README for Akk INseq

Last update March 17, 2022

Data analysis workflow

Overview

1. Pre-processing to format primary sequencing data for Goodman INseq perl scripts
2. INseq scripts to map reads and generate read count tables (counts per Tn insertion-site)
3. Quality control and filtering
4. ARTIST-based method for normalization
5. Multi-sample statistics, and annotation with TRANSIT

Step-1: Pre-processing sequencing data

Download sequencing data

- Keep a copy of compressed data somewhere safe for permanent storage (Data Commons)
- Do all the following work on a **copy** of the primary compressed data

Set-up for running pre-processing script

- Make sure you have Python installed (v3 works; not sure about 2.7 that comes with macOS)
- Make sure you have Biopython installed too
- Directory structure (recommended)
    - curDir (wherever you want; must be the current directory in Terminal when calling script)
        - python script, "repair_barcodes.py"
        - note: could also put elsewhere and specify path when calling script
    - curDir/Input directory contains
        - all raw data files listed in sample map
        - 3-column sample mapping file (see below)
    - curDir/Output directory (empty for now)

Sample mapping file

This file assigns sequence reads to Samples based on Barcode sequence in data. Make very

sure this table is correct.
3-column table: Barcode <---> SampleID <---> Raw data filename

- be sure text is plain, avoid hidden characters that might come from Excel, etc.
- tab delimiter to separate fields
- raw data filename must be exact match to filename that will be accessed for data, "XYZ.fastq.gz"

SampleID defines file names for output data tables

- avoid use of underscores
- to simplify bulk data import of array plate data, use following format: "PlateID", where "ID" is a number or letter

```
BARCODE    SampleID    raw_data_fileName
```
e.g.
```
ACGTAC    Pool7    Sample_Lane1_blahBlah.fastq.gz
```

NOTE: "repair_barcodes.py" generates a 2-column table that should be used for Step-2, Goodman INseq perl scripts.

Run pre-processing script: "repair_barcodes.py" from command line

- `python3 repair_barcodes.py Input/sampleMap.txt Output/outName.fasta --fastqdir Input/`
- `/Users/pmalkus/Documents/PYTHON/inVitro_Exp5_seqData`
    - Input and Output directories as above
    - "outName.fasta" is the destination for the processed sequencing data; choose a name you like
- debug option
    - highly recommended prior to analyzing full data set
    - append to command above: `--debug integer`
    - where "integer" is the number of reads to analyze from each of the raw data files, 10 should suffice
    - check that "outName.fasta" has expected content
- Output directory contents
    - outName.fasta
        - all Sample sequencing data, concatenated (simplifies subsequent step)
        - barcode reading errors identified by Illumina barcode grouping are corrected based on 3-col sample map
        - barcodes and 2-column sample map used by Goodman perl scripts to create separate Sample files
    - 2-column sample map
        - can be directly used for Goodman INseq pipeline
        - naming convention appends "_to2column.csv" to sampleMap provided

(e.g. "sampleMap_to2column.csv")
    - note: Goodman readme calls flor .txt file, but .csv will work too

---

Step-2: Identifying and enumerating Tn insertion sites

Special thanks to Andrew Goodman for reagents, scripts, and advice.
Methods and tools are described in, or based on: https://pubmed.ncbi.nlm.nih.gov/22094732/

Directory structure

- see Goodman_README_v2.txt for more information
- Goodman analysis directory (scripts & index files) can be anywhere, but need to stay where they're placed. Must contain:
    - PERL scripts
    - config.txt
    - Index directory
        - Genome specific directories associated with Bowtie mapping (see Annotation files, below)
- Experiment/Output directory (specific to experiment), must have:
    - 2-column sample map file (tab delimited; .txt or .csv), with a row for each sample:
        - barcode
        - sample name
    - sequence data (fasta format)
        - note: formatting of raw data may vary depending on how/where it was sequenced
        - repair_barcodes.py and INseq_pipeline_v2 work together based on raw data from 2020 Illumina fastq output format

Annotation files

- protein table (.ptt) required for Goodman 'pipeline' script is no longer provided by NCBI
    - Matlab script was used to generate a .ptt file from a .gb or .gbff file downloaded from NCBI
        - fields extracted from gbk to fill the ptt are described in the header of the script
        - for BAA-835 (handles sequencing error at 1704819, see below)
            - for RefSeq resource .gb file
                - gbk2ptt_refSeq.m
            - for GenBank .gb fiile

