PoolQ

Mark Tomko, John Sullivan, Shuba Gopal.

This documentation was covers PoolQ version 2.2.0 (last updated 02/26/2016).

PoolQ is a counter for indexed samples from next-gen sequencing of pooled DNA.

Background

The Broad Institute Genetic Perturbation Platform (GPP) uses Illumina sequencing to tally the results of pooled screens. PoolQ can process pooled screens representing a variety of genetic perturbation techniques, including CRISPR/Cas9, RNAi, and barcoded ORFs. We perform the following steps to generate one or more files of sequencing reads (or "reads files") for the pooled screen:

1. We start with multiple genomic DNA samples (conditions) - gathered either from different pools, or from the same pool at different time points.
2. We apply PCR to the genomic DNA with primers that are designed to attach to the genomic DNA within the vector sequence at a fixed distance from the start of the construct barcode sequence. The primer contains a fixed-length DNA barcode that is unique to the genomic DNA sample. In this way, all the amplification products will contain the inserted sample barcode.
3. We mix all the samples, normalized to equalize barcode representation, and run them in a single sequencing lane.
4. The sequencing process generates an output file for each sequencing lane.

PoolQ comes into play after the reads file has been generated. A single PoolQ run processes one sequencing run. It attempts to parse out the barcodes representing the sample and the construct from each read. It maps sample barcodes to conditions and construct barcodes to construct IDs, and generates a matrix with the conditions as columns, and the constructs as rows.

PoolQ Use Cases

In general, usage of PoolQ falls into one of four basic scenarios, defined by the number of sequencing files it needs to read and the way it will locate construct barcodes within those files. PoolQ can process sequencing results with reads wholly contained within one sequencing file, or with each read partially represented within two sequencing files. It can locate construct barcodes that occur at a constant offset within the reads or that occur at a variable offset within the reads (by locating a short DNA prefix known to precede the construct barcode).

We describe the combination of these two dimensions in the following way:

- Scenario 1: Fixed offset, one sequencing file
- Scenario 2: Variable offset, one sequencing file
- Scenario 3: Fixed offset, two sequencing files
- Scenario 4: Variable offset, two sequencing files

These scenarios require different command-line options. The PoolQ distribution contains sample data representing each scenario and presents example command lines for processing each case.

PoolQ Inputs

PoolQ requires 3 input files to run: the (FASTQ or SAM/BAM) file containing the reads (the reads file); a file mapping barcodes to conditions (the conditions file); and a file mapping construct barcode sequences to construct IDs (the reference file). PoolQ can also take an optional input file (the platform reference file) describing known barcodes that are not expected to be present in the sequencing reads. There are also a variety of optional settings that specify how PoolQ processes the reads.

The Reads File

The *reads file* is a either a standard FASTQ file or a SAM/BAM file, containing the reads. It may or may not be compressed with gzip. If the file is a FASTQ file, then the file extension should be `.fastq` or `.txt` (case-insensitive). If it is a BAM file, then the file extension should be `.bam`, and for a SAM file, the extension should be `.sam` (case-insensitive). Files that do not follow these naming conventions can still be used, but you must specify the file type explicitly.

If the file is compressed with gzip, then the file must have an extra `.gz` extension after the `.fastq`, `.txt`, or `.bam` extension.

A barcode representing a sample and a construct are parsed out of each read. The length of the sample barcode is inferred from the contents of the conditions file. The length of the construct barcode is inferred from the contents of the reference file. By default, the sample barcode starts at the first base of the read, and the construct barcode starts at the 17th base of the read, but both of these defaults can be overridden with an optional flag.

In some cases, reads are divided into two separate FASTQ files (BAM/SAM is not supported in this case). One file contains the sample barcodes, while the other contains the construct barcodes. In this case, the file with shorter reads is assumed to contain the sample barcodes, while the file with longer reads is assumed to contain the construct barcodes. This mode of operation requires that FASTQ record IDs match between the two input files.

