly in FASTA

- **metadata** (*dict*) – Input file meta data associated with their roles

  genome : str

- **output_files** (*dict*) – Output file locations

  bam [str] Output bam file location

Returns

output_files – Output file locations associated with their roles, for the output

index : str

Return type dict

3.6 BS Seeker2 Aligner

This pipeline aligns FASTQ paired end reads using BS Seeker2 and Bowtie2.

3.6.1 Running from the command line

Parameters

config [str] Configuration JSON file

in_metadata [str] Location of input JSON metadata for files

out_metadata [str] Location of output JSON metadata for files

Returns

bam [file] Aligned Bam file

bai [file] Aligned Bam index file

Example

When running the pipeline on a local machine without COMPSs:

```
python process_bs_seeker_aligner.py                              \
   --config tests/json/config_wgbs_align.json \
   --in_metadata tests/json/input_wgbs_align_metadata.json \
   --out_metadata tests/json/output_wgbs_align.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                    \
   --lang=python             \
   --library_path=${HOME}/bin \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug         \
   process_bs_seeker_aligner.py         \
      --config tests/json/config_wgbs_align.json \
      --in_metadata tests/json/input_wgbs_align_metadata.json \
      --out_metadata tests/json/output_wgbs_align.json
```

3.6.2 Methods

class process_bs_seeker_aligner.**process_bs_seeker_aligner**(*configuration=None*)

Functions for downloading and processing whole genome bisulfate sequencings (WGBS) files. Files are filtered, aligned and analysed for points of methylation

run(*input_files*, *metadata*, *output_files*)

This pipeline processes paired-end FASTQ files to identify methylated regions within the genome.

Parameters

- **input_files** (*dict*) – List of strings for the locations of files. These should include:

  genome_fa [str] Genome assembly in FASTA

  fastq1 [str] Location for the first filtered FASTQ file for single or paired end reads

  fastq2 [str] Location for the second filtered FASTQ file if paired end reads

  index [str] Location of the index file

- **metadata** (*dict*) – Input file meta data associated with their roles

  genome_fa : dict fastq1 : dict fastq2 : dict index : dict

- **output_files** (*dict*) – bam : str bai : str

Returns **bam|bai** – Location of the alignment bam file and the associated index

Return type str

3.7 BiSulphate Sequencing Filter

This pipeline processes FASTQ files to filter out duplicate reads.

3.7.1 Running from the command line

Parameters

config [str] Configuration JSON file

in_metadata [str] Location of input JSON metadata for files

out_metadata [str] Location of output JSON metadata for files

Returns

fastq1_filtered|fastq1_filtered [str] Locations of the filtered FASTQ files from which alignments were made

fastq2_filtered|fastq2_filtered [str] Locations of the filtered FASTQ files from which alignments were made

Example

When running the pipeline on a local machine without COMPSs:

```
python process_bs_seeker_filter.py                    \
   --config tests/json/config_wgbs_filter.json \
   --in_metadata tests/json/input_wgbs_filter_metadata.json \
   --out_metadata tests/json/output_metadata.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                                  \
   --lang=python                           \
   --library_path=${HOME}/bin              \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug                       \
   process_bs_seeker_filter.py                          \
      --config tests/json/config_wgbs_filter.json \
      --in_metadata tests/json/input_wgbs_filter_metadata.json \
      --out_metadata tests/json/output_metadata.json
```

3.7.2 Methods

class process_bs_seeker_filter.**process_bsFilter**(*configuration=None*)

Functions for filtering FASTQ files. Files are filtered for removal of duplicate reads. Low quality reads in qseq file can also be filtered.

run(*input_files*, *metadata*, *output_files*)

This pipeline processes FASTQ files to filter duplicate entries

Parameters

- **input_files** (*dict*) – List of strings for the locations of files. These should include:

  fastq1 [str] Location for the first FASTQ file for single or paired end reads

  fastq2 [str] Location for the second FASTQ file if paired end reads [OPTIONAL]

- **metadata** (*dict*) – Input file meta data associated with their roles

  fastq1 : str

  fastq2 [str] [OPTIONAL]

- **output_files** (*dict*) – fastq1_filtered : str

  fastq2_filtered [str] [OPTIONAL]

Returns

- **fastq1_filtered|fastq1_filtered** (*str*) – Locations of the filtered FASTQ files from which alignments were made
- **fastq2_filtered|fastq2_filtered** (*str*) – Locations of the filtered FASTQ files from which alignments were made

3.8 BS Seeker2 Methylation Peak Caller

3.9 BWA Alignment - bwa aln

This pipeline aligns FASTQ reads to a given indexed genome. The pipeline can handle single-end and paired-end reads.

3.9.1 Running from the command line

Parameters

config [str] Configuration JSON file

in_metadata [str] Location of input JSON metadata for files

out_metadata [str] Location of output JSON metadata for files

Returns

bam [file] Aligned reads in bam file

Example

REQUIREMENT - Needs the indexing step to be run first

When running the pipeline on a local machine without COMPSs:

```
python process_align_bwa.py                              \
   --config tests/json/config_chipseq.json \
   --in_metadata tests/json/input_chipseq.json \
   --out_metadata tests/json/output_chipseq.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                     \
   --lang=python              \
   --library_path=${HOME}/bin \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug          \
   process_align_bwa.py         \
      --config tests/json/config_bwa_aln_single.json \
      --in_metadata tests/json/input_bwa_aln_single_metadata.json \
      --out_metadata tests/json/output_bwa_aln_single.json
```

```
runcompss                     \
   --lang=python              \
   --library_path=${HOME}/bin \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug          \
   process_align_bwa.py         \
      --config tests/json/config_bwa_aln_paired.json \
```

(continues on next page)

(continued from previous page)

```
8       --in_metadata tests/json/input_bwa_aln_paired_metadata.json \
9       --out_metadata tests/json/output_bwa_aln_paired.json
```

3.9.2 Methods

class process_align_bwa.**process_bwa**(*configuration=None*)

Functions for aligning FastQ files with BWA ALN

run(*input_files*, *metadata*, *output_files*)

Main run function for aligning FastQ reads with BWA ALN.

Parameters

- **input_files** (*dict*) – Location of the initial input files required by the workflow

  genome [str] Genome FASTA file

  index [str] Location of the BWA archived index files

  loc [str] Location of the FASTQ reads files

  fastq2 [str] [OPTIONAL] Location of the FASTQ reads file for paired end data

- **metadata** (*dict*) – Input file meta data associated with their roles

  genome : str index : str loc : str fastq2 : str

- **output_files** (*dict*) – Output file locations

  bam [str] Output bam file location

Returns

- **output_files** (*dict*) – Output file locations associated with their roles, for the output

  bam [str] Aligned FASTQ short read file locations

- **output_metadata** (*dict*) – Output metadata for the associated files in output_files

  bam : Metadata

3.10 BWA Alignment - bwa mem

This pipeline aligns FASTQ reads to a given indexed genome. The pipeline can handle single-end and paired-end reads.

3.10.1 Running from the command line

Parameters

config [str] Configuration JSON file

in_metadata [str] Location of input JSON metadata for files

out_metadata [str] Location of output JSON metadata for files

Returns

bam [file] Aligned reads in bam file

Example

REQUIREMENT - Needs the indexing step to be run first

When running the pipeline on a local machine without COMPSs:

```
python process_align_bwa.py                              \
   --config tests/json/config_chipseq.json \
   --in_metadata tests/json/input_chipseq.json \
   --out_metadata tests/json/output_chipseq.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                     \
   --lang=python              \
   --library_path=${HOME}/bin \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug          \
   process_align_bwa_mem.py         \
      --config tests/json/config_bwa_mem_single.json \
      --in_metadata tests/json/input_bwa_mem_single_metadata.json \
      --out_metadata tests/json/output_bwa_mem_single.json
```

```
runcompss                     \
   --lang=python              \
   --library_path=${HOME}/bin \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug          \
   process_align_bwa_mem.py         \
      --config tests/json/config_bwa_mem_paired.json \
      --in_metadata tests/json/input_bwa_mem_paired_metadata.json \
      --out_metadata tests/json/output_bwa_mem_paired.json
```

3.10.2 Methods

class process_align_bwa_mem.**process_bwa_mem**(*configuration=None*)

Functions for aligning FastQ files with BWA MEM

run(*input_files*, *metadata*, *output_files*)

Main run function for aligning FastQ data with BWA MEM.

Parameters

- **input_files** (*dict*) – Location of the initial input files required by the workflow

  genome [str] Genome FASTA file

  index [str] Location of the BWA archived index files

  loc [str] Location of the FASTQ reads files

  fastq2 [str] [OPTIONAL] Location of the FASTQ reads file for paired end data

- **metadata** (*dict*) – Input file meta data associated with their roles

  genome : str index : str loc : str fastq2 : str

- **output_files** (*dict*) – Output file locations

  bam [str] Output bam file location

Returns

- **output_files** (*dict*) – Output file locations associated with their roles, for the output

  bam [str] Aligned FASTQ short read file locations

- **output_metadata** (*dict*) – Output metadata for the associated files in output_files

  bam : Metadata

3.11 ChIP-Seq Analysis

This pipeline can process FASTQ to identify protein-DNA binding sites.

3.11.1 Running from the command line

Parameters

config [str] Configuration JSON file

in_metadata [str] Location of input JSON metadata for files

out_metadata [str] Location of output JSON metadata for files

Returns

bed [file] Bed files with the locations of transcription factor binding sites within the genome

Example

REQUIREMENT - Needs the indexing step to be run first

When running the pipeline on a local machine without COMPSs:

```
python process_chipseq.py                          \
   --config tests/json/config_chipseq.json \
   --in_metadata tests/json/input_chipseq.json \
   --out_metadata tests/json/output_chipseq.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                     \
   --lang=python              \
   --library_path=${HOME}/bin \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug          \
```

(continues on next page)

(continued from previous page)

```
process_chipseq.py         \
   --config tests/json/config_chipseq.json \
   --in_metadata tests/json/input_chipseq.json \
   --out_metadata tests/json/output_chipseq.json
```

3.11.2 Methods

class process_chipseq.**process_chipseq**(*configuration=None*)

Functions for processing Chip-Seq FastQ files. Files are the aligned, filtered and analysed for peak calling

run(*input_files*, *metadata*, *output_files*)

Main run function for processing ChIP-seq FastQ data. Pipeline aligns the FASTQ files to the genome using BWA. MACS 2 is then used for peak calling to identify transcription factor binding sites within the genome.

Currently this can only handle a single data file and a single background file.

Parameters

- **input_files** (*dict*) – Location of the initial input files required by the workflow

  genome [str] Genome FASTA file

  index [str] Location of the BWA archived index files

  loc [str] Location of the FASTQ reads files

  fastq2 [str] Location of the paired end FASTQ file [OPTIONAL]

  bg_loc [str] Location of the background FASTQ reads files [OPTIONAL]

  fastq2_bg [str] Location of the paired end background FASTQ reads files [OPTIONAL]

- **metadata** (*dict*) – Input file meta data associated with their roles

  genome : str index : str

  bg_loc [str] [OPTIONAL]

- **output_files** (*dict*) – Output file locations

  bam [, "bam_bg"] : str filtered [, "filtered_bg"] : str narrow_peak : str summits : str broad_peak : str gapped_peak : str

Returns

- **output_files** (*dict*) – Output file locations associated with their roles, for the output

  bam [, "bam_bg"] [str] Aligned FASTQ short read file [and aligned background file] locations

  filtered [, "filtered_bg"] [str] Filtered versions of the respective bam files

  narrow_peak [str] Results files in bed4+1 format

  summits [str] Results files in bed6+4 format

  broad_peak [str] Results files in bed6+3 format

  gapped_peak [str] Results files in bed12+3 format

- **output_metadata** (*dict*) – Output metadata for the associated files in output_files

  bam [, "bam_bg"] : Metadata filtered [, "filtered_bg"] : Metadata narrow_peak : Metadata summits : Metadata broad_peak : Metadata gapped_peak : Metadata

3.12 iDamID-Seq Analysis

This pipeline can process FASTQ to identify protein-DNA binding sites.

3.12.1 Running from the command line

Parameters

config [str] Configuration JSON file

in_metadata [str] Location of input JSON metadata for files

out_metadata [str] Location of output JSON metadata for files

Returns

bigwig [file] Bigwig file of the binding profile of transcription factors

Example

REQUIREMENT - Needs the indexing step to be run first

When running the pipeline on a local machine without COMPSs:

```
python process_damidseq.py                              \
   --config tests/json/config_idamidseq.json \
   --in_metadata tests/json/input_idamidseq.json \
   --out_metadata tests/json/output_idamidseq.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                     \
   --lang=python              \
   --library_path=${HOME}/bin \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug          \
   process_damidseq.py         \
      --config tests/json/config_idamidseq.json \
      --in_metadata tests/json/input_idamidseq.json \
      --out_metadata tests/json/output_idamidseq.json
```

3.12.2 Methods

class process_damidseq.**process_damidseq**(*configuration=None*)

Functions for processing Chip-Seq FastQ files. Files are the aligned, filtered and analysed for peak calling

run (*input_files*, *metadata*, *output_files*)

Main run function for processing DamID-seq FastQ data. Pipeline aligns the FASTQ files to the genome using BWA. iDEAR is then used for peak calling to identify transcription factor binding sites within the genome.

Currently this can only handle a single data file and a single background file.

Parameters

- **input_files** (*dict*) – Location of the initial input files required by the workflow

  genome [str] Genome FASTA file

  index [str] Location of the BWA archived index files

  fastq_1 [str] Location of the FASTQ reads files

  fastq_2 [str] Location of the FASTQ repeat reads files

  bg_fastq_1 [str] Location of the background FASTQ reads files

  bg_fastq_2 [str] Location of the background FASTQ repeat reads files

- **metadata** (*dict*) – Input file meta data associated with their roles

  genome : str index : str fastq_1 : str fastq_2 : str bg_fastq_1 : str bg_fastq_2 : str

- **output_files** (*dict*) – Output file locations

  bam [, "bam_bg"] : str filtered [, "filtered_bg"] : str

Returns

- **output_files** (*dict*) – Output file locations associated with their roles, for the output

  bam [, "bam_bg"] [str] Aligned FASTQ short read file [and aligned background file] locations

  filtered [, "filtered_bg"] [str] Filtered versions of the respective bam files

  bigwig [str] Location of the bigwig peaks

- **output_metadata** (*dict*) – Output metadata for the associated files in output_files

  bam [, "bam_bg"] : Metadata filtered [, "filtered_bg"] : Metadata bigwig : Metadata

3.13 iNPS

This pipeline can process bam file to identify nucleosome positions.

3.13.1 Running from the command line

Parameters

config [str] Configuration JSON file

in_metadata [str] Location of input JSON metadata for files

out_metadata [str] Location of output JSON metadata for files

Returns

bed [file] Bed files with the locations of nucleosome binding sites within the genome

Example

REQUIREMENT - Needs the indexing step to be run first

When running the pipeline on a local machine without COMPSs:

```
python process_iNPS.py                              \
   --config tests/json/config_inps.json \
   --in_metadata tests/json/input_iNPS_metadata.json \
   --out_metadata tests/json/output_iNPS.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                                       \
   --lang=python                                \
   --library_path=${HOME}/bin                   \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug                            \
   process_iNPS.py                          \
       --config tests/json/config_inps.json \
       --in_metadata tests/json/input_iNPS_metadata.json \
       --out_metadata tests/json/output_iNPS.json
```

3.13.2 Methods

class process_iNPS.**process_iNPS**(*configuration=None*)

Functions for improved nucleosome positioning algorithm (iNPS). Bam Files are analysed for peaks for nucleosome positioning

run(*input_files*, *metadata*, *output_files*)

This pipeline processes bam files to identify nucleosome regions within the genome and generates bed files.

Parameters

- **input_files** (*dict*) – bam_file : str Location of the aligned sequences in bam format
- **output_files** (*dict*) – peak_bed : str Location of the collated bed file of nucleosome peak calls

Returns **peak_bed** – Location of the collated bed file of nucleosome peak calls

Return type str

3.14 MACS2 Analysis

Transcript binding site peak caller for ChIP-seq data

3.14.1 Running from the command line

Parameters

config [str] Configuration JSON file

in_metadata [str] Location of input JSON metadata for files

out_metadata [str] Location of output JSON metadata for files

Returns

bam [file] Aligned reads in bam file

Example

REQUIREMENT - Needs the indexing step to be run first

When running the pipeline on a local machine without COMPSs:

```
python process_align_bwa.py                              \
   --config tests/json/config_macs2.json \
   --in_metadata tests/json/input_macs2.json \
   --out_metadata tests/json/output_macs2.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                     \
   --lang=python              \
   --library_path=${HOME}/bin \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug          \
   process_macs2.py         \
      --config tests/json/config_macs2_single.json \
      --in_metadata tests/json/input_macs2_metadata.json \
      --out_metadata tests/json/output_macs2.json
```

```
runcompss                     \
   --lang=python              \
   --library_path=${HOME}/bin \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug          \
   process_macs2.py         \
      --config tests/json/config_macs2_bgd_paired.json \
      --in_metadata tests/json/input_macs2_bgd_paired_metadata.json \
      --out_metadata tests/json/output_macs2_bgd.json
```

3.14.2 Methods

class process_macs2.**process_macs2**(*configuration=None*)

Functions for processing Chip-Seq FastQ files. Files are the aligned, filtered and analysed for peak calling

run (*input_files*, *metadata*, *output_files*)

Main run function for processing ChIP-seq FastQ data. Pipeline aligns the FASTQ files to the genome using BWA. MACS 2 is then used for peak calling to identify transcription factor binding sites within the genome.

Currently this can only handle a single data file and a single background file.

Parameters

- **input_files** (*dict*) – Location of the initial input files required by the workflow

  bam [str] Location of the aligned reads file

  bam_bg [str] Location of the background aligned FASTQ reads file [OPTIONAL]

- **metadata** (*dict*) – Input file meta data associated with their roles

  bam : str

  bam_bg [str] [OPTIONAL]

- **output_files** (*dict*) – Output file locations

  narrow_peak : str summits : str broad_peak : str gapped_peak : str

Returns

- **output_files** (*dict*) – Output file locations associated with their roles, for the output

  narrow_peak [str] Results files in bed4+1 format

  summits [str] Results files in bed6+4 format

  broad_peak [str] Results files in bed6+3 format

  gapped_peak [str] Results files in bed12+3 format

- **output_metadata** (*dict*) – Output metadata for the associated files in output_files

  narrow_peak : Metadata summits : Metadata broad_peak : Metadata gapped_peak : Metadata

3.15 Mnase-Seq Analysis

This pipeline can process FASTQ to identify nucleosome binding sites.

