process_radtags

This program examines raw reads from an Illumina sequencing run and first, checks that the barcode and the RAD cutsite are intact, and demultiplexes the data. If there are errors in the barcode or the RAD site within a certain allowance **process_radtags** can correct them. Second, it slides a window down the length of the read and checks the average quality score within the window. If the score drops below 90% probability of being correct (a raw phred score of 10), the read is discarded. This allows for some seqeuncing errors while elimating reads where the sequence is degrading as it is being sequenced. By default the sliding window is 15% of the length of the read, but can be specified on the command line (the threshold and window size can be adjusted).

The **process_radtags** program can:

- handle data that is barcoded, either inline or using an index, or unbarcoded.
- use combinatorial barcodes.
- check and correct for a restriction enzyme cutsite for single or double-digested data.
- filter adapter sequence while allowing for sequencing error in the adapter pattern.
- process individual files or whole directories of files.
- directly read/write gzipped data
- filter reads based on Illumina's Chastity filter.
- name output files according to their sample names instead of barcode names, if supplied.

Below you will find additional information on how to:

1. Run **process_radtags** with Illumina HiSeq data.
2. Run **process_radtags** with generic FASTQ data.
3. Run **process_radtags** with Illumina BUSTARD/GERALD data.
4. Choose the appropriate flags for your barcode type.

Program Options

```
process_radtags [-f in_file | -p in_dir [-P] [-I] | -1 pair_1 -2 pair_2] -b barcode_file -o out_d

                [-c] [-q] [-r] [-t len] [-D] [-w size] [-s lim] [-h]

    f   — path to the input file if processing single-end seqeunces.

    i   — input file type, either 'bustard' for the Illumina BUSTARD format,
          'bam', 'fastq' (default), or 'gzfastq' for gzipped FASTQ.

    y   — output type, either 'fastq', 'gzfastq', 'fasta', or 'gzfasta'
          (default is to match the input file type).

    p   — path to a directory of files.

    P   — files contained within directory specified by '-p' are paired.

    I   — specify that the paired-end reads are interleaved in single files.

    1   — first input file in a set of paired-end sequences.

    2   — second input file in a set of paired-end sequences.

    o   — path to output the processed files.
```

b — path to a file containing barcodes for this run.

c — clean data, remove any read with an uncalled base.

q — discard reads with low quality scores.

r — rescue barcodes and RAD-Tags.

t — truncate final read length to this value.

E — specify how quality scores are encoded, 'phred33' (Illumina 1.8+, Sanger, default) or 'phred64' (Illumina 1.3 - 1.5).

D — capture discarded reads to a file.

w — set the size of the sliding window as a fraction of the read length, between 0 and 1 (default 0.15).

s — set the score limit. If the average score within the sliding window drops below this value, the read is discarded (default 10).

h — display this help messsage.

Barcode options:

--inline_null: barcode is inline with sequence, occurs only on single-end read (default).

--index_null: barcode is provded in FASTQ header, occurs only on single-end read.

--inline_inline: barcode is inline with sequence, occurs on single and paired-end read.

--index_index: barcode is provded in FASTQ header, occurs on single and paired-end read.

--inline_index: barcode is inline with sequence on single-end read, occurs in FASTQ header for paired-end read.

--index_inline: barcode occurs in FASTQ header for single-end read, is inline with sequence on paired-end read.

Restriction enzyme options:

-e [enz], --renz_1 [enz]: provide the restriction enzyme used (cut site occurs on single-end read)

--renz_2 [enz]: if a double digest was used, provide the second restriction enzyme used (cut site occurs on the paired-end read).

Currently supported enzymes include:

'aciI', 'ageI', 'aluI', 'apeKI', 'apoI', 'aseI', 'bamHI', 'bfaI', 'bgIII', 'bspDI', 'bstYI', 'claI', 'ddeI', 'dpnII', 'eaeI', 'ecoRI', 'ecoRV', 'ecoT22I', 'hindIII', 'kpnI', 'mluCI', 'mseI', 'mspI', 'ndeI', 'nheI', 'nlaIII', 'notI', 'nsiI', 'pstI', 'rsaI', 'sacI', 'sau3AI', 'sbfI', 'sexAI', 'sgrAI', 'speI', 'sphI', 'taqI', 'xbaI', or 'xhoI'.

Adapter options:

--adapter_1 [sequence]: provide adaptor sequence that may occur on the single-end read for filtering.

