**Changes to incorporate in documentation/scripts:**

* Add support for compressed (.gz-formatted) sequencing results
* Explicitly describe protocol for sharing Google spreadsheet to ensure users don't have problems accessing their records from the pipeline
* Mapping file must have 3 columns (scripts should verify on startup)
* Update genome names in genomecodes.pm; need better way to maintain master list so that it's easy to update a central repository; long-term: write/modify subroutines for pulling down references from NCBI by accession # (best strategy, but non-trivial to implement)
* Add support for specifying different sets of genomes for different pools from the same analysis group. All pools within a group will currently be aligned against the set of all genomes found across all groups. This is non-obvious in the documentation.
* No-match filtering should be heavily reworked:
  + Make this step optional for those who don't care what non-matching sequences align to (speeds up workflow substantially)
  + Allow for customizable NM reference genomes; the ones I typically use will not be relevant to many users.
  + Illumina adapter reference sequence(s) in NM set need updating

**Running the COPRO-Seq Pipeline**

(Amendments in progress; complete up to IGS table section)

1. ***Checking out the software (via GitHub)***
   1. ***Obtaining a copy of the mouse reference genome***
2. ***Preparing the inputs***
   1. ***Google spreadsheet***
   2. ***Mapping file***
   3. ***Pre-calculated IGS table***
   4. ***References not available in microbialomics***
3. ***Initiating the analysis (batch\_coproseq.pl)***
4. ***Retrieving your sequencing results (getdata.sh)***
5. ***Calculating your IGS table (calcIGS.sh)***
6. ***Submitting your jobs (align.jobs)***
7. ***Aggregating your results (summarize.sh)***
8. ***Cleaning up (cleanup.sh)***
9. ***Interpreting the output files***
10. ***Troubleshooting***

***I. Checking out the software (via GitHub)***

The easiest way to obtain or update the files you’ll need to analyze COPRO-Seq data is to fetch them from the public "COPRO-Seq" GitHub repository. The commands below should be executed from within the directory where you want to install the pipeline. Please verify you do not have a pre-existing folder called "COPRO-Seq" in this folder before carrying out the steps below (I'm not sure if the git utility will overwrite a pre-existing folder's contents or not, but best not to chance it).

Note: In order to fetch the COPRO-Seq repository using the commands below, you'll need to have the 'git' utility installed on your system. This utility is available on the cluster already, but probably won't be available on your local computer by default. It can be downloaded (free of charge) at:

http://git-scm.com/downloads

*If you’ve never checked out the COPRO-Seq repository before, type:*

git clone https://github.com/nmcnulty/COPRO-Seq.git

After the download has completed, you should have a folder in your home directory named “COPRO-Seq”. This folder will contain (almost) all of the scripts, perl modules and support files that you’ll need to analyze your data.

***Obtaining a copy of the mouse reference genome***

In its current implementation, the COPRO-Seq workflow requires access to a copy of the mouse genome. Unfortunately, the mouse genome is large, and a file containing this information exceeds GitHub's limits on supported file sizes. It's therefore necessary to create a copy of the mouse reference files in a specific location after you've cloned the COPRO-Seq repository. These files are made available on the cluster for easy access. Once you've cloned the repository, just execute the following command:

cp /home/comp/jglab/nmcnulty/COPRO-Seq/filteringrefs/mouse/\* ~/COPRO-Seq/filteringrefs/mouse

If you're accessing these files from outside the Gordon Lab cluster environment, use the "get" or "wget" utility to pull them from a publicly available folder:

cgs.wustl.edu/~nmcnulty/mouse\_ref/

<what utility to use to get these files?>

Note: this transfer will take some time (probably at least a minute), given that the files being moved are quite large.