        - gbk2ptt_genBank.m
- Genome sequence, fasta format (.fna)
- NOTE for handling sequencing error in NCBI version of BAA-835 genome. Extra base (G) at 1704819 interrupts recG ORF (Amuc_1422), which is conserved across Akkermansia. See "BAA-835_NCBI-sequence-error.pdf" in GitHub repository.
    - use fasta/fna genome sequence file that has been corrected
    - Matlab scripts above adjust genome coordinates output to .ptt file
    - manually update Amuc_1422 entry in .ptt file using plain text editor

Bowtie mapping
Modified from Goodman README.txt

- Download bowtie from http://bowtie-bio.sourceforge.net/index.shtml and install properly to your system,
- Edit the "config.txt" in the analysis package. Change the bowtie_dir variable to the path where your bowtie installation is.
- Go to the directory where you unpacked the INseq analysis package, go to the "indexes" directory, and create a new directory with your organism's name, e.g. AkkBlah. Put the multifasta file and the .ptt files in the new directory, go into the new directory and construct a bowtie index by typing the command:

```
$    <PATH-to-BOWTIE_DIRECTORY>/bowtie-build
<REFERENCE_GENOME.fna>  <REF_GENOME_DIRECTORY>
```
examples:
```
 $    /Users/mwu/bowtie-0.12.7/bowtie-build AkkBlah.fna AkkBlah

$ /Applications/bowtie-1.2.2-macos-x86_64/bowtie-
build AmCP001071_w1422.fna AmCP001071_w1422
```

From Terminal, curDir = directory with sequence data & sample map (for example = "Output"):
```
$ perl <PATH-to-GOODMAN_ANALYSIS_DIRECTORY>/INSeq_pipeline_v2.pl -i
<SEQUENCE-DATA-FILENAME> -m <SAMPLE_MAP> -s <INDEX_GENOMENAME> -d 0.9
```
For example:
```
$ perl /Users/jDoe/Documents/Goodman_INseq_home/INSeq_pipeline_v2.pl
-i outName.fasta -m sampleMap_to2column.csv -s AmCP001071_w1422 -d
0.9
```

Run mapping jobs to generate read tables, cpm normalized count tables, and mapped (CDS annotated) tables
```
$ sh mappingjobsXXX.job
```
Example for one Sample (consult Sample mapping table for Sample-Barcode correspondence):
```
$ sh mappingjobs_INSEQ_experiment.scarf_ACGTAC.job
```

Or do for all Samples in Experiment directory, using:

```
$ for file in *.job; do printf "$file\n"; sh "$file"; done
```

Prints mappingjobs filename (includes barcode) above output stats from running job for each Sample

- Copy QC information displayed in Terminal to QC spreadsheet

Will also generate "results" folder containing:

- INSEQ_experiment.scarf_SampleID.bowtiemap: Raw mapping output file from bowtie
- INSEQ_experiment.scarf_SampleID.bowtiemap_processed.txt_chromosomename: Read count table containing Tn insertion site (genome coordinate) and read count per site (Left, Right, Total)
- INSEQ_experiment.scarf_SampleID.bowtiemap_processed.txt_chromosomename_filter_cpm.txt: Filtered (total reads <3) and normalized to counts per million reads (cpm)
- INSEQ_experiment.scarf_SampleID.bowtiemap_processed.txt_chromosomename_filter_cpm.txt_mapped: Annotated to genes (CDS only, positional threshold set in "INSeq_pipeline_v2.pl" call)

Step-3: Quality Control and filtering

We experienced numerous problems with bad sequencing data. The source of these problems was not conclusively identified, but may have been excessive Mg2+ in the amplification steps of the library prep (result of replacing the polymerase described in the original protocol, with a polymerase+reagents pre-mix and continuing to add the PCR reagents described in the protocol).
The most frequent, and consequential, problems we observed in the sequencing data were:

- large number of spurious reads, which reduced overall read-depth in individual samples
    - high number of non-TA reads
    - reads from sites adjacent to real Tn-sites (typically one-sided)
- large degree of variation between reads coming from Left and Right sides of Tn insertion site

To address these problems we developed scripts for:

- data browsing, plotting various features of the data (INseq_read_filter_v3.m, INseq_read_filter_v4.m)
- filtering, based on explicit criteria, and guided by plotting data features
- tabulating quality control metrics (INseq_QCsummary_v3.m)