--adapter_2 [sequence]: provide adaptor sequence that may occur on the paired-read for filtering.

--adapter_mm [mismatches]: number of mismatches allowed in the adapter sequence.

Output options:

--retain_header: retain unmodified FASTQ headers in the output.

--merge: if no barcodes are specified, merge all input files into a single output file.

Advanced options:

--filter_illumina: discard reads that have been marked by Illumina's chastity/purity filter as failing.

--disable_rad_check: disable checking if the RAD site is intact.

--barcode_dist: provide the distace between barcodes to allow for barcode rescue (default 2).

Example Usage

The **process_radtags** program is designed to work on several types of data. The latest versions of the Illumina analysis pipeline output all reads from the sequencer in a series of FASTQ formatted files. The FASTQ ID in these files contains a flag as to whether the read passed Illumina's interal quality filters and may contain a barcode (or index).

Prior Illumina analysis pipelines output the data either from the **BUSTARD** pipeline (data are unfiltered), in a series of tab-separated files, or from the **GERALD** pipeline, which is quality filtered by Illumina's internal filter. The **GERALD** output consists of a single file (or pair of files for paired-end data) in a FASTQ formatted file, despite having a ".txt" extension. Finally, **process_radtags** should work with generic, FASTQ formatted data.

If your data **do not contain barcodes**, simply omit the barcodes file, and **process_radtags** will place the filtered files in the output directory with the same name as the input files.

Illumina HiSeq Data

1. If your data are **single-end, Illumina HiSeq data**, in a directory called `raw`:

```
~/raw% ls
lane3_NoIndex_L003_R1_001.fastq  lane3_NoIndex_L003_R1_006.fastq  lane3_NoIndex_L003_R1_011
lane3_NoIndex_L003_R1_002.fastq  lane3_NoIndex_L003_R1_007.fastq  lane3_NoIndex_L003_R1_012
lane3_NoIndex_L003_R1_003.fastq  lane3_NoIndex_L003_R1_008.fastq  lane3_NoIndex_L003_R1_013
lane3_NoIndex_L003_R1_004.fastq  lane3_NoIndex_L003_R1_009.fastq
lane3_NoIndex_L003_R1_005.fastq  lane3_NoIndex_L003_R1_010.fastq
```

   Then you can run **process_radtags** in the following way:

```
% process_radtags -p ./raw/ -o ./samples/ -b ./barcodes/barcodes_lane3 \
                  -e sbfI -E phred33 -r -c -q
```

2. If your data are **paired-end, Illumina HiSeq data**, in a directory called `raw`:

```
~/raw% ls
lane4_NoIndex_L004_R1_001.fastq  lane4_NoIndex_L004_R1_009.fastq  lane4_NoIndex_L004_R2_005
lane4_NoIndex_L004_R1_002.fastq  lane4_NoIndex_L004_R1_010.fastq  lane4_NoIndex_L004_R2_006
lane4_NoIndex_L004_R1_003.fastq  lane4_NoIndex_L004_R1_011.fastq  lane4_NoIndex_L004_R2_007
lane4_NoIndex_L004_R1_004.fastq  lane4_NoIndex_L004_R1_012.fastq  lane4_NoIndex_L004_R2_008
lane4_NoIndex_L004_R1_005.fastq  lane4_NoIndex_L004_R2_001.fastq  lane4_NoIndex_L004_R2_009
lane4_NoIndex_L004_R1_006.fastq  lane4_NoIndex_L004_R2_002.fastq  lane4_NoIndex_L004_R2_010
lane4_NoIndex_L004_R1_007.fastq  lane4_NoIndex_L004_R2_003.fastq  lane4_NoIndex_L004_R2_011
lane4_NoIndex_L004_R1_008.fastq  lane4_NoIndex_L004_R2_004.fastq  lane4_NoIndex_L004_R2_012
```

   Then you simply add the `-P` flag. **process_radtags** understands the Illumina naming scheme and will figure out how to properly pair the files together:

```
% process_radtags -P -p ./raw/ -o ./samples/ -b ./barcodes/barcodes_lane4 \
                  -e sbfI -E phred33 -r -c -q
```