***II. Preparing the inputs***

* Google spreadsheet formatted for COPRO-seq records
* Mapping files specifying barcode assignments
* (optional) Previously calculated IGS table
* (optional) Fasta-formatted sequence files for reference genomes not available in the microbialomics database you’re using (optional)

***Google spreadsheet***

* The COPRO-Seq pipeline scripts will obtain most of the information they require from a Google spreadsheet in which you’ll maintain records of your sequencing runs.
* Navigate to the 'COPRO-Seq Template' using this URL:

https://docs.google.com/spreadsheet/ccc?key=0AhsSO\_Vep9tqdFlkSjBFWktTajl6R2hBcjJ0bDE2aGc#gid=0

* After opening the template go to “File -> Make a Copy” and rename it with your own preferred name (this will make it available in your Google Docs home page).
* The instructions for filling out each field of the spreadsheet can be found in the “Instructions” tab of the template.
* **Important: You must make your spreadsheet available to anyone with its associated URL before proceeding. In the next window, under 'Who has access', click 'Change...' and select the 'Anyone with the link' radio button.**
* You’ll probably also find the Google spreadsheet “Genomes Key (COPRO-Seq)” useful when specifying which genomes to include in your analysis. That document is located here:

http://spreadsheets.google.com/ccc?key=0AhsSO\_Vep9tqdHNLLUs2WG84Nl9PZVNxS3hFR3BVTUE&hl=en

* + Note that the microbialomics database from which the accession numbers are derived is specified on the spreadsheet tab (i.e. '\_v1\_Build2\_dannonAll' would be the genomes key for the 'microbialomics\_v1\_Build2\_dannonAll' database).
  + The 'microbialomics\_v1\_Build2\_dannonAll' database is a good database to start with if you don't have a customized database (it has most/all of the genomes you're likely to need).
  + This spreadsheet is updated periodically as lab members’ needs change. If you find that a species you work with isn’t on the list or doesn’t have a shortcut abbreviation, talk to Nate.

***Mapping file***

* All of the pools you wish to include in a particular analysis must be described in a single “mapping file”. This file is headerless and contains one row for each sample sequenced. Each of these rows contains the following information separated by tabs:

Sequencing pool name (e.g. “NM1200\_1”)

* + Try to stick to letters, numbers, hyphens and underscores for your pool names

Sample name

* + All sample names must be unique (even across different pools) to avoid ambiguities during the final data summarization steps
  + Tip: If your samples are numbered you may find sorting your results later on to be easier if you use leading zeros (e.g. if your samples are 1-1, 5-10, and 15-5, consider changing them to 01-01, 05-10, and 15-05)

Barcode

* + Barcodes can be literal barcode sequences (e.g. AATC, GAGA, etc.) or one of the supported barcode abbreviations
    - There are now multiple sets of barcoded adapters. As of right now, 3 sets are supported based on information in the 'barcodes.pm' module that comes with the COPRO-Seq workflow.
    - Barcode abbreviations are formatted as either XBCY (for single-read adapters), or PEXBCY (for paired-end adapters), where X is the # of base pairs in the barcode and Y is the lab-adopted number used to designate that barcode (e.g. PE8BC1, 4BC1, 8BC1, etc.).
  + Mixtures of abbreviations and literal sequences in the same mapping file are permitted
  + Only one instance of a barcode in a given pool is permitted

*Note: Things get a bit tricky if you sequence the same pool multiple times. It’s recommended that in such situations you provide a line in the mapping file for each instance in which a pool/sample was sequenced, appending some kind of suffix to the pool and sample names to help distinguish them from one another (use a consistent nomenclature to avoid confusion, e.g. append –r1, -r2, -r3, …, or alternatively, append the run/lane designation for samples run multiple times).*

For example:

exp1\_pool1 sample1 4BC1

exp1\_pool1 sample2 4BC2

exp1\_pool1 sample3 4BC3

exp1\_pool2-r1 sample4-r1 4BC1

exp1\_pool2-r1 sample5-r1 4BC2

exp1\_pool2-r1 sample6-r1 4BC3

exp1\_pool2-r2 sample4-r2 4BC1

exp1\_pool2-r2 sample5-r2 4BC2

exp1\_pool2-r2 sample6-r2 4BC3

*…*

*Alternatively, you may describe each pool just once in your mapping file if you never combine replicate sequencing runs of the same pool in an analysis group on the Google spreadsheet.*

* Your mapping file can include samples you will not be looking at in your analysis, so feel free to keep one master mapping file for each of your experiments (or even one master mapping file for multiple experiments if there will be no overlap in the sample names)
* It’s usually easiest to create the mapping file in Excel and then save it as a tab-separated text file. Just be careful not to introduce hidden characters (software such as Excel for Mac is notoriously bad about this). To verify your mapping file looks okay you can open it from the command line with less:

less <mapping file name>

* + If you see any instances of “^M” or your entries fail to show up on separate lines, you’ll need to open your mapping file in a robust text editor (like TextWrangler for Mac OS X or EditPlus++ for Windows) and save your file as tab-delimited in Unix format.

