Short-Read Archive (SRA) Submission Tutorial:
Analyzing microbial communities using high-throughput 16S rRNA sequencing data.

J. Gregory Caporaso[1*], Justin Kuczynski[2*], Jesse Stombaugh[1*], Kyle Bittinger[3], Frederic D. Bushman[3], Elizabeth K. Costello[1], Noah Fierer[4], Antonio Gonzalez Peña[5], Julia K. Goodrich[5], Jeff I. Gordon[6], Gavin Huttley[7], Scott T. Kelley[8], Dan Knights[5], Jeremy E. Koenig[9], Ruth E. Ley[9], Cathy A. Lozupone[1], Daniel McDonald[1], Brian D. Muegge[6], Megan Pirrung[1], Jens Reeder[1], Joel R. Sevinsky[10], Peter J. Turnbaugh[6], Will Van Treuren[1], William A. Walters[2], Jeremy Widmann[1], Tanya Yatsunenko[6], Jesse Zaneveld[2] and Rob Knight[1,11**]

1. Department of Chemistry and Biochemistry, UCB 215, University of Colorado, Boulder, CO 80309
2. Department of Molecular, Cellular and Developmental Biology, UCB 347, University of Colorado, Boulder, CO 80309
3. Department of Microbiology, Johnson Pavilion 425, University of Pennsylvania, Philadelphia, PA 19104
4. Cooperative Institute for Research in Environmental Sciences, University of Colorado, Boulder, CO 80309, USA.; Department of Ecology and Evolutionary Biology, University of Colorado, Boulder, CO 80309, USA.
5. Department of Computer Science, University of Colorado, Boulder, Colorado, USA.
6. Center for Genome Sciences, Washington University School of Medicine, St. Louis, MO 63108
7. Computational Genomics Laboratory, John Curtin School of Medical Research, The Australian National University, Canberra, Australian Capital Territory, Australia.
8. Department of Biology, San Diego State University, San Diego CA 92182
9. Department of Microbiology, Cornell University, Ithaca NY 14853
10. Luca Technologies, 500 Corporate Circle, Suite C, Golden, Colorado 80401
11. Howard Hughes Medical Institute

Table of Contents

Introduction

This Tutorial provides an overview of the SRA submission preparation procedure, using as an example the data from the Fierer et al. 2008 hand dataset. The SRA submission example shown here, is from the first barcoded SRA submission, accession #: SRS001216. It outlines the same two-stage procedure we will use for submitting the Human Microbiome Project (HMP) data to SRA. This draft is intended to be passed on to the Data Analysis and Coordination Center (DACC) for further editing and revision with guidance from the centers.

The data shown in this tutorial can be downloaded from: http://tajmahal.colorado.edu/tmp/knight_handstudy_demo.zip

Philosophy of this Document and the Accompanying Code

This document and code does not seek to capture the full complexity of what is possible in the SRA. Instead, the goal is to provide a simple pathway for submission to the SRA of the most common types of data. For example, more normalization could be provided through requiring additional tables, but this additional complexity is not justified when the appropriate rows can be copied/pasted/updated in Excel in a few seconds. More flexibility and better normalization is certainly possible and may be supported in future: the focus here is on getting something that works now.

Originally, the plan was to do this as a series of standalone scripts, but this ended up with an unacceptable level of reimplementation of things that are already in QIIME ("Quantitative Insights Into Microbial Ecology", our 454 analysis package). We have therefore contributed the code into QIIME. You can get this package from sourceforge here:

http://sourceforge.net/projects/qiime/

You will need several other packages to complete this analysis:

- BLAST, http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download
- cd-hit, http://www.bioinformatics.org/cd-hit/
- pycogent, http://sourceforge.net/projects/pycogent/
- numpy, http://numpy.scipy.org/
- matplotlib (required for pycogent but not used here), http://matplotlib.sourceforge.net/
- python, http://www.python.org/download/

The pycogent install guide is fairly helpful in showing you the dependencies and how to get them; the QIIME install guide has some information about installing the 3rd-party applications it wraps. Note that some of the unit tests will fail if you don't install all the applications but the subset of the analysis shown here should work (e.g. you won't be able to do sequence alignment or taxonomy assignment without additional work).

