SRA Barcoding Guide

Version 1.1 Mar 10 2010

National Center for Biotechnology Information – National Library of Medicine

EMBL European Bioinformatics Institute

DNA Databank of Japan

1 Contents

SRA Barcoding Guide 1

Version 1.1 Mar 10 2010 1

National Center for Biotechnology Information – National Library of Medicine 1

EMBL European Bioinformatics Institute 1

DNA Databank of Japan 1

2 Overview 1

3 Use Cases 2

3.1 Default 2

3.2 Sample Pool 2

3.3 Sample Pool with Barcodes 2

3.3.1 Sample pool with barcodes de-multiplexed by the submitter 3

3.3.2 Sample pool with barcodes de-multiplexed by the Archive 6

3.4 Study Pool 6

4 Features 6

4.1 Multi-dimensional Barcodes 6

4.2 Overloading Barcodes 8

4.3 Primer-Barcode Pairs 8

4.4 Overlapping Pools 8

5 Data Preparation 8

5.1 454 Barcode Libraries 8

5.1.1 Submitter Demultiplexing 8

2 Overview

This document reviews the features and submission requirements for SRA barcoded experiments and resulting sequencing data.

3 Use Cases

3.1 Default

The default SRA submission use case is for each experiment to have exactly one sample.

```
<EXPERIMENT>
  <DESIGN>
    <SAMPLE_DESCRIPTOR refname="Sample_1" refcenter="XYZ" />
```

where *Sample_1* is a SAMPLE record defined using the *SRA.sample.xsd* schema.

3.2 Sample Pool

In this use case one sequencing library is prepared from a pool of samples. The samples can be identified in the pool but the resulting sequencing data cannot distinguish the pool members except by secondary analysis such as alignment, which would occur outside of the SRA.

```
<EXPERIMENT>
  <DESIGN>
    <SAMPLE_DESCRIPTOR>
       <POOL>
          <MEMBER> member_name="BAC_1" accession="SRS000001"/>
          <MEMBER> member_name="BAC_2" accession="SRS000002"/>
           ...
```

Each member is defined by referencing its sample record. The member name has scope within the experiment and its child runs.

3.3 Sample Pool with Barcodes

A sample pool with barcodes is set up as follows:

1. Define the sample pool within the Sample Descriptor block. The SAMPLE_DESCRIPTOR attributes (accession or refname) can be used to define the default sample (the one that reads are assigned to if their barcode values cannot be decoded because of sequencing error or some other artifact).

```
<EXPERIMENT>
  <DESIGN>
    <SAMPLE_DESCRIPTOR refname="unassigned_bacs" refcenter="XYZ" >
       <POOL>
          <MEMBER> member_name="BAC_1" accession="SRS000001">
             <READ_LABEL read_group_tag="ACTGTT">barcode_tag</READ_LABEL>
           </MEMBER>
          <MEMBER> member_name="BAC_2" accession="SRS000002">
             <READ_LABEL read_group_tag="TAGTTG">barcode_tag</READ_LABEL>
           </MEMBER>
           ...
```

2. Next, define the SPOT_DESCRIPTOR to include a barcode tag as one of the “technical reads”. In this example, the barcode tag appears at the end of the read, and is decoded by substring matching.

```
<SPOT_DESCRIPTOR>
  <SPOT_DECODE_SPEC>
    <READ_SPEC>
      <READ_INDEX>0</READ_INDEX>
      <READ_CLASS>Application Read</READ_CLASS>
      <READ_TYPE>Forward</READ_TYPE>
      <BASE_COORD>1</BASE_COORD>
    </READ_SPEC>
    <READ_SPEC>
      <READ_INDEX>1</READ_INDEX>
      <READ_LABEL>barcode_tag</READ_LABEL>
      <READ_CLASS>Technical Read</READ_CLASS>
      <READ_TYPE>BarCode</READ_TYPE>
      <EXPECTED_BASECALL_TABLE
        ...
      />
  </SPOT_DECODE_SPEC>
</SPOT_DESCRIPTOR>
```

3. Next, define the lookup table that will associate a bar code pattern match with a member of the sample pool:

```
<EXPECTED_BASECALL_TABLE>
  <BASECALL read_group_tag="ACTGTT" min_match="6"
            max_mismatch="0" match_edge="full" >ACTGTT</BASECALL>
  <BASECALL read_group_tag="TAGTGG" min_match="6"
            max_mismatch="0" match_edge="full" >TAGTGG</BASECALL>
  ...
</EXPECTED_BASECALL_TABLE>
```

3.3.1 Sample pool with barcodes de-multiplexed by the submitter

The submitter takes care to split the reads within each run and reconstitute the submission container files in such a way that all the reads associated with a given member are contiguous and receive that member’s reference.

