Applied BioInformatics Laboratory

A k-mer based approach for faster gene identification

Date: 5th September 2017

Table of Contents

Summary

The Mycotics Disease Branch (MDB) of CDC has a need for rapid profiling of antimicrobial resistance genes directly from raw fungal whole genome sequence reads. This report describes ABiL's adoption of a modified version of stringMLST for addressing this specific need. The current workflow for gene identification from raw sequence reads entails assembly of the genome sequences followed by sequence similarity searching. The assembly process is the bottleneck in this workflow and presents a significant challenge in scaling up to several hundreds of isolates. We demonstrate here that the stringMLST based approach is a faster and reliable alternative to this commonly used assembly based pipeline. We tested stringMLST on 48 *Candida auris* isolates and five genes (ERG11, ERG2, ERG3, FKS1 and MSH2) provided by the MDB. The modified version of stringMLST, **`abil-genecaller`**, was able to correctly predict all the alleles of the genes and took on average 20 seconds per isolate. **`abil-genecaller`** has been deployed at the MDB's working group folder on the scientific computing infrastructure at CDC (BioLINUX and Aspen).

Accessing the Script

The script is located on the SciComp shared folder: /scicomp/groups/OID/NCEZID/DFWEB/MDB/abil/abil-genecaller.py

For instructions on accessing and using SciComp resources, please refer to the extensive documentation provided by SciComp on this page: http://info.biotech.cdc.gov/?page_id=479

Algorithm Overview

The stringMLST algorithm works on exact k-mer matching paradigm. The algorithm fragments both the database gene sequences as well as the input read sequences. A k-mer counter tracks the number of k-mers from each read that matched each gene in the database. A minimum k-mer depth cutoff is determined based on the number of k-mers observed (directly proportional to sequence read depth) and the genes with the k-mer depth exceeding the cutoffs are reported as present. This approach is sensitive enough to distinguish between different allelic variants of a gene sequence. For further details and discussion of the original algorithm, please refer to Gupta et al Bioinformatics 2017 (PubMed ID: 27605103).

After testing the algorithm on a small set of 48 *C. auris* genomes, the following two major changes were implemented:

1. **`abil-genecaller`** verifies that every base in the called gene is covered and no positions have abnormally low coverage. Abnormally low coverage of individual positions or short segments (<5bp) may indicate a new allele or mixed population within the sequenced library. This change is borrowed from our on-going development of a related tool – STing.
2. **`abil-genecaller`** produces assemblies from the filtered reads. Using the filtered subset, which is typically on the order of a few thousands of reads instead of several millions of reads, significantly decreases the computational and time requirements for assembly. The resulting assemblies can be directly processed by the already deployed abil-blast pipeline for rapid annotation and analysis.

Benchmarking Results on *Candida auris* Genome Sequences

`abil-genecaller` was tested on 48 *C. auris* genome sequences and the results obtained were verified by genome assembly followed by BLAST based similarity searching. The early implementation of **`abil-genecaller`** was able to detect all of the AMR genes correctly (100% accuracy) and 19 seconds/isolate and utilizing a peak memory of 3.33GB (minimum peak memory recorded = 171.77MB; see table 1). As noted above, **`abil-genecaller`** was later supplemented to assemble sequence reads that were detected to contain the gene of interest. This added approximately 2 min/isolate of additional runtime which was still significantly lower than that of an assembly based approach (>2 hours/isolate).

Table 1. Prediction results from **`abil-genecaller`** on 48 *C. auris* isolate samples.

Sample	# Reads	ERG11	ERG2	ERG3	FKS1	MSH2	Time (in sec)
SRR3883426	7,378,048	WT	WT	WT	WT	WT	24.9
SRR3883427	6,750,800	WT	WT	WT	WT	WT	21.63
SRR3883428	4,869,764	WT	WT	WT	WT	WT	15.68
SRR3883429	5,275,382	R	WT	WT	WT	WT	16.74
SRR3883430	5,385,520	R	WT	WT	WT	WT	16.89
SRR3883431	5,484,030	WT	WT	WT	WT	WT	17.94
SRR3883432	5,467,638	WT	WT	WT	WT	WT	17.26
SRR3883433	6,006,818	WT	WT	WT	WT	WT	19.36
SRR3883434	7,516,626	R	WT	WT	WT	WT	23.98
SRR3883435	7,979,960	R	WT	WT	WT	WT	25.87
SRR3883436	7,067,462	R	WT	WT	WT	WT	22.64
SRR3883437	6,201,112	WT	WT	WT	WT	WT	19.88
SRR3883438	3,856,010	R	WT	WT	WT	WT	12.83
SRR3883439	6,175,248	R	WT	WT	WT	WT	20.35
SRR3883440	6,382,642	R	WT	WT	WT	WT	21.08
SRR3883441	6,936,578	WT	WT	WT	WT	WT	21.74
SRR3883442	5,528,992	WT	WT	WT	WT	WT	19.14
SRR3883443	6,543,742	WT	WT	WT	WT	WT	20.86
SRR3883444	7,761,356	WT	WT	WT	WT	WT	24.9
SRR3883445	5,365,210	WT	WT	WT	WT	WT	17.04
SRR3883446	6,966,864	WT	WT	WT	WT	WT	22.77
SRR3883447	5,812,088	WT	WT	WT	WT	WT	18.87
SRR3883448	5,786,280	WT	WT	WT	WT	WT	17.58
SRR3883449	3,923,556	R	WT	WT	WT	WT	12.42
SRR3883450	7,586,682	WT	WT	WT	WT	WT	22.89
SRR3883451	6,710,252	R	WT	WT	WT	WT	21.52
SRR3883452	7,055,498	R	WT	WT	WT	WT	22.1
SRR3883453	6,236,022	R	WT	WT	WT	WT	20.03
SRR3883454	5,755,636	R	WT	WT	WT	WT	18.57
SRR3883455	4,843,360	R	WT	WT	WT	WT	15.32
SRR3883456	5,568,522	R	WT	WT	WT	WT	17.52
SRR3883457	5,867,242	R	WT	WT	WT	WT	18.92
SRR3883458	4,823,366	R	WT	WT	WT	WT	16.62
SRR3883459	6,486,732	R	WT	WT	WT	WT	30.03
SRR3883460	6,067,922	WT	WT	WT	WT	WT	19.6
SRR3883461	7,248,118	R	WT	WT	WT	WT	23.33
SRR3883462	6,696,652	R	WT	WT	WT	WT	21.11
SRR3883463	5,125,488	R	WT	WT	WT	WT	16.41

Sample	# Reads	ERG11	ERG2	ERG3	FKS1