The Conditions File

The conditions file maps sample barcodes to samples or experimental conditions:

- It is a file with at least two columns
- The first column contains the sample barcodes sequence and the second column contains the condition descriptions
- All other columns are ignored
- Every row must have the same number of columns
- The columns can be separated by either commas or tabs
- The file may not include any column headers or any extra columns
- Barcodes must contain only A, T, C or G
- A barcode cannot occur more than once in the file
- Every barcode in the conditions file must have the same length
- You can have multiple barcodes mapping to the same condition, but be aware that if you do so, the reads for those barcodes will be counted together, resulting in a single column in the scores file for that condition

The Reference File

The *reference file* maps from construct barcodes to construct IDs:

- It is a file with at least two columns
- The first column contains the construct barcode sequences, and the second column contains the construct IDs
- All other columns are ignored
- Every row must have the same number of columns
- The columns can be separated by either commas or tabs
- The file may not include any column headers or any extra columns
- Barcode sequences must contain only A, T, C or G
- A construct ID cannot occur more than once in the file
- Every construct barcode in the reference file must have the same length
- You can have multiple construct IDs mapping to the same barcode sequence; the scores file will report the scores for the construct barcode alongside a comma-separated list of associated construct IDs.

The Platform Reference File

The *platform reference file* is an optional input file whose format is identical to that of the reference file. It consists of a master list of known construct sequences and their construct IDs. This file is used to provide construct IDs for barcodes encountered during the PoolQ run that were not expected to occur. A construct is expected to occur only if it is present in the reference file.

Include Non PF Reads

This is an optional input flag that, when present, indicates that PoolQ should include reads from a BAM file that fail the purity filter (PF) quality control check. This flag has no effect on the PoolQ results for FASTQ files, since the annotation is not present in FASTQ files. For more information on this flag, please see the documentation for Picard and SAMtools.

Barcode Start Index

This is an optional input flag that, when present, specifies the index of the first base of the sample barcode within the read. This is a 0-based index, which means that the first base of the read has index 0, the second base of the read has index 1, etc.

Construct Start Index

This is an optional input flag that, when present, specifies the index of the first base of the construct barcode within the read. This is a 0-based index, which means that the first base of the read has index 0, the second base of the read has index 1, etc.

Construct Search Prefix

Optionally, PoolQ can search for the construct barcode sequence by looking for a short DNA prefix that always precedes the construct barcode. This behavior is enabled by passing the construct search prefix argument *instead of* the construct start index argument. The construct barcode extracted will be the sequence immediately following the prefix.

Construct Search Start Index

This is an optional input flag that, when present, specifies the index at which PoolQ begins searching for the construct search

prefix. This is used to restrict the search to within an acceptable range, to avoid false positives early in the read.

Construct Search End Index

This is an optional input flag that, when present, specifies the last index at which the construct search prefix may begin in the reads. This is used to restrict the search to within an acceptable range, to avoid false positives late in the read.

Unexpected Sequence Threshold

This is an optional input flag that, when present, specifies the number of unexpected construct and sample barcodes to include in the unexpected sequence report. PoolQ will report the most frequently occurring barcodes. The default value is 100, meaning the report will contain the 100 most frequently occurring construct barcodes and the 100 most frequently occurring sample barcodes.

Exact Match

This is an optional input flag that, when present, indicates that fuzzy matching of construct barcodes found in reads to the construct barcodes in the reference file should be disabled. Only barcodes that exactly match a construct barcode in the reference file will be counted. The default behavior (when this flag is not present) is to allow single base mismatches when matching construct barcodes to the reference file.

Include Ambiguous

This is an optional input flag that, when present, controls the handling of barcodes encountered in reads that fuzzy-match to more than one construct in the reference file. The default behavior is to discard ambiguously matching reads, so they will not be included in the scores file. If this behavior is not desired, specify this flag and all ambiguous reads will be counted for every possible matching construct barcode. As a consequence of this behavior, the sum of the scores for a given column may add up to more than the number of reads for a particular condition.

Reads File Type

This is an optional input flag that, when present, specifies how PoolQ should treat the reads file type. By default, PoolQ will attempt to guess whether the reads file is a FASTQ, BAM, or text file based on the file name. If the filename is misleading, you can specify the file type explicitly using this flag. Valid values include BAM, FASTQ, and RAW (for plain text).

Skip Short Reads

This is an optional input flag that, when present, specifies that PoolQ should simply ignore reads that are too short to contain both barcode sequences. By default, PoolQ considers files containing short reads to be badly formed and exits. By specifying this flag, you indicate that PoolQ should simply skip these short reads; a count of the number of skipped short reads will be available in the quality file.