These scripts can be executed on individual read count data files, or in some cases on a

directory of such files.
Outputs were used to:

- define criteria to include samples in downstream analysis (see Methods)
- filter individual samples using a set of pre-defined, high-confidence Tn insertion sites

High-confidence Tn-insertion sites in the the Arrayed Library were identified by sequencing individual 96-well plates. The combination of 1) extremely high sequencing depth, 2) an expected number or true Tn-sites, 3) and plotting data features, provided a basis for defining a set of "real", high-confidence Tn-insertion sites in the Arrayed Library. An analogous set of high-confidence Tn-sites was identified in the Pooled Library, by performing multiple deep sequencing runs of the Input library and combining these data for additional depth.

Step-4: ARTIST-based normalization

ARTIST publication: https://journals.plos.org/plosgenetics/article?id=10.1371/journal.pgen.1004782
ARTIST resources, Waldor lab,
HMS: https://drive.google.com/drive/u/0/folders/0Bz7imAGWYK6nfl80LUk1QnVoVnZaWThudi1zXzVVM3F2VWVmOXMwN09JOFRIU2RvNUQtd2M

Normalization methods developed for RNAseq are widely available, and are often applied to TNseq data. However, these methods normalize around a sample-wide average, applying the assumption that while the transcription of individual genes may change dramatically, the overall level of transcriptional activity will be similar in all conditions tested. This assumption does not apply to transposon mutant populations tested for conditional growth, where under restrictive conditions (mouse colonization, minimal medium) a large number of clones with disruptions in essential genes will be depleted, resulting in reduced library complexity. The normalization method used by the ARTIST package attempts to account for these changes in library complexity.

Numerous steps in the ARTIST workflow described in the links above were not relevant/appropriate for our analysis and were bypassed using custom Matlab scripts. Minor modifications were also made to ARTIST scripts for computational efficiency, for limited library size, and for low-quality sequencing data.

Directory structure, contents
The following should be placed in a single directory

- All "Sample" read count data tables output from Step-2, that passed QC metrics in Step-3
- "Input" read count data table output from Step-2
- "Filter" read count data table, from Step-3 (note that only genome coordinates are used from this table)

All data tables should be provided in the format of read count tables output from Goodman perl scripts. Goodman naming convention should be preserved.

- other files required
    - GENOME_TAsites.txt
    - GENOME_TAsitesID.txt

Note: "GENOME" in file name is specific to user

- GENOME_TAsites.txt
    - List of all TA sites in genome
    - to generate use: TAfinder.py
        - input: genome sequence fasta file
- GENOME_TAsitesID.txt
    - Annotation table formatted for ARTIST
    - to generate use: annotate_Akk_TAsites_Artist.m
        - inputs:
            - GENOME_TAsites.txt
            - GenBank resource .gb annotation file

Execute modified ARTIST-based normalization:
Run from CurDir

- For a single Sample
    - Artist_run_v2.m
    - inputs: (genome, sampleName, inputName, filterName)
- For all Samples in the directory
    - Artist_run_directory.m
    - inputs: (genome, inputName, filterName)
- Output is Matlab files containing results as workspace variables:
    - file NAME: "inputName_sampleName_filterName.mat"
    - workspace variables from ARTIST-based calculation and simulation
        - "MWUdone_NAME.mat"
    - spreadsheet of log-fold change and MWU stats for Sample-Input pairs
        - "MWUstats_NAME.txt"

Step-5: TRANSIT

TRANSIT: **https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004401**

Formatting for Transit
ARTIST is designed to act on single Sample-Input pairs, so an additional multi-sample normalization step was required to use TRANSIT for analysis of multiple Samples (e.g.

biological replicates) prepared using a single Input population. Note: all Transit normalization features were set to OFF.

- Data tables
    - Artist_multiSample_normalization_exact.m
    - input:
        - method (see header in script for more detail)
        - files to export (specify NAME, "MWUdone_NAME.mat" files, all in current directory)
    - output spreadsheet
        - multi-sample normalized read count data table
        - columns: Tn-site / Input / Sample1 / Sample2 / Sample3
- Annotation tables for TRANSIT were generated using
    - Akk_Transit_table.m
    - input: GenBank or RefSeq resource .gb annotation file

Data and annotation tables were then provided to TRANSIT. Additional information on settings used for TRANSIT can be found in [Supplemental] Methods.