***Pre-calculated IGS table (optional)***

The COPRO-seq pipeline requires an IGS table in its later stages for the data normalization process. The preparation of this table as one of the steps in the workflow (described later) is made fairly painless for the user, but it is one of the most time-consuming steps. If you’ve run the COPRO-seq scripts on data before using the exact same genomes (no more, no less) and the same read length you can recycle your previously-calculated IGS table to save time. This can be very useful if you are processing your samples in batches or if you often work with the same collection of genomes.

Note that the pipeline will only generate the minimum IGS table needed for the analysis you are requesting (i.e. it will not calculate IGSs for every possible read length, but only the read length you are using). But, sometimes it’s informative to compare COPRO-seq results generated from several different read lengths. For example, you will likely find that using a longer read length will generate fewer false positive hits (i.e. hits to genomes that aren’t actually present), but greater numbers of no-match reads that don’t seem to hit any of your references (presumably due to errors at later bases in the read).

If you want to perform such an analysis, or if you often work with a very specific set of genomes in your experiments, you may find it useful to calculate one ‘master IGS table’ for those genomes that summarizes IGSs for all your genomes at a wide range of relevant k-mer lengths. The steps to calculate such a table are as follows (please run these commands from a sandbox):

1. Create a temporary directory with any name of your choosing.
2. Copy the nucleotide FASTA-formatted genome files for all species you wish to include in the table to this temporary directory. Note that the FASTA files must be single-entry files (i.e. draft genomes must be collapsed to a single contig/entry).
3. Copy the IGScalc.pl script in your ~/coproseq/ directory to this temp directory
4. From within the temporary diretory run:

*perl IGScalc.pl –f . –p –c*

Note: you can also add a –k option to specify the k-mer read lengths to utilize (multiple k-mer lengths should be comma-separated with no space; e.g. –k 15,20,25). By default IGScalc.pl will calculate IGSs for all k-mers from 10-32bp.

1. Two outputs will be generated:

*IGS.table* - this is the IGS table you can optionally pass to the COPRO-seq pipeline

*genome\_uniqueness.table* – describes the percentage of k-mers from each genome that were unique among all genomes analyzed

***Including reference genomes not available in microbialomics (optional)***

In some cases, you may want to include a genome in your analysis that isn’t available in your local microbialomics database. This isn’t a problem at all. In fact, if you want, you can rely exclusively on user-supplied genomes for your alignments and leave the microbialomics server out of the equation entirely. To include reference genomes that are not present in microbialomics:

1. Get a FASTA-formatted nucleotide file for each genome of interest. Each FASTA file must contain a *single* entry. In the case of draft genomes comprising many contigs, you'll need to concatenate the contigs together.
2. Open up your Google analysis spreadsheet and record the full path of each additional reference genome in the "External genome paths" field for the pool being analyzed. In cases where you're using multiple additional reference files, the paths should be comma-separated.
3. Invoke batch\_coproseq.pl as you normally would for an analysis involving only microbialomics-derived references.

***III. Initiating the analysis (batch\_coproseq.pl)***

The pipeline is initiated by invoking the “batch\_coproseq.pl” script which will set up most of the files/folders needed for subsequent steps to be run. Briefly, the script pulls down the experiment/sample information you’ve stored in your Google spreadsheet (using its unique Google key), pulls down reference genomes from microbialomics (when available), “squashes” the references into the format required for the ELAND aligner, and creates a series of .sh files that you will run in sequence to carry out the COPRO-seq analysis.