PoolQ Outputs

PoolQ generates output files representing the matrix of read counts (or scores) for expected sequences, a report of read counts for unexpected sequences, and a report containing simple metrics used to help assess the overall quality of the sequencing data. There are two optional output files that contain alternative representations of the scores matrix. One contains the scores in log normalized form and the other contains read counts by barcode rather than by condition.

The Scores File

The *scores file* is a text file that contains a simple matrix of the read counts. The columns of the matrix represent the experimental conditions, and the rows of the matrix correspond the construct barcode sequences. The individual values in each row are separated by tabs.

If you plan on loading the scores file into a spreadsheet application such as Excel, then we recommend using a file extension such as `.txt`, that your spreadsheet application will recognize as being a text file. When opening the file in Excel, you will probably be prompted with a dialog asking you to describe the structure of the file. In the section about separator options, be sure that the checkbox for "Tab" is selected.

The Scores File in GCT Format

PoolQ can also produce the scores file in GCT format. This format is required for upload into GENE-E to perform a RIGER type analysis. Simply choose a '.gct' or '.GCT' file extension when selecting the name of the file.

The Quality Report

The *quality report* is a simple text file containing some extra information gathered during the PoolQ run. The information

reported here is intended to help you assess the quality of your data, and spot problems such as an unacceptably high frequency of uncounted reads, or mistakes in barcode tracking. We currently report:

- The total number of reads
- The total number of reads that were successfully counted
- Out of the counted reads, the total number of single base mismatches to the construct barcode
- Out of the counted reads, the percent that matched to both a known sample barcode and a known construct barcode
- The average frequency of unknown sample barcode sequences
- The log-normalized frequency of unknown sample barcode sequences
- For each sample barcode mapped to a condition, we report:
    - the barcode
    - the condition
    - the total number of reads matching the sample barcode plus an expected construct barcode
    - the total number of reads matching the sample barcode
    - the percent of the reads for the sample barcode that matched an expected construct barcode
    - the log normalized number of matches
- For each sample barcode not mapped to a condition, we report the barcode and the total number of reads
- For construct barcodes mapping to multiple construct IDs, we report the construct barcodes and the construct IDs they map to. A construct barcode can map to multiple construct IDs for two reasons:
    - The construct barcode may have occurred twice in the reference file
    - If the construct barcode sequences are truncated in the reads in the reads file, then two otherwise unique barcodes could end up being the same after truncation

The Barcode Scores File

The barcode scores file has a similar format to the scores file, except that the columns in the matrix represent the read counts for individual DNA barcodes rather than for experimental conditions. If, based on the quality file, a particular PCR appear to have been of low quality, it is possible to reaggregate scores by condition by excluding the scores from the barcode corresponding to the failed PCR. The barcode scores file is an optional output intended to provide support for loading PoolQ data into the RNAi Informatics database. However, it is available for any consumer of PoolQ data.

The Log Normalized Scores File

The log normalized scores file has the same format as the scores file, but every score is normalized according to the following procedure:

1. Take the raw read count for the construct ID and the condition
2. Divide by the total number of reads for that condition that matched a construct barcode found in a reference file
3. Multiply by a constant factor of 1 million
4. Add one
5. Take the log base 2

The Unexpected Sequence File

The unexpected sequence file contains a report that describes briefly the collection of sequences found in the position where a construct barcode was expected during the run. It is an optional output. Tracking unexpected sequences currently results in a substantial performance penalty. We recommend that you only generate an unexpected sequence for forensic or investigative purposes.

The unexpected sequence report contains two sections. The first section consists of a table whose rows correspond to unexpected construct barcode sequences and whose columns indicate the number of times each sequence was found for each barcode. An additional column lists the construct IDs for these sequences, if the IDs are known. These construct IDs can be provided to PoolQ via the platform reference file, described above.

The second section describes unexpected sample barcodes and the number of times an unexpected sequence appeared with each unexpected barcode. The unexpected sample barcodes are listed in descending order of the number of occurrences.