3. If your data are **gzipped, paired-end, Illumina HiSeq data**, in a directory called `raw`:

```
~/raw% ls
lane4_NoIndex_L004_R1_001.fastq.gz  lane4_NoIndex_L004_R1_009.fastq.gz  lane4_NoIndex_L004_R
lane4_NoIndex_L004_R1_002.fastq.gz  lane4_NoIndex_L004_R1_010.fastq.gz  lane4_NoIndex_L004_R
lane4_NoIndex_L004_R1_003.fastq.gz  lane4_NoIndex_L004_R1_011.fastq.gz  lane4_NoIndex_L004_R
lane4_NoIndex_L004_R1_004.fastq.gz  lane4_NoIndex_L004_R1_012.fastq.gz  lane4_NoIndex_L004_R
lane4_NoIndex_L004_R1_005.fastq.gz  lane4_NoIndex_L004_R2_001.fastq.gz  lane4_NoIndex_L004_R
lane4_NoIndex_L004_R1_006.fastq.gz  lane4_NoIndex_L004_R2_002.fastq.gz  lane4_NoIndex_L004_R
lane4_NoIndex_L004_R1_007.fastq.gz  lane4_NoIndex_L004_R2_003.fastq.gz  lane4_NoIndex_L004_R
lane4_NoIndex_L004_R1_008.fastq.gz  lane4_NoIndex_L004_R2_004.fastq.gz  lane4_NoIndex_L004_R
```

Then you specify the input file type using the `-i` flag:

```
% process_radtags -P -p ./raw/ -o ./samples/ -b ./barcodes/barcodes_lane4 \
                  -e sbfI -E phred33 -r -c -q -i gzfastq
```

4. If your data are **double-digested, paired-end, Illumina HiSeq data using combinatorial barcodes**, in a directory called `raw`:

```
~/raw% ls
GfddRAD1_005_ATCACG_L007_R1_001.fastq.gz  GfddRAD1_005_ATCACG_L007_R2_001.fastq.gz
GfddRAD1_005_ATCACG_L007_R1_002.fastq.gz  GfddRAD1_005_ATCACG_L007_R2_002.fastq.gz
GfddRAD1_005_ATCACG_L007_R1_003.fastq.gz  GfddRAD1_005_ATCACG_L007_R2_003.fastq.gz
GfddRAD1_005_ATCACG_L007_R1_004.fastq.gz  GfddRAD1_005_ATCACG_L007_R2_004.fastq.gz
GfddRAD1_005_ATCACG_L007_R1_005.fastq.gz  GfddRAD1_005_ATCACG_L007_R2_005.fastq.gz
GfddRAD1_005_ATCACG_L007_R1_006.fastq.gz  GfddRAD1_005_ATCACG_L007_R2_006.fastq.gz
GfddRAD1_005_ATCACG_L007_R1_007.fastq.gz  GfddRAD1_005_ATCACG_L007_R2_007.fastq.gz
GfddRAD1_005_ATCACG_L007_R1_008.fastq.gz  GfddRAD1_005_ATCACG_L007_R2_008.fastq.gz
GfddRAD1_005_ATCACG_L007_R1_009.fastq.gz  GfddRAD1_005_ATCACG_L007_R2_009.fastq.gz
```

Then you specify both restriction enzymes using the `--renz_1` and `--renz_2` flags. You must also specify the type combinatorial barcoding used, such as inline/inline, or inline/index, specifying the type of barcodes to look for on the single and paired-end read:

```
% process_radtags -P -p ./raw  -b ./barcodes/barcodes_lane4 -o ./samples/ \
                  -c -q -r --inline_index --renz_1 nlaIII --renz_2 mluCI -i gzfastq
```

See below on how to format the barcodes file.

5. If your data **may contain adapter sequence**, and are **Illumina HiSeq data**, in a directory called `raw`:

```
~/raw% ls
lane4_NoIndex_L004_R1_001.fastq  lane4_NoIndex_L004_R1_009.fastq  lane4_NoIndex_L004_R2_005
lane4_NoIndex_L004_R1_002.fastq  lane4_NoIndex_L004_R1_010.fastq  lane4_NoIndex_L004_R2_006
lane4_NoIndex_L004_R1_003.fastq  lane4_NoIndex_L004_R1_011.fastq  lane4_NoIndex_L004_R2_007
lane4_NoIndex_L004_R1_004.fastq  lane4_NoIndex_L004_R1_012.fastq  lane4_NoIndex_L004_R2_008
lane4_NoIndex_L004_R1_005.fastq  lane4_NoIndex_L004_R2_001.fastq  lane4_NoIndex_L004_R2_009
lane4_NoIndex_L004_R1_006.fastq  lane4_NoIndex_L004_R2_002.fastq  lane4_NoIndex_L004_R2_010
lane4_NoIndex_L004_R1_007.fastq  lane4_NoIndex_L004_R2_003.fastq  lane4_NoIndex_L004_R2_011
lane4_NoIndex_L004_R1_008.fastq  lane4_NoIndex_L004_R2_004.fastq  lane4_NoIndex_L004_R2_012
```