1. Navigate to ~/coproseq/
2. Create a directory for your experiment
3. Navigate to within the new directory.
4. Run (from either the frontend or a sandbox):

perl ../batch\_coproseq.pl –g *<group code from Google spreadsheet>* -k *<Google spreadsheet key>* -m *<mapping file>*

Additionally, any of the following options can be used if you wish to deviate from defaults:

–e : number of errors (mismatches) to allow when aligning reads to references

–i : location of pre-calculated IGS table (if available)

–l : length of read (after trimming barcode) to use in alignment

–o : store output in a directory other than the current directory

Other options that are available (but you probably won’t need) include:

–a : leave all intermediate files generated during the

–p : page # where data is stored in your Google spreadsheet if moved from the first page (this is usually not needed)

Tip: If you keep most/all of your COPRO-seq experiment/sample records in the same Google spreadsheet, it’s usually easier to modify one line in the batch\_coproseq.pl script to store a new default for the spreadsheet key (so that you don’t have to pass the complicated key to the script each time).

Change: my $google\_key = "0AhsSO\_Vep9tqdDlzVXlJSWRPUVgxNWxkVVFJOE4tQ0E";

To: my $google\_key = “*your key here*”;

Note: it's often a little confusing trying to figure out which part of the Google URL for your spreadsheet defines the key. Here is an example of a URL with the key highlighted in bold (note that everything after the first ampersand has been excluded):

https://spreadsheets.google.com/ccc?key=**0AhsSO\_Vep9tqdDlzVXlJSWRPUVgxNWxkVVFJOE4tQ0E**&hl=en#gid=0

Several new files and a directory should appear in your folder:

* One or more “.bc” files
* “align.jobs”
* “cleanup.sh”
* Directory “genomes” containing:
  + One or more FASTA files, one per genome
  + A “squashed” directory, containing:
    - One or more .vld and .2bpb files, one per genome
* “getdata.sh”
* “project.info”
* “summarize.sh”
* “calcIGS.sh” (this file will not be created if you invoke the –i option above)

Note: a tracking module has recently been added to the start of the first script in the workflow to alert you to whether or not this script is up-to-date. If you receive a warning that your scripts are out of date, or if you want to verify that all the components of the COPRO-Seq pipeline are current, type (from within your "COPRO-Seq" folder):

git pull

***IV. Retrieving your sequencing results (getdata.sh)***

Next, create symbolic links to the original sequencing files so that the COPRO-seq pipeline knows from where to grab the read data (think of this as being analogous to downloading your data). To do so, type the following command from within your experiment folder:

sh getdata.sh

One .scarf file will be created in your experiment folder for each lane/run specified in your original Google spreadsheet.

***V. Calculating your IGS table (calcIGS.sh)***

Next, generate your IGS table. To do so, log into one of the sandboxes and type the following command from within your experiment folder:

sh calcIGS.sh

Note: this step is not required if you passed an IGS table to batch\_coproseq.pl using the –i option. No calcIGS.sh file will exist in such a case.

Once complete, a new folder “IGS” should be present in your “genomes” folder. It will contain a IGS.table and a genome\_uniqueness.table file. The IGS.table file can be saved and used in later analyses if you wish (see section “IGS table” in section (II) for details). Be sure to copy this table to a new location before running the cleanup.sh script in section (VIII) if you wish to keep it.

***VI. Submitting your jobs (align.jobs)***

Next, submit your sequences to the cluster for COPRO-seq analysis. To do so, just type the following command from within your experiment folder:

nq align.jobs | qsub –P long –l h\_vmem=XG

The memory you’ll need to allocate above (signified by “X”; X should be substituted with the # of gigabytes of memory desired) will be dependent on different variables, including the # of bases of sequence you are aligning and the number/size of the reference genomes involved. Out-of-memory crashes are not always obvious and can be difficult to diagnose, so it’s generally recommended that you allocate more than you think you’ll need.

It is very important that you allow all of the jobs created by this cluster submission to finish running before you proceed to the next step. Type “qstat” at the prompt to check on the status of your jobs periodically while you’re waiting. You may notice that the number of jobs inflates and contracts over time. This is because following the initial alignment of your sequences to the references, follow-up alignments against adapter sequences and the mouse genome are also performed (using those sequences that do not align to your references). The mouse alignment can take awhile, so plan to allocate 30-45m on average for your runs to complete.

