bactpipeline Documentation

Release 2.0.2

Tyghe Vallard, Michael panciera

December 23, 2015

Contents

1 Install **3**
- 1.1 Requirements . . . 3
- 1.2 Installation . . . 3

2 Running the Pipeline **5**
- 2.1 Running a single sample . . . 5
- 2.2 Running multiple samples . . . 5
- 2.3 runsample . . . 6
- 2.4 fix_fastq . . . 8

3 Indices and tables **9**

Version: 2.0.2

Contents:

CHAPTER 1

Install

1.1 Requirements

- Roche 454 Analysis Software(gsAssembler, runProject)
- Everything should be bundled with the pipeline

1.2 Installation

1. Clone repo

```
git clone https://github.com/VDBWRAIR/bactpipeline.git
```

2. Enter code directory

```
cd bactpipeline
```

3. Install python virtualenv

```
virtualenv env
. env/bin/activate
```

4. Install bactpipeline

```
pip install -r requirements.txt
python setup.py install
```

CHAPTER 2

Running the Pipeline

If you have not already, make sure to check out the Install page first.

You will have to re-source the virtualenv when you want to use the pipeline, using

```
. bactpipeline/bin/activate
```

2.1 Running a single sample

Running a single sample is fairly straight forward assuming you have your miseq reads parsed out into individual folders for each sample.

2.1.1 Usage Example

For this example, assume you have a directory /home/username/reads/sample1 that contains MiSeq reads for sample1:

- sample1_S1_L001_R1_001.fastq
- sample1_S1_L001_R2_001.fastq

Simply run the following command to run the pipeline on sample1's data

```
runsample -o sample1 /home/username/reads/sample1
```

2.2 Running multiple samples

Multiple samples can be run with the `--sample-sheet` parameter.

```
./runsample --sample-sheet samplesheet.csv -o outdir
./runsample -s samplesheet.csv -o outdir
```

2.2.1 Sample Sheet Syntax

```
sample_directory,sample_id,primer_file
test/fixtures/fix_fastq,SampleA,
```

2.2.2 PBS/Torque Example

Here is a quick example on how to run all your samples using the qsub command

Here we are using a bash while loop to loop through all lines in the samplesheet.csv file to spawn a new job for each line.

We use qsub's -j option to ensure standard output and standard error are joined into one stream.

We then use qsub's -V option to ensure that your current environment gets forwarded on to each job. This is important as your current environment should already include your virtualenv's variables as well as having the newbler executables in your PATH.

First we will define where all project directories will be created:

```
OUTDIR="outdir"
```

Now we can run our loop over the samplesheet.csv file

```
mkdir -p $OUTDIR
while IFS=',' read path sn primer; do \
    [ "$sn" == "sample_id" ] && continue; \
    echo "cd \$PBS_O_WORKDIR; runsample -o $OUTDIR/${sn} $path" | qsub -j oe -N $sn -V; \
done < samplesheet.csv
```

Once all jobs are completed you can build your full report and aggregate all contig files into a single directory named after each sample.

```
grep -h sample $OUTDIR/*/summary.tsv | head -1 > $OUTDIR/full_summary.tsv
grep -h -v sample $OUTDIR/*/summary.tsv >> $OUTDIR/full_summary.tsv
mkdir -p $OUTDIR/contigs
for c in $OUTDIR/*/top_contigs.fasta; do sn=$(basename $(dirname $c)); ln -s ../${sn}/top_contigs.fas
```

2.3 runsample

This script takes care of putting all the pieces of the pipeline together

2.3.1 Pipeline Flow

- fix_fastq
- flash
- btrim
- Newbler(runAssembly)

Usage

- **Output Directory**
  - *-o* or *–output*
  - Specifies where to put all the resulting output directories for the various stages
  - Default: output
- **primer**

    - *-p* or *–primer*
    - Specify a primer trimming file to use for the newbler assembly. It is passed using *-vt* to *runProject*
- **sample sheet**
    - *-s* or *–sample-sheet*
    - Specify a samplesheet to parse containing your input files and primer files
- **truseq**
    - *-t* or *–truseq*
    - Specify the path to a truseq.txt file for adapter trimming
- **readdir**
    - Specifies a directory that contains the paired MiSeq reads

```
runsample --help
```

Output Files

- **fix_fastq**
    - fastq files with same name as were in the readdir argument but with sequence id modified for Newbler
- **flash/**
    - **out.extendedFrags.fastq**
        * paired reads combined together
    - **out.notCombined_1.fastq**
        * R1 reads that did not combine
    - **out.notCombined_2.fastq**
        * R2 reads that did not combine
    - **out.hist**
        * Combined read lengths
    - **out.histogram**
        * Combined read lengths visual
- **btrim**
    - fastq files with same name as out.*.fastq from flash, but with .btrim.fastq at end
- **newbler_assembly**
    - **gsAssembler project directory**
        * See Newbler documentation about contents of this directory.
- **top_contigs.fasta** Contains the top 100 contigs from newbler_assembly/assembly/454AllContigs.fna sorted by sequence length
- **summary.tsv** Summary file that contains quick easy summary to view about all the contigs including their length, number of reads used to compose them, N50, % of total reads from after btrim ran that compose each contig

2.4 fix_fastq

This script handles renaming sequence identifiers in Illumina reads such that Newbler will use them as paired end correctly.

It addresses this

2.4.1 Usage

```
fix_fastq [-o outdir] fastq [fastq ...]
```

Example usage

You essentially supply the script with the location of any fastq files you want and it will replace the sequence id in each and copy the modified version into an output directory.

If you have a bunch of fastq files in a directory, lets say /home/username/reads, then you could run it as follows:

```
fix_fastq -o newbler_reads /home/username/reads/*.fastq
```

All modified reads would then be placed in a directory called newbler_reads in the current directory.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search