Then you specify the the adapter sequence you expext to be present in the front read and optionally the adapter seqeunce expected to be present on the paired-end read, and the number of mismatches you want to allow in the adapter sequence (if any):

```
% process_radtags -P -p ./raw/ -o ./samples/ -b ./barcodes/barcodes_lane4 \
                  -e sbfI -E phred33 -r -c -q \
                  --adapter_1 GATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCGATCTCGTATGCCGTCTTCTGCTTG \
```

```
--adapter_2 AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGATCTCGGTGGTCGCCGTATCATT \
--adapter_mm 2
```

Generic FASTQ Data

1. If your data are **paired-end** but don't use the Illumina naming scheme, or were renamed, you can specify the pairs explicitly. If your data are in a directory called `raw`:

   ```
   ~/raw% ls
   Raw_Rad_data_R1.fastq  Raw_Rad_data_R2.fastq
   ```

   Then you use the `-1` and `-2` parameters to specify a pair of files. If you have multiple pairs of files, you can run **process_radtags** multiple times (using a shell script) and concatenate the outputs together (or you can concatenate the input files together as well).

   ```
   % process_radtags -1 ./raw/Raw_Rad_data_R1.fastq -2 ./raw/Raw_Rad_data_R2.fastq \
                     -o ./samples/ -b ./barcodes/barcodes -e sbfI -r -c -q
   ```

2. If your data are **single-end** but don't use the Illumina naming scheme, or were renamed, you can specify the single file explicitly. If the file is in a directory called `raw`:

   ```
   ~/raw% ls
   rad_data.fq
   ```

   Then you use the `-f` parameter.

   ```
   % process_radtags -f ./raw/rad_data.fq -o ./samples/ -b ./barcodes/barcodes -e sbfI -r -c -
   ```

Illumina BUSTARD/GERALD Data

1. Earlier versions of the Illumina BUSTARD pipeline provided unfiltered, tab-separated files containing the raw reads. There is generally one file per sequencer tile, up to 120 files total. Stacks refers to this file type as 'bustard' format. These files can be processed with **process_radtags** by specifying an input type file with the `-i` parameter.

   Given **single-end BUSTARD-formatted** data in a directory called `raw`:

   ```
   ~/raw% ls
   s_8_1_0001_qseq.txt  s_8_1_0025_qseq.txt  s_8_1_0049_qseq.txt  s_8_1_0073_qseq.txt  s_8_1_0
   s_8_1_0002_qseq.txt  s_8_1_0026_qseq.txt  s_8_1_0050_qseq.txt  s_8_1_0074_qseq.txt  s_8_1_0
   s_8_1_0003_qseq.txt  s_8_1_0027_qseq.txt  s_8_1_0051_qseq.txt  s_8_1_0075_qseq.txt  s_8_1_0
   s_8_1_0004_qseq.txt  s_8_1_0028_qseq.txt  s_8_1_0052_qseq.txt  s_8_1_0076_qseq.txt  s_8_1_0
   s_8_1_0005_qseq.txt  s_8_1_0029_qseq.txt  s_8_1_0053_qseq.txt  s_8_1_0077_qseq.txt  s_8_1_0
   s_8_1_0006_qseq.txt  s_8_1_0030_qseq.txt  s_8_1_0054_qseq.txt  s_8_1_0078_qseq.txt  s_8_1_0
   s_8_1_0007_qseq.txt  s_8_1_0031_qseq.txt  s_8_1_0055_qseq.txt  s_8_1_0079_qseq.txt  s_8_1_0
   s_8_1_0008_qseq.txt  s_8_1_0032_qseq.txt  s_8_1_0056_qseq.txt  s_8_1_0080_qseq.txt  s_8_1_0
   ...
   s_8_1_0019_qseq.txt  s_8_1_0043_qseq.txt  s_8_1_0067_qseq.txt  s_8_1_0091_qseq.txt  s_8_1_0
   s_8_1_0020_qseq.txt  s_8_1_0044_qseq.txt  s_8_1_0068_qseq.txt  s_8_1_0092_qseq.txt  s_8_1_0
   s_8_1_0021_qseq.txt  s_8_1_0045_qseq.txt  s_8_1_0069_qseq.txt  s_8_1_0093_qseq.txt  s_8_1_0
   s_8_1_0022_qseq.txt  s_8_1_0046_qseq.txt  s_8_1_0070_qseq.txt  s_8_1_0094_qseq.txt  s_8_1_0
   s_8_1_0023_qseq.txt  s_8_1_0047_qseq.txt  s_8_1_0071_qseq.txt  s_8_1_0095_qseq.txt  s_8_1_0
   s_8_1_0024_qseq.txt  s_8_1_0048_qseq.txt  s_8_1_0072_qseq.txt  s_8_1_0096_qseq.txt  s_8_1_0
   ```