1. The SRA Run must be configured so that the SRA Loader will associate individual reads with members of the sample pool. This is done in the DATA_BLOCK/member attribute.

```
<DATA_BLOCK
  name = "FMSX0OV"
  region = "1"
  member_name = "BAC_1"
>
  <FILES>
    <FILE filename="BAC_1.sff"
          filetype="sff"
          checksum_method="MD5"
          checksum="4026fc6b91ed2ffbef374a665e02802b" />
          …
  </FILES>
</DATA_BLOCK>
<DATA_BLOCK
  name = "FMSX0OV"
  region = "1"
  member_name = "BAC_2"
>
 <FILES>
    <FILE filename="BAC_2.sff"
           filetype="sff"
           checksum_method="MD5"
           checksum="7f7ba170dbc6a25409a5eb6d845da88f" />
           …
  </FILES>
</DATA_BLOCK>
```

2. The EXPECTED_BASECALL_TABLE serves to document what was done in order to split up the run, but is not used to load the run. Barcode design is crucial to the success of the experiment and its processing, so users of the archive data may wish to repeat the de-multiplexing step. Therefore, indude the EXPECTED_BASECALL_TABLE in the experiment's SPOT_DESCRIPTOR.
3. An additional file containing auxiliary tag location information necessary to the loading of the data may be required. This tab delimited text file contains information about how to locate the barcode(s) and other technical tags within each raw spot sequence.

   Fields are:
   INSDC:read_name: String value used to join with native read name in run data file. For example, EQYRFS112HPIGW
   INSDC:read_seg: A vector of start-stop coordinates (basis 1, inclusive) that partitions a particular raw spot sequence among the tags defined for this run. The read_seg vector is expressed like this: [1-4],[5-12],[13-]. This tells the SRA loader that first tag has start coordinate 1 and end coordinate 4, and the tag starts at coordinate 13 and goes to the end of the raw spot sequence. The expression [0] indicates that the tag is not present in the sequence.

   Here are some example entries:

   ```
   INSDC:read_name      INSDC:read_seg
   EQYRFS112HPIGW       [1-4][5-12][13-]
   EQYRFS112HPIHA       [1-4][0][5-]
   EQYRFS112HPIMX       [1-4][0][5-]
   ```

   The FILE block must contain an additional entry that specifies the submission of the auxiliary read segments file:

```
<DATA_BLOCK
  name = "FMSX0OV"
  region = "1"
  member_name = "BAC_1"
>
  <FILES>
    <FILE filename="BAC_1.sff"
           filetype="sff"
           checksum_method="MD5"
           checksum="4026fc6b91ed2ffbef374a665e02802b"
    </FILE>
    <FILE filename="BAC_1.readseg.tab"
           filetype="tab"
```

```
           checksum_method="MD5"
           checksum="fc6b91ed2ffbef374a665e02802b4026"
           <DATA_SERIES_LABEL>INSDC:read_seg</DATA_SERIES_LABEL>
    </FILE>
  </FILES>
</DATA_BLOCK>
```

4. An addition file containing auxiliary clipping information necessary to the annotation of the data may be required for submitter-loads. This tab delimited text file contains information about how to locate the barcode(s) and other technical reads within each raw spot sequence.

   Fields are:
   INSDC:read_name: String value used to join with native read name in run data file. For example, EQYRFS112HPIGW
   INSDC:clip_quality_left: A coordinate (basis 1, inclusive) indicating the start of good quality biological sequence.
   INSDC:clip_quality_right: A coordinate (basis 1, inclusive) indicating the end of good quality biological sequence.