3.15.1 Running from the command line

Parameters

config [str] Configuration JSON file

in_metadata [str] Location of input JSON metadata for files

out_metadata [str] Location of output JSON metadata for files

Returns

bed [file] Bed files with the locations of nucleosome binding sites within the genome

Example

REQUIREMENT - Needs the indexing step to be run first

When running the pipeline on a local machine without COMPSs:

```
python process_mnaseseq.py                              \
   --config tests/json/config_mnaseseq.json \
   --in_metadata tests/json/input_mnaseseq.json \
   --out_metadata tests/json/output_mnaseseq.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                                          \
   --lang=python                                   \
   --library_path=${HOME}/bin                      \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug                               \
   process_mnaseseq.py                             \
      --config tests/json/config_mnaseseq.json \
      --in_metadata tests/json/input_mnaseseq.json \
      --out_metadata tests/json/output_mnaseseq.json
```

3.15.2 Methods

class process_mnaseseq.**process_mnaseseq**(*configuration=None*)

Functions for downloading and processing Mnase-seq FastQ files. Files are downloaded from the European Nucleotide Archive (ENA), then aligned, filtered and analysed for peak calling

run(*input_files*, *metadata*, *output_files*)

Main run function for processing MNase-Seq FastQ data. Pipeline aligns the FASTQ files to the genome using BWA. iNPS is then used for peak calling to identify nucleosome position sites within the genome.

Parameters

- **files_ids** (*list*) – List of file locations
- **metadata** (*list*) – Required meta data

Returns **outputfiles** – List of locations for the output bam, bed and tsv files

Return type list

3.16 RNA-Seq Analysis

This pipeline can process FASTQ to quantify the level of expression of cDNAs.

3.16.1 Running from the command line

Parameters

config [str] Configuration JSON file

in_metadata [str] Location of input JSON metadata for files

out_metadata [str] Location of output JSON metadata for files

Returns

bed [file] WIG file with the levels of expression for genes

Example

When running the pipeline on a local machinewithout COMPSs:

```
python process_rnaseq.py                                          \
   --config tests/json/config_rnaseq.json \
   --in_metadata tests/json/input_rnaseq.json \
   --out_metadata tests/json/output_rnaseq.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                                       \
   --lang=python                                \
   --library_path=${HOME}/bin                   \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug                            \
   process_rnaseq.py                            \
      --config tests/json/config_rnaseq.json \
      --in_metadata tests/json/input_rnaseq.json \
      --out_metadata tests/json/output_rnaseq.json
```

3.16.2 Methods

class process_rnaseq.**process_rnaseq**(*configuration=None*)

Functions for downloading and processing RNA-seq FastQ files. Files are downloaded from the European Nucleotide Archive (ENA), then they are mapped to quantify the amount of cDNA

run(*input_files*, *metadata*, *output_files*)

Main run function for processing RNA-Seq FastQ data. Pipeline aligns the FASTQ files to the genome using Kallisto. Kallisto is then also used for peak calling to identify levels of expression.

Parameters

- **files_ids** (*dict*) – List of file locations (genome FASTA, FASTQ_01, FASTQ_02 (for paired ends))
- **metadata** (*list*) – Required meta data
- **output_files** (*list*) – List of output file locations
- **input_files** (*list*) – List of file locations
- **metadata** – Required meta data
- **output_files** – List of output file locations

Returns **outputfiles** – List of locations for the output bam, bed and tsv files

Return type list

Returns

- **outputfiles** (*dict*) – List of locations for the output index files
- **output_metadata** (*dict*) – Metadata about each of the files

3.17 TrimGalore

This pipeline can process FASTQ to trim poor base quality or adapter contamination.

3.17.1 Running from the command line

Parameters

config [str] Configuration JSON file

in_metadata [str] Location of input JSON metadata for files

out_metadata [str] Location of output JSON metadata for files

Returns

fastq_trimmed [file] Location of a fastq file containing the sequences after poor base qualities or contamination trimming

A full description of the Trim Galore files can be found at https://github.com/FelixKrueger/TrimGalore

Example

When running the pipeline on a local machine without COMPSs:

```
python process_trim_galore.py                    \
   --config tests/json/config_trimgalore.json \
   --in_metadata tests/json/input_trimgalore_metadata.json \
   --out_metadata tests/json/output_trimgalore.json \
   --local
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                                  \
   --lang=python                           \
   --library_path=${HOME}/bin              \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug                       \
   process_trim_galore.py                          \
      --config tests/json/config_trimgalore.json \
      --in_metadata tests/json/input_trimgalore_metadata.json \
      --out_metadata tests/json/output_trimgalore.json
```

3.17.2 Methods

class process_trim_galore.**process_trim_galore**(*configuration=None*)

Functions for filtering FASTQ files. Files are filtered for removal of duplicate reads. Low quality reads in qseq file can also be filtered.

run(*input_files*, *metadata*, *output_files*)

This pipeline processes FASTQ files to trim low quality base calls and adapter sequences

Parameters **input_files** (*dict*) – List of strings for the locations of files. These should include:

fastq [str] Location for the first FASTQ file for single or paired end reads

metadata [dict] Input file meta data associated with their roles

output_files : dict

fastq_trimmed : str

Returns **fastq_trimmed|fastq_trimmed** – Locations of the filtered FASTQ files from which trimmings were made

Return type str

3.18 Whole Genome BiSulphate Sequencing Analysis

3.19 Hi-C Analysis

This pipeline can process paired end FASTQ files to identify structural interactions that occur so that the genome can fold into the confines of the nucleus

3.19.1 Running from the command line

Parameters

genome [str] Location of the genomes FASTA file

genome_gem [str] Location of the genome GEM index file

taxon_id [int] Species taxonomic ID

assembly [str] Genomic assembly ID

file1 [str] Location of FASTQ file 1

file2 [str] Location of FASTQ file 2

resolutions [str] Comma separated list of resolutions to calculate the matrix for. [DEFAULT : 1000000,10000000]

enzyme_name [str] Name of the enzyme used to digest the genome (example 'MboI')

window_type [str] iter | frag. Analysis windowing type to use

windows1 [str] FASTQ sampling window sizes to use for the first paired end FASTQ file, the default is to use *[[1,25], [1,50], [1,75], [1,100]]*. This would be represented as *1,25,50,75,100* as input for this variable

windows2 [str] FASTQ sampling window sizes to use for the second paired end FASTQ file, the default is to use *[[1,25], [1,50], [1,75], [1,100]]*. This would be represented as *1,25,50,75,100* as input for this variable

normalized [int] 1 | 0. Determines whether the counts of alignments should be normalized

tag [str] Name for the experiment output files to use

Returns

Adjacency List : file HDF5 Adjacency Array : file

Example

REQUIREMENT - Needs the indexing step to be run first

When running the pipeline on a local machine:

```
python process_hic.py                                    \
   --genome /<dataset_dir>/Homo_sapiens.GRCh38.fasta     \
   --genome_gem /<dataset_dir>/Homo_sapiens.GRCh38.gem   \
   --assembly GCA_000001405.25                           \
   --taxon_id 9606                                       \
   --file1 /<dataset_dir>/<file_name>_1.fastq            \
   --file2 /<dataset_dir>/<file_name>_2.fastq            \
   --resolutions 1000000,10000000                        \
   --enzyme_name MboI                                    \
   --windows1 1,100                                      \
   --windows2 1,100                                      \
   --normalized 1                                        \
   --tag Human.SRR1658573                              \
   --window_type frag
```

When using a local version of the [COMPS virtual machine](https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/):

```
runcompss                       \
   --lang=python                \
   --library_path=${HOME}/bin   \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/ \
   --log_level=debug            \
   process_hic.py               \
      --taxon_id 9606           \
      --genome /<dataset_dir>/.Human.GCA_000001405.22_gem.fasta \
      --assembly GRCh38         \
      --file1 /<dataset_dir>/Human.SRR1658573_1.fastq \
      --file2 /<dataset_dir>/Human.SRR1658573_2.fastq \
      --genome_gem /<dataset_dir>/Human.GCA_000001405.22_gem.fasta.gem \
      --enzyme_name MboI        \
      --resolutions 10000,100000 \
      --windows1 1,100          \
      --windows2 1,100          \
      --normalized 1            \
      --tag Human.SRR1658573    \
      --window_type frag
```

3.19.2 Methods

class process_hic.**process_hic**(*configuration=None*)

Functions for downloading and processing Mnase-seq FastQ files. Files are downloaded from the European Nucleotide Archive (ENA), then aligned, filtered and analysed for peak calling

run(*input_files*, *metadata*, *output_files*)

Main run function for processing MNase-Seq FastQ data. Pipeline aligns the FASTQ files to the genome using BWA. iNPS is then used for peak calling to identify nucleosome position sites within the genome.

Parameters

- **files_ids** (*list*) – List of file locations
- **metadata** (*list*) – Required meta data
- **output_files** (*list*) – List of output file locations

Returns **outputfiles** – List of locations for the output bam, bed and tsv files

Return type list

CHAPTER 4

Tools for processing FastQ files

4.1 File Validation

Pipelines and functions assessing the quality of input files.

4.1.1 FastQC

class tool.validate_fastqc.**fastqcTool**(*configuration=None*)

Tool for running indexers over a genome FASTA file

run(*input_files*, *input_metadata*, *output_files*)

Tool for assessing the quality of reads in a FastQ file

Parameters

- **input_files** (*dict*) –

  fastq [str] List of file locations

- **metadata** (*dict*) –

  fastq [dict] Required meta data

- **output_files** (*dict*) –

  report [str] Location of the HTML

Returns **array** – First element is a list of the index files. Second element is a list of the matching metadata

Return type list

validate(***kwargs*)

FastQC Validator

Parameters

- **FastQC_file** (*str*) – Location of the FastQ file

- **report_loc** (*str*) – Location of the output report file

4.1.2 TrimGalore

class tool.trimgalore.**trimgalore**(*configuration=None*)
Tool for trimming FASTQ reads that are of low quality

static get_trimgalore_params(*params*)
Function to handle for extraction of commandline parameters

Parameters **params** (*dict*) –

Returns

Return type list

run(*input_files*, *input_metadata*, *output_files*)
The main function to run TrimGalore to remove low quality and very short reads. TrimGalore uses CutAdapt and FASTQC for the analysis.

Parameters

- **input_files** (*dict*) –

  fastq1 [string] Location of the FASTQ file

  fastq2 [string] [OPTIONAL] Location of the paired end FASTQ file

- **metadata** (*dict*) – Matching metadata for the inpit FASTQ files

Returns

- **output_files** (*dict*) –

  fastq1_trimmed [str] Location of the trimmed FASTQ file

  fastq2_trimmed [str] [OPTIONAL] Location of a trimmed paired end FASTQ file

- **output_metadata** (*dict*) – Matching metadata for the output files

trimgalore_paired(***kwargs*)
Trims and removes low quality subsections and reads from paired-end FASTQ files

Parameters

- **fastq_file_in** (*str*) – Location of the input fastq file
- **fastq_file_out** (*str*) – Location of the output fastq file
- **params** (*dict*) – Parameters to use in TrimGalore

Returns Indicator of the success of the function

Return type bool

trimgalore_single(***kwargs*)
Trims and removes low quality subsections and reads from a singed-ended FASTQ file

Parameters

- **fastq_file_in** (*str*) – Location of the input fastq file
- **fastq_file_out** (*str*) – Location of the output fastq file
- **params** (*dict*) – Parameters to use in TrimGalore

Returns Indicator of the success of the function

Return type bool

trimgalore_version(***kwargs*)
Trims and removes low quality subsections and reads from a singed-ended FASTQ file

Parameters

- **fastq_file_in** (*str*) – Location of the input fastq file
- **fastq_file_out** (*str*) – Location of the output fastq file
- **params** (*dict*) – Parameters to use in TrimGalore

Returns Indicator of the success of the function

Return type bool

4.2 Indexers

4.2.1 Bowtie 2

class tool.bowtie_indexer.**bowtieIndexerTool**(*configuration=None*)
Tool for running indexers over a genome FASTA file

bowtie2_indexer(***kwargs*)
Bowtie2 Indexer

Parameters

- **file_loc** (*str*) – Location of the genome assembly FASTA file
- **idx_loc** (*str*) – Location of the output index file

run(*input_files*, *input_metadata*, *output_files*)
Tool for generating assembly aligner index files for use with the Bowtie 2 aligner

Parameters

- **input_files** (*list*) – List with a single str element with the location of the genome assembly FASTA file
- **metadata** (*list*) –

Returns **array** – First element is a list of the index files. Second element is a list of the matching metadata

Return type list

4.2.2 BSgenome Index

class tool.forge_bsgenome.**bsgenomeTool**(*configuration=None*)
Tool for peak calling for iDamID-seq data

bsgenome_creater(***kwargs*)
Make BSgenome index files.Uses an R script that wraps the required code.

Parameters

- **genome** (*str*) –
- **circo_chrom** (*str*) – Comma separated list of chromosome ids that are circular in the genome

- **seed_file_param** (*dict*) – Parameters required for the function to build the seed file
- **genome_2bit** (*str*) –
- **chrom_size** (*str*) –
- **seed_file** (*str*) –
- **bsgenome** (*str*) –

static genome_to_2bit(*genome*, *genome_2bit*)
Generate the 2bit genome file from a FASTA file

Parameters

- **genome** (*str*) – Location of the FASRA genome file
- **genome_2bit** (*str*) – Location of the 2bit genome file

Returns True if successful, False if not.

Return type bool

static get_chrom_size(*genome_2bit*, *chrom_size*, *circ_chrom*)
Generate the chrom.size file and identify the available chromosomes in the 2Bit file.

Parameters

- **genome_2bit** (*str*) – Location of the 2bit genome file
- **chrom_size** (*str*) – Location to save the chrom.size file to
- **circ_chrom** (*list*) – List of chromosomes that are known to be circular

Returns

- *If successful 2 lists* – [0] : List of the linear chromosomes in the 2bit file [1] : List of circular chromosomes in the 2bit file
- *Returns (False, False) if there is an IOError*

run(*input_files*, *input_metadata*, *output_files*)
The main function to run iNPS for peak calling over a given BAM file and matching background BAM file.

Parameters

- **input_files** (*list*) – List of input bam file locations where 0 is the bam data file and 1 is the matching background bam file
- **metadata** (*dict*) –

Returns

- **output_files** (*list*) – List of locations for the output files.
- **output_metadata** (*list*) – List of matching metadata dict objects

4.2.3 BS-Seeker2 Indexer

class tool.bs_seeker_indexer.**bssIndexerTool**(*configuration=None*)
Script from BS-Seeker2 for building the index for alignment. In this case it uses Bowtie2.

bss_build_index(***kwargs*)
Function to submit the FASTA file for the reference sequence and build the required index file used by the aligner.

Parameters

- **fasta_file** (*str*) – Location of the genome FASTA file
- **aligner** (*str*) – Aligner to use by BS-Seeker2. Currently only bowtie2 is available in this build
- **aligner_path** (*str*) – Location of the aligners binary file
- **bss_path** – Location of the BS-Seeker2 libraries
- **idx_out** (*str*) – Location of the output compressed index file

Returns **bam_out** – Location of the output bam alignment file

Return type str

static get_bss_index_params(*params*)
Function to handle to extraction of commandline parameters and formatting them for use in the aligner for BWA ALN

Parameters **params** (*dict*) –

Returns

Return type list

run(*input_files*, *input_metadata*, *output_files*)
Tool for indexing the genome assembly using BS-Seeker2. In this case it is using Bowtie2

Parameters

- **input_files** (*list*) – FASTQ file
- **metadata** (*list*) –

Returns **array** – Location of the filtered FASTQ file

Return type list

4.2.4 BWA

class tool.bwa_indexer.**bwaIndexerTool**(*configuration=None*)
Tool for running indexers over a genome FASTA file

bwa_indexer(***kwargs*)
BWA Indexer

Parameters

- **file_loc** (*str*) – Location of the genome assebly FASTA file
- **idx_out** (*str*) – Location of the output index file

Returns

Return type bool

run(*input_files*, *input_metadata*, *output_files*)
Function to run the BWA over a genome assembly FASTA file to generate the matching index for use with the aligner

Parameters

- **input_files** (*dict*) – List containing the location of the genome assembly FASTA file

- **meta_data** (*dict*) –
- **output_files** (*dict*) – List of outpout files generated

Returns

- **output_files** (*dict*) –

  index [str] Location of the index file defined in the input parameters
- **output_metadata** (*dict*) –

  index [Metadata] Metadata relating to the index file

4.2.5 GEM

class tool.gem_indexer.**gemIndexerTool**(*configuration=None*)

Tool for running indexers over a genome FASTA file

gem_indexer(***kwargs*)

GEM Indexer

Parameters

- **genome_file** (*str*) – Location of the genome assembly FASTA file
- **idx_loc** (*str*) – Location of the output index file

run(*input_files*, *input_metadata*, *output_files*)

Tool for generating assembly aligner index files for use with the GEM indexer

Parameters

- **input_files** (*list*) – List with a single str element with the location of the genome assembly FASTA file
- **input_metadata** (*list*) –

Returns **array** – First element is a list of the index files. Second element is a list of the matching metadata

Return type list

4.2.6 Kallisto

class tool.kallisto_indexer.**kallistoIndexerTool**(*configuration=None*)

Tool for running indexers over a genome FASTA file

kallisto_indexer(***kwargs*)

Kallisto Indexer

Parameters

- **file_loc** (*str*) – Location of the cDNA FASTA file for a genome
- **idx_loc** (*str*) – Location of the output index file

run(*input_files*, *input_metadata*, *output_files*)

Tool for generating assembly aligner index files for use with Kallisto

Parameters

- **input_files** (*list*) – FASTA file location will all the cDNA sequences for a given genome

- **input_metadata** (*list*) –

Returns **array** – First element is a list of the index files. Second element is a list of the matching metadata

Return type list

4.3 Aligners

4.3.1 Bowtie2

class tool.bowtie_aligner.**bowtie2AlignerTool**(*configuration=None*)

Tool for aligning sequence reads to a genome using BWA

bowtie2_aligner_paired(***kwargs*)

Bowtie2 Aligner - Paired End

Parameters

- **genome_file_loc** (*str*) – Location of the genomic fasta
- **read_file_loc1** (*str*) – Location of the FASTQ file
- **read_file_loc2** (*str*) – Location of the FASTQ file
- **bam_loc** (*str*) – Location of the output aligned bam file
- **bt2_1_file** (*str*) – Location of the <genome>.1.bt2 index file
- **bt2_2_file** (*str*) – Location of the <genome>.2.bt2 index file
- **bt2_3_file** (*str*) – Location of the <genome>.3.bt2 index file
- **bt2_4_file** (*str*) – Location of the <genome>.4.bt2 index file
- **bt2_rev1_file** (*str*) – Location of the <genome>.rev.1.bt2 index file
- **bt2_rev2_file** (*str*) – Location of the <genome>.rev.2.bt2 index file
- **aln_params** (*dict*) – Alignment parameters

Returns **bam_loc** – Location of the output file

Return type str

bowtie2_aligner_single(***kwargs*)

Bowtie2 Aligner - Single End

Parameters

- **genome_file_loc** (*str*) – Location of the genomic fasta
- **read_file_loc1** (*str*) – Location of the FASTQ file
- **bam_loc** (*str*) – Location of the output aligned bam file
- **bt2_1_file** (*str*) – Location of the <genome>.1.bt2 index file
- **bt2_2_file** (*str*) – Location of the <genome>.2.bt2 index file
- **bt2_3_file** (*str*) – Location of the <genome>.3.bt2 index file
- **bt2_4_file** (*str*) – Location of the <genome>.4.bt2 index file
- **bt2_rev1_file** (*str*) – Location of the <genome>.rev.1.bt2 index file

- **bt2_rev2_file** (*str*) – Location of the <genome>.rev.2.bt2 index file
- **aln_params** (*dict*) – Alignment parameters

Returns **bam_loc** – Location of the output file

Return type str

static get_aln_params(*params*, *paired=False*)

Function to handle to extraction of commandline parameters and formatting them for use in the aligner for Bowtie2

Parameters

- **params** (*dict*) –
- **paired** (*bool*) – Indicate if the parameters are paired-end specific. [DEFAULT=False]

Returns

Return type list

run(*input_files*, *input_metadata*, *output_files*)

The main function to align bam files to a genome using Bowtie2

Parameters

- **input_files** (*dict*) – File 0 is the genome file location, file 1 is the FASTQ file
- **metadata** (*dict*) –
- **output_files** (*dict*) –

Returns

- **output_files** (*dict*) – First element is a list of output_bam_files, second element is the matching meta data
- **output_metadata** (*dict*)

untar_index(***kwargs*)

Extracts the Bowtie2 index files from the genome index tar file.