   You can run **process_radtags** like this:

```
% process_radtags -p ./raw/ -o ./samples/ -b ./barcodes/barcodes_lane8 -e sbfI -r -c -q -i f
```

2. Given **paired-end BUSTARD-formatted** data in a directory called `raw` add the `-P` parameter:

```
~/raw% ls
s_7_1_0001_qseq.txt  s_7_1_0049_qseq.txt  s_7_1_0097_qseq.txt  s_7_2_0025_qseq.txt  s_7_2_0(
s_7_1_0002_qseq.txt  s_7_1_0050_qseq.txt  s_7_1_0098_qseq.txt  s_7_2_0026_qseq.txt  s_7_2_0(
s_7_1_0003_qseq.txt  s_7_1_0051_qseq.txt  s_7_1_0099_qseq.txt  s_7_2_0027_qseq.txt  s_7_2_0(
s_7_1_0004_qseq.txt  s_7_1_0052_qseq.txt  s_7_1_0100_qseq.txt  s_7_2_0028_qseq.txt  s_7_2_0(
s_7_1_0005_qseq.txt  s_7_1_0053_qseq.txt  s_7_1_0101_qseq.txt  s_7_2_0029_qseq.txt  s_7_2_0(
s_7_1_0006_qseq.txt  s_7_1_0054_qseq.txt  s_7_1_0102_qseq.txt  s_7_2_0030_qseq.txt  s_7_2_0(
s_7_1_0007_qseq.txt  s_7_1_0055_qseq.txt  s_7_1_0103_qseq.txt  s_7_2_0031_qseq.txt  s_7_2_0(
...
s_7_1_0041_qseq.txt  s_7_1_0089_qseq.txt  s_7_2_0017_qseq.txt  s_7_2_0065_qseq.txt  s_7_2_0
s_7_1_0042_qseq.txt  s_7_1_0090_qseq.txt  s_7_2_0018_qseq.txt  s_7_2_0066_qseq.txt  s_7_2_0
s_7_1_0043_qseq.txt  s_7_1_0091_qseq.txt  s_7_2_0019_qseq.txt  s_7_2_0067_qseq.txt  s_7_2_0
s_7_1_0044_qseq.txt  s_7_1_0092_qseq.txt  s_7_2_0020_qseq.txt  s_7_2_0068_qseq.txt  s_7_2_0
s_7_1_0045_qseq.txt  s_7_1_0093_qseq.txt  s_7_2_0021_qseq.txt  s_7_2_0069_qseq.txt  s_7_2_0
s_7_1_0046_qseq.txt  s_7_1_0094_qseq.txt  s_7_2_0022_qseq.txt  s_7_2_0070_qseq.txt  s_7_2_0
s_7_1_0047_qseq.txt  s_7_1_0095_qseq.txt  s_7_2_0023_qseq.txt  s_7_2_0071_qseq.txt  s_7_2_0
s_7_1_0048_qseq.txt  s_7_1_0096_qseq.txt  s_7_2_0024_qseq.txt  s_7_2_0072_qseq.txt  s_7_2_0
```

You can run **process_radtags** like this:

```
% process_radtags -P -p ./raw/ -o ./samples/ -b ./barcodes/barcodes_lane7 -e sbfI -r -c -q
```

3. Given **paired-end GERALD-formatted** data in a directory called `raw`:

```
~/raw% ls
s_3_1_sequence.txt  s_3_2_sequence.txt
```