Overview of the Submission Process

An SRA submission can consist of metadata only, or of metadata together with sequence data. Martin Shumway and Chris O'Sullivan have recently added support for submission of barcoded pyrosequencing runs. This document describes how to prepare such submissions in a two-stage process:

1. Submission of the study and sample metadata, including clinical metadata (this will be performed by the DACC).
2. Submission of the experiment and run data, and associated technical metadata (this is expected to be performed by the DACC for the pilot, although there is no technical barrier to having it performed by the Centers). It is, at this point, expected that the Centers will submit their own data after the pilot phase, although there are also arguments for submission through the DACC for consistency of QA processes.

Updates are handled by regenerating the submission xml files (and optionally the .sff files), at which point they will be reloaded into the SRA. The current pipeline does not provide a mechanism for doing partial updates (e.g. adding an experiment or a run to an existing submission) -- all the metadata and/or data for the submission must be replaced with clean files. It may be possible in the future to allow incremental submissions but this is not presently supported.

Questions about the Submission Process

Q1. *Can I submit multiplexed pyrosequnecing runs now?*

A1. Yes.

Q2. *Can I combine mock and clinical samples on the same 454 plate? (or, more generally, can I combine samples from different studies on the same 454 plate?)*

A2. Yes, but you must specify in the library.txt input file which samples go with which study.

Q3. *Can I associate the same sample (and thus reads) with more than one study?*

A3. No.

Q4. *Can I combine samples that use different primers on the same run?*

A4. Yes, but you must specify in the library.txt input file which primer was used for each "member" of the pooled library.

Q5. *Who will submit what?*

A5. At this stage, we expect the DACC to submit both the sample/study metadata and the experiment/library metadata and sequence data for the pilot. Later, the Centers will have the capacity to submit their own data. Centers will be credited with their data appropriately regardless of the mechanism by which the submission is actually performed. The submission will be a two-stage process: (1) the creation of study and sample records by the DACC, (2) the submission of sequence data and associated metadata by the DACC and/ or the Centers.

Q6. *Can I associate the same sample with more than one barcode and/or primer?*

A6. Yes, but you must specify a unique identifier for each "member" of the pool that associates the sample, primer and barcode.

Q7. *What is the distinction between a STUDY, an EXPERIMENT, and a RUN?*

A7. As SRA uses the terms, a STUDY is a collection of EXPERIMENTS. An EXPERIMENT is a LIBRARY (potentially a library of many samples that form a POOL, if multiplexing was used -- each MEMBER of a pool is associated with a sample, a primer, and a barcode) that was sequenced using one or more instrument runs. A RUN is the sequencing of a particular MEMBER of a pooled library on a particular instrument at a particular time. Thus, a single instrument run gives rise to many RUN entries in SRA.

Q8. *Is there an intermediate level between STUDY and EXPERIMENT?*

A8. Not for practical purposes. SRA will eventually allow a hierarchy of STUDY entries but this is not yet implemented.

Q9. *Do I really have to make a separate sff file for every MEMBER of every POOL for every instrument run?*

A9. Yes, and you also have to reset the quality trimming to correspond to the primer that was used for that particular member. The SRA will, in future, provide the demultiplexing service, but for now requires that the submissions be demultiplexed in advance. Fortunately, the accompanying scripts assist with this process.

Q10. *Is it OK for primers to be different lengths on the same 454 run?*

A10. Yes, but not within the same MEMBER of a library (i.e. if you have primers of different lengths, the different lengths are considered different MEMBER entries and should be marked as such in library.txt).

Q11. *How should degenerate primers be handled?*

A11. All possible sequences that match the degenerate primer should be allowed using the EXPECTED_BASECALL_TABLE mechanism in experiment.xml (see example).

What files and information are needed to prepare a submission?