Parameters

- **genome_file_name** (*str*) – Location string of the genome fasta file
- **genome_idx** (*str*) – Location of the Bowtie2 index file
- **bt2_1_file** (*str*) – Location of the <genome>.1.bt2 index file
- **bt2_2_file** (*str*) – Location of the <genome>.2.bt2 index file
- **bt2_3_file** (*str*) – Location of the <genome>.3.bt2 index file
- **bt2_4_file** (*str*) – Location of the <genome>.4.bt2 index file
- **bt2_rev1_file** (*str*) – Location of the <genome>.rev.1.bt2 index file
- **bt2_rev2_file** (*str*) – Location of the <genome>.rev.2.bt2 index file

Returns Boolean indicating if the task was successful

Return type bool

4.3.2 BWA - ALN

class tool.bwa_aligner.**bwaAlignerTool**(*configuration=None*)
Tool for aligning sequence reads to a genome using BWA

bwa_aligner_paired(***kwargs*)
BWA ALN Aligner - Paired End

Parameters

- **genome_file_loc** (*str*) – Location of the genomic fasta
- **read_file_loc1** (*str*) – Location of the FASTQ file
- **read_file_loc2** (*str*) – Location of the FASTQ file
- **bam_loc** (*str*) – Location of the output aligned bam file
- **amb_file** (*str*) – Location of the amb index file
- **ann_file** (*str*) – Location of the ann index file
- **bwt_file** (*str*) – Location of the bwt index file
- **pac_file** (*str*) – Location of the pac index file
- **sa_file** (*str*) – Location of the sa index file
- **aln_params** (*dict*) – Alignment parameters

Returns **bam_loc** – Location of the output file

Return type str

bwa_aligner_single(***kwargs*)
BWA ALN Aligner - Single Ended

Parameters

- **genome_file_loc** (*str*) – Location of the genomic fasta
- **read_file_loc** (*str*) – Location of the FASTQ file
- **bam_loc** (*str*) – Location of the output aligned bam file
- **amb_file** (*str*) – Location of the amb index file
- **ann_file** (*str*) – Location of the ann index file
- **bwt_file** (*str*) – Location of the bwt index file
- **pac_file** (*str*) – Location of the pac index file
- **sa_file** (*str*) – Location of the sa index file
- **aln_params** (*dict*) – Alignment parameters

Returns **bam_loc** – Location of the output file

Return type str

static get_aln_params(*params*)
Function to handle to extraction of commandline parameters and formatting them for use in the aligner for BWA ALN

Parameters **params** (*dict*) –

Returns

Return type list

run(*input_files*, *input_metadata*, *output_files*)
The main function to align bam files to a genome using BWA

Parameters

- **input_files** (*dict*) – File 0 is the genome file location, file 1 is the FASTQ file
- **metadata** (*dict*) –
- **output_files** (*dict*) –

Returns

- **output_files** (*dict*) – First element is a list of output_bam_files, second element is the matching meta data
- **output_metadata** (*dict*)

untar_index(***kwargs*)
Extracts the BWA index files from the genome index tar file.

Parameters

- **genome_file_name** (*str*) – Location string of the genome fasta file
- **genome_idx** (*str*) – Location of the BWA index file
- **amb_file** (*str*) – Location of the amb index file
- **ann_file** (*str*) – Location of the ann index file
- **bwt_file** (*str*) – Location of the bwt index file
- **pac_file** (*str*) – Location of the pac index file
- **sa_file** (*str*) – Location of the sa index file

Returns Boolean indicating if the task was successful

Return type bool

4.3.3 BWA - MEM

class tool.bwa_mem_aligner.**bwaAlignerMEMTool**(*configuration=None*)
Tool for aligning sequence reads to a genome using BWA

bwa_aligner_paired(***kwargs*)
BWA MEM Aligner - Paired End

Parameters

- **genome_file_loc** (*str*) – Location of the genomic fasta
- **read_file_loc1** (*str*) – Location of the FASTQ file
- **read_file_loc2** (*str*) – Location of the FASTQ file
- **bam_loc** (*str*) – Location of the output aligned bam file
- **amb_file** (*str*) – Location of the amb index file
- **ann_file** (*str*) – Location of the ann index file
- **bwt_file** (*str*) – Location of the bwt index file
- **pac_file** (*str*) – Location of the pac index file

- **sa_file** (*str*) – Location of the sa index file
- **mem_params** (*dict*) – Alignment parameters

Returns **bam_loc** – Location of the output file

Return type str

bwa_aligner_single(***kwargs*)
BWA MEM Aligner - Single Ended

Parameters

- **genome_file_loc** (*str*) – Location of the genomic fasta
- **read_file_loc** (*str*) – Location of the FASTQ file
- **bam_loc** (*str*) – Location of the output aligned bam file
- **amb_file** (*str*) – Location of the amb index file
- **ann_file** (*str*) – Location of the ann index file
- **bwt_file** (*str*) – Location of the bwt index file
- **pac_file** (*str*) – Location of the pac index file
- **sa_file** (*str*) – Location of the sa index file
- **mem_params** (*dict*) – Alignment parameters

Returns **bam_loc** – Location of the output file

Return type str

static get_mem_params(*params*)
Function to handle to extraction of commandline parameters and formatting them for use in the aligner for BWA MEM

Parameters **params** (*dict*) –

Returns

Return type list

run(*input_files*, *input_metadata*, *output_files*)
The main function to align bam files to a genome using BWA

Parameters

- **input_files** (*dict*) – File 0 is the genome file location, file 1 is the FASTQ file
- **metadata** (*dict*) –
- **output_files** (*dict*) –

Returns

- **output_files** (*dict*) – First element is a list of output_bam_files, second element is the matching meta data
- **output_metadata** (*dict*)

untar_index(***kwargs*)
Extracts the BWA index files from the genome index tar file.

Parameters

- **genome_file_name** (*str*) – Location string of the genome fasta file

- **genome_idx** (*str*) – Location of the BWA index file
- **amb_file** (*str*) – Location of the amb index file
- **ann_file** (*str*) – Location of the ann index file
- **bwt_file** (*str*) – Location of the bwt index file
- **pac_file** (*str*) – Location of the pac index file
- **sa_file** (*str*) – Location of the sa index file

Returns Boolean indicating if the task was successful

Return type bool

4.3.4 BS-Seeker2 Aligner

class tool.bs_seeker_aligner.**bssAlignerTool**(*configuration=None*)

Script from BS-Seeker2 for building the index for alignment. In this case it uses Bowtie2.

bs_seeker_aligner(***kwargs*)

Alignment of the paired ends to the reference genome

Generates bam files for the alignments

This is performed by running the external program rather than reimplementing the code from the main function to make it easier when it comes to updating the changes in BS-Seeker2

Parameters

- **input_fastq1** (*str*) – Location of paired end FASTQ file 1
- **input_fastq2** (*str*) – Location of paired end FASTQ file 2
- **aligner** (*str*) – Aligner to use
- **aligner_path** (*str*) – Location of the aligner
- **genome_fasta** (*str*) – Location of the genome FASTA file
- **genome_idx** (*str*) – Location of the tar.gz genome index file
- **bam_out** (*str*) – Location of the aligned bam file

Returns **bam_out** – Location of the BAM file generated during the alignment.

Return type file

bs_seeker_aligner_single(***kwargs*)

Alignment of the paired ends to the reference genome

Generates bam files for the alignments

This is performed by running the external program rather than reimplementing the code from the main function to make it easier when it comes to updating the changes in BS-Seeker2

Parameters

- **input_fastq1** (*str*) – Location of paired end FASTQ file 1
- **input_fastq2** (*str*) – Location of paired end FASTQ file 2
- **aligner** (*str*) – Aligner to use
- **aligner_path** (*str*) – Location of the aligner
- **genome_fasta** (*str*) – Location of the genome FASTA file

- **genome_idx** (*str*) – Location of the tar.gz genome index file
- **bam_out** (*str*) – Location of the aligned bam file

Returns **bam_out** – Location of the BAM file generated during the alignment.

Return type file

static get_aln_params(*params*, *paired=False*)
Function to handle to extraction of commandline parameters and formatting them for use in the aligner for Bowtie2

Parameters

- **params** (*dict*) –
- **paired** (*bool*) – Indicate if the parameters are paired-end specific. [DEFAULT=False]

Returns

Return type list

run(*input_files*, *input_metadata*, *output_files*)
Tool for indexing the genome assembly using BS-Seeker2. In this case it is using Bowtie2

Parameters

- **input_files** (*list*) – FASTQ file
- **output_files** (*list*) – Results files.
- **metadata** (*list*) –

Returns **array** – Location of the filtered FASTQ file

Return type list

run_aligner(*genome_idx*, *bam_out*, *script*, *params*)
Run the aligner

Parameters

- **genome_idx** (*str*) – Location of the genome index archive
- **bam_out** (*str*) – Location of the output bam file
- **script** (*str*) – Location of the BS Seeker2 aligner script
- **params** (*list*) – Parameter list for the aligner

Returns True if the function completed successfully

Return type bool

4.4 Filters

4.4.1 BioBamBam Filter

class tool.biobambam_filter.**biobambam**(*configuration=None*)
Tool to sort and filter bam files

biobambam_filter_alignments(***kwargs*)
Sorts and filters the bam file.

It is important that all duplicate alignments have been removed. This can be run as an intermediate step, but should always be run as a check to ensure that the files are sorted and duplicates have been removed.

Parameters

- **bam_file_in** (*str*) – Location of the input bam file
- **bam_file_out** (*str*) – Location of the output bam file
- **tmp_dir** (*str*) – Tmp location for intermediate files during the sorting

Returns **bam_file_out** – Location of the output bam file

Return type str

run(*input_files*, *input_metadata*, *output_files*)
The main function to run BioBAMBAMfilter to remove duplicates and spurious reads from the FASTQ files before analysis.

Parameters

- **input_files** (*dict*) – List of input bam file locations where 0 is the bam data file
- **metadata** (*dict*) – Matching meta data for the input files
- **output_files** (*dict*) – List of output file locations

Returns

- **output_files** (*dict*) – Filtered bam fie.
- **output_metadata** (*dict*) – List of matching metadata dict objects

4.4.2 BS-Seeker2 Filter

class tool.bs_seeker_filter.**filterReadsTool**(*configuration=None*)
Script from BS-Seeker2 for filtering FASTQ files to remove repeats

bss_seeker_filter(***kwargs*)
This is optional, but removes reads that can be problematic for the alignment of whole genome datasets.

If performing RRBS then this step can be skipped

This is a function that is installed as part of the BS-Seeker installation process.

Parameters **infile** (*str*) – Location of the FASTQ file

Returns **outfile** – Location of the filtered FASTQ file

Return type str

run(*input_files*, *input_metadata*, *output_files*)
Tool for filtering duplicate entries from FASTQ files using BS-Seeker2

Parameters

- **input_files** (*list*) – FASTQ file
- **input_metadata** (*list*) –

Returns **array** – Location of the filtered FASTQ file

Return type list

4.4.3 Trim Galore

class tool.trimgalore.**trimgalore**(*configuration=None*)
Tool for trimming FASTQ reads that are of low quality

static get_trimgalore_params(*params*)
Function to handle for extraction of commandline parameters

Parameters **params** (*dict*) –

Returns

Return type list

run(*input_files*, *input_metadata*, *output_files*)
The main function to run TrimGalore to remove low quality and very short reads. TrimGalore uses CutAdapt and FASTQC for the analysis.

Parameters

- **input_files** (*dict*) –

  fastq1 [string] Location of the FASTQ file

  fastq2 [string] [OPTIONAL] Location of the paired end FASTQ file

- **metadata** (*dict*) – Matching metadata for the inpit FASTQ files

Returns

- **output_files** (*dict*) –

  fastq1_trimmed [str] Location of the trimmed FASTQ file

  fastq2_trimmed [str] [OPTIONAL] Location of a trimmed paired end FASTQ file

- **output_metadata** (*dict*) – Matching metadata for the output files

trimgalore_paired(***kwargs*)
Trims and removes low quality subsections and reads from paired-end FASTQ files

Parameters

- **fastq_file_in** (*str*) – Location of the input fastq file
- **fastq_file_out** (*str*) – Location of the output fastq file
- **params** (*dict*) – Parameters to use in TrimGalore

Returns Indicator of the success of the function

Return type bool

trimgalore_single(***kwargs*)
Trims and removes low quality subsections and reads from a singed-ended FASTQ file

Parameters

- **fastq_file_in** (*str*) – Location of the input fastq file
- **fastq_file_out** (*str*) – Location of the output fastq file
- **params** (*dict*) – Parameters to use in TrimGalore

Returns Indicator of the success of the function

Return type bool

trimgalore_version(***kwargs*)
Trims and removes low quality subsections and reads from a singed-ended FASTQ file

Parameters

- **fastq_file_in** (*str*) – Location of the input fastq file
- **fastq_file_out** (*str*) – Location of the output fastq file
- **params** (*dict*) – Parameters to use in TrimGalore

Returns Indicator of the success of the function

Return type bool

4.5 Peak Calling

4.5.1 BS-Seeker2 Methylation Caller

4.5.2 iDEAR

class tool.idear.**idearTool**(*configuration=None*)
Tool for peak calling for iDamID-seq data

idear_peak_calling(***kwargs*)
Make iDamID-seq peak calls. These are saved as bed files That can then get displayed on genome browsers. Uses an R script that wraps teh iDEAR protocol.

Parameters

- **sample_name** (*str*) –
- **bg_name** (*str*) –
- **sample_bam_tar_file** (*str*) – Location of the aligned sequences in bam format
- **bg_bam_tar_file** (*str*) – Location of the aligned background sequences in bam format
- **species** (*str*) – Species name for the alignments
- **assembly** (*str*) – Assembly used for teh aligned sequences
- **peak_bed** (*str*) – Location of the peak bed file

Returns **peak_bed** – Location of the collated bed file

Return type str

run(*input_files*, *input_metadata*, *output_files*)
The main function to run iNPS for peak calling over a given BAM file and matching background BAM file.

Parameters

- **input_files** (*list*) – List of input bam file locations where 0 is the bam data file and 1 is the matching background bam file
- **metadata** (*dict*) –

Returns

- **output_files** (*list*) – List of locations for the output files.

- **output_metadata** (*list*) – List of matching metadata dict objects

4.5.3 iNPS

class tool.inps.**inps**(*configuration=None*)

Tool for peak calling for MNase-seq data

inps_peak_calling(***kwargs*)

Convert Bam to Bed then make Nucleosome peak calls. These are saved as bed files That can then get displayed on genome browsers.

Parameters

- **bam_file** (*str*) – Location of the aligned sequences in bam format
- **peak_bed** (*str*) – Location of the collated bed file of nucleosome peak calls

Returns **peak_bed** – Location of the collated bed file of nucleosome peak calls

Return type str

run(*input_files*, *input_metadata*, *output_files*)

The main function to run iNPS for peak calling over a given BAM file and matching background BAM file.

Parameters

- **input_files** (*list*) – List of input bam file locations where 0 is the bam data file and 1 is the matching background bam file
- **metadata** (*dict*) –

Returns

- **output_files** (*list*) – List of locations for the output files.
- **output_metadata** (*list*) – List of matching metadata dict objects

4.5.4 Kallisto Quantification

class tool.kallisto_quant.**kallistoQuantificationTool**(*configuration=None*)

Tool for quantifying RNA-seq alignments to calculate expression levels of genes within a genome.

kallisto_quant_paired(***kwargs*)

Kallisto quantifier for paired end RNA-seq data

Parameters

- **idx_loc** (*str*) – Location of the output index file
- **fastq_file_loc_01** (*str*) – Location of the FASTQ sequence file
- **fastq_file_loc_02** (*str*) – Location of the paired FASTQ sequence file

Returns **wig_file_loc** – Location of the wig file containing the levels of expression

Return type loc

kallisto_quant_single(***kwargs*)

Kallisto quantifier for single end RNA-seq data

Parameters

- **idx_loc** (*str*) – Location of the output index file

- **fastq_file_loc** (*str*) – Location of the FASTQ sequence file

Returns **wig_file_loc** – Location of the wig file containing the levels of expression

Return type loc

kallisto_tsv2bed(***kwargs*)
So that the TSV file can be viewed within the genome browser it is handy to convert the file to a BigBed file

kallisto_tsv2gff(***kwargs*)
So that the TSV file can be viewed within the genome browser it is handy to convert the file to a BigBed file

static load_gff_ensembl(*gff_file*)
Function to extract all of the genes and their locations from a GFF file generated by ensembl

static load_gff_ucsc(*gff_file*)
Function to extract all of the genes and their locations from a GFF file generated by ensembl

run(*input_files*, *input_metadata*, *output_files*)
Tool for calculating the level of expression

Parameters

- **input_files** (*list*) – Kallisto index file for the FASTQ file for the experiemtnal alignments
- **input_metadata** (*list*) –

Returns **array** – First element is a list of the index files. Second element is a list of the matching metadata

Return type list

static seq_read_stats(*file_in*)
Calculate the mean and standard deviation of the reads in a fastq file

Parameters **file_in** (*str*) – Location of a FASTQ file

Returns mean : Mean length of sequenced strands std : Standard deviation of lengths of sequenced strands

Return type dict

4.5.5 MACS2

class tool.macs2.**macs2**(*configuration=None*)
Tool for peak calling for ChIP-seq data

static get_macs2_params(*params*)
Function to handle to extraction of commandline parameters and formatting them for use in the aligner for BWA ALN

Parameters **params** (*dict*) –

Returns

Return type list

macs2_peak_calling(***kwargs*)
Function to run MACS2 for peak calling on aligned sequence files and normalised against a provided background set of alignments.

Parameters

- **name** (*str*) – Name to be used to identify the files
- **bam_file** (*str*) – Location of the aligned FASTQ files as a bam file
- **bai_file** (*str*) – Location of the bam index file
- **bam_file_bgd** (*str*) – Location of the aligned FASTQ files as a bam file representing background values for the cell
- **bai_file_bgd** (*str*) – Location of the background bam index file
- **narrowpeak** (*str*) – Location of the output narrowpeak file
- **summits_bed** (*str*) – Location of the output summits bed file
- **broadpeak** (*str*) – Location of the output broadpeak file
- **gappedpeak** (*str*) – Location of the output gappedpeak file
- **chromosome** (*str*) – If the tool is to be run over a single chromosome the matching chromosome name should be specified. If None then the whole bam file is analysed

Returns

- **narrowPeak** (*file*) – BED6+4 file - ideal for transcription factor binding site identification
- **summitPeak** (*file*) – BED4+1 file - Contains the peak summit locations for everypeak
- **broadPeak** (*file*) – BED6+3 file - ideal for histone binding site identification
- **gappedPeak** (*file*) – BED12+3 file - Contains a merged set of the broad and narrow peak files
- *Definitions defined for each of these files have come from the MACS2*
- **documentation described in the docs at https** (*//github.com/taoliu/MACS*)

macs2_peak_calling_nobgd(***kwargs*)

Function to run MACS2 for peak calling on aligned sequence files without a background dataset for normalisation.