You can run **process_radtags** like this:

```
% process_radtags -1 ./raw/s_3_1_sequence.txt -2 ./raw/s_3_2_sequence.txt -o ./samples/ \
                  -b ./barcodes/barcodes_lane3 -e sbfI -r -c -q -i fastq
```

4. Given **single-end GERALD-formatted** data in a directory called `raw`:

```
~/raw% ls
s_3_sequence.txt
```

You can run **process_radtags** like this:

```
% process_radtags -f ./raw/s_3_sequence.txt -o ./samples/ -b ./barcodes/barcodes_lane4
                  -e sbfI -r -c -q -i fastq
```

Specifying the Barcode Type

1. If your data are **single-end** or **paired-end**, with an inline barcode present only on the single-end (marked in red):

```
@HWI-ST0747:188:C09HWACXX:1:1101:2968:2083 1:N:0:
TTATGATGCAGGACCAGGATGACGTCAGCACAGTGCGGGTCCTCCATGGATGCTCCTCGGTCGTGGTTGGGGGAGGAGGCA
+
```

```
@@@DDDDDBHHFBF@CCAGEHHHBFGIIFGIIGIEDBBGFHCGIIGAEEEDCC;A?;;5,:@A?=B5559999B@BBBBBA
@HWI-ST0747:188:C09HWACXX:1:1101:2863:2096 1:N:0:
TTATGATGCAGGCAAATAGAGTTGGATTTTGTGTCAGTAGGCGGTTAATCCCATACAATTTTACACTTTATTCAAGGTGGA
+
CCCFFFFFHHHHHJJGHIGGAHHIIGGIIJDHIGCEGHIFIJIH7DGIIIAHIJGEDHIDEHJJHFEEECEFEFFDECDDD
@HWI-ST0747:188:C09HWACXX:1:1101:2837:2098 1:N:0:
GTGCCTTGCAGGCAATTAAGTTAGCCGAGATTAAGCGAAGGTTGAAAATGTCGGATGGAGTCCGGCAGCAGCGAATGTAAA
```

Then you can specify the `--inline_null` flag to **process_radtags**. This is also the default behavior and the flag can be ommitted in this case.

2. If your data are **single-end** or **paired-end**, with a single index barcode (in blue):

```
@9432NS1:54:C1K8JACXX:8:1101:6912:1869 1:N:0:ATGACT
TCAGGCATGCTTTCGACTATTATTGCATCAATGTTCTTTGCGTAATCAGCTACAATATCAGGTAATATCAGGCGCA
+
CCCFFFFFHHHHHJJJJJJJJIJJJJJJJJJJJHIIJJJJJIJJJJJJJJJJJJJJJJJJGIJJJJJJJHHHFF
@9432NS1:54:C1K8JACXX:8:1101:6822:1873 1:N:0:ATGACT
CAGCGCATGAGCTAATGTATGTTTTACATTCCAGAAAGAGAGCTACTGCTGCAGGTTGTGATAAAATAAAGTAAGA
+
B@@FFFFFHFFHHJJJJFHIJHGGGHIJIIJIJCHJIIGGIIIGGIJEHIJJHII?FFHICHFFGGHIIGG@DEHH
@9432NS1:54:C1K8JACXX:8:1101:6793:1916 1:N:0:ATGACT
TTTCGCATGCCCTATCCTTTTATCACTCTGTCATTCAGTGTGGCAGCGGCCATAGTGTATGGCGTACTAAGCGAAA
+
@C@DFFFFHGHHHGIGHHJJJJJJJGIJIJJIGIJJJJHIGGGHGII@GEHIGGHDHEHIHD6?493;AAA?;=;=
```

Then you can specify the `--index_null` flag to **process_radtags**.