A submission consists of a submission.xml, metadata file, which references other xml metadata files and optionally tarballs of sequence data files, as follows:

1. Submission of Study and Sample Metadata.

Input:

a. study.txt - tabular metadata about the study (this is used to accession the study).
b. sample.txt - tabular metadata about each sample (this is used to accession samples).
c. study_template.xml - xml tempate for study data (located in Qiime/tests/sra_xml_templates/ directory)
d. sample_template.xml - xml template for sample data (located in Qiime/tests/sra_xml_templates/ directory)
e. submission_template.xml - xml template for submission (located in Qiime/tests/sra_xml_templates/ directory)

Output:

a. study.xml - xml-format metadata about the study.
b. sample.xml - xml-format metadata about each sample
c. submission.xml - xml-format metadata about the study and sample submission

NOTE: There will be two STUDY entries associated with the pilot: HMP_PILOT_CLINICAL for the clinical data, and HMP_PILOT_MOCK for the mock community data. There will be additional STUDY entries associated with the demonstration projects, in the namespace HMP_DEMO_X where X is the name of the demonstration project (the DACC will help demonstration projects submit their data using the process described in this document, and/or provide the relevant scripts to the demonstration projects). The SRA STUDY concept is intended to map more or less onto a paper, but the SRA EXPERIMENT concept maps onto a library: there will in the future be a way of associating SRA STUDY entries with each other in a hierarchical way but it does not yet exist. The initial mechanism is that SRA will provide a web page indexing all the HMP STUDY entries as a portal.

NOTE: Martin stresses that it is VERY IMPORTANT that the centers do not make up new accessions or include samples that have not been accessioned by the DACC in their HMP submissions. This will cause load problems, failed submissions, etc. The recommendation is to separate out the HMP from the non-HMP data when a run contains both, and to do independent submissions (they will be matched up to the same run via the run name attribute in SRA).

Example:

```
$ python ~/Qiime/qiime/make_sra_submission.py -a sample.txt -t study.txt -u submission.txt -T
        ~/Qiime/tests/sra_xml_templates/study_template.xml -A
        ~/Qiime/tests/sra_xml_templates/sample_template.xml -U
        ~/Qiime/tests/sra_xml_templates/submission_template.xml
```

Produces sample.xml, study.xml, submission.xml from the tab-delimited text files.

2. Submission of Experiment and Run Metadata.

Input:

a. experiment.txt - tabular metadata about the contents of each combination of library and sff file.
b. data - directory of multiple sff files containing the actual sequence data

Output:

a. experiment.xml - xml-format metadata about the set of experiments described in library.txt
b. run.xml - xml-format metadata about each run, i.e. the association betweena specific member of a pool and a specific xml file.
c. data.tgz - a gzipped tar archive containing individual sff files for each SRA RUN (see the Questions above if you are unclear on the distinction between the SRA RUN concept and the concept of an instrument run).

Example:

Step 1: Get fasta and qual from sff files

This step converts the sff files into text formats that are more usable. Note that in this example the .fna and .qual files are already in there to eliminate the requirement for the off-machine apps, so they will simply be overwritten with identical files

by this script. If you do not have these apps, please skip to the Step 2.

```
$ python ~/Qiime/qiime/process_sff.py sff_files/
```

Output: makes .fna and .qual files for each sff file.

Step 2: Produce valid mapping file for library demultiplexing

This step converts the input experiment file into separate mapping files for each combination of STUDY and RUN_PREFIX (separating by run prefix is necessary when the same barcodes are used in different runs). This allows demultiplexing of the separate studies, which will then be sent in as separate submissions, and of the different barcoded plates, which will be demultiplexed separately.

Note: the LINKER field is no longer required in the spreadsheet.

```
$ python ~/Qiime/qiime/sra_spreadsheet_to_map_files.py experiment.txt
```

Output: produces valid mapping files per 454 plate: fierer_hand_study_E86FECS.map fierer_hand_study_FA6P1OK.map

Step 3: Demultiplex libraries

This step assigns each sequence to a library, dropping low-quality sequences and producing a log explaiing why specific sequences were dropped.