Parameters

- **name** (*str*) – Name to be used to identify the files
- **bam_file** (*str*) – Location of the aligned FASTQ files as a bam file
- **bai_file** (*str*) – Location of the bam index file
- **narrowpeak** (*str*) – Location of the output narrowpeak file
- **summits_bed** (*str*) – Location of the output summits bed file
- **broadpeak** (*str*) – Location of the output broadpeak file
- **gappedpeak** (*str*) – Location of the output gappedpeak file
- **chromosome** (*str*) – If the tool is to be run over a single chromosome the matching chromosome name should be specified. If None then the whole bam file is analysed

Returns

- **narrowPeak** (*file*) – BED6+4 file - ideal for transcription factor binding site identification
- **summitPeak** (*file*) – BED4+1 file - Contains the peak summit locations for everypeak
- **broadPeak** (*file*) – BED6+3 file - ideal for histone binding site identification

- **gappedPeak** (*file*) – BED12+3 file - Contains a merged set of the broad and narrow peak files
- *Definitions defined for each of these files have come from the MACS2*
- **documentation described in the docs at https** (*//github.com/taoliu/MACS*)

run (*input_files*, *input_metadata*, *output_files*)
The main function to run MACS 2 for peak calling over a given BAM file and matching background BAM file.

Parameters

- **input_files** (*dict*) – List of input bam file locations where 0 is the bam data file and 1 is the matching background bam file
- **metadata** (*dict*) –

Returns

- **output_files** (*dict*) – List of locations for the output files.
- **output_metadata** (*dict*) – List of matching metadata dict objects

4.6 Hi-C Parsing

The following tools are a split out of the Hi-C pipelines generated to use the TADbit library.

4.6.1 FASTQ mapping

class tool.tb_full_mapping.**tbFullMappingTool**
Tool for mapping fastq paired end files to the GEM index files

run (*input_files*, *input_metadata*, *output_files*)
The main function to map the FASTQ files to the GEM file over different window sizes ready for alignment

Parameters

- **input_files** (*list*) –

  gem_file [str] Location of the genome GEM index file

  fastq_file_bgd [str] Location of the FASTQ file
- **metadata** (*dict*) –

  windows [list] List of lists with the window sizes to be computed

  enzyme_name [str] Restriction enzyme used [OPTIONAL]

Returns

- **output_files** (*list*) – List of locations for the output files.
- **output_metadata** (*list*) – List of matching metadata dict objects

tb_full_mapping_frag (***kwargs*)
Function to map the FASTQ files to the GEM file based on fragments derived from the restriction enzyme that was used.

Parameters

- **gem_file** (*str*) – Location of the genome GEM index file

- **fastq_file_bgd** (*str*) – Location of the FASTQ file
- **enzyme_name** (*str*) – Restriction enzyme name (MboI)
- **windows** (*list*) – List of lists with the window sizes to be computed
- **window_file** (*str*) – Location of the first window index file

Returns **window_file** – Location of the window index file

Return type str

tb_full_mapping_iter(***kwargs*)

Function to map the FASTQ files to the GEM file over different window sizes ready for alignment

Parameters

- **gem_file** (*str*) – Location of the genome GEM index file
- **fastq_file_bgd** (*str*) – Location of the FASTQ file
- **windows** (*list*) – List of lists with the window sizes to be computed
- **window1** (*str*) – Location of the first window index file
- **window2** (*str*) – Location of the second window index file
- **window3** (*str*) – Location of the third window index file
- **window4** (*str*) – Location of the fourth window index file

Returns

- **window1** (*str*) – Location of the first window index file
- **window2** (*str*) – Location of the second window index file
- **window3** (*str*) – Location of the third window index file
- **window4** (*str*) – Location of the fourth window index file

4.6.2 Map Parsing

class tool.tb_parse_mapping.**tbParseMappingTool**

Tool for parsing the mapped reads and generating the list of paired ends that have a match at both ends.

run(*input_files*, *input_metadata*, *output_files*)

The main function to map the aligned reads and return the matching pairs. Parsing of the mappings can be either iterative of fragment based. If it is to be iteractive then the locations of 4 output file windows for each end of the paired end window need to be provided. If it is fragment based, then only 2 window locations need to be provided along within an enzyme name.

Parameters

- **input_files** (*list*) –

  genome_file [str] Location of the genome FASTA file

  window1_1 [str] Location of the first window index file

  window1_2 [str] Location of the second window index file

  window1_3 [str] [OPTIONAL] Location of the third window index file

  window1_4 [str] [OPTIONAL] Location of the fourth window index file

  window2_1 [str] Location of the first window index file

window2_2 [str] Location of the second window index file

window2_3 [str] [OPTIONAL] Location of the third window index file

window2_4 [str] [OPTIONAL] Location of the fourth window index file

- **metadata** (*dict*) –

  windows [list] List of lists with the window sizes to be computed

  enzyme_name [str] Restricture enzyme name

  mapping [list] The mapping function used. The options are iter or frag.

Returns

- **output_files** (*list*) – List of locations for the output files.
- **output_metadata** (*dict*) – Dict of matching metadata dict objects

Example

Iterative:

```
from tool import tb_parse_mapping

genome_file = 'genome.fasta'

root_name_1 = "/tmp/data/expt_source_1".split
root_name_2 = "/tmp/data/expt_source_2".split
windows = [[1,25], [1,50], [1,75], [1,100]]

windows1 = []
windows2 = []

for w in windows:
    tail = "_full_" + w[0] + "-" + w[1] + ".map"
    windows1.append('/'.join(root_name_1) + tail)
    windows2.append('/'.join(root_name_2) + tail)

files = [genome_file] + windows1 + windows2

tpm = tb_parse_mapping.tb_parse_mapping()
metadata = {'enzyme_name' : 'MboI', 'mapping' : ['iter', 'iter'], 'expt_name'␣
↪= 'test'}
tpm_files, tpm_meta = tpm.run(files, metadata)
```

Fragment based mapping:

```
from tool import tb_parse_mapping

genome_file = 'genome.fasta'

root_name_1 = "/tmp/data/expt_source_1".split
root_name_2 = "/tmp/data/expt_source_2".split
windows = [[1,100]]

start = windows[0][0]
end   = windows[0][1]
```

(continues on next page)

(continued from previous page)

```
window1_1 = '/'.join(root_name_1) + "_full_" + start + "-" + end + ".map"
window1_2 = '/'.join(root_name_1) + "_frag_" + start + "-" + end + ".map"

window2_1 = '/'.join(root_name_2) + "_full_" + start + "-" + end + ".map"
window2_2 = '/'.join(root_name_2) + "_frag_" + start + "-" + end + ".map"

files = [
    genome_file,
    window1_1, window1_2,
    window2_1, window2_2,
]

tpm = tb_parse_mapping.tb_parse_mapping()
metadata = {'enzyme_name' : 'MboI', 'mapping' : ['frag', 'frag'], 'expt_name'
↪= 'test'}
tpm_files, tpm_meta = tpm.run(files, metadata)
```

tb_parse_mapping_frag(***kwargs*)

Function to map the aligned reads and return the matching pairs

Parameters

- **genome_seq** (*dict*) – Object containing the sequence of each of the chromosomes
- **enzyme_name** (*str*) – Name of the enzyme used to digest the genome
- **window1_full** (*str*) – Location of the first window index file
- **window1_frag** (*str*) – Location of the second window index file
- **window2_full** (*str*) – Location of the first window index file
- **window2_frag** (*str*) – Location of the second window index file
- **reads** (*str*) – Location of the reads thats that has a matching location at both ends of the paired reads

Returns **reads** – Location of the intersection of mapped reads that have matching reads in both pair end files

Return type str

tb_parse_mapping_iter(***kwargs*)

Function to map the aligned reads and return the matching pairs

Parameters

- **genome_seq** (*dict*) – Object containing the sequence of each of the chromosomes
- **enzyme_name** (*str*) – Name of the enzyme used to digest the genome
- **window1_1** (*str*) – Location of the first window index file
- **window1_2** (*str*) – Location of the second window index file
- **window1_3** (*str*) – Location of the third window index file
- **window1_4** (*str*) – Location of the fourth window index file
- **window2_1** (*str*) – Location of the first window index file
- **window2_2** (*str*) – Location of the second window index file
- **window2_3** (*str*) – Location of the third window index file

- **window2_4** (*str*) – Location of the fourth window index file
- **reads** (*str*) – Location of the reads thats that has a matching location at both ends of the paired reads

Returns **reads** – Location of the intersection of mapped reads that have matching reads in both pair end files

Return type str

4.6.3 Filter Aligned Reads

class tool.tb_filter.**tbFilterTool**(*configuration=None*)

Tool for filtering out experimetnal artifacts from the aligned data

run(*input_files*, *input_metadata*, *output_files*)

The main function to filter the reads to remove experimental artifacts

Parameters

- **input_files** (*list*) –

  reads [str] Location of the reads thats that has a matching location at both ends of the paired reads

- **metadata** (*dict*) –

  conservative [bool] Level of filtering to apply [DEFAULT : True]

Returns

- **output_files** (*list*) – List of locations for the output files.
- **output_metadata** (*list*) – List of matching metadata dict objects

tb_filter(***kwargs*)

Function to filter out expoerimental artifacts

Parameters

- **reads** (*str*) – Location of the reads thats that has a matching location at both ends of the paired reads
- **filtered_reads_file** (*str*) – Location of the filtered reads
- **conservative** (*bool*) – Level of filtering [DEFAULT : True]

Returns **filtered_reads** – Location of the filtered reads

Return type str

4.6.4 Identify TADs and Compartments

class tool.tb_segment.**tbSegmentTool**

Tool for finding tads and compartments in an adjacency matrix

run(*input_files*, *input_metadata*, *output_files*)

The main function to the predict TAD sites and compartments for a given resolution from the Hi-C matrix

Parameters

- **input_files** (*list*) –

  bamin [str] Location of the tadbit bam paired reads

biases [str] Location of the pickle hic biases

- **metadata** (*dict*) –

  resolution [int] Resolution of the Hi-C

  workdir [str] Location of working directory

  ncpus [int] Number of cpus to use

Returns

- **output_files** (*list*) – List of locations for the output files.
- **output_metadata** (*list*) – List of matching metadata dict objects

tb_segment (***kwargs*)

Function to find tads and compartments in the Hi-C matrix

Parameters

- **bamin** (*str*) – Location of the tadbit bam paired reads
- **biases** (*str*) – Location of the pickle hic biases
- **resolution** (*int*) – Resolution of the Hi-C
- **callers** (*str*) – 1 for ta calling, 2 for compartment calling
- **workdir** (*str*) – Location of working directory
- **ncpus** (*int*) – Number of cpus to use

Returns

- **compartments** (*str*) – Location of tsv file with compartment definition
- **tads** (*str*) – Location of tsv file with tad definition
- **filtered_bins** (*str*) – Location of filtered_bins png

4.6.5 Normalize paired end reads file

class tool.tb_normalize.**tbNormalizeTool**

Tool for normalizing an adjacency matrix

run (*input_files*, *input_metadata*, *output_files*)

The main function for the normalization of the Hi-C matrix to a given resolution

Parameters

- **input_files** (*list*) –

  bamin [str] Location of the tadbit bam paired reads

- **metadata** (*dict*) –

  normalization: str normalization(s) to apply. Order matters. Choices: [Vanilla, oneD]

  resolution [str] Resolution of the Hi-C

  min_perc [str] lower percentile from which consider bins as good.

  max_perc [str] upper percentile until which consider bins as good.

  workdir [str] Location of working directory

  ncpus [str] Number of cpus to use

min_count [str] minimum number of reads mapped to a bin (recommended value could be 2500). If set this option overrides the perc_zero

fasta: str Location of the fasta file with genome sequence, to compute GC content and number of restriction sites per bin. Required for oneD normalization

mappability: str Location of the file with mappability, required for oneD normalization

rest_enzyme: str For oneD normalization. Name of the restriction enzyme used to do the Hi-C experiment

Returns

- **output_files** (*list*) – List of locations for the output files.
- **output_metadata** (*list*) – List of matching metadata dict objects

tb_normalize (***kwargs*)

Function to normalize to a given resolution the Hi-C matrix

Parameters

- **bamin** (*str*) – Location of the tadbit bam paired reads
- **normalization** (*str*) – normalization(s) to apply. Order matters. Choices: [Vanilla, oneD]
- **resolution** (*str*) – Resolution of the Hi-C
- **min_perc** (*str*) – lower percentile from which consider bins as good.
- **max_perc** (*str*) – upper percentile until which consider bins as good.
- **workdir** (*str*) – Location of working directory
- **ncpus** (*str*) – Number of cpus to use
- **min_count** (*str*) – minimum number of reads mapped to a bin (recommended value could be 2500). If set this option overrides the perc_zero
- **fasta** (*str*) – Location of the fasta file with genome sequence, to compute GC content and number of restriction sites per bin. Required for oneD normalization
- **mappability** (*str*) – Location of the file with mappability, required for oneD normalization
- **rest_enzyme** (*str*) – For oneD normalization. Name of the restriction enzyme used to do the Hi-C experiment

Returns

- **hic_biases** (*str*) – Location of HiC biases pickle file
- **interactions** (*str*) – Location of interaction decay vs genomic distance pdf
- **filtered_bins** (*str*) – Location of filtered_bins png

4.6.6 Extract binned matrix from paired end reads file

class tool.tb_bin.**tbBinTool**

Tool for binning an adjacency matrix

run (*input_files*, *input_metadata*, *output_files*)

The main function to the predict TAD sites for a given resolution from the Hi-C matrix

Parameters

- **input_files** (*list*) –

  bamin [str] Location of the tadbit bam paired reads

  biases [str] Location of the pickle hic biases

- **input_metadata** (*dict*) –

  resolution [int] Resolution of the Hi-C

  coord1 [str] Coordinate of the region to retrieve. By default all genome, arguments can be either one chromosome name, or the coordinate in the form: "-c chr3:110000000-120000000"

  coord2 [str] Coordinate of a second region to retrieve the matrix in the intersection with the first region.

  norm [str] [['raw']] normalization(s) to apply. Order matters. Choices: [norm, decay, raw]

  workdir [str] Location of working directory

  ncpus [int] Number of cpus to use

Returns

- **output_files** (*list*) – List of locations for the output files.
- **output_metadata** (*list*) – List of matching metadata dict objects

tb_bin(***kwargs*)

Function to bin to a given resolution the Hi-C matrix

Parameters

- **bamin** (*str*) – Location of the tadbit bam paired reads
- **biases** (*str*) – Location of the pickle hic biases
- **resolution** (*int*) – Resolution of the Hi-C
- **coord1** (*str*) – Coordinate of the region to retrieve. By default all genome, arguments can be either one chromosome name, or the coordinate in the form: "-c chr3:110000000-120000000"
- **coord2** (*str*) – Coordinate of a second region to retrieve the matrix in the intersection with the first region.
- **norm** (*list*) – [['raw']] normalization(s) to apply. Order matters. Choices: [norm, decay, raw]
- **workdir** (*str*) – Location of working directory
- **ncpus** (*int*) – Number of cpus to use

Returns

- **hic_contacts_matrix_raw** (*str*) – Location of HiC raw matrix in text format
- **hic_contacts_matrix_nrm** (*str*) – Location of HiC normalized matrix in text format
- **hic_contacts_matrix_raw_fig** (*str*) – Location of HiC raw matrix in png format
- **hic_contacts_matrix_norm_fig** (*str*) – Location of HiC normalized matrix in png format

4.6.7 Save Matrix to HDF5 File

class tool.tb_save_hdf5_matrix.**tbSaveAdjacencyHDF5Tool**
Tool for filtering out experimetnal artifacts from the aligned data

run(*input_files*, *output_files*, *metadata=None*)
The main function save the adjacency list from Hi-C into an HDF5 index file at the defined resolutions.

Parameters

- **input_files** (*list*) –

  adj_list [str] Location of the adjacency list

  hdf5_file [str] Location of the HDF5 output matrix file

- **metadata** (*dict*) –

  resolutions [list] Levels of resolution for the adjacency list to be daved at

  assembly [str] Assembly of the aligned sequences

  normalized [bool] Whether the dataset should be normalised before saving

Returns

- **output_files** (*list*) – List of locations for the output files.
- **output_metadata** (*list*) – List of matching metadata dict objects

tb_matrix_hdf5(***kwargs*)
Function to the Hi-C matrix into an HDF5 file

This has to be run sequentially as it is not possible for multiple streams to write to the same HDF5 file. This is a run once and leave operatation. There also needs to be a check that no other process is writing to the HDF5 file at the same time. This should be done at the stage and unstaging level to prevent to file getting written to by multiple processes and generating conflicts.

This needs to include attributes for the chromosomes for each resolution - See the mg-rest-adjacency hdf5_reader for further details about the requirement. This prevents the need for secondary storage details outside of the HDF5 file.