3. If your data are **single-end** with both an inline barcode (in red) and an index barcode (in blue):

```
@9432NS1:54:C1K8JACXX:8:1101:6912:1869 1:N:0:ATCACG
TCACGCATGCTTTCGACTATTATTGCATCAATGTTCTTTGCGTAATCAGCTACAATATCAGGTAATATCAGGCGCA
+
CCCFFFFFHHHHHJJJJJJJJIJJJJJJJJJJJHIIJJJJJIJJJJJJJJJJJJJJJJJJGIJJJJJJJHHHFF
@9432NS1:54:C1K8JACXX:8:1101:6822:1873 1:N:0:ATCACG
GTCCGCATGAGCTAATGTATGTTTTACATTCCAGAAAGAGAGCTACTGCTGCAGGTTGTGATAAAATAAAGTAAGA
+
B@@FFFFFHFFHHJJJJFHIJHGGGHIJIIJIJCHJIIGGIIIGGIJEHIJJHII?FFHICHFFGGHIIGG@DEHH
@9432NS1:54:C1K8JACXX:8:1101:6793:1916 1:N:0:ATCACG
GTCCGCATGCCCTATCCTTTTATCACTCTGTCATTCAGTGTGGCAGCGGCCATAGTGTATGGCGTACTAAGCGAAA
+
@C@DFFFFHGHHHGIGHHJJJJJJJGIJIJJIGIJJJJHIGGGHGII@GEHIGGHDHEHIHD6?493;AAA?;=;=
```

Then you can specify the `--inline_index` flag to **process_radtags**.

4. If your data are **paired-end** with an inline barcode on the single-end (in red) and an index barcode (in blue):

```
@9432NS1:54:C1K8JACXX:7:1101:5584:1725 1:N:0:CGATGT
ACTGGCATGATGATCATAGTATAACGTGGGATACATATGCCTAAGGCTAAAGATGCCTTGAAGCTTGGCTTATGTT
+
#1=DDDFFHFHFHIFGIEHIEHGIIHFFHICGGGIIIIIIIIAEIGIGHAHIEGHHIHIIGFFFGGIIIGIIIEE7
@9432NS1:54:C1K8JACXX:7:1101:5708:1737 1:N:0:CGATGT
TTCGACATGTGTTTACAACGCGAACGGACAAAGCATTGAAAATCCTTGTTTTGGTTTCGTTACTCTCTCCTAGCAT
+
#1=DFFFFHHHHHJJJJJJJJJJJJJJJIIJIJJJJJJJJJIIJJHHHHHFEFEEDDDDDDDDDDDDDDDDD@
```

```
@9432NS1:54:C1K8JACXX:7:1101:5584:1725 2:N:0:CGATGT
AATTTACTTTGATAGAAGAACAACATAAGCCAAGCTTCAAGGCATCTTTAGCCTTAGGCATATGTATCCCACGTTA
+
@@@DFFFFHGHDHIIJJJGGIIIEJJJCHIIIGIJGGEGGIIGGGIJIJIHIIJJJJIJJJIIIGGIIJJJIICEH
@9432NS1:54:C1K8JACXX:7:1101:5708:1737 2:N:0:CGATGT
AGTCTTGTGAAAAACGAAATCTTCCAAAATGCTAGGAGAGAGTAACGAAACCAAAACAAGGATTTTCAATGCTTTG
```

```
+
C@CFFFFFHHHHHJJJJJIJJJJJJJJJJJJJIJJJHIJJFHIIJJJJIIJJJJJJJJJHGHHHHFFFFFFFED
```

Then you can specify the `--inline_index` flag to **process_radtags**.

5. If your data are **paired-end** with indexed barcodes on the single and paired-ends (in blue):

```
@9432NS1:54:C1K8JACXX:7:1101:5584:1725 1:N:0:ATCACG
ACTGGCATGATGATCATAGTATAACGTGGGATACATATGCCTAAGGCTAAAGATGCCTTGAAGCTTGGCTTATGTT
+
#1=DDDFFHFHFHIFGIEHIEHGIIHFFHICGGGIIIIIIIIAEIGIGHAHIEGHHIHIIGFFFGGIIIGIIIEE7
@9432NS1:54:C1K8JACXX:7:1101:5708:1737 1:N:0:ATCACG
TTCGACATGTGTTTACAACGCGAACGGACAAAGCATTGAAAATCCTTGTTTTGGTTTCGTTACTCTCTCCTAGCAT
+
#1=DFFFFHHHHHJJJJJJJJJJJJJJJJIIJIJJJJJJJJJJIIJJHHHHHFEFEEDDDDDDDDDDDDDDDD@
```

```
@9432NS1:54:C1K8JACXX:7:1101:5584:1725 2:N:0:CGATGT
AATTTACTTTGATAGAAGAACAACATAAGCCAAGCTTCAAGGCATCTTTAGCCTTAGGCATATGTATCCCACGTTA
+
@@@DFFFFHGHDHIIJJJGGIIIEJJJCHIIIGIJGGEGGIIGGGIJIJIHIIJJJJIJJJIIIGGIIJJJIICEH
@9432NS1:54:C1K8JACXX:7:1101:5708:1737 2:N:0:CGATGT
AGTCTTGTGAAAAACGAAATCTTCCAAAATGCTAGGAGAGAGTAACGAAACCAAAACAAGGATTTTCAATGCTTTG
+
C@CFFFFFHHHHHJJJJJJIJJJJJJJJJJJJJIJJJHIJJFHIIJJJJIIJJJJJJJJJHGHHHHFFFFFFFED
```

Then you can specify the `--index_index` flag to **process_radtags**.