NOTE: The SRA requests that you deposit ALL your sequence data, including bad reads, unless there is an IRB reason not to do so (i.e. human contamination). Therefore the quality and length filtering should be turned off.

```
$ python ~/Qiime/qiime/split_libraries.py -h
```

Output: shows you the help for split_libaries.py

```
$ python ~/Qiime/qiime/split_libraries.py -s 5 -l 30 -L 1000 -b 12 -H 1000 -M 100 -a 1000 -f
        sff_files/E86FECS01.fna,sff_files/E86FECS02.fna -q
        sff_files/E86FECS01.qual,sff_files/E86FECS02.qual -m fierer_hand_study_E86FECS.map -o
        E86FECS_demultiplex

$ python ~/Qiime/qiime/split_libraries.py -s 5 -l 50 -L 1000 -b 12 -H 1000 -M 100 -a 1000 -f
        sff_files/FA6P1OK01.fna,sff_files/FA6P1OK02.fna -q
        sff_files/FA6P1OK01.qual,sff_files/FA6P1OK02.qual -m fierer_hand_study_FA6P1OK.map -o
        FA6P1OK_demultiplex -r
```

Output: produces two files: seqs.fna with valid sequences assigned to samples via barcodes, and split_libraries_log.txt with info about which sequences failed QC. The parameters above are essentially turning off the default quality filters and require an average qual score of at least 5, a minimum sequence length of 30 (basically just the primer_barcode), a maximum sequence length of 1000, max homopolymer run of 1000, up to 100 errors in the primer, etc. to let everything through, and specify that we are using 12-base barcodes (turning off the Golay error-correction, which would be specified with -b golay_12), specify the (comma-delimited) paths to the fasta and mapping files (note: no spaces are allowed around the commas), and finally specify the mapping file as one of the map files we produced in step 2 (taking care to use the right map file for each run). Note: you can turn off the quality filtering steps if you want to make sure that all the sequences appear in the output. The -r True flag removes unassigned sequences from the fasta file and, if added, will make the analysis run substantially faster. In this case we use -r on the second run but not on the first run because all the reads on the first run were from this study, but only some of the reads from the second run were from this study, and we can't tell a valid read from another study apart from a bad read from this one.

Step 4: reduce sequence complexity by picking OTUs with cd-hit

This step reduces the number of sequences to do the human screen by picking OTUs at 95%. We make the simplifying assumption that sequences that are identical over the first 100 bases will fall into the same OTU.

Note: this step requires that you have cd-hit installed.

$ python ~/Qiime/qiime/pick_otus.py -M 4000 -n 100 -s 0.95 -o E86FECS_demultiplex -i E86FECS_demultiplex/seqs.fna

Produces two files: E86FECS_demultiplex/seqs_otus.txt and E86FECS_demultiplex/seqs_otus.log (which have the OTUs and the log file describing the analysis respectively).

$ python ~/Qiime/qiime/pick_otus.py -M 4000 -n 100 -s 0.95 -o FA6P1OK_demultiplex/ -i FA6P1OK_demultiplex/seqs.fna

Repeat the same procedure for the other library.

Step 5: pick a representative sequence for each OTU

This step gets the actual sequences for each OTU picked in Step 4.

$ python ~/Qiime/qiime/pick_rep_set.py -i E86FECS_demultiplex/seqs_otus.txt -f E86FECS_demultiplex/seqs.fna

Produces E86FECS_demultiplex/seqs.fna_rep_set.fasta

$ python ~/Qiime/qiime/pick_rep_set.py -i FA6P1OK_demultiplex/seqs_otus.txt -f FA6P1OK_demultiplex/seqs.fna

Produces FA6P1OK_demultiplex/seqs.fna_rep_set.fasta

Step 6: blast the representative set sequences against 95% OTUs in greengenes to eliminate sequences that aren't really 16S rRNA

This step performs a human/contaminant screen the "safe" way by identifying and excluding sequences that aren't 16S rRNA.

$ python ~/Qiime/qiime/exclude_seqs_by_blast.py -d greengenes_unaligned.fasta-OTUs_at_0.05.fasta -i E86FECS_demultiplex/seqs.fna_rep_set.fasta -W 10 -p 0.25 -o E86FECS_demultiplex/blast_results -e 1e-20

We are using blastn with a word size of 10, requiring 25% coverage of the sequence, and an E-value of 1e-20. Our tests suggest that this is sufficient to screen out human genomic reads (the human 18S sequence hits bacterial 16S with E-value between 1e-18 and 1e-10 depending on lineage).