   Here are some example entries:

   ```
   INSDC:read_name      INSDC:clip_quality_left     INSDC:clip_quality_right
   EQYRFS112HPIGW       13                          278
   EQYRFS112HPIHA       13                          280
   EQYRFS112HPIMX       5                           277
   ```

   The FILE block must contain an additional entry that specifies the submission of the auxiliary clips file:

```
<DATA_BLOCK
  name = "FMSX0OV"
  region = "1"
  member_name = "BAC_1"
>
  <FILES>
    <FILE filename="BAC_1.sff"
           filetype="sff"
           checksum_method="MD5"
           checksum="4026fc6b91ed2ffbef374a665e02802b"
    </FILE>
    <FILE filename="BAC_1.clips.tab"
           filetype="tab"
           checksum_method="MD5"
           checksum="fc6b91ed2ffbef374a665e02802b4026"
           <DATA_SERIES_LABEL>INSDC:clip_quality_left</DATA_SERIES_LABEL>
           <DATA_SERIES_LABEL>INSDC:clip_quality_right</DATA_SERIES_LABEL>

    </FILE>
  </FILES>
</DATA_BLOCK>
```

5. Steps 3, 4 may be combined into one auxiliary data file.

3.3.2 Sample pool with barcodes de-multiplexed by the Archive

Submission is simpler in the case where the Archive is asked to perform the demultiplexing according to the instructions in the SPOT_DESCRIPTOR.

1. First, enter one file or set of files per data block. Do NOT use a member_name attribute:

```
<DATA_BLOCK
  name = "FMSX0OV"
  region = "1"
>
 <FILES>
    <FILE filename="all.bacs.sff"
           filetype="sff"
           checksum_method="MD5"
           checksum="7f7ba170dbc6a25409a5eb6d845da88f" />
  </FILES>
</DATA_BLOCK>
```

2. Create the EXPECTED_BASECALL_TABLE to tell the loader how to recognize each barcode tag and assign individual reads to a sample pool member.

3.4 Study Pool

The pooling of studies and experiments in a single run is not currently supported. The sequencing center is expected to de-multiplex the data for each study and return the appropriate subset to each investigator. From there it will be possible to make wholly distinct submissions. The SRA_LINK can be used to identify the relationship of several SRA runs to a single production run.

4 Features

4.1 Multi-dimensional Barcodes

More than one dimension of barcoding can be used with each tuple decoded to yield a member.

```
<EXPERIMENT>
  <DESIGN>
    <SAMPLE_DESCRIPTOR refname="unassigned_bacs" refcenter="XYZ" >
       <POOL>
          <MEMBER> member_name="site1_fraction1" accession="SRS000001">
             <READ_LABEL read_group_tag="site1">barcode_tag_a</READ_LABEL>
             <READ_LABEL read_group_tag="fraction1">barcode_tag_b</READ_LABEL>
           </MEMBER>
          <MEMBER> member_name="site1_fraction2" accession="SRS000002">
             <READ_LABEL read_group_tag="site1">barcode_tag_a</READ_LABEL>
             <READ_LABEL read_group_tag="fraction2">barcode_tag_b</READ_LABEL>
           </MEMBER>
          <MEMBER> member_name="site2_fraction1" accession="SRS000003">
             <READ_LABEL read_group_tag="site2">barcode_tag_a</READ_LABEL>
             <READ_LABEL read_group_tag="fraction1">barcode_tag_b</READ_LABEL>
           </MEMBER>
          <MEMBER> member_name="site2_fraction1" accession="SRS000004">
```

```
             <READ_LABEL read_group_tag="site2">barcode_tag_a</READ_LABEL>
             <READ_LABEL read_group_tag="fraction2">barcode_tag_b</READ_LABEL>
           </MEMBER>


           ...

      <SPOT_DESCRIPTOR>
        <SPOT_DECODE_SPEC>
          <READ_SPEC>
            <READ_INDEX>0</READ_INDEX>
            <READ_CLASS>Application Read</READ_CLASS>
            <READ_TYPE>Forward</READ_TYPE>
            <BASE_COORD>1</BASE_COORD>
          </READ_SPEC>
          <READ_SPEC>
            <READ_INDEX>1</READ_INDEX>
            <READ_LABEL>barcode_tag_a</READ_LABEL>
            <READ_CLASS>Technical Read</READ_CLASS>
            <READ_TYPE>BarCode</READ_TYPE>
            <EXPECTED_BASECALL_TABLE
              ...
            />
          <READ_SPEC>
            <READ_INDEX>2</READ_INDEX>
            <READ_LABEL>barcode_tag_b</READ_LABEL>
            <READ_CLASS>Technical Read</READ_CLASS>
            <READ_TYPE>BarCode</READ_TYPE>
            <EXPECTED_BASECALL_TABLE
              ...
            />

        </SPOT_DECODE_SPEC>
      </SPOT_DESCRIPTOR>
```

with the EXPECTED_BASECALL_TABLE for *barcode_tag_a* and for *barcode_tag_b* set up in the SPOT_DESCRIPTOR.

