Analyzing dual-barcoded sequences from targeted amplicons

Matthew Settles and Alida Gerritsen

February 12, 2015

Contents

1 Overview

This document provides instructions for the installation and execution of the *dbcAmplicons* package for the analysis of targeted genomic regions. Amplicon sequencing can be customized to very specific regions via PCR primers and dual-barcoded for a high degree of multiplexing within a single sequencing run. However, the amount of data generated as well as the complexity of the information can be an obstacle when it comes to analysis and interpretation. This package is designed to take the researcher from sequence data to abundance tables in a simple pipeline.

2 Installation

Minimum requirements for the *dbcAmplicons* package: Python 2.X is required to run any and all of the elements in the package. Some applications within the package rely on flash or the RDP classifier to complete, so the user must determine of those external tools are necessary if the entire pipeline will not be run.

A modified executable *flash2* is included in the installation of *dbcAmplicons*, so the user may determine which version they would like to use.

2.1 Scripts

All executable scripts are located in the github repository dbcAmplicons.

2.2 Programs

External programs used in the pipeline:
flash
RDP classifier

Windows Systems
dbcAmplicons has not been tested on Windows systems, although Windows Powershell (active in XP systems and later) may be sufficient to run the analysis. It is highly recommended, however, that the user install a Linux-based virtual machine to run the pipeline.

Mac OSX
Mac users should first download and install Xcode and then download the app via a terminal by:

```
git clone https://github.com/msettles/dbcAmplicons.git
cd dbcAmplicons
sudo python setup.py install
dbcAmplicons
```

If installation was successful, this will bring up the usage arguments.

Linux
For Linux users, some python dependencies may be required before the installation is succesful. To install, open a terminal and type

```
sudo apt-get install python-dev
```

And type the `sudo` password at the prompt. Then, download the app via a terminal by:

```
git clone https://github.com/msettles/dbcAmplicons.git
cd dbcAmplicons
sudo python setup.py install
dbcAmplicons
```

If installation was successful, this will bring up the usage arguments.

3 Running *dbcAmplicons*

There are 4 levels of analysis within the *dbcAmplicons* package. The package was developed for command line usage to analyze high volumes of microbial 16S rRNA sequences. All components of the package are designed to flow directly into the next with no intermediate steps required.

```
dbcAmplicons [-h] [--version] {preprocess,join,classify,abundance}
```

3.1 *dbcAmplicons preprocess*

dbcAmplicons preprocess takes four fastq files (Read 1, Index 1, Read 2, and Index 2), the assigned barcode indices for each sample,the sequences of the template-specific primers, and the individual sample names along with an assigned project folder. *dbcAmplicons preprocess* will then assign each read in the fastq file to a specific project according to both barcode and primer sequences, and return 2 fastq files with all of the identifying information in the read header. If a project directory is specified, *dbcAmplicons preprocess* will create a directory with the indicated output prefix. The syntax for usage is as follows:

```
dbcAmplicons preprocess [-h] -B barcodesFile [-d BARCODEDIFF]
        [-P primerFile] [-D PRIMERDIFF] [-e PRIMEREND]
        [-S sampleFile] [-q MINQ] [-l MINL]
        [-b BATCHSIZE] [-O PREFIX] [-U] [-u] [-v]
        [-1 read1 [read1 ...]]  [-2 read2 [read2 ...]]
        [-3 read3 [read3 ...]][-4 read4 [read4 ...]]  [--test]
        [--keepPrimers] [--debug]
```

Options:
`-h, --help` Shows this help message and exits.
`-B barcodesFile, --barcodes_file` Barcodes file with barcodes used in the run
`-d BARCODEDIFF, --integer` The maximum number of mismatches allowed per barcode (default:1)
`-P primerFile, --primer_file` File with primer sequences
`-D PRIMERDIFF, --integer` The maximum number of mismatched allowed per primer (default:4)
`-e PRIMEREND, --integer` The required number of matching bases at end of primer (default:4)
`-S sampleFile, --samples_metadata` File denoting samples with primer and barcode asignments
`-q MINQ, --minQ` Trim 3' end of sequences to minQ (default:None)
`-l MINL, --minL` If minQ is not None, only keep reads that are at least minL length (default:0)
`-b BATCHSIZE, --integer` Batch size in which to process reads (default:100,000)
`-O PREFIX, --output_prefix` Output file basename (default:fastq_prefix)
`-U, --output_unidentified` Keep and output unidentified reads (default:False)
`-u, --uncompressed` Leave output files uncompressed (default:False)
`-v, --silent` Verbose output (default:False)
`-1 read1 [read1 ...]` Path to read 1 of an amplicon fastq four file set
`-2 read2 [read2 ...]` Path to read 2 of an amplicon fastq four file set
`-3 read3 [read3 ...]` Path to read 3 of an amplicon fastq four file set
`-4 read4 [read4 ...]` Path to read 4 of an amplicon fastq four file set
`--test` Exit after the first batch in order to test the inputs
`--keepPrimers` No primer trimming, leave as a part of the read (default:0)
`--debug` Show traceback on error