Parameters

- **hic_data** (*hic_data*) – Hi-C data object
- **hdf5_file** (*str*) – Location of the HDF5 output matrix file
- **resolution** (*int*) – Resolution to read teh Hi-C adjacency list at
- **chromosomes** (*list*) – List of listsd of the chromosome names and their size in the order that they are presented for indexing

Returns **hdf5_file** – Location of the HDF5 output matrix file

Return type str

4.6.8 Generate TAD Predictions

class tool.tb_generate_tads.**tbGenerateTADsTool**
Tool for taking the adjacency lists and predicting TADs

run(*input_files*, *output_files*, *metadata=None*)
The main function to the predict TAD sites for a given resolution from the Hi-C matrix

Parameters

- **input_files** (*list*) –

  adj_list [str] Location of the adjacency list

- **metadata** (*dict*) –

  resolutions [list] Levels of resolution for the adjacency list to be daved at

  assembly [str] Assembly of the aligned sequences

Returns

- **output_files** (*list*) – List of locations for the output files.
- **output_metadata** (*list*) – List of matching metadata dict objects

tb_generate_tads (***kwargs*)
Function to the predict TAD sites for a given resolution from the Hi-C matrix

Parameters

- **expt_name** (*str*) – Location of the adjacency list
- **matrix_file** (*str*) – Location of the HDF5 output matrix file
- **resolution** (*int*) – Resolution to read the Hi-C adjacency list at
- **tad_file** (*str*) – Location of the output TAD file

Returns **tad_file** – Location of the output TAD file

Return type str

tb_hic_chr (***kwargs*)
Get the list of chromosomes in the adjacency list

tb_merge_tad_files (***kwargs*)
Merge 2 TAD adjacnecny list files

4.6.9 Generate 3D models from binned interaction matrix

class tool.tb_model.**tbModelTool**
Tool for normalizing an adjacency matrix

run (*input_files*, *input_metadata*, *output_files*)
The main function for the normalization of the Hi-C matrix to a given resolution

Parameters

- **input_files** (*list*) –

  hic_contacts_matrix_norm [str] Location of the tab-separated normalized matrix

- **metadata** (*dict*) –

  optimize_only: bool True if only optimize, False for computing the models and stats

  gen_pos_chrom_name [str] Coordinates of the genomic region to model.

  resolution [str] Resolution of the Hi-C

  gen_pos_begin [int] Genomic coordinate from which to start modeling.

  gen_pos_end [int] Genomic coordinate where to end modeling.

  num_mod_comp [int] Number of models to compute for each optimization step.

num_mod_comp [int] Number of models to keep.

max_dist [str] Range of numbers for optimal maxdist parameter, i.e. 400:1000:100; or just a single number e.g. 800; or a list of numbers e.g. 400 600 800 1000.

upper_bound [int] Range of numbers for optimal upfreq parameter, i.e. 0:1.2:0.3; or just a single number e.g. 0.8; or a list of numbers e.g. 0.1 0.3 0.5 0.9.

lower_bound [int] Range of numbers for optimal low parameter, i.e. -1.2:0:0.3; or just a single number e.g. -0.8; or a list of numbers e.g. -0.1 -0.3 -0.5 -0.9.

cutoff [str] Range of numbers for optimal cutoff distance. Cutoff is computed based on the resolution. This cutoff distance is calculated taking as reference the diameter of a modeled particle in the 3D model. i.e. 1.5:2.5:0.5; or just a single number e.g. 2; or a list of numbers e.g. 2 2.5.

workdir [str] Location of working directory

ncpus [str] Number of cpus to use

Returns

- **output_files** (*list*) – List of locations for the output files.
- **output_metadata** (*list*) – List of matching metadata dict objects

tb_model (***kwargs*)

Function to normalize to a given resolution the Hi-C matrix

Parameters

- **optimize_only** (*bool*) – True if only optimize, False for computing the models and stats
- **hic_contacts_matrix_norm** (*str*) – Location of the tab-separated normalized matrix
- **resolution** (*str*) – Resolution of the Hi-C
- **gen_pos_chrom_name** (*str*) – Coordinates of the genomic region to model.
- **gen_pos_begin** (*int*) – Genomic coordinate from which to start modeling.
- **gen_pos_end** (*int*) – Genomic coordinate where to end modeling.
- **num_mod_comp** (*int*) – Number of models to compute for each optimization step.
- **num_mod_comp** – Number of models to keep.
- **max_dist** (*str*) – Range of numbers for optimal maxdist parameter, i.e. 400:1000:100; or just a single number e.g. 800; or a list of numbers e.g. 400 600 800 1000.
- **upper_bound** (*int*) – Range of numbers for optimal upfreq parameter, i.e. 0:1.2:0.3; or just a single number e.g. 0.8; or a list of numbers e.g. 0.1 0.3 0.5 0.9.
- **lower_bound** (*int*) – Range of numbers for optimal low parameter, i.e. -1.2:0:0.3; or just a single number e.g. -0.8; or a list of numbers e.g. -0.1 -0.3 -0.5 -0.9.
- **cutoff** (*str*) – Range of numbers for optimal cutoff distance. Cutoff is computed based on the resolution. This cutoff distance is calculated taking as reference the diameter of a modeled particle in the 3D model. i.e. 1.5:2.5:0.5; or just a single number e.g. 2; or a list of numbers e.g. 2 2.5.
- **workdir** (*str*) – Location of working directory
- **ncpus** (*str*) – Number of cpus to use

Returns

- **tadkit_models** (*str*) – Location of TADkit json file
- **modeling_stats** (*str*) – Location of the folder with the modeling files and stats

CHAPTER 5

Utility Functions

5.1 Common Functions

The following functions are ones that have been used across multiple tools for transformations of the data when requried.

class tool.common.**cd**(*newpath*)
Context manager for changing the current working directory

5.2 Alignment Utilities

class tool.aligner_utils.**alignerUtils**
Functions for downloading and processing N-seq FastQ files. Functions provided allow for the downloading and indexing of the genome assemblies.

static bowtie2_align_reads(*genome_file*, *bam_loc*, *params*, *reads_file_1*, *reads_file_2=None*)
Map the reads to the genome using BWA.

Parameters

- **genome_file** (*str*) – Location of the assembly file in the file system
- **reads_file** (*str*) – Location of the reads file in the file system
- **bam_loc** (*str*) – Location of the output file

bowtie2_untar_index(*genome_name*, *tar_file*, *bt2_1_file*, *bt2_2_file*, *bt2_3_file*, *bt2_4_file*, *bt2_rev1_file*, *bt2_rev2_file*)
Extracts the BWA index files from the genome index tar file.

Parameters

- **genome_file_name** (*str*) – Location string of the genome fasta file
- **tar_file** (*str*) – Location of the Bowtie2 index file

- **bt2_1_file** (*str*) – Location of the amb index file
- **bt2_2_file** (*str*) – Location of the ann index file
- **bt2_3_file** (*str*) – Location of the bwt index file
- **bt2_4_file** (*str*) – Location of the pac index file
- **bt2_rev1_file** (*str*) – Location of the sa index file
- **bt2_rev2_file** (*str*) – Location of the sa index file

Returns Boolean indicating if the task was successful

Return type bool

static bowtie_index_genome(*genome_file*)

Create an index of the genome FASTA file with Bowtie2. These are saved alongside the assembly file.

Parameters **genome_file** (*str*) – Location of the assembly file in the file system

bwa_aln_align_reads_paired(*genome_file*, *reads_file_1*, *reads_file_2*, *bam_loc*, *params*)

Map the reads to the genome using BWA.

Parameters

- **genome_file** (*str*) – Location of the assembly file in the file system
- **reads_file** (*str*) – Location of the reads file in the file system
- **bam_loc** (*str*) – Location of the output file

bwa_aln_align_reads_single(*genome_file*, *reads_file*, *bam_loc*, *params*)

Map the reads to the genome using BWA. :param genome_file: Location of the assembly file in the file system :type genome_file: str :param reads_file: Location of the reads file in the file system :type reads_file: str :param bam_loc: Location of the output file :type bam_loc: str

static bwa_index_genome(*genome_file*)

Create an index of the genome FASTA file with BWA. These are saved alongside the assembly file. If the index has already been generated then the locations of the files are returned

Parameters **genome_file** (*str*) – Location of the assembly file in the file system

Returns

- **amb_file** (*str*) – Location of the amb file
- **ann_file** (*str*) – Location of the ann file
- **bwt_file** (*str*) – Location of the bwt file
- **pac_file** (*str*) – Location of the pac file
- **sa_file** (*str*) – Location of the sa file

Example

```
from tool.aligner_utils import alignerUtils
au_handle = alignerUtils()

indexes = au_handle.bwa_index_genome('/<data_dir>/human_GRCh38.fa.gz')
print(indexes)
```

static bwa_mem_align_reads(*genome_file*, *bam_loc*, *params*, *reads_file_1*, *reads_file_2=None*)
Map the reads to the genome using BWA.

Parameters

- **genome_file** (*str*) – Location of the assembly file in the file system
- **reads_file** (*str*) – Location of the reads file in the file system
- **bam_loc** (*str*) – Location of the output file

bwa_untar_index(*genome_name*, *tar_file*, *amb_file*, *ann_file*, *bwt_file*, *pac_file*, *sa_file*)
Extracts the BWA index files from the genome index tar file.

Parameters

- **genome_file_name** (*str*) – Location string of the genome fasta file
- **genome_idx** (*str*) – Location of the BWA index file
- **amb_file** (*str*) – Location of the amb index file
- **ann_file** (*str*) – Location of the ann index file
- **bwt_file** (*str*) – Location of the bwt index file
- **pac_file** (*str*) – Location of the pac index file
- **sa_file** (*str*) – Location of the sa index file

Returns Boolean indicating if the task was successful

Return type bool

static gem_index_genome(*genome_file*, *index_name=None*)
Create an index of the genome FASTA file with GEM. These are saved alongside the assembly file.

Parameters **genome_file** (*str*) – Location of the assembly file in the file system

static replaceENAHeader(*file_path*, *file_out*)
The ENA header has pipes in the header as part of the stable_id. This function removes the ENA stable_id and replaces it with the final section after splitting the stable ID on the pipe.

5.3 Bam Utilities

class tool.bam_utils.**bamUtils**
Tool for handling bam files

static bam_copy(*bam_in*, *bam_out*)
Wrapper function to copy from one bam file to another

Parameters

- **bam_in** (*str*) – Location of the input bam file
- **bam_out** (*str*) – Location of the output bam file

static bam_count_reads(*bam_file*, *aligned=False*)
Wrapper to count the number of (aligned) reads in a bam file

static bam_filter(*bam_file*, *bam_file_out*, *filter_name*)
Wrapper for filtering out reads from a bam file

Parameters

- **bam_file** (*str*) –
- **bam_file_out** (*str*) –
- **filter** (*str*) –

  One of: duplicate - Read is PCR or optical duplicate (1024) supplementary - Reads that are chimeric, fusion or non linearly aligned (2048) unmapped - Read is unmapped or not the primary alignment (260)

static bam_index(*bam_file*, *bam_idx_file*)
Wrapper for the pysam SAMtools index function

Parameters

- **bam_file** (*str*) – Location of the bam file that is to be indexed
- **bam_idx_file** (*str*) – Location of the bam index file (.bai)

static bam_list_chromosomes(*bam_file*)
Wrapper to list the chromosome names that are present within the bam file

Parameters **bam_file** (*str*) – Location of the bam file

Returns List of the names of the chromosomes that are present in the bam file

Return type list

static bam_merge(**args*)
Wrapper for the pysam SAMtools merge function

Parameters

- **bam_file_1** (*str*) – Location of the bam file to merge into
- **bam_file_2** (*str*) – Location of the bam file that is to get merged into bam_file_1

static bam_paired_reads(*bam_file*)
Wrapper to test if a bam file contains paired end reads

static bam_sort(*bam_file*)
Wrapper for the pysam SAMtools sort function

Parameters **bam_file** (*str*) – Location of the bam file to sort

static bam_split(*bam_file_in*, *bai_file*, *chromosome*, *bam_file_out*)
Wrapper to extract a single chromosomes worth of reading into a new bam file

Parameters

- **bam_file_in** (*str*) – Location of the input bam file
- **bai_file** (*str*) – Location of the bam index file. This needs to be in the same directory as the bam_file_in
- **chromosome** (*str*) – Name of the chromosome whose alignments are to be extracted
- **bam_file_out** (*str*) – Location of the output bam file

static bam_stats(*bam_file*)
Wrapper for the pysam SAMtools flagstat function

Parameters **bam_file** (*str*) – Location of the bam file

Returns **list** – qc_passed : int qc_failed : int description : str

Return type dict

static bam_to_bed(*bam_file*, *bed_file*)
Function for converting bam files to bed files

static check_header(*bam_file*)
Wrapper for the pysam SAMtools for checking if a bam file is sorted

Parameters **bool** – True if the file has been sorted

static sam_to_bam(*sam_file*, *bam_file*)
Function for converting sam files to bam files

class tool.bam_utils.**bamUtilsTask**
Wrappers so that the function above can be used as part of a @task within COMPSs avoiding the files being copied around the infrastructure too many times

bam_copy(***kwargs*)
Wrapper function to copy from one bam file to another

Parameters

- **bam_in** (*str*) – Location of the input bam file
- **bam_out** (*str*) – Location of the output bam file

bam_filter(***kwargs*)
Wrapper for filtering out reads from a bam file

Parameters

- **bam_file** (*str*) –
- **bam_file_out** (*str*) –
- **filter** (*str*) –

  One of: duplicate - Read is PCR or optical duplicate (1024) unmapped - Read is unmapped or not the primary alignment (260)

bam_index(***kwargs*)
Wrapper for the pysam SAMtools merge function

Parameters

- **bam_file** (*str*) – Location of the bam file that is to be indexed
- **bam_idx_file** (*str*) – Location of the bam index file (.bai)

bam_list_chromosomes(***kwargs*)
Wrapper to get the list of chromosomes in a given bam file

Parameters **bam_file** (*str*) – Location of the bam file

Returns **chromosome_list** – List of the chromosomes in the bam file

Return type list

bam_merge(*in_bam_job_files*)
Wrapper task taking any number of bam files and merging them into a single bam file.

Parameters **bam_job_files** (*list*) – List of the locations of the separate bam files that are to be merged The first file in the list will be taken as the output file name

bam_merge_10(***kwargs*)
Wrapper for the pysam SAMtools merge function

Parameters

- **bam_file_1** (*str*) – Location of the bam file to merge into
- **bam_file_2** (*str*) – Location of the bam file that is to get merged into bam_file_1
- **bam_file_3** (*str*) – Location of the bam file that is to get merged into bam_file_1
- **bam_file_4** (*str*) – Location of the bam file that is to get merged into bam_file_1
- **bam_file_5** (*str*) – Location of the bam file that is to get merged into bam_file_1
- **bam_file_6** (*str*) – Location of the bam file that is to get merged into bam_file_1
- **bam_file_7** (*str*) – Location of the bam file that is to get merged into bam_file_1
- **bam_file_8** (*str*) – Location of the bam file that is to get merged into bam_file_1
- **bam_file_9** (*str*) – Location of the bam file that is to get merged into bam_file_1
- **bam_file_10** (*str*) – Location of the bam file that is to get merged into bam_file_1

bam_merge_2(***kwargs*)
Wrapper for the pysam SAMtools merge function

Parameters

- **bam_file_1** (*str*) – Location of the bam file to merge into
- **bam_file_2** (*str*) – Location of the bam file that is to get merged into bam_file_1

bam_merge_3(***kwargs*)
Wrapper for the pysam SAMtools merge function

Parameters

- **bam_file_1** (*str*) – Location of the bam file to merge into
- **bam_file_2** (*str*) – Location of the bam file that is to get merged into bam_file_1
- **bam_file_3** (*str*) – Location of the bam file that is to get merged into bam_file_1

bam_merge_4(***kwargs*)
Wrapper for the pysam SAMtools merge function

Parameters

- **bam_file_1** (*str*) – Location of the bam file to merge into
- **bam_file_2** (*str*) – Location of the bam file that is to get merged into bam_file_1
- **bam_file_3** (*str*) – Location of the bam file that is to get merged into bam_file_1
- **bam_file_4** (*str*) – Location of the bam file that is to get merged into bam_file_1

bam_merge_5(***kwargs*)
Wrapper for the pysam SAMtools merge function

Parameters

- **bam_file_1** (*str*) – Location of the bam file to merge into
- **bam_file_2** (*str*) – Location of the bam file that is to get merged into bam_file_1
- **bam_file_3** (*str*) – Location of the bam file that is to get merged into bam_file_1
- **bam_file_4** (*str*) – Location of the bam file that is to get merged into bam_file_1
- **bam_file_5** (*str*) – Location of the bam file that is to get merged into bam_file_1

bam_paired_reads(***kwargs*)
Wrapper for the pysam SAMtools view function to identify if a bam file contains paired end reads

Parameters **bam_file** (*str*) – Location of the bam file that is to be indexed

Returns True if the bam file contains paired end reads

Return type bool

bam_sort(***kwargs*)
Wrapper for the pysam SAMtools sort function

Parameters **bam_file** (*str*) – Location of the bam file to sort

bam_stats(***kwargs*)
Wrapper for the pysam SAMtools flagstat function

Parameters

- **bam_file** (*str*) – Location of the bam file that is to be indexed
- **bam_idx_file** (*str*) – Location of the bam index file (.bai)

check_header(***kwargs*)
Wrapper for the pysam SAMtools merge function

Parameters

- **bam_file_1** (*str*) – Location of the bam file to merge into
- **bam_file_2** (*str*) – Location of the bam file that is to get merged into bam_file_1

5.4 FASTQ Functions

The following functions are ones that are used for the manipulation of FASTQ files.

5.4.1 Reading

The following functions are to provide easy access for iterating through entries within a FASTQ file(s) both single and paired.

class tool.fastqreader.**fastqreader**
Module for reading single end and paired end FASTQ files

closeFastQ()
Close file handles for the FastQ files.

closeOutputFiles()
Close the output file handles

createOutputFiles(*tag=''*)
Create and open the file handles for the output files

Parameters **tag** (*str*) – Tag to identify the output files (DEFAULT: '')

eof(*side=1*)
Indicate if the end of the file has been reached

Parameters **side** (*int*) – 1 or 2

incrementOutputFiles()
Increment the counter and create new files for splitting the original FastQ paired end files.

next(*side=1*)
Get the next read element for the specific FastQ file pair

Parameters **side** (*int*) – 1 or 2 to get the element from the relevant end (DEFAULT: 1)

Returns

id [str] Sequence ID

seq [str] Called sequence

add [str] Plus sign

score [str] Base call score

Return type dict

openFastQ(*file1*, *file2=None*)
Create file handles for reading the FastQ files

Parameters

- **file1** (*str*) – Location of the first FASTQ file
- **file2** (*str*) – Location of a paired end FASTQ file.

writeOutput(*read*, *side=1*)
Writer to print the extracted lines

Parameters

- **read** (*dict*) – Read is the dictionary object returned from self.next()
- **side** (*int*) – The side that the read has coe from (DEFAULT: 1)

Returns False if a value other than 1 or 2 is entered for the side.

Return type bool

5.4.2 Splitting

This tool has been created to aid in splitting FASTQ files into manageable chunks for parallel processing. It is able to work on single and paired end files.

class tool.fastq_splitter.**fastq_splitter**(*configuration=None*)
Script for splitting up FASTQ files into manageable chunks

paired_splitter(***kwargs*)
Function to divide the paired-end FastQ files into separte sub files of 1000000 sequences so that the aligner can run in parallel.

Parameters

- **in_file1** (*str*) – Location of first paired end FASTQ file
- **in_file2** (*str*) – Location of second paired end FASTQ file
- **tag** (*str*) – DEFAULT = tmp Tag used to identify the files. Useful if this is getting run manually on a single machine multiple times to prevent collisions of file names

Returns

- **Returns** (*Returns a list of lists of the files that have been generated.*) – Each sub list containing the two paired end files for that subset.
- **paired_files** (*list*) – List of lists of pair end files. Each sub list containing the two paired end files for that subset.

run (*input_files*, *input_metadata*, *output_files*)
The main function to run the splitting of FASTQ files (single or paired) so that they can aligned in a distributed manner

Parameters

- **input_files** (*dict*) – List of input fastq file locations
- **metadata** (*dict*) –
- **output_files** (*dict*) –

Returns

- **output_file** (*str*) – Location of compressed (.tar.gz) of the split FASTQ files
- **output_names** (*list*) – List of file names in the compressed file

single_splitter (***kwargs*)
Function to divide the FastQ files into separate sub files of 1000000 sequences so that the aligner can run in parallel.

Parameters

- **in_file1** (*str*) – Location of first FASTQ file
- **tag** (*str*) – DEFAULT = tmp Tag used to identify the files. Useful if this is getting run manually on a single machine multiple times to prevent collisions of file names

Returns

- **Returns** (*Returns a list of the files that have been generated.*) – Each sub list containing the two paired end files for that subset.
- **paired_files** (*list*) – List of lists of pair end files. Each sub list containing the two paired end files for that subset.