A toy example,

```
          <READ_SPEC>
            <READ_INDEX>1</READ_INDEX>
            <READ_LABEL>barcode_tag_a</READ_LABEL>
            <READ_CLASS>Technical Read</READ_CLASS>
            <READ_TYPE>BarCode</READ_TYPE>
            <EXPECTED_BASECALL_TABLE
<BASECALL read_group_tag="site1" min_match="4" max_mismatch="0" match_edge="full" >GCAT</BASECALL>
<BASECALL read_group_tag="site2" min_match="4" max_mismatch="0" match_edge="full" >CTTG</BASECALL>
            />
          <READ_SPEC>
            <READ_INDEX>2</READ_INDEX>
            <READ_LABEL>barcode_tag_b</READ_LABEL>
            <READ_CLASS>Technical Read</READ_CLASS>
            <READ_TYPE>BarCode</READ_TYPE>
            <EXPECTED_BASECALL_TABLE
<BASECALL read_group_tag="fraction1" min_match="4" max_mismatch="0" match_edge="full" >AACA</BASECALL>
<BASECALL read_group_tag="fraction2" min_match="4" max_mismatch="0" match_edge="full" >GTAT</BASECALL>
            />
```

4.2 Overloading Barcodes

More than one barcode can be used to resolve a particular member:

```
<EXPECTED_BASECALL_TABLE>
  <BASECALL read_group_tag="site_1" min_match="6"
            max_mismatch="0" match_edge="full" >ACTGTT</BASECALL>
  <BASECALL read_group_tag="site_1" min_match="6"
            max_mismatch="0" match_edge="full" >TAGTGG</BASECALL>
  ...
</EXPECTED_BASECALL_TABLE>
```

4.3 Primer-Barcode Pairs

Using the same mechanism of subsequence matching, a combination of primer and barcode tags can be defined for each spot. These definitions are made in the SPOT_DESCRIPTOR block. Then, a table of tag value pairs can be defined that maps each combination of primer and barcode to a particular sample pool member.

4.4 Overlapping Pools

Two sample pools may contain the same sample(s). Each sample pool will have one library constructed for it. A distinct SRA Experiment should be defined for each pool.

5 Data Preparation

5.1 454 Barcode Libraries

You should download the *sfftools* toolkit from Roche Diagnostics Corporation (license required). This will allow you to work with SFF files to dump their contents and the partition and recombine the files.

5.1.1 Submitter Demultiplexing

Using the utility *sfffinfo*, obtain the list of reads and their sequences from each plate's worth of run data.

```
sffinfo –s EQYRFS1.sff > EQYRFS1.fasta
```

Use a substring alignment program to match against a set of barcode sequences. One such program that works well for short, nearly exact matches is the MUMmer package (www.mummer.sourceforge.net). For example, these commands work well to find instances of barcode exact matches in the sequencing data.

```
nucmer –maxmatch –g 0 –c 12 –l 12 EQYRFS1reads.fasta barcodes.fasta –p EQYRFS1-barcodes
show-coords –cTH EQYRFS1-barcodes > EQYRFS1-barcodes.coords
```

The latter file can be used to generate the set of individual read-barcode hits as well as the auxiliary readseg tab file.

```
grep barcode01 EQYRFS1-barcodes.coords | awk '{ print $(NF-1) }' > barcode01.seqs
```

taking care to identify 100% matches, eliminate duplicate hits, and choose between multiple barcode hits.

Finally, each of the hit files can be applied to the master SFF file to extract the subset of records associated with a certain bar code.

```
sfffile –i barcode01.seqs –o EQYRFS1-barcode01 EQYRFS1.sff
…
```

The command *sfffile –t <filename>* can also be used to reset the quality_clip_left parameter. If only one mapping tag (barcode/primer etc) is being used, then this is sufficient for the loader to recognize the barcode tag boundaries and the auxiliary readseg tab file is not needed.