3.1.1 Inputs

At a minimum, *dbcAmplicons preprocess* requires a properly formatted reference barcode sequence file to demultiplex raw sequences. For maximum information, the following three inputs should be used, all in a tab-separated text format:

1. Reference barcode sequences file.
2. Reference primer sequences file.

3. Sample metadata file with sample IDs, primer IDs, and barcode IDs.

1. Reference barcode sequences file

dbcAmplicons preprocess requires a tab-separated text file with as few extraneous characters as possible. The barcode sequence reference file should have three columns: the first column is the user-defined barcode name; the second column is the Read 2 sequence, also referred to as Index 1 on the P7 end of the sequence; and the third column is the Read 3 sequence, also known as Index 2 on the P5 end of the sequence. Example format is below:

#BarcodeID	Read2	Read3
Barcode1	TAAGGCGA	TAGATCGC
Barcode2	CGTACTAG	CTCTCTAT
Barcode3	TAAGGCGA	TATCCTCT
Barcode4	CGTACTAG	AGAGTAGA
Barcode5	TAAGGCGA	GTAAGGAG

dbcAmplicons preprocess searches through the Read 2 fastq file from CASAVA for Index 1. As noted below (Figure 1), Index 2 is sequenced in reverse complement orientation relative to the P7 adaptor end of the sequence, so use of *dbcAmplicons* requires that the user know the orientation of the barcodes that were attached to the P7 end. Nextera kit users will often need to reverse complement their Index 1 sequences. TruSeq kit users will often not need to reverse complement their Index 1 sequences.

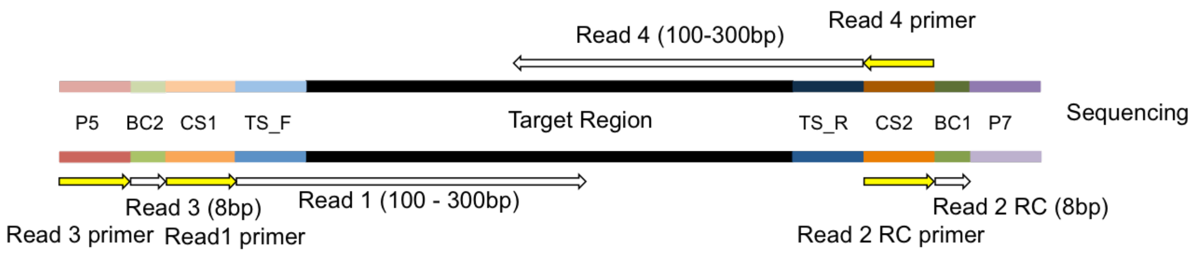

Figure 1: The orientation of sequencing reads on the Illumina Miseq. Note that Read 2 (Barcode 1) is sequenced in reverse complement orientation relative to Read 4.

2. Reference primer sequences file

The reference primer sequences file, similar to the barcode reference file, should also be a tab-separated text file. An example primer sheet is below:

#Read	Pair_ID	Primer_ID	Sequence
P5	PrimerPair1	Primer1Forward	GTAGAGTTTGATCCTGGCTCAG
P5	PrimerPair2	Primer2Forward	CGTAGAGTTTGATCATGGCTCAG
P5	PrimerPair3	Primer3Forward	ACGTAGAGTTTGATTCTGGCTCAG
P5	DegeneratePair1	Degenerate1Forward	GTGARTCATCGAATCTTTG
P5	DegeneratePair2	Degenerate2Forward	CGTGARTCATCGAATCTTTG
P7	PrimerPair1	Primer1Reverse	GTCCTCCGCTTATTGATATGC
P7	PrimerPair2	Primer2Reverse	TGTCCTCCGCTTATTGATATGC
P7	PrimerPair3	Primer3Reverse	ATGTCCTCCGCTTATTGATATGC
P7	DegeneratePair1	Degenerate1Reverse	GGGACTACHVGGGTWTCTAAT
P7	DegeneratePair2	Degenerate2Reverse	TGGGACTACHVGGGTWTCTAAT

There are 4 necessary columns in the primer reference sheet. *dbcAmplicons* will ignore any columns that are outside of the 4 named columns above, so the user may include additional information to each sample if necessary. Column 1 (Read) indicates the orientation of the primer. The user can denote this by the Illumina-specific adapter (P7 or P5), or the user may also use R1, R2, Read1, Read2, F, R, Forward, and Reverse. The "Pair_ID" column is the user-specified name for the primer pair that was used. The "Primer_ID" column denotes the name of each individual primer, and should correspond with the "Read" column. The "Sequence" column contains the full primer sequence **without** the CS-tag sequences. Note that the "Sequence" column accepts IUPAC nucleotide codes and treats each variation as a separate primer for searching.

3. Sample metadata file

The sample metadata file contains information to help split each sample into different projects along with the sample, primer, and barcode identifications. This feature used when the user wants to split out a single run's worth of data for multiple projects and can be done at several points along the *dbc Amplicons* analysis pipeline, in *dbcAmplicons classify* and also in *dbcAmplicons abundance*. The correct format for the sample metadata file is below:

SampleID	PrimerPairID	BarcodeID	ProjectID
Amp1	PrimerPair1	Barcode1	Idaho/amplicon
Amp2	PrimerPair2	Barcode2	Idaho/amplicon
Amp3	PrimerPair3	Barcode3	Idaho/amplicon
Car1	DegeneratePair1	Barcode4	Idaho/car
Car2	DegeneratePair2	Barcode5	Idaho/car

As in the primer reference sequences sheet, there are 4 columns in the sample metadata file. The "SampleID" column is the user's designation for each sample that was sequenced. The "PrimerPairID" column and the "BarcodeID" column should assign each of the primers and barcodes used to individual samples. Lastly, the "ProjectID" column denotes the output path for each sample. In the above table, sequences identified from all 5 samples will be placed in a folder named "Idaho" and from there will be divided into folders called "amplicon" and "car." If the user wants all primer pairs to be searched for all samples, *dbcAmplicons preprocess* will accept the regular expression "*" as a command to search all reference primers. If the user does not wish to use a primer reference sheet, the regular expression "-" will be accepted as a command to ignore the primers.

3.1.2 Outputs

Once *dbcAmplicons preprocess* has completed, a summary `IdentifiedBarcodes.txt` file is automatically generated within the executing directory that contains the number of reads that were assigned to each barcode and primer combination. Additionally, two `fastq.gz` files will be created for each user-specified output folder. Each of the fastq files will contain all of the information that was input at the beginning of *dbcAmplicons preprocess* for each identified read within the header of a sequence. Any primers found in the sequences are trimmed from the final output files unless the user specifies to keep them (Figure 2).

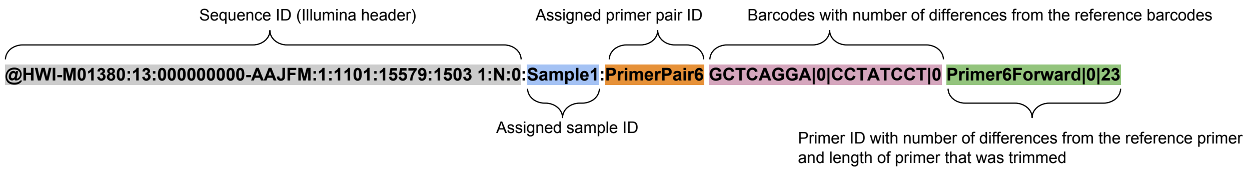
Sequence ID (Illumina header)
Assigned primer pair ID
Barcodes with number of differences from the reference barcodes
@HWI-M01380:13:000000000-AAJFM:1:1101:15579:1503 1:N:0:Sample1:PrimerPair6 GCTCAGGA|0|CCTATCCT|0 Primer6Forward|0|23
Assigned sample ID
Primer ID with number of differences from the reference primer and length of primer that was trimmed

Figure 2: Example output header from *dbcAmplicons preprocess* with annotations for sample ID, barcode ID and primer ID.