The Correlation File

The correlation file contains a pairwise correlation matrix comparing the per-construct scores for each experimental condition. The correlation metric is the Pearson product-moment correlation.

Running PoolQ

There are two different ways you can run PoolQ: using the Graphical User Interface (GUI), or using the Command Line Interface (CLI). But before you can run it, you need to download the zip file and unzip it.

Prerequisites

PoolQ is built for Java 8. To run PoolQ, you will need a JRE for version 8 or later. To compile PoolQ you will need a Java 8 JDK. You can download an appropriate JRE or JDK from Oracle at:

> http://www.oracle.com/technetwork/java/javase/downloads/index.html

Downloading and Unzipping PoolQ

You can download PoolQ from an as yet undetermined location. The file you download is a ZIP file that you will need to unzip. In most cases, this is as simple as right-clicking on the zip file, and selecting something like "extract contents" from the popup menu. This will create a new folder on your computer named `poolq-2.1.4`, with the following contents:

- `poolq.jar`
- `poolq.bat`
- `poolq.sh`
- `test-data/`

Feel free to rename the folder, and to move it to wherever you want. Be aware, however, that the `.sh` and `.bat` files will only function properly if they can find the `poolq.jar` file in the same folder.

The test-data folder contains four sets of sample input data and a Makefile that will run PoolQ on each set, printing the full command line in the process. If you have GNU Make installed, you can test all four sample datasets by running:

```
  % cd test-data
% make
```

You can also test them individually by running:

```
  % cd test-data
% make test-scenario1
% make test-scenario2
% make test-scenario3
% make test-scenario4
```

Recommended JVM Settings

We recommend the following JVM settings be provided when running PoolQ:

- `-Xmx4G`
- `-XX:+UseG1GC`
- `-XX:+UseFastAccessorMethods`
- `-XX:+UseCompressedOops`

This document contains a number of example command-lines for running PoolQ; however, we only list the full JVM options once, since typing the full command becomes unwieldy and the JVM options distract somewhat from the command-line arguments that are passed to PoolQ itself. You can copy and paste the full Java command from here:

```
java -Xmx4G -XX:+UseG1GC -XX:+UseFastAccessorMethods -XX:+UseCompressedOops -jar
poolq.jar
```

Running PoolQ

You can run PoolQ from any Windows, Mac, or Linux machine, but it requires some understanding about how to launch programs from the command line on your given operating system.

1. Open a terminal window for your operating system
2. Change directories to the `poolq-2.1.4` directory
3. On Windows, run:

   ```
   poolq.bat
   ```

- Or, on a UNIX-based machine, run:

  ```
  ./poolq.sh
  ```

- Or, on any machine, run:

```
java -jar poolq.jar
```

If you successfully launched PoolQ, you should see a usage message explaining all of the command-line options:

```
Usage: poolq [options]
Options:
  -barcode-scores, --barcode-scores
     An optional output CSV file with the sample barcodes as columns, the
     construct barcodes as rows, and the read counts as scores.
  -barcode-start, --barcode-start
     The index of the start of a sample barcode within a read. Defaults to 0.
     Default: 0
* -conditions, --conditions
     An input file with two columns: the barcode, and the condition.
  -construct-barcode-match-length, --construct-barcode-match-length
     The length of construct barcodes that should be used in matching.
     Default: -1
  -construct-barcode-search-end, --construct-barcode-search-end
     The last base (counting from 0) where PoolQ should search for construct
     barcodes.
  -construct-barcode-search-prefix, --construct-barcode-search-prefix
     The DNA prefix that precedes the construct barcode in the reads file and
     is used to locate the construct barcode.
  -construct-barcode-search-start, --construct-barcode-search-start
     The first base (counting from 0) where PoolQ should search for construct
     barcodes.
     Default: 7
  -construct-start, --construct-start, -hairpin-start, --hairpin-start
     The index of the start of a construct barcode within a read. Defaults to
     16.
     Default: 16
  -correlation, --correlation
     An optional output file with the correlation matrix of the log normalized
     scores by condition.
  -exact-match, --exact-match
     Use exact matches for construct barcodes in the reference file
     Default: false
  -include-ambiguous, --include-ambiguous
     Score reads that match ambiguously to all matching constructs. Defaults
     to false, in which case ambiguous matches are discarded.
     Default: false
  -include-non-pf, --include-non-pf
     Include non-PF reads [SAM or BAM files only]
     Default: true
  -norm-scores, --norm-scores
     An optional output CSV file with the conditions as columns, the construct
     barcodes as rows, and the read counts as the log normalized scores.
  -platform-reference, --platform-reference
     An optional input file with two columns: a construct barcode that is
     known to exist, and the associated construct ID.
* -quality, --quality
     An output text file containing a basic report of the quality control
     information gathered while processing the reads.
* -reads, --reads
     The file containing the sequencing reads in either FASTQ or BAM format.
     The file may be gzipped or not; if the file is gzipped, it should end with
     the .gz suffix.
     Default: []
  -reads-file-type, --reads-file-type
     Override the reads file type. One of [BAM, FASTQ, RAW].
* -reference, --reference
     An input file with two columns: the construct barcode that are contained
     in the reads, and the construct IDs.
* -scores, --scores
     An output CSV file with the conditions as columns, the construct barcodes
     as rows, and the read counts as the scores.
  -skip-short-reads, --skip-short-reads
     Skip reads too short to contain both a barcode and construct. Defaults to
```

```
   false.
   Default: false
-unexpected-sequence-threshold, --unexpected-sequence-threshold
   The maximum number of unexpected sequences to report
   Default: 100
-unexpected-sequences, --unexpected-sequences
   An optional output text file containing a report of sequences found in
   the reads but not mapped to construct IDs by the reference file. If a
   platform reference file is provided, any constructs contained there will be
   identified by IDs in this file.
-help, --help
   Output this message.
   Default: false
```

At this point, you are ready to run PoolQ for real, supplying file names and locations for the 3 file inputs and 2 or 3 file outputs. For example:

- On Windows, run:

  ```
  poolq.bat --reads reads.txt --conditions conditions.txt --reference reference.txt --
  scores scores.txt --quality quality.txt
  ```

- Or, on a UNIX-based machine, run:

  ```
  ./poolq.sh --reads reads.txt --conditions conditions.txt --reference reference.txt --
  scores scores.txt --quality quality.txt
  ```

- Or, on any machine, run:

  ```
  java -jar poolq.jar --reads reads.txt --conditions conditions.txt --reference
  reference.txt --scores scores.txt --quality quality.txt
  ```

The Scoring Algorithm

The *reads file* contains the sequencing reads. PoolQ supports any of the following formats:

- FASTQ (including Solexa/Illumina variants)
- SAM
- BAM
- Plain text (one read per line)

PoolQ currently ignores any read sequence content besides the sample barcode and the construct barcode. In the future, we may check the remaining sequence to help confirm the quality of the read.

Most often, the entire construct barcode is included in the read. However, for files with very short read lengths the construct barcode sequence may be truncated. In the case of truncated construct barcode sequences, PoolQ will attempt to match based on the available construct barcode sequence prefix.

If the read does not have a sample barcode that is an exact match to a barcode found in the conditions file, then the line is not counted, except in the section of the quality report devoted to counting reads for barcodes not found in the conditions file.

Counting Reads that Match a Barcode

The PoolQ scoring algorithm always attempts to match construct barcodes exactly to to one of the sequences provided in the reference file first. If an exact match is found, then only the exact match is counted.

If an exact match is not found, PoolQ will attempt to match to a known construct barcode sequence allowing a single nucleotide mismatch. An N in the construct barcode sequence is considered a single nucleotide mismatch. The exact match setting allows you to override the single nucleotide mismatch behavior and score only exact matches.

If a construct barcode sequence is a single nucleotide mismatch to two or more different barcodes, PoolQ will discard the read by default. It is possible to override this behavior as well with the include ambiguous setting, in which case PoolQ scores the read for every construct barcode sequence that is a single nucleotide mismatch.

If PoolQ matches a read to a sample barcode that is mapped to a condition, and a construct barcode that is mapped to one or more construct IDs, then the counts are incremented for all of the matching condition/construct ID pairs.

Construct barcodes are counted as unexpected sequences if they are not successfully matched by the above procedure.

Contact Us

Your feedback of any kind is much appreciated. Please email us at rnaiinformatics@broadinstitute.org.