Once all jobs are complete, you may notice quite a few new directories in your experiment folder (elandresults, filteredseqs, hitratios, mappingstats, NM, seqcountsbybc). Avoid moving any of their contents, as it will interfere with step (VII) below.

***VII. Aggregating your results (summarize.sh)***

The COPRO-seq pipeline will generate quite a few output files by the time it’s finished running. The final step is to aggregate all of these results into single tables that are easier to digest and import into other software. To do so, run:

sh summarize.sh

***VIII. Cleaning up (cleanup.sh)***

Finally, you just need to clean-up intermediate and unneeded files to keep things tidy. Just run:

sh cleanup.sh

***IX. Interpreting the output files***

Once clean-up is complete, you’ll probably find only a handful of files/directories in your experiment folder:

* Your original mapping file
* Your IGS table (if it was provided when running batch\_coproseq.pl)
* “filteredseqs” directory : contains FASTA-formatted sequence files for trimmed reads that did not map to any of your references. These are divided into “adapter”, “mouse”, and “unknown” sequences. The “unknown” sequences may be useful for subsequent BLAST searches if you suspect a contamination event in your experiment (e.g. based on the percent.mappingstats summary file)
* “summaries” directory, containing:
  + .mappingstats files : provide count and percentage breakdowns of how your data mapped to the references.
    - U(0,1,2): mapped uniquely to one location in one of the references when 0, 1 or 2 base changes were incorporated
    - R(0,1,2): mapped redundantly to more than one genome and/or location within a genome when 0, 1 or 2 base changes were incorporated
    - NM: did not match one of the references
    - NM-adapter: did not match one of the references, but was subsequently found to match the Illumina adapter sequences (i.e. most likely represents an adapter-dimer product)
    - NM-mouse: did not match one of the references or the adapter, but was subsequently found to match the mouse (host) genome (i.e. most likely represents DNA from mouse epithelial cells in the sample; this # can be especially high near the beginning of a colonization)
    - NM-unknown: did not match one of the references, nor did it match the adapter or mouse sequences (i.e. most likely represents reads with in-dels or multiple base errors, or possibly represents contaminating sequence; usually one can expect 1-2% of sequences to fall into this category, but numbers over 2% should be investigated further to rule out contamination)
    - QC: no matching done because sequence failed ELAND quality control criteria (probably too many N’s)
    - RM: no matching done because repeat was masked (this should always be 0% given the way ELAND is run)
  + .bcdist files : barcode distribution files describing the relative proportion of all barcodes observed in your data (not just those you actually used). This provides information on barcode distribution in your pool and what % of your sequences matched one of the barcodes used.
  + .profile files : raw\_counts.profile and norm\_counts.profile describe the number of raw or normalized hits, respectively, that mapped to each reference genome. **The “norm\_percent.profile” file is the final output for the pipeline and describes the relative proportion of each species in each sample.**

***X. Troubleshooting***

*Problem*

You have additional FASTA files for additional reference genomes you would like to include in your analysis, but they do not have the correct formatting (i.e. they are either multi-entry or have unwanted newline characters introduced by a text editor).

*Suggestion*

Copy “contigs2genome.pl” from /home/comp/jglab/nmcnulty/scripts and run it on the problematic FASTA file:

perl contigs2genome.pl <input> <output>

*Problem*

Hidden end-of-line characters in one of your input files (e.g. your mapping file) are disrupting one or more steps in the pipeline. This is most often a problem when one creates a mapping file in Excel (even if it’s Excel running on Mac OS X).

*Suggestion*

Excel and other programs (especially Windows software) will introduce hidden characters that cause problems in Unix-based environments, even when you save your spreadsheet as tab-delimited text. This is a consequence of the fact that different operating systems have different ways of signifying the end of a line in a text file. For more info on this issue, see:

*http://en.wikipedia.org/wiki/Newline*

You may try running the “dos2unix” utility on the problematic file at the terminal command prompt:

dos2unix <filename>

Alternatively, you can open the file in a robust text editor (e.g. TextWrangler for Mac OS X, EditPlus++ or EditPadPro for Windows) and re-save the file as tab-delimited text (Unix format for line breaks).