3.2 *dbcAmplicons join*

dbcAmplicons join takes the two fastq files generated from *dbcAmplicons preprocess* and joins them into a single long read using flash. The syntax is as follows:

```
dbcAmplicons join [-h] [-O PREFIX] [-u] [-x NUM] [-t NTHREADS]
        [-p path] [-v] [-1 read1] [-2 read2]
```

Options:
`-h, --help` Show this help message and exit
`-O PREFIX, --output_path` Path for output files (default:joined)
`-u, --uncompressed` Leave output files uncompressed (default:False)

`-x NUM, --max-mismatch-density` Maximum allowed ratio between the number of mismatched base pairs and the overlap length. Two reads will not be combined with a given overlap if that overlap results in a mismatched base density higher than this value. Note: Any occurence of an "N" in either read is ignored and not counted towards the mismatches or overlap length. (default:0.25)
`-t NTHREADS, --integer` Set the number of worker threads. (default:1)
`-p path, --flash_path` Path to the application flash (default: flash)
`-v, --verbose` Verbose output (default:False)
`-1 read1` Path to read1 of an amplicon fastq (or fastq.gz) two file set
`-2 read2` Path to read2 of an amplicon fastq (or fastq.gz) two file set

3.2.1 Outputs

There are 5 output files from *dbcAmplicons join*. The sucessfully combined sequences will be in a fastq file with an `.extendedFrags.fastq.gz` extension. Any sequences that were not combined will be in the fastq files with the `.notCombined_1/2` extension. Two text files, `joined.hist` and `joined.histogram` list the number of sequences and total lengths of the combined reads; `joined.hist` by integer, and `joined.histogram` graphically. For more information about flash output, please refer to the developer site.

3.3 *dbcAmplicons classify*

The current iteration of *dbcAmplicons classify* uses the RDP classifier for fungal LSU, bacterial 16S, and archaeal 16S to identify sequences. Usage is as follows:

```
dbcAmplicons classify [-h] [-b BATCHSIZE] [-q MINQ] [-l MINL]
        [-p PROCS] [--rdpPath PATH] [-g <arg>]
        [-S sampleFile] [-O outputPrefix]
        [-1 read1 [read1 ...]]  [-2 read2 [read2 ...]]
        [-U single [single ...]]  [-v] [--debug]
```

Options:
`-h, --help` Show this help message and exit.
`-b BATCHSIZE, --integer` Batch size for read processing (default:100,000)
`-q MINQ, --minQ` Trim 3' end of sequences to minQ (paired-end reads only) (default:None)
`-l MINL, --minL` If minQ is not None, only keep reads that are at least minL length (paired-end reads only) (default:0)
`-p PROCS, --integer` Number of processors to use (default:1)
`--rdpPath` Path to the RDP classifier
`-g <arg>, --arg` Choice of classification between 16S microbial rRNA or fungal LSU (default:16SrRNA)
`-S sampleFile, --sample_metadata` Sample metadata file with primer and barcode designations
`-O outputPrefix, --output_path` Path for output files (default:classify)
`-1 read1` Read 1 of an amplicon fastq two file set
`-2 read2` Read 2 of an amplicon fastq two file set
`-U single-end amplicon` Output from *dbcAmplicons join* or other joined paired reads
`-v, --silent` Verbose output (default:False)
`--debug` Show traceback on error

3.3.1 Outputs

For more information on the RDP classifier and its classification algorithm, please refer to the RDP Wiki. The RDP classifier within *dbcAmplicons classify* outputs a single file with each sequence classified according to its closest identified taxonomic rank using bootstrap values for confidence. The read is classified within the most specific taxonomic rank that meets the boostrap threshold of 50%.