6. If your data are **paired-end** with inline barcodes on the single and paired-ends (in red):

```
@9432NS1:54:C1K8JACXX:7:1101:5584:1725 1:N:0:
ACTGGCATGATGATCATAGTATAACGTGGGATACATATGCCTAAGGCTAAAGATGCCTTGAAGCTTGGCTTATGTT
+
#1=DDDFFHFHFHIFGIEHIEHGIIHFFHICGGGIIIIIIIIAEIGIGHAHIEGHHIHIIGFFFGGIIIGIIIEE7
@9432NS1:54:C1K8JACXX:7:1101:5708:1737 1:N:0:
TTCGACATGTGTTTACAACGCGAACGGACAAAGCATTGAAAATCCTTGTTTTGGTTTCGTTACTCTCTCCTAGCAT
+
#1=DFFFFHHHHHJJJJJJJJJJJJJJJJIIJIJJJJJJJJJJIIJJHHHHHFEFEEDDDDDDDDDDDDDDDD@
```

```
@9432NS1:54:C1K8JACXX:7:1101:5584:1725 2:N:0:
AATTTACTTTGATAGAAGAACAACATAAGCCAAGCTTCAAGGCATCTTTAGCCTTAGGCATATGTATCCCACGTTA
+
@@@DFFFFHGHDHIIJJJGGIIIEJJJCHIIIGIJGGEGGIIGGGIJIJIHIIJJJJIJJJIIIGGIIJJJIICEH
@9432NS1:54:C1K8JACXX:7:1101:5708:1737 2:N:0:
AGTCTTGTGAAAAACGAAATCTTCCAAAATGCTAGGAGAGAGTAACGAAACCAAAACAAGGATTTTCAATGCTTTG
+
C@CFFFFFHHHHHJJJJJJIJJJJJJJJJJJJJIJJJHIJJFHIIJJJJIIJJJJJJJJJHGHHHHFFFFFFFED
```

Then you can specify the `--inline_inline` flag to **process_radtags**.

Barcode File Format

The barcode file is a very simple format — one barcode per line.

```
% more barcodes_lane3
CGATA
CGGCG
GAAGC
GAGAT
TAATG
TAGCA
AAGGG
```

```
ACACG
ACGTA
```

Combinatorial barcodes are specified, one per column, separated by a tab:

```
% more barcodes_lane07
CGATA<tab>ACGTA
CGGCG     ACGTA
GAAGC     ACGTA
GAGAT     ACGTA
CGATA     TAGCA
CGGCG     TAGCA
GAAGC     TAGCA
GAGAT     TAGCA
```

Here is an example that includes sample names. The **process_radtags** program will demultiplex reads according to the barcode, but will write them to an output file with the sample name you specify in the barcodes file in an additional, tab separated column.

```
% more barcodes_run01_lane01
CGATA<tab>spruce_site_12-01
CGGCG     spruce_site_12-02
GAAGC     spruce_site_12-03
GAGAT     spruce_site_12-04
TAATG     spruce_site_06-01
TAGCA     spruce_site_06-02
AAGGG     spruce_site_06-03
ACACG     spruce_site_06-04
```

Combinatorial barcodes are specified, one per column, separated by a tab:

```
% more barcodes_run01_lane06
CGATA<tab>ACGTA<tab>sample-01
CGGCG     ACGTA     sample-02
GAAGC     ACGTA     sample-03
```

Other Pipeline Programs

Raw Reads

- process_radtags
- process_shortreads
- clone_filter
- kmer_filter

Core

- ustacks
- pstacks
- cstacks
- sstacks
- genotypes
- populations
- rxstacks

Execution control

- denovo_map.pl
- ref_map.pl
- load_radtags.pl

Utilities

- index_radtags.pl
- export_sql.pl
- sort_read_pairs.pl
- exec_velvet.pl