5.4.3 Entry Functions

The following functions allow for manipulating FASTQ files.

class tool.fastq_utils.**fastqUtils**
Set of methods to help with the management of FastQ files.

static fastq_match_paired_ends (*fastq_1*, *fastq_2*)
Take 2 fastq files and remove ends that don't have a matching pair. Requires that the FastQ files are ordered correctly.

Mismatches can occur if there is a filtering step that removes one of the paired ends

Parameters

- **fastq_1** (*str*) – Location of FastQ file
- **fastq_2** (*str*) – Location of FastQ file

static fastq_randomise (*fastq*, *output=None*)
Randomising the order of reads withim a FastQ file

Parameters

- **fastq** (*str*) – Location of the FastQ file to randomise
- **output** (*str*) – [OPTIONAL] Location of the output FastQ file. If left blank then the randomised output is saved to the same location as *fastq*

static fastq_sort_file(*fastq*, *output=None*)
Sorting of a FastQ file

Parameters

- **fastq** (*str*) – Location of the FastQ file to sort
- **output** (*str*) – [OPTIONAL] Location of the output FastQ file. If left blank then the sorted output is saved to the same location as *fastq*

CHAPTER 6

Continuous Integration with Travis

This document summarizes the steps involved in setting up a continuous integration suite with Travis for our pipelines.

6.1 Getting Started

Login with your gitHub account details on travis.ci. Add the "Multiscale Genomics" to your organization to gain access to its repositories. (you would have to send a request to be added to this organization).

Follow the onscreen instructions on travis.ci.

```
Flick the repository switch to "on".
Add .travis.yml to your repository.
Sync your travis account to your GitHub account.
Pushing to Git will trigger the first Travis build after adding the above file.
```

6.2 Making .travis.yml File

To your added .travis.yml file in your GitHub repository include:

- "python" in "language". With version/s specified in "python: "
- All packages required for running the pipelines in "addons: apt: packages: ". (For more information on the packages please see Full Installation from : http://multiscale-genomics.readthedocs.io/projects/mg-process-fastq/en/latest/full_installation.html#setup-the-system-environment)
- Docker in "services" to tell Travis it needs to have docker installed.
- **services:**
  - docker

- All tools to be installed within "install:" section (For more information on the packages please see Full Installation from : http://multiscale-genomics.readthedocs.io/projects/mg-process-fastq/en/latest/full_installation.html#setup-the-system-environment)

Note: libtbb did not seem to be installing correctly when put in "apt: packages: ". It has therefore been done with sudo in "install:"

- Setup all symlinks in "before_script: "
- Change execution permissions on shims folder and harness.sh
- Add the shims folder path to your $PATH
- Include harness.sh to your "script" to be executed

6.3 Making harness.sh File

Include all test scripts to be tested with pytest to your harness.sh file. As iNPS does not work with Python < 3, a conditional check is present to ensure that it does not run unless Travis is running Python 3

6.4 Running Docker container

To run docker within Travis, it has been included within the "services" in the .yml file. Every time travis runs docker it will pull the latest publicly available image from DockerHub and run it. This gets done from within the shims files. The pre-built container can also be pulled from : https://hub.docker.com/r/multiscalegenomics/mgprocessfastq/

using command :

```
docker pull multiscalegenomics/mgprocessfastq:testdocker
```

For more details on the docker container, please refer to : https://github.com/Multiscale-Genomics/mg-process-fastq/blob/master/docs/docker.rst

6.5 Setting up Shims

Libmaus2 and Biobambam2 have had to be installed within a docker container, as they were causing Travis to time out. As the container is non-interactable while on Travis and is not hosting live server. An intermediate layer to access the contents within docker from travis has been introduced in the form of shims. To make these files, take the list of all biobambam2 modules and construct bash script files with the following command for each of the modules :

```
exec docker run -it  multiscalegenomics/mgprocessfastq:biobambamimage ~/lib/
↪biobambam2/bin/biobambam_module_name $@
```

The .travis.yml file used for testing the mg-process-fastq pipelines can be found at : https://github.com/Multiscale-Genomics/mg-process-fastq/blob/master/.travis.yml

CHAPTER 7

Setting up and using a Docker Container

7.1 Our reason for using a container

While working with Travis, the installation of libmaus2 and biobambam2 took up more than 45 minutes (accumulative with the rest of the installations), which caused Travis to time out. We therefore resorted to putting both tools in a container and accessing the commands from there. This document summarizes the steps involved in making a docker image for the above two tools and running a container from that image. As well as uploading your image to docker hub to make it publicly accessible.

This document has been prepared keeping macOS Sierra in mind, although many of the commands are cross platform (*nix) compliant.

7.2 Getting Started

To be able to build a docker container you must have :

a) Docker installed on your machine

b) An account on one of the docker repositories (Docker Hub or Quay). We have used Docker Hub as this was free access.

7.2.1 a) Installing docker to your machine

For this work I had installed a command line based docker, along with the Virtual machine boot2docker. There is however a GUI distribution available for MAC as well. You may install boot2docker using :

```
brew install boot2docker
```

7.2.2 b) Setting up account on Docker Hub

Go to https://hub.docker.com and setup an account with Docker Hub. Add Multiscale Genomics to your organizations and create a repository : mgprocessfastq. You will be uploading your docker images to this repository later.

7.3 Constructing a docker container

Run the following preliminary commands to get your boot2docker running:

```
boot2docker up
eval "$(boot2docker shellinit)"
```

You would also need to have :

- docker-machine
- Virtual Box

installed on your Mac. After these, execute the following commands:

```
docker-machine create -d virtualbox dev
eval $(docker-machine env dev)
```

To ensure your docker is running:

```
docker
```

7.3.1 Making the Dockerfile for libmaus2 and biobambam2

You may want to create a new folder for this purpose, as the docker command compiles the Dockerfile with the given path to the folder. Create a new file with the name of "Dockerfile". Include the following lines within this file:

```
FROM ubuntu:14.04

RUN apt-get update && \
      apt-get -y install sudo

RUN  sudo apt-get install -y make build-essential libssl-dev zlib1g-dev \
libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm libncurses5-dev \
libncursesw5-dev xz-utils tk-dev unzip mcl libgtk2.0-dev r-base-core     \
libcurl4-gnutls-dev python-rpy2 git

RUN mkdir Mug  \
 && cd Mug  \
 && apt-get -y install git \
 && git config --global user.name "your_username"    \
 && git config --global user.email "your_emailId"    \
 && pwd      \
 && mkdir bin lib code       \
 && cd lib   \
 && git clone https://github.com/gt1/libmaus2.git
 && cd libmaus2  \
 && sudo apt-get -y install libtool m4 automake \
 && libtoolize \
 && aclocal  \
```

(continues on next page)

(continued from previous page)

```
&& autoheader       \
&& automake --force-missing --add-missing   \
&& autoconf \
&& ./configure --prefix=/Mug/lib/libmaus2   \

&& make  \
&& make install \
&& cd /Mug/lib      \


&& git clone https://github.com/gt1/biobambam2.git  && cd biobambam2        \
&& autoreconf -i -f \
&& ./configure --with-libmaus2=/Mug/lib/libmaus2 --prefix=/Mug/lib/biobambam2       \
&& make install
```

7.3.2 Making the docker image

Build a docker image from this file using:

```
cd /path/to/your/dockerfile
docker build -t multiscalegenomics/mgprocessfastq/biobambamimage.
```

Login with your docker hub account details :

```
docker login
```

Push the above image to your docker hub repository

```
docker push multiscalegenomics/mgprocessfastq:biobambamimage
```

7.3.3 Running a docker container

You should be able to run the above image locally on your machine as well as pulling it elsewhere (on a system which has docker):

```
docker pull multiscalegenomics/mgprocessfastq:biobambamimage
```

and then running a container via :

```
docker run --name name_you_want multiscalegenomics/mgprocessfastq:biobambamimage
```

Our Travis build pulls the image from our mgprocessfastq repository from within the shims files, and runs the containers using the commands within.

CHAPTER 8

Architectural Design Record

8.1 2017-08-15 - Implementation of pigz

It was highlighted that the archival step using the python module tarfile was taking a long time when archiving and compressing large volumes of data. This is an issue in the WGBS pipeline where the FASTQ files are broken down into small sections and are then aligned individually.

The issue was in relation to the compression step. tar, and therefore tarfile, runs as a single process. In linear time to compress a 14GB file takes 18 minutes, which is a long time for the user that have to wait. The archival step is less than 1 min. As a result the compression has moved to using pigz after the archival step to perform the compression in a parallel fashion across all available cores on a machine. On a 4 core machine this allowed the compression of the same file to take only 7 min.

The resultant compressed file remains accessible via gzip. After decompression files have the same structure, so this should be an invisible change.

8.2 2018-01-26 - Disable no-self-use for @tasks

The disabling of pylint test for determining if a function should be a static method was made as this affected the behaviour of pycompss preventing it from functioning correctly. As a result all @task functions do not require the no-self-use test to be run.

8.3 2018-02-28 - BAM Merge Strategy

Based on benchmarking of the pipeline the procedure for merging of bam files has been modified to get an optimal balance between running as much as possible in parallel vs the cost of spinning up new jobs to perform the merging. It was found that running each job merging 10 files provided the break even point between the cost of creating a new job and getting the maximum throughput through the system. It also reduces the number of iterative merge procedures which is beneficial when there are alignments that are difficult to merge.

8.4 2018-04-26 - BAM Splitting

Added the required functions for tools to be able to split a bam by the number of chromosomes so that the analysis can be done in parallel. As an initial run this has been implemented in the MACS2 tool, where the indexing of the bam file is performed by the tool. The bam and bai files are passed to the jobs so that they can then extract the chromosome that they are required to analyse.

In the future the creation of the bai file could be done at alignment time, but in the case of MACS2 there is a filtering step on the aligned bam file, so a new index would be required.

8.5 2018-05-01 - Compression of FASTQ

Added compression of the split FASTQ files to reduce the amount of space required when processing the data. There is also the removal of the tmp directory after the completion of the splitter to save on space.

8.6 2018-05-09 - Handling aligner index decompression

The code has been modified so that there is a single decompression of the BWA and Bowtie2 common indexes. The index files are then explicitly handed to the alignment task rather than handing over the compressed index. The decompression is performed as a @task so that the index files are already in the COMPSs system. This means that handing the index files to the alignment tasks creates a single symlink in the sandbox temporary file directory rather than duplicating the whole of the index structure for each job.

8.7 2018-05-22 - GEM Naming

Update so that the gem files are name <genome-file>.gem.gz inline with requests from WP7 partners so that the name of the index is picked up correctly

8.8 2018-05-22 - TrimGalore

To try and improve the quality of the reads that are used for numerous pipelines, TrimGalore has been included as a pipeline to aid in the clipping and removal of low quality regions of reads. The pipeline can be run on single or paired end FASTQ files. A report of the trimmed data is also returned for the user to identify what changes were made.

8.9 2018-05-31 - Public genomes and indexes

The VRE is making it possible to use centrally stored genomic sequences and the pre-built indexes. This relies on a second key-value pair for the genome and indexes. Pipelines that interact with the public files need to be able to test for the present of “genome_public” and “index_public”, if they are not present then they should be able to safely fall back on “genome” and “index” keys-value pairs. With the separation of the tools and the pipelines this means that the changes should only need to happen in the pipelines as this can perform the translation between the “genome_public” to “genome” ready for the tool.

8.10 2018-06-01 - Separated WGBS Code Testing

To bring down the run time for the TravisCI, the WGBS has been moved to a separate track. This has the benefit of getting the testing started earlier and allowing the other tests to finish sooner.

8.11 2018-06-01 - Travis Caching

Travis CI is now able to cache the lib directory so that the build phase is reduced to improve test speeds. If there are changes to the lib then these need to be refreshed in the TravisCI settings to ensure that the new libraries are included, or flushed if there are changes to the versions of packages in the lib.

There is also caching of the pip directory to reduce the load time.

8.12 2018-06-04 - Split the WGBS test scripts

Split the testing of the WGBS pipeline and tool chains so that they 2 sets can run in parallel. Both take too long when run in series.

8.13 2018-06-05 - Use of the logger PROGRESS

Added in the use of the logger.progress to indicate the progression of a process.

8.14 2018-06-14 - Paired end alignment

The aligner pipelines has been modified the pass through all the input and metadata to the aligner tools, this simplifies the the passing of a second fastq file and also make using these pipelines for alignment of paired end data possible.

8.15 2018-06-18 - Branch tidying during alignment

Modified the way that the alignment pipelines manage the temporary files. These are now deleted once the pipeline has finished using them. The purpose of this is to save space on the file system and prevent large jobs taking up too much space.

There have also been changes to the handling of paired end files for the alignment pipelines improving the clarity of what is happening and simplifying the passing of parameters. There are also changes to the tests to allow for the removal of temporary files and there are tests to make sure that the output bam files are single or paired end.

Other changes include: - Simplification of the untarring functions - Modifications to the Bowtie2 index file for consistency with the BWA index file - Refactored the BWA ALN sai file generation to reduce redundancy to allow for multi-processing when there is paired-end data - Improved the handling of the suffixes for FASTQ and FASTA files so that it can handle variants

8.16 2018-06-27 - Remove reads marked as duplicate by BioBamBam

BioBamBam only marks reads as duplicate, but does not remove the after. The Tool has been updated to remove the flagged duplicates using samtools with the parameter *-F 1024*. This matches the pipeline used within the Blueprints project.

Also performed some tidying of the code to annotate issues that had been highlighted by pylint.

8.17 2018-07-11 - Changes FASTQ splitter file management

The previous splitter would split the FASTQ files into separate changes, then create the tar file and then the Gzip file. This results in a large amount of wasted tmp space, which is a limited resource. The changes implemented incrementally add the sub-FASTQ files to the archive file, deleting them once they have been added. The whole archive file is then compressed. This has a large advantage when handling larger human datasets.

There has also been some refactoring of the handling of the archiving and compression steps to reduce the duplication of code within the repository.

8.18 2018-07-16 - Modified handling of file locations

Updated the handling of file locations to use os.path.join and os.path.split to allow for compatibility between different operating systems for the pipelines and tools.

8.19 2018-08-02 - Added in Paired End BAM file handling for MACS2

MACS2 is able to automatically handle the files that are handed to it except for paired-end BAM and BED files (BAMPE and BEDPE respectively). The MACS2 tool only accepts BAM files so a check was implemented to determine if the BAM file contained paired-end reads.

There has also been a major rewrite of the MACS2 tool to remove code duplication.

8.20 2018-07-16 - Modified handling of file locations

Updated the handling of file locations to use os.path.join and os.path.split to allow for compatibility between different operating systems for the pipelines and tools.

8.21 2018-08-07 - Storing tool parameters as part of the metadata

To improve the amount of information that is stored about the run of a tool, the parameters that were used are now being included as part of the metadata.

8.22 2018-08-07 - Extra output files from MACS2

MACS2 is able to generate a plot of the results as well as a bedGraph. These have now been integrated as part of the output files from teh tool.

8.23 2018-08-13 - Normalised the use of OSError

IOError was depricated in favour of OSError when moving to py3, but to maintian backwards compatibility IOError also needs to be supported. There were places in the code where this was not true and other places that relied on just OSError. Instances of just IOError have been converted to testing for both IOError and OSError and visa versa.

8.24 2018-08-15 - Use the config.json execution path

Using the directory of the input file for building the location of the working directory with outside of a task is not a viable option as it can write data to the wrong directory. The execution path provided in the config.json arguments is the right place. This location is also the location for output files. This issue occurred as the FASTQ splitter was generating a tar file that the aligners were downloading to the wrong location. Even though this was tidied up this was still not the right place to put this file.

8.25 2018-08-16 - Prevent further duplicate filtering by MACS2

In the process_chipseq.py pipeline the duplicates have already been filtered by BioBamBam2 and samtools so there is no need for further filtering to be done by MACS2.

8.26 2018-09-04 - Adding functionality to bam_utils and MACS2

MACS2 was previously set to work with the BAMPE option for the -f/–format parameter. Additional functionality has been added to bam_utils and macs2 mode to incorporate the BEDPE option. This has been done for the Atac Seq pipeline to incorporate the processing of bed file rather than bam files if the user would need changes to the result files generated.

8.27 2018-09-17 - Updates to tool and pipline run()

Changes to the pipelines so that the run() function matches the definitions within the Tool API. There have also been a number of changes so that the pipeline and tool code is python 3 compatible

8.28 2018-08-22 - Improvement of tadbit tools wrappers

A json with the matrix was included in the outputs of tadbit bin New normalization method OneD in tadbit normalize Code update to use last features of the development branch of tadbit tools api The wrapper of tadbit model was rebuilt to allow the modelling of full genomes, mainly for yeast General reshape of all the code according to pylint Inclusion of tests for the wrappers and tools of the tadbit pipelines

8.29 2018-09-25 - Converting the Kallisto TSV file to BED

To display the scores on the genome browser the abundance tsv is used to generate a bed file where the score matches the transcripts per million column from the abundance.tsv output from Kallisto. This module requires the presence of the ensembl gff3 file for the matching assembly. This should be passed by the VRE when passing the FASTA file for the transcripts.

8.30 2018-10-18 - Multi File handling for the DamID-seq Pipeline

Ability to handle multiple input single/paired end data and background data files and process them in an orderly fashion. Ability to handle the resultant multiples of generated bam files in the idear tool and its matching individual pipeline.

8.31 2018-10-25 - WGBS Pipeline Create BigWig files as standard

The output wig files from BS Seeker2 are now converted to BigWig files by default rather than returning wig files. This is so that they are easier to visualise on the JBrowse interface.

8.32 2018-10-31 - Modify the file names for docker script

There were inconsistencies in the names, these have been updated to be more consistent with the consortium naming instead.

8.33 2018-11-08 - Modifications for the movement of files

To keep the memory usage low the files are now moved in N byte chunks rather than as a whole file.

8.34 2018-11-16 - Reading of Gzipped FASTQ files

Due to the limitations of space it is necessary for the users to be able to load only the gzipped versions of the FASTQ files. Added the ability to read the gzipped FASTQ files to the FASTQ reader. This is behind the aligners and the splitting procedure so has a wide ranging benefit for pipelines.

CHAPTER 9

License

Apache License Version 2.0, January 2004 http://www.apache.org/licenses/

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

   "Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

> To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "{}" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

> http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

CHAPTER 10

Testing

To ensure that the code written to process the data within the MuG VRE works it is essential to have a testing framework to ensure that as the code evolves it does not break the funcitonality of the pipelines and tools. There are 2 key parts to this:

1. Sample data
2. Runnable scripts

For each tool within the mg-process-fastq repository there is an initial set of sample data and matching tests for each tool and pipeline.

10.1 Sample Data

10.1.1 Test Data for ChIP-seq pipeline

The following document is for the preparation of data set required for testing the ChIP-seq pipeline. The document has been written with macOS Sierra in mind, although many of the commands are cross platform (*nix) compliant.

You would need to have the tools listed in "Prerequisites" installed on your system. For more details on installing the tools for this pipeline please refer to

http://multiscale-genomics.readthedocs.io/projects/mg-process-fastq/en/latest/full_installation.html

If you already have certain packages installed feel free to skip over certain steps. Likewise the bin, lib and code directories are relative to the home dir; if this is not the case for your system then make the required changes when running these commands.