3.4 *dbcAmplicons abundance*

dbcAmplicons abundance takes the output from *dbcAmplicons classify* and builds abundance tables from the information. Usage:

```
dbcAmplicons abundance [-h] [-r <arg>] [-t VALUE] [-S FILE]
       [-O FILE_PREFIX] -F FILE [FILE ...]  [-v] [--debug]
```

Options:
`-h, --help` Show this help message and exit
`-r <arg>, --rank` Taxonomic rank from which to build table, allowable values: domain, phylum, class, order, family, genus, and species (if performed), (default:genus)
`-t VALUE, --integer` Threshold bootstrap value to use for assignment, first taxon level greater than threshold
`-S FILE, --sample_metadata` File with primers
`-O FILE_PREFIX, --output_path` Path for output files (default:table)
`-F FILE [FILE ...]` Fixrank formatted classification file generated by *dbcAmplicons classify*
`-v, --silent` Verbose output (default:False)
`--debug` Show traceback on error

3.4.1 Outputs

dbcAmplicons abundance will generate three `.txt` files. The `abundance.txt` file lists the taxon abundance with the read counts. The `proportions.txt` lists the proportions of reads per sample, and the `taxa_counts.txt` lists a summary of the lowest level classification and the number of counts for that level.

4 Supplementary Scripts

4.1 reduce_amplicons

A script for separating the already processed amplicons into 1 or more of the following categories: consensus, ambiguities, occurrence. The consensus catagory will take the sequence that occurs most frequently as the consensus and return that sequence. The ambiguities category will return a sequence that includes IUPAC ambiguity codes if specified by the user. The last category, occurrence, returns the number of times a given sequence was found that matched the minimum requirements listed by the user.

```
reduce_amplicons
```

Options:
`-h, --help` Show this help message and exit
`-p PROGRAM` Comma separated list of the functions of consensus, ambiguities, occurrence (default: consensus)
`-s MIN-SEQ, --integer` Minimum number of reads per bin (default:5)
`-f MIN-FREQ, --frequency` Minimum frequency of reads per bin (default:0.05)

`--trim-1=TRIM-1` Number of bases to trim from the end of read 1 (default:0)
`--trim-2=TRIM-2` Number of bases to trim from the end of read 2 (default:0)
`-r, --reuse` Reuse the reads within the folder (default:False)
`-o OUTPUTDIR, --outputDir` Directory to place output files (default:(basename).reduced)
`-c CPUS, --integer` Number of processors to use; choose 0 to use all available cores (default:0)

4.2 convert2ReadTo4Read

A script for back-coverting two Illumina sequence files that have been processed for their barcodes back to a four read set to be reprocessed using *dbcAmplicons*. Usage:

```
convert2ReadTo4Read.py [-h] [--version] [-b BATCHSIZE]
          [-O PREFIX] [-u] [-v] [-1 read1 [read1 ...]]
          [-2 read2 [read2 ...]]  [--debug]

Options:
```

`-h, --help` Show this help message and exit
`--version` Show program's version number and exit
`-b BATCHSIZE, --batchsize` Batch size for read processing (default:100,000)
`-O PREFIX, --output_path` Path for output files (default:None)
`-u, --uncompressed` Leave output files uncompressed (default:False)
`-v, --silent` Verbose output (default:False)
`-1 read1 [read1 ...]` Read 1 of an amplicon fastq two file set
`-2 read2 [read2 ...]` Read2 of an amplicon fastq two file set
`--debug` Show traceback on error

4.3 splitReadsBySample

splitReadsBySample, a python script for splitting out Illumina sequence reads into individual sample files. The files must have already been processed by *dbcAmplicons*. Usage:

```
splitReadsBySample.py [-h] [--version] [-b BATCHSIZE] [-O PREFIX] [-u]
          [-v] [-1 read1 [read1 ...]]
          [-2 read2 [read2 ...]]
          [-U singleRead [singleRead ...]]  [--debug]
```

Options:
`-h, --help` Show this help message and exit
`--version` Show program's version number and exit
`-b BATCHSIZE, --batchsize` Batch size for read processing (default: 100,000)
`-O PREFIX, --output_path` Filenames for output files (default: DBCSample)
`-u, --uncompressed` Leave output files uncompressed (default: False)
`-v, --silent` Verbose output (default: False)
`-1 read1 [read1 ...]` Read1 of an amplicon fastq two file set
`-2 read2 [read2 ...]` Read2 of an amplicon fastq two file set
`-U singleRead [singleRead ...]` Fastq from a 1 file Illlumina run
`--debug` Show traceback on error