This produces a bunch of log files and output; the file of screened seqs (i.e. that failed to hit a known 16S rRNA with even relaxed criteria)

Repeat the same procedure for the other library:

$ python ~/Qiime/qiime/exclude_seqs_by_blast.py -d greengenes_unaligned.fasta-OTUs_at_0.05.fasta -i FA6P1OK_demultiplex/seqs.fna_rep_set.fasta -W 10 -p 0.25 -o FA6P1OK_demultiplex/blast_results -e 1e-20

Step 7: make per-library files of "good" ids to pass to sfffile

This step maps the ids of the representative set back onto the ids of the OTUs they came from so that we can get all the members of the OTUs that had a representative that matched a known 16S rRNA.

$ python ~/Qiime/qiime/make_library_id_lists.py -i E86FECS_demultiplex/seqs.fna -s E86FECS_demultiplex/blast_results.screened -u E86FECS_demultiplex/seqs_otus.txt -o E86FECS_demultiplex/per_lib_info

This makes a new directory called E86FECS_demultiplex/per_lib_idlists, which contains a separate file with an id list for each library.

$ python ~/Qiime/qiime/make_library_id_lists.py -i FA6P1OK_demultiplex/seqs.fna -s FA6P1OK_demultiplex/blast_results.screened -u FA6P1OK_demultiplex/seqs_otus.txt -o

FA6P1OK_demultiplex/per_lib_info

This makes a new directory called FA6P1OK_demultiplex/per_lib_idlists, which contains a separate file with an id list for each library.

Step 8: use sfffile to make per-library sff files

This step takes the good lists of ids from step 7 and extracts a separate sff file for each of those lists.

```
$ python ~/Qiime/qiime/make_per_library_sff.py -i sff_files/E86FECS01.sff,sff_files/E86FECS02.sff -l
    E86FECS_demultiplex/per_lib_info/

$ python ~/Qiime/qiime/make_per_library_sff.py -i sff_files/FA6P1OK01.sff,sff_files/FA6P1OK02.sff -l
    FA6P1OK_demultiplex/per_lib_info/
```

Step 9: use sfffile to quality-trim the barcodes, primers and linkers

The SRA requires that the user reset the left trim in the sff file to eliminate the technical reads (barcode, primer, linker if present). This means figuring out the length of the technical parts of the read, the length of the current read, writing out a text file with the per-id info, and running sfffile to reset the read lengths.

```
$ python ~/Qiime/qiime/trim_sff_primers.py -m fierer_hand_study_E86FECS.map -l
    E86FECS_demultiplex/per_lib_info/

$ python ~/Qiime/qiime/trim_sff_primers.py -m fierer_hand_study_FA6P1OK.map -l
    FA6P1OK_demultiplex/per_lib_info/
```

Step 10: move files around and make archive, which will be automated in the future releases.

```
mkdir per_run_sff
mkdir per_run_sff/FA6P1OK
mkdir per_run_sff/E86FECS
cp FA6P1OK_demultiplex/per_lib_info/*.sff per_run_sff/FA6P1OK/
cp E86FECS_demultiplex/per_lib_info/*.sff per_run_sff/E86FECS/
mv per_run_sff/E86FECS/Unassigned.sff per_run_sff/E86FECS/fierer_hand_study_default_E86FECS.sff
mv per_run_sff/FA6P1OK/Unassigned.sff per_run_sff/FA6P1OK/fierer_hand_study_default_FA6P1OK.sff
tar cvfz all_hand_sff_data.tgz per_run_sff/
```

Step 11: finally make the second-stage submission!

```
$ python ~/Qiime/qiime/make_sra_submission.py -u submission_second_stage.txt -e experiment.txt -s
    per_run_sff -T ~/Qiime/tests/sra_xml_templates/study_template.xml -A
    ~/Qiime/tests/sra_xml_templates/sample_template.xml -U
    ~/Qiime/tests/sra_xml_templates/submission_template.xml
```

produces files:
experiment.xml
run.xml
submission_second_stage.xml

...and the process is complete.

Note: SRA prefers you give the individual files more meaningful names than the defaults, so suggest not just using generic names like experiment etc.