Raw data processing

The raw data processing includes the necessary steps to convert SFF files into FASTA (sequence data) and QUAL (quality scores) files.

Applicable for the following GS Sequencer platforms/kits:

- GS 20
- GS FLX
- GS FLX Titanium
- GS Junior

Content:

- Necessary resources
- Split SFF files (Multiplexed data only)
- Join SFF files
- Extract FASTA+QUAL data from SFF file

Background:

The GS Sequencer records a set of raw digital images representing the light detected over the PTP device, during each reagent flow of the sequencing run. This information is stored as raw image files in .pif format. The image processing is performed in real time during the sequencing run as the images are being captured. During the image processing, the raw signals for each nucleotide flow are extracted from each active well in the PTP device. The uncorrected flow signals are stored in composite wells format (.cwf) with metadata about the sequencing run and data processing. During the signal processing, the flow signal data is normalized, corrected, quality filtered and then converted into flowgrams for each read. These flowgrams include basecalls and quality scores for individual reads and are stored in standard flowgram format (.sff) files (one per region). Information about reads which pass filters, as well metrics associated with each filter can be found in the two files "454BaseCallerMetrics" and "454QualityFilterMetrics".

Necessary resources

Hardware

Computer with a Linux/Unix or MAC OSX operating system

Software

sffinfo and sfffile (part of 454/Roche SFF Tools)

Files

SFF file(s)

MIDConfig.parse (for multiplexed data)

Notes

The SFF Tools are the programs of choice as they were specifically designed for this kind of data and incorporate error models currently allowing up to two mismatches in MID tags. Alternative software such as sff_extract can be used if the SFF Tools are not accessible. However, this particular program showed problems of incorrectly processed outputs in the past and might only be used with caution. Using self-written programs should be used with caution too, as they likely do not implement any error model for MID tags to separate the sequences correctly and might not trim the MID tags from the sequence reads, resulting in additional steps during data preprocessing.

Split SFF files (Multiplexed data only)

Multiplexed data requires this additional step. Multiplexed data contains 10-12 nt sequence tags (MIDs) that are used to determine the source of the reads. The MIDs are part of the adaptors used for library preparation and are added between the sequencing key and the template-specific primer. This step should be performed on single SFF files instead of multiple files. Sequencing errors can alter the MIDs beyond the two allowed changes and result in an alternative MID. Processing each file separately allows the identification of altered MIDs (see "Example" below for more detail).

1. Open a shell/terminal window and go to the directory with the SFF file(s) as the output will by default be written to the current directory

   ```
   $ cd directory_with_sff_files
   ```

2. Use the following command to split the SFF file by MID tags and generate separate SFF files for each MID tag found

   ```
   $ sfffile –s file.sff
   ```

Notes

The output of the program is written to the file "454Reads.sff" in the current working directory, or the filename given by the "-o" option. The per-MID output files will append the MID names to the filename (i.e., by default for the GSMIDS set, the output files will be "454Reads.MID1.sff", "454Reads.MID2.sff", ...).

By default, a "manifest" listing the set of 454 runs that were the source of the reads in an SFF file are written into the index section of the SFF file (to provide an explicit traceback to the origin of the reads). This includes the run name (the R_... name), the analysis name (the D_... name) and the full path to the analysis directory used in the conversion from 454 run data to SFF file(s). If the manifest should not be written to the output SFF files, use the "-nmft" option.

Example output

```
$ sfffile -s example.sff
Reading the input SFF file(s)...
Generating the split SFF file(s)...
  RL1:   220183 reads written into the SFF file.
  RL2:   251824 reads written into the SFF file.
  RL3:   24 reads written into the SFF file.
  RL4:   3 reads written into the SFF file.
  RL5:  0 reads found.
  RL6:  0 reads found.
  RL7:  0 reads found.
  RL8:  0 reads found.
  RL9:   1 reads written into the SFF file.
  RL10:  0 reads found.
  RL11:  0 reads found.
  RL12:  0 reads found.
$ ls
454Reads.MID1.sff  454Reads.MID2.sff  454Reads.MID3.sff
454Reads.MID4.sff  454Reads.MID9.sff  example.sff
```

As the example output shows, there are 24 sequence reads for MID 3, three for MID 4 and one for MID 9. Processing this SFF file with an SFF file that contains sequences with any of these three MIDs will result in datasets with sequences from different samples. In most cases, you will ignore the SFF files for wrong MIDs (here 3, 4 and 9), as they most likely contain sequences with high error rates that should not be used for downstream analysis.