Prerequisites

- BWA
- MACS 2
- Biobambam

- Samtools

Data set for genome file

Filtering for required coverage

Download the genome file from

```
wget "http://www.ebi.ac.uk/ena/data/view/CM000663.2,CM000664.2,CM000665.2,CM000666.2,
↪CM000667.2,CM000668.2,CM000669.2,CM000670.2,CM000671.2,CM000672.2,CM000673.2,
↪CM000674.2,CM000675.2,CM000676.2,CM000677.2,CM000678.2,CM000679.2,CM000680.2,
↪CM000681.2,CM000682.2,CM000683.2,CM000684.2,CM000685.2,CM000686.2,J01415.2&
↪display=fasta&download=fasta&filename=entry.fasta" -O GCA_000001405.22.fasta
```

Checkout https://github.com/Multiscale-Genomics/mg-misc-scripts/blob/master/ChIPSeq_Scripts/extract_chromosomeForChIP.py and extract chromosome 22 from the above file using the following command.

```
python extract_chromosomeForChIP.py path/to/your/input/file path/to/output/file
```

Download the fastq file from

```
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/DRR000/DRR000150/DRR000150.fastq.gz
```

Unzip this file.

```
gunzip DRR000150.fastq.gz
```

Index the fasta file

```
bwa index GCA_000001405.22.chr22.fa.fasta
```

Align the fastq file

```
bwa aln GCA_000001405.22.chr22.fa.fasta DRR000150.chr22.fastq >GCA_000001405.22.chr22.
↪sai
```

And make the sam file

```
bwa samse GCA_000001405.22.chr22.fa.fasta GCA_000001405.22.chr22.sai DRR000150.chr22.
↪fastq >GCA_000001405.22.chr22.sam
```

Sort the sam file

```
samtools sort GCA_000001405.22.chr22.sam >GCA_000001405.22.chr22.sorted.sam
```

Find the depths of coverage from the sorted file

```
samtools depth GCA_000001405.22.chr22.sorted.sam >GCA_000001405.22.chr22.dp
```

From the depth file, find regions with >= 70 depth, spanning over >=55 base pairs. You may get the script for this from https://github.com/Multiscale-Genomics/mg-misc-scripts/blob/master/ChIPSeq_Scripts/traverseForCoverageRegion_ChIP.py. Run it using:

```
python traverseForCoverageRegion_ChIP.py path/to/GCA_000001405.22.chr22.dp
```

Running this script would print the spanning regions. If it is a continuous region, you may only take the first starting base pair and the last ending base pair, as inputs for the next step. (Take out 1000 and add in 1000 to these respectively to get upstream and downstream spanning bases)

Extract the corresponding fasta sequence from the chromosome file for the positions retrieved from the above step. Checkout file from https://github.com/Multiscale-Genomics/mg-misc-scripts/blob/master/ChIPSeq_Scripts/extractChromosomalRegion.py and run using command:

```
python extractChromosomalRegion.py path/to/original/fasta/file path/to/output/file/
↪for/region/macs2.Human.GCA_000001405.22.fasta starting_base_position ending_base_
↪position
```

Making the Fastq file

Index the fasta file for the selected region

```
bwa index macs2.Human.GCA_000001405.22.fasta
```

Align the fastq file

```
bwa aln macs2.Human.GCA_000001405.22.fasta DRR000150.chr22.fastq >macs2.Human.GCA_
↪000001405.22.sai
```

And make the sam file

```
bwa samse macs2.Human.GCA_000001405.22.fasta macs2.Human.GCA_000001405.22.sai␣
↪DRR000150.chr22.fastq >macs2.Human.GCA_000001405.22.sam
```

Filter this sam file for the reads which aligned with chromosome 22 using the following command:

```
awk '$3 == chr22' macs2.Human.GCA_000001405.22.sam >macs2.Human.GCA_000001405.22.22.
↪sam
```

From the filtered reads from the above output file, extract the corresponding entries in fastq file. You may do this using the file at :

```
https://github.com/Multiscale-Genomics/mg-misc-scripts/blob/master/ChIPSeq_Scripts/
↪makeFastQFiles.py
```

and running it via command line:

```
python makeFastQFiles.py --samfile macs2.Human.GCA_000001405.22.22.sam --fastQfile /
↪path/to/DRR000150.chr22.fastq --pathToOutput /path/to/save/output/fastq/file/to/ --
↪fastqOut macs2.Human.DRR000150.22.fastq
```

The fastq file in the above step and fasta file macs2.Human.GCA_000001405.22.fasta together make up the data set for ChIP-seq pipeline

10.1.2 iDamID-Seq Test Data

Test Data

Dataset

Stable ID	SRR3714775
Citation	Franks et al 2016

Genome

Assembly	GRCh38
Chromosome	22
Start	15000000
End	19999999

Method

The full dataset was downloaded from ENA aligned to the genome using GEM.

```
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR371/005/SRR3714775/SRR3714775.fastq.gz
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR371/005/SRR3714775/SRR3714776.fastq.gz
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR371/005/SRR3714775/SRR3714777.fastq.gz
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR371/005/SRR3714775/SRR3714778.fastq.gz

wget "http://www.ebi.ac.uk/ena/data/view/CM000684.2&display=fasta&download=fasta&
↪filename=entry.fasta" -O GCA_000001405.22.chr22.fasta

bwa index GCA_000001405.22.chr22.fasta

bwa aln -q 5 -f SRR3714775.fastq.sai
bwa aln -q 5 -f SRR3714776.fastq.sai
bwa aln -q 5 -f SRR3714777.fastq.sai
bwa aln -q 5 -f SRR3714778.fastq.sai

bwa samse -f SRR3714775.fastq.sam GCA_000001405.22.chr22.fasta SRR3714775.fastq.sai␣
↪SRR3714775.fastq
bwa samse -f SRR3714776.fastq.sam GCA_000001405.22.chr22.fasta SRR3714776.fastq.sai␣
↪SRR3714776.fastq
bwa samse -f SRR3714777.fastq.sam GCA_000001405.22.chr22.fasta SRR3714777.fastq.sai␣
↪SRR3714777.fastq
bwa samse -f SRR3714778.fastq.sam GCA_000001405.22.chr22.fasta SRR3714778.fastq.sai␣
↪SRR3714778.fastq

samtools view -h -o SRR3714775.sam SRR3714775.bam
samtools view -h -o SRR3714776.sam SRR3714776.bam
samtools view -h -o SRR3714777.sam SRR3714777.bam
samtools view -h -o SRR3714778.sam SRR3714778.bam

awk 'BEGIN { FS = "[[:space:]]+" } ; $3 ~ /CM000684/ {print $1}' SRR3714775.sam >␣
↪SRR3714775.chr22.sam
awk 'BEGIN { FS = "[[:space:]]+" } ; $3 ~ /CM000684/ {print $1}' SRR3714776.sam >␣
↪SRR3714776.chr22.sam
```

(continues on next page)

(continued from previous page)

```
awk 'BEGIN { FS = "[[:space:]]+" } ; $3 ~ /CM000684/ {print $1}' SRR3714777.sam > SRR3714777.chr22.sam
awk 'BEGIN { FS = "[[:space:]]+" } ; $3 ~ /CM000684/ {print $1}' SRR3714778.sam > SRR3714778.chr22.sam

python scripts/ExtractRowsFromFASTQs.py --input_1 tests/data/SRR3714775.fastq --rows tests/data/SRR3714775.chr22.sam --output_tag profile
python scripts/ExtractRowsFromFASTQs.py --input_1 tests/data/SRR3714776.fastq --rows tests/data/SRR3714776.chr22.sam --output_tag profile
python scripts/ExtractRowsFromFASTQs.py --input_1 tests/data/SRR3714777.fastq --rows tests/data/SRR3714777.chr22.sam --output_tag profile
python scripts/ExtractRowsFromFASTQs.py --input_1 tests/data/SRR3714778.fastq --rows tests/data/SRR3714778.chr22.sam --output_tag profile
```

The alignments were then filtered with BioBamBam and peak calling was performed with iDEAR and a suitable region with a number of peaks was identified. The chromosomal region was extract and the matching reads to this region were identified. To reduce the number of reads that matched so that it could be used as a test set for the code base every other read was selected so that a reasonable number of reads we present. This mean that there are results when running the test, but generating the alignments does not take too long to compute.

```
sed -n 539916,556583p GCA_000001405.chr22.fasta > GCA_000001405.chr22ss.fasta
sed -n 5002,11669p idear.Human.GCA_000001405.22.fasta >> idear.Human.GCA_000001405.22.subset.fasta
```

Test Scripts

The following are the tests for checking that the tools in the Hi-C pipeline are functioning correctly.

The tests should be run in this order so that the required input files are generated at the correct stage.

```
pytest -m idamidseq tests/test_bwa_indexer.py
pytest -m idamidseq tests/test_bwa_aligner.py
pytest -m idamidseq tests/test_biobambam.py
pytest -m idamidseq tests/test_bsgenome.py
pytest -m idamidseq tests/test_idear.py
```

These can be called as part of a single tool chain with:

```
python tests/test_toolchains.py --pipeline idamidseq
```

10.1.3 Test Data for MNase-seq pipeline

The following document is for the preparation of data set required for testing the MNase-seq pipeline. The document has been written with macOS Sierra in mind, although many of the commands are cross platform (*nix) compliant.

You would need to have the tools listed in "Prerequisites" installed on your system. For more details on installing the tools for this pipeline please refer to

http://multiscale-genomics.readthedocs.io/projects/mg-process-fastq/en/latest/full_installation.html

If you already have certain packages installed feel free to skip over certain steps. Likewise the bin, lib and code directories are relative to the home dir; if this is not the case for your system then make the required changes when running these commands.

Prerequisites

- BWA
- Samtools
- iNPS

Note: iNPS will only run within a Python 3 or above environment

Data set for genome file

Filtering for required coverage

Download the fasta file for Mouse chromosome 19 from

```
wget "http://www.ebi.ac.uk/ena/data/view/CM001012&display=fasta&download=fasta&
↪filename=CM001012.fasta" -O Mouse.CM001012.2.fasta
```

Download the fastq file from

```
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/DRR000/DRR000386/DRR000386.fastq.gz
```

Unzip this file.

```
gunzip DRR000386.fastq.gz
```

Index the fasta file

```
bwa index Mouse.CM001012.2.fasta
```

Align the fastq file

```
bwa aln Mouse.CM001012.2.fasta DRR000386.fastq >Mouse.CM001012.2.sai
```

And make the sam file

```
bwa samse Mouse.CM001012.2.fasta Mouse.CM001012.2.sai DRR000386.fastq >Mouse.CM001012.
↪2.sam
```

Filter out the aligned reads from the above sam file.

```
awk '$3 != "*"' Mouse.CM001012.2.sam >Mouse.CM001012.2.filtered.sam
```

Sort the sam file

```
samtools sort Mouse.CM001012.2.filtered.sam >Mouse.CM001012.2.sorted.sam
```

Find the depths of coverage from the sorted file

```
samtools depth Mouse.CM001012.2.sorted.sam >Mouse.CM001012.2.dp
```

From the depth file, find regions with >= 70 depth, spanning over >=55 base pairs. You may get the script for this from: https://github.com/Multiscale-Genomics/mg-misc-scripts/blob/master/MNaseSeq_Scripts/traverseForCoverageRegion_MNase.py

Run it using:

```
python traverseForCoverageRegion_MNase.py path/to/Mouse.CM001012.2.dp
```

Running this script would print the spanning regions. Running this script for this data set gives multiple regions. The output is in the format : start - end - depth. The one at the end has a maximal coverage from this data set. Since it is a continuous region, you may take the first starting base pair and the last ending base pair, as inputs for the next step. (Take out 1000 and add in 1000 to these respectively to get upstream and downstream spanning bases)

Extract the corresponding fasta sequence from the chromosome file for the positions retrieved from the above step. Checkout file from https://github.com/Multiscale-Genomics/mg-misc-scripts/blob/master/MNaseSeq_Scripts/extractChromosomalRegion.py and run using command:

```
python extractChromosomalRegion.py path/to/original/fasta/file path/to/output/file/
↪for/region/inps.Mouse.GRCm38.fasta starting_base_position ending_base_position
```

Making the Fastq file

Index the fasta file for the selected region

```
bwa index inps.Mouse.GRCm38.fasta
```

Align the fastq file

```
bwa aln inps.Mouse.GRCm38.fasta DRR000386.fastq >inps.Mouse.GRCm38.sai
```

And make the sam file

```
bwa samse inps.Mouse.GRCm38.fasta inps.Mouse.GRCm38.sai DRR000386.fastq >inps.Mouse.
↪GRCm38.sam
```

Filter this sam file for the reads which aligned with chromosome 19 using the following command:

```
awk '$3 != "*"' inps.Mouse.GRCm38.sam >inps.Mouse.GRCm38.sam.19.sam
```

From the filtered reads from the above output file, extract the corresponding entries in fastq file. You may do this using the file at https://github.com/Multiscale-Genomics/mg-misc-scripts/blob/master/MNaseSeq_Scripts/makeFastQFiles.py

and running it via command line:

```
python makeFastQFiles.py --samfile path/to/inps.Mouse.GRCm38.sam.19.sam --fastQfile /
↪path/to/DRR000386.fastq --pathToOutput /path/to/save/output/fastq/file/to/ --
↪fastqOut DRR000386.MNaseseq.fastq
```

Shorten this file by running the script at https://github.com/Multiscale-Genomics/mg-misc-scripts/blob/master/MNASeq_Scripts/randomSeqSelector.py

using

```
python randomSeqSelector.py DRR000386.MNaseseq.fastq inps.Mouse.DRR000386.fastq
```

The fastq file in the above step and fasta file inps.Mouse.GRCm38.fasta together make up the data set for MNase-seq pipeline

10.1.4 Test Data for RNA-seq pipeline

The following document is for the preparation of data set required for testing the RNA-seq pipeline. The document has been written with macOS Sierra in mind, although many of the commands are cross platform (*nix) compliant.

You would need to have the tools listed in "Prerequisites" installed on your system. For more details on installing the tools for this pipeline please refer to

http://multiscale-genomics.readthedocs.io/projects/mg-process-fastq/en/latest/full_installation.html

If you already have certain packages installed feel free to skip over certain steps. Likewise the bin, lib and code directories are relative to the home dir; if this is not the case for your system then make the required changes when running these commands.

Prerequisites

- Kallisto
- Samtools

Data set for genome file

Go to Ensemble website >> Human >> Example gene

```
http://www.ensembl.org/Homo_sapiens/Gene/Summary?g=ENSG00000139618;r=13:32315474-
↪32400266
```

Copy the chromosome number and coordinates given in the "location" field. Go to BioMart (top panel), and select Filters from the left panel. Expand Regions and enter the information retrieved above.

Click on "Attributes" in the left panel. Select Gene stable ID, Transcript stable ID from Features. Select cDNA sequences from Sequences radio button.

Click on the Results button above the left panel. Export results to fasta file.

Index this file using Kallisto indexer:

```
kallisto index -i kallisto.Human.GRCh38.fasta.idx /path/to/file/exportSequences.fasta
```

Download the fastq files

```
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR030/ERR030872/ERR030872_1.fastq.gz
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR030/ERR030872/ERR030872_2.fastq.gz
```

Run the Kallisto quantifier using command:

```
kallisto quant -i kallisto.Human.GRCh38.fasta.idx -o out --pseudobam /path/to/
↪ERR030872_1.fastq.gz /path/to/ERR030872_2.fastq.gz  >kallisto.ERR030872.sam
```

Filter the aligned sequence entries from the above sam file:

```
awk '$3 != "*"' kallisto.ERR030872.sam >kallisto.ERR030872.filtered.sam
```

Unzip the fastq files.

```
unzip ERR030872_1.fastq.gz
unzip ERR030872_2.fastq.gz
```

Checkout https://github.com/Multiscale-Genomics/mg-misc-scripts/blob/master/RNASeq_Scripts/makeFastQFiles.py and use the following command to generate the fastq files:

```
python /path/to/makeFastQFiles.py --samfile kallisto.ERR030872.filtered.sam --
↪fastQfile ERR030872_1.fastq --pathToOutput /path/to/make/fastqFile/ --fastqOut␣
↪ERR030872_1.RNAseq.fastq
python /path/to/makeFastQFiles.py --samfile kallisto.ERR030872.filtered.sam --
↪fastQfile ERR030872_2.fastq --pathToOutput /path/to/make/fastqFile/ --fastqOut␣
↪ERR030872_2.RNAseq.fastq
```

Shorten these files by running the script at https://github.com/Multiscale-Genomics/mg-misc-scripts/blob/master/RNASeq_Scripts/randomSeqSelector.py using

```
python PythonScripts/randomSeqSelector.py ERR030872_1.RNAseq.fastq kallisto.Human.
↪ERR030872_1.fastq
python PythonScripts/randomSeqSelector.py ERR030872_2.RNAseq.fastq kallisto.Human.
↪ERR030872_2.fastq
```

Then zip them:

```
gzip kallisto.Human.ERR030872_1.fastq
gzip kallisto.Human.ERR030872_2.fastq
```

10.1.5 WGBS Test Data

Test Data

Dataset

Stable ID	SRR892982
Citation	Sun et al 2014

Genome

Assembly	GRChm8, chr19
Chromosome	19
Start	3000000
End	4020000

Method

The full dataset was downloaded from ENA aligned to the genome using GEM.

```
1 wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR892/SRR892982/SRR892982_1.fastq.gz
2 wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR892/SRR892982/SRR892982_2.fastq.gz
3 gunzip SRR892982_1.fastq.gz
4 gunzip SRR892982_2.fastq.gz
5
6 wget "http://www.ebi.ac.uk/ena/data/view/CM001012&display=fasta&download=fasta&
  ↪filename=entry.fasta" -O mouse.GRCm38.19.fasta
```

(continues on next page)

(continued from previous page)

```

sed -n 40022200,41046060p GCA_000001635.6.fasta > mouse.GRCm38.19.fasta

bowtie2-build mouse.GRCm38.19.fasta mouse.GRCm38.19.idx
bowtie2 -x genome/mouse.GRCm38.19.idx -U data/SRR892982_1.fastq > SRR892982_1.sam
samtools view -h -o SRR892982_1.sam SRR892982_1.bam

awk 'BEGIN { FS = "[[:space:]]+" } ; $3 ~ /CM001012/ {print $1}' SRR892982_1.sam >␣
↪SRR892982_1.chr19.rows

python scripts/ExtractRowsFromFASTQs.py --input_1 tests/data/SRR892982_1.fastq --
↪input_2 tests/data/SRR892982_2.fastq --rows SRR892982_1.chr19.rows --output_tag␣
↪profile

mv data/tmp/SRR892982_1.profile_0.fastq data/SRR892982_1.chr19.fastq
mv data/tmp/SRR892982_2.profile_0.fastq data/SRR892982_2.chr19.fastq

bowtie2 -x genome/mouse.GRCm38.19.idx -1 data/SRR892982_1.chr19.fastq -2 data/
↪SRR892982_2.chr19.fastq > SRR892982_1.chr19.sam
```

There are a range of positive and negative peaks between 3000000 and 4020000. Subselect the genome and matching FASTQ reads for the required subsection:

```
head -n 1 mouse.GRCm38.19.fasta > 3M/mouse.GRCm38.19.3M.fasta
sed -n 50001,67001p mouse.GRCm38.19.fasta >> 3M/mouse.GRCm38.19.3M.fasta

bowtie2-build 3M/mouse.GRCm38.19.3M.fasta 3M/mouse.GRCm38.19.3M.idx
bowtie2 -p 4 -x 3M/mouse.GRCm38.19.3M.idx -1 SRR892982_1.chr19.fastq -2 SRR892982_2.
↪chr19.fastq > 3M/SRR892982.chr19.3M.sam
samtools view -h -o 3M/SRR892982.chr19.3M.sam 3M/SRR892982.chr19.3M.bam

python scripts/ExtractRowsFromFASTQs.py --input_1 SRR892982_1.chr19.fastq --input_2␣
↪SRR892982_2.chr19.fastq --rows 3M/SRR892982.chr19.3M.rows --output_tag subset

mv tmp/SRR892982_1.chr19.subset_0.fastq 3M/SRR892982_1.chr19.3M.fastq
mv tmp/SRR892982_2.chr19.subset_0.fastq 3M/SRR892982_2.chr19.3M.fastq
```

This has enough load to test the system, while also generating results in the output files.