Possible problems

1. The MID tag information cannot be found by sfffile:

   If you have write access for the sfffile installation directory, create a folder with the name "config" and copy the MIDConfig.parse file into this directory. Alternatively, you can specify the location of the MIDConfig.parse file using the "–mcf" option as follows:

   ```
   $ sfffile –mcf location_of_MID_file –s file.sff
   ```

2. You do not have enough memory to process the SFF file and will see something like this:

   ```
   Reading the input SFF file(s)...
   terminate called after throwing an instance of
   'std::bad_alloc'
         what(): St9bad_alloc
   Aborted
   ```

   Currently, the only way to get around this is to find a computer with more memory and try again.

Note on errors in MID

Approximately 3% of the reads will have errors in their MID. About half of the errors are deletions, one-third are replacements and the rest are insertions. The SFF Tools are able to uniquely identify MIDs with up to 2 or 3 errors (depending on the MID used). If you want to exclude sequences with errors in the MID from downstream analysis, you can filter out those sequences after you extracted the *untrimmed* sequence and quality data (see below). Make sure that you trim MID tags from the 3' as well if you use untrimmed data with RLMIDs (Rapid Library MIDs).

According to a study by Huse *et al.* ("Accuracy and quality of massively parallel DNA pyrosequencing", Genome Biology 2007), MIDs with three or more errors were indicative of low quality sequences downstream of the MID. MIDs with fewer errors were not a good predictor of overall error of the read. This basically means that if the same applies to later generations of the sequencing platform, you can safely use the sequences trimmed with sfffile.

Join SFF files

This step should be performed if you run the same sample in different regions (e.g. as technical replicates or in separate sequencing runs) and want to join the multiple SFF files into a single SFF file for this specific sample.

1. Open a shell/terminal window and go to the directory with the SFF files as the output will by default be written to the current directory

```
$ cd directory_with_sff_files
```

2. Use the following command to join the SFF files and generate a single SFF file

```
$ sfffile file1.sff file2.sff file3.sff
```

Notes

The output of the program is written to the file "454Reads.sff" in the current working directory, or the filename given by the "-o" option. Make sure you do not have a SFF file with the same name (e.g. 454Reads.sff) in this directory, as this file will be overwritten with the new data without warning.

Example output

```
$ sfffile file1.sff file2.sff file3.sff
Reading the input SFF file(s)...
Generating the SFF file...
   7834 reads written into the SFF file.
$ ls
454Reads.sff  file1.sff  file2.sff  file3.sff

$ sfffile -o allfiles.sff file1.sff file2.sff file3.sff
Reading the input SFF file(s)...
Generating the SFF file...
   7834 reads written into the SFF file.
$ ls
allfiles.sff  file1.sff  file2.sff  file3.sff
```

Extract FASTA+QUAL data from SFF file

The signal flowgrams of the reads are basecalled and quality scores for each base are calculated based on the signal properties relative to the statistical distributions of the Control DNA fragment signals. The base-called files for each region are outputted as FASTA files, while the associated quality files which contains the quality score for each base is outputted as QUAL file. (Quality scores are phred-based since version 1.1.03; ranging from 0 to 40)

1. Open a shell/terminal window and go to the directory with the SFF file(s)

```
$ cd directory_with_sff_files
```

2. Use the following command to extract the FASTA data from the SFF file

```
$ sffinfo -s file.sff > file.fasta
```

3. Use the following command to extract the QUAL data from the SFF file

```
$ sffinfo -q file.sff > file.qual
```

4. Use the following command to extract the manifest from the SFF file (if required)

```
$ sffinfo -m file.sff > file.mft
```

If you have to extract data from multiple .sff files, you can use some Perl to process multiple files as follows:

```
$ perl -e
'foreach("file_MID1","file_MID2","file_something","some_other_fil
e"){`sffinfo -s $_.sff > $_.fasta`;`sffinfo -q $_.sff >
$_.qual`;`sffinfo -m $_.sff > $_.mft`;}'
```

Note that the file names of the .sff files do not contain the .sff extension.

Notes

To extract the untrimmed sequence and quality data, add an additional `-n` after or before the `-s` and `-q` in the `sffinfo` command. The untrimmed sequences will be soft-masked. Lower case letters represent the regions that would be trimmed off and upper case letters the trimmed region that would be returned without the `-n` option.

gact**acagacgact**TGATTTCGACAGGTTCGACGAAGGCCGCCGGGATGCGCCGCAGAACAAGCA
CAGCGAATCGATCCTGTCCGTCCATCCGGCGTgaacgtactgcaaggcgacacaggggagtagng

Depending if your sequences contain an MID tag (bold) and if you separated the SFF files by MID, the MID tag will either be marked as trimmed (lower case) or untrimmed (upper case).