These files are then saved in the tests/data directory as:

```
bsSeeker.Mouse.GRCm38.fasta
bsSeeker.Mouse.SRR892982_1.fastq.gz
bsSeeker.Mouse.SRR892982_2.fastq.gz
```

These files were too large for use within the Travis tst environment, so the number of entries was reduced by taking every other read:

```
sed -n -e '0~9{N;N;N;N;p}' tests/data/bsSeeker.Mouse.SRR892982_1.fastq > tests/data/
↪test_bsSeeker.Mouse.SRR892982_1.fastq
sed -n -e '0~9{N;N;N;N;p}' tests/data/bsSeeker.Mouse.SRR892982_2.fastq > tests/data/
↪test_bsSeeker.Mouse.SRR892982_2.fastq

mv tests/data/test_bsSeeker.Mouse.SRR892982_1.fastq tests/data/bsSeeker.Mouse.
↪SRR892982_1.fastq
mv tests/data/test_bsSeeker.Mouse.SRR892982_2.fastq tests/data/bsSeeker.Mouse.
↪SRR892982_2.fastq
```

(continues on next page)

(continued from previous page)

```

gzip tests/data/bsSeeker.Mouse.SRR892982_1.fastq
gzip tests/data/bsSeeker.Mouse.SRR892982_2.fastq
```

Test Scripts

The following are the tests for checking that the tools in the WGBS pipeline are functioning correctly.

The tests should be run in this order so that the required input files are generated at the correct stage.

```
pytest -m wgbs tests/test_fastqc_validation.py
pytest -m wgbs tests/test_bs_seeker_filter.py
pytest -m wgbs tests/test_bs_seeker_indexer.py
pytest -m wgbs tests/test_bs_seeker_aligner.py
pytest -m wgbs tests/test_bs_seeker_methylation_caller.py
```

These can be called as part of a single tool chain with:

```
python tests/test_toolchains.py --pipeline wgbs
```

10.1.6 Hi-C Test Data

Test Data

Dataset

Stable ID	SRR1658573
Citation	Rao et al 2014

Genome

Assembly	GRCh38
Chromosome	21
Start	15000000
End	19999999

Method

The full dataset was downloaded from ENA aligned to the genome using GEM.

```
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR165/003/SRR1658573/SRR1658573_1.fastq.gz
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR165/003/SRR1658573/SRR1658573_2.fastq.gz

wget "http://www.ebi.ac.uk/ena/data/view/CM000663.2,CM000664.2,CM000665.2,CM000666.2,
↪CM000667.2,CM000668.2,CM000669.2,CM000670.2,CM000671.2,CM000672.2,CM000673.2,
↪CM000674.2,CM000675.2,CM000676.2,CM000677.2,CM000678.2,CM000679.2,CM000680.2,
↪CM000681.2,CM000682.2,CM000683.2,CM000684.2,CM000685.2,CM000686.2,J01415.2&
↪display=fasta&download=fasta&filename=entry.fasta" -O GCA_000001405.22.fasta
```

```
from tool.common import common

genome_file = "GCA_000001405.22.fasta"
new_genome_file = "GCA_000001405.22_gem.fasta"

common_handle = common()
common_handle.replaceENAHeader(genome_file, new_genome_file)

idx_loc = common_handle.gem_index_genome(new_genome_file, new_genome_file)

from pytadbit.mapping.mapper import full_mapping

fastq_file_1 = "SRR1658573_1.fastq"
fastq_file_2 = "SRR1658573_2.fastq"
output_dir = "mapped_reads"
windows = None
enzyme_name = "MboI"

map_files_1 = full_mapping(
    idx_loc, fastq_file, output_dir,
    r_enz=enzyme_name, windows=windows, frag_map=True, nthreads=32,
    clean=True, temp_dir='/tmp/'
)
map_files_2 = full_mapping(
    idx_loc, fastq_file, output_dir,
    r_enz=enzyme_name, windows=windows, frag_map=True, nthreads=32,
    clean=True, temp_dir='/tmp/'
)
```

A list of FASTQ ids that had aligned to the genome were then extracted:

```
grep chr21 mapped_reads/SRR1658573_1_full_1-100.map | tr "\t" "~" | cut -d"~" -f1 -f5
↪| tr ":" "\t" | awk '(NR==1) || (($4>15000000) && ($4<20000000))' | tr "\t" "~" |
↪cut -d "~" -f1 > SRR1658573_1_chr21_1-100.row
grep chr21 mapped_reads/SRR1658573_2_full_1-100.map | tr "\t" "~" | cut -d"~" -f1 -f5
↪| tr ":" "\t" | awk '(NR==1) || (($4>15000000) && ($4<20000000))' | tr "\t" "~" |
↪cut -d "~" -f1 > SRR1658573_2_chr21_1-100.row
```

The IDs were filtered to ensure matching pairs in both files:

```
grep -Fx -f SRR1658573_1_chr21_1-100.row SRR1658573_2_chr21_1-100.row > SRR1658573_
↪chr21.row
```

The *split_paired_fastq.py* was used to divide the original FASTQ files into chunks of 1000000 reads. The *ExtractRowsFromFASTQ.py* script was then used to extract the matching FASTQ pairs from each of the FASTQ files in parallel. All of the individual FASTQ files were then concatenated together to form the final 2 FASTQ test files. The commands for this were:

```
python split_paired_fastq.py --input_1 SRR1658573_1.fastq --input_2 SRR1658573_1.fastq
```

Test Scripts

The following are the tests for checking that the tools in the Hi-C pipeline are functioning correctly.

The tests should be run in this order so that the required input files are generated at the correct stage.

```
pytest tests/test_gem_indexer.py
pytest tests/test_tb_full_mapping.py
pytest tests/test_tb_parse_mapping.py
pytest tests/test_tb_filter.py
pytest tests/test_tb_generate_tads.py
pytest tests/test_tb_save_hdf5_matrix.py
```

These can be called as part of a single tool chain with:

```
python tests/test_toolchains.py --pipeline hic
```

10.2 Pipelines

There is a test for each of the tools. This uses the "process" scripts to run each of the tools. This is to ensure that the pipeline scripts are able to call out to each of the tools and the correct parameters are handed to each one.

10.2.1 ChIP-Seq

To run the pipeline test:

```
pytest tests/test_pipeline_chipseq.py
```

Methods

tests.test_pipeline_chipseq.**test_chipseq_pipeline_00**()
Test case to ensure that the ChIP-seq pipeline code works.

Running the pipeline with the test data from the command line:

```
runcompss                                                          \
   --lang=python                                                   \
   --library_path=${HOME}/bin                                      \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/  \
   --log_level=debug                                               \
   process_chipseq.py                                              \
      --taxon_id 9606                                              \
      --genome /<dataset_dir>/Human.GCA_000001405.22.fasta         \
      --assembly GRCh38                                            \
      --file /<dataset_dir>/DRR000150.22.fastq
```

tests.test_pipeline_chipseq.**test_chipseq_pipeline_01**()
Test case to ensure that the ChIP-seq pipeline code works.

Running the pipeline with the test data from the command line:

```
runcompss                                                          \
   --lang=python                                                   \
   --library_path=${HOME}/bin                                      \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/  \
   --log_level=debug                                               \
   process_chipseq.py                                              \
      --taxon_id 9606                                              \
```

(continues on next page)

(continued from previous page)

```
      --genome /<dataset_dir>/Human.GCA_000001405.22.fasta        \
      --assembly GRCh38                                           \
      --file /<dataset_dir>/DRR000150.22.fastq
```

Sample Data

Test Data for ChIP-seq pipeline

10.2.2 iDamID-Seq

To run the pipeline test:

```
pytest tests/test_pipeline_idamidseq.py
```

Methods

tests.test_pipeline_idamidseq.**test_idamidseq_pipeline_00**()

Test case to ensure that the ChIP-seq pipeline code works.

Running the pipeline with the test data from the command line:

```
runcompss                                                          \
   --lang=python                                                   \
   --library_path=${HOME}/bin                                      \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/  \
   --log_level=debug                                               \
   process_damidseq.py                                              \
      --taxon_id 9606                                              \
      --genome /<dataset_dir>/Human.GCA_000001405.22.fasta         \
      --assembly GRCh38                                            \
      --file /<dataset_dir>/DRR000150.22.fastq
```

tests.test_pipeline_idamidseq.**test_idamidseq_pipeline_01**()

Test case to ensure that the ChIP-seq pipeline code works.

Running the pipeline with the test data from the command line:

```
runcompss                                                          \
   --lang=python                                                   \
   --library_path=${HOME}/bin                                      \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/  \
   --log_level=debug                                               \
   process_damidseq.py                                              \
      --taxon_id 9606                                              \
      --genome /<dataset_dir>/Human.GCA_000001405.22.fasta         \
      --assembly GRCh38                                            \
      --file /<dataset_dir>/DRR000150.22.fastq
```

Sample Data

Test Data for ChIP-seq pipeline

10.2.3 Genome Indexing

To run the pipeline test:

```
pytest tests/test_pipeline_genome.py
```

Methods

tests.test_pipeline_genome.**test_genome_pipeline_00**()
Test case to ensure that the Genome indexing pipeline code works.

Running the pipeline with the test data from the command line:

```
runcompss                                                          \
   --lang=python                                                   \
   --library_path=${HOME}/bin                                      \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/  \
   --log_level=debug                                               \
   process_genome.py                                               \
      --taxon_id 9606                                              \
      --genome /<dataset_dir>/Human.GCA_000001405.22.fasta         \
      --assembly GRCh38                                            \
      --file /<dataset_dir>/DRR000150.22.fastq
```

tests.test_pipeline_genome.**test_genome_pipeline_01**()
Test case to ensure that the Genome indexing pipeline code works.

Running the pipeline with the test data from the command line:

```
runcompss                                                          \
   --lang=python                                                   \
   --library_path=${HOME}/bin                                      \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/  \
   --log_level=debug                                               \
   process_genome.py                                               \
      --taxon_id 9606                                              \
      --genome /<dataset_dir>/Human.GCA_000001405.22.fasta         \
      --assembly GRCh38                                            \
      --file /<dataset_dir>/DRR000150.22.fastq
```

Sample Data

Uses the genome sequences required by all the tools and pipelines

10.2.4 Hi-C

To run the pipeline test:

```
pytest tests/test_pipeline_tb.py
```

Methods

tests.test_pipeline_tb.**test_tb_pipeline**()
Test case to ensure that the Hi-C pipeline code works.

Running the pipeline with the test data from the command line:

```
runcompss                                                                    \
   --lang=python                                                             \
   --library_path=/home/compss/bin                                           \
   --pythonpath=/<pyenv_virtenv_dir>//lib/python2.7/site-packages/           \
   --log_level=debug                                                         \
   process_hic.py                                                            \
      --taxon_id 9606                                                        \
      --genome /<dataset_dir>/tb.Human.GCA_000001405.22_gem.fasta            \
      --assembly GRCh38                                                      \
      --file1 /<dataset_dir>/tb.Human.SRR1658573_1.fastq                     \
      --file2 /<dataset_dir>/tb.Human.SRR1658573_2.fastq                     \
      --genome_gem /<dataset_dir>/tb.Human.GCA_000001405.22_gem.fasta.gem    \
      --taxon_id 9606                                                        \
      --enzyme_name MboI                                                     \
      --resolutions 10000,100000                                             \
      --windows1 1,100                                                       \
      --windows2 1,100                                                       \
      --normalized 1                                                         \
      --tag tb.Human.SRR1658573                                              \
      --window_type frag                                                     \
```

Sample Data

Hi-C Test Data

10.2.5 MNase-Seq

To run the pipeline test:

```
pytest tests/test_pipeline_mnaseseq.py
```

Methods

tests.test_pipeline_mnaseseq.**test_mnaseseq_pipeline**()
Test case to ensure that the MNase-seq pipeline code works.

Running the pipeline with the test data from the command line:

```
runcompss                                                          \
   --lang=python                                                   \
   --library_path=${HOME}/bin                                      \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/  \
   --log_level=debug                                               \
   process_mnaseseq.py                                             \
      --taxon_id 10090                                             \
      --genome /<dataset_dir>/Mouse.GRCm38.fasta                   \
      --assembly GRCm38                                            \
      --file /<dataset_dir>/DRR000386.fastq
```

Sample Data

Test Data for MNase-seq pipeline

10.2.6 RNA-Seq

To run the pipeline test:

```
pytest tests/test_pipeline_rnaseq.py
```

Methods

tests.test_pipeline_rnaseq.**test_rnaseq_pipeline**()

Test case to ensure that the RNA-seq pipeline code works.

Running the pipeline with the test data from the command line:

```
runcompss                                                          \
   --lang=python                                                   \
   --library_path=${HOME}/bin                                      \
   --pythonpath=/<pyenv_virtenv_dir>/lib/python2.7/site-packages/  \
   --log_level=debug                                               \
   process_rnaseq.py                                               \
      --taxon_id 9606                                              \
      --genome /<dataset_dir>/Human.GRCh38.fasta                   \
      --assembly GRCh38                                            \
      --file /<dataset_dir>/ERR030872_1.fastq                      \
      --file2 /<dataset_dir>/ERR030872_2.fastq
```

Sample Data

Test Data for RNA-seq pipeline

10.2.7 Whole Genome Bisulfate Sequencing (WGBS)

To run the pipeline test:

```
pytest tests/test_pipeline_wgbs.py
```

Methods

Sample Data

WGBS Test Data

10.3 Tools

As the data stored is only the raw data, each of the sets of tools has been packaged up into a tool chain to run each of the tools without failing. This has been done with the tests.test_toolchains.py script.

```
python tests/test_toolchains.py --pipeline [genome | chipseq | hic | mnaseseq |␣
↪rnaseq | wgbs]
```

This script automates the running of each of the tools that are required for a given pipeline.

10.3.1 Methods

tests.test_toolchains.**all_toolchain**(*verbose=False*)
Runs the tests for all of the tools

This set is only required for determining code coverage.

tests.test_toolchains.**biobambam_toolchain**(*verbose=False*)
Runs the tests for all of the tools from the BWA pipeline

Runs the following tests:

```
pytest -m chipseq tests/test_fastqc_validation.py
pytest -m chipseq tests/test_bwa_indexer.py
pytest -m chipseq tests/test_bwa_aligner.py
pytest -m chipseq tests/test_biobambam.py
```

tests.test_toolchains.**bowtie2_toolchain**(*verbose=False*)
Runs the tests for all of the tools from the BWA pipeline

Runs the following tests:

```
pytest -m bowtie2 tests/test_fastqc_validation.py
pytest -m bowtie2 tests/test_bowtie_indexer.py
pytest -m bowtie2 tests/test_bowtie2_aligner.py
```

tests.test_toolchains.**bwa_toolchain**(*verbose=False*)
Runs the tests for all of the tools from the BWA pipeline

Runs the following tests:

```
pytest -m bwa tests/test_fastqc_validation.py
pytest -m bwa tests/test_bwa_indexer.py
pytest -m bwa tests/test_bwa_aligner.py
```

tests.test_toolchains.**chipseq_toolchain**(*verbose=False*)
Runs the tests for all of the tools from the ChIP-seq pipeline

Runs the following tests:

```
pytest -m chipseq tests/test_fastqc_validation.py
pytest -m chipseq tests/test_bwa_indexer.py
pytest -m chipseq tests/test_bwa_aligner.py
pytest -m chipseq tests/test_biobambam.py
pytest -m chipseq tests/test_macs2.py
```

tests.test_toolchains.**genome_toolchain**(*verbose=False*)
Runs the tests for all of the tools from the Genome indexing pipeline

Runs the following tests:

```
pytest -m genome tests/test_bowtie_indexer.py
pytest -m genome tests/test_bwa_indexer.py
pytest -m genome tests/test_gem_indexer.py
```

tests.test_toolchains.**hic_toolchain**(*verbose=False*)
Runs the tests for all of the tools from the Hi-C pipeline

Runs the following tests:

```
pytest -m hic tests/test_fastqc_validation.py
pytest -m hic tests/test_gem_indexer.py
pytest -m hic tests/test_tb_full_mapping.py
pytest -m hic tests/test_tb_parse_mapping.py
pytest -m hic tests/test_tb_filter.py
pytest -m hic tests/test_tb_normalize.py
pytest -m hic tests/test_tb_segment.py
pytest -m hic tests/test_tb_generate_tads.py
pytest -m hic tests/test_tb_bin.py
pytest -m hic tests/test_tb_save_hdf5_matrix.py
```

tests.test_toolchains.**idamidseq_toolchain**(*verbose=False*)
Runs the tests for all of the tools from the iDamID-seq pipeline

Runs the following tests:

```
pytest -m idamidseq tests/test_bwa_indexer.py
pytest -m idamidseq tests/test_bwa_aligner.py
pytest -m idamidseq tests/test_biobambam.py
pytest -m idamidseq tests/test_bsgenome.py
pytest -m idamidseq tests/test_idear.py
```

tests.test_toolchains.**mnaseseq_toolchain**(*verbose=False*)
Runs the tests for all of the tools from the MNase-seq pipeline

Runs the following tests:

```
pytest -m mnaseseq tests/test_fastqc_validation.py
pytest -m mnaseseq tests/test_bwa_indexer.py
pytest -m mnaseseq tests/test_bwa_aligner.py
pytest -m mnaseseq tests/test_inps.py
```

tests.test_toolchains.**rnaseq_toolchain**(*verbose=False*)
Runs the tests for all of the tools from the RNA-seq pipeline

Runs the following tests:

```
pytest -m rnaseq tests/test_fastqc_validation.py
pytest -m rnaseq tests/test_kallisto_indexer.py
pytest -m rnaseq tests/test_kallisto_quant.py
```

tests.test_toolchains.**tidy_data**()
Runs the tidy_data.sh script

tests.test_toolchains.**wgbs_toolchain**(*verbose=0*)
Runs the tests for all of the tools from the WGBS pipeline

Runs the following tests:

```
pytest -m wgbs tests/test_fastqc_validation.py
pytest -m wgbs tests/test_bs_seeker_filter.py
pytest -m wgbs tests/test_bs_seeker_indexer.py
pytest -m wgbs tests/test_bs_seeker_aligner.py
pytest -m wgbs tests/test_bs_seeker_methylation_caller.py
```

CHAPTER 11

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

t

Index

C

E

F

G

R

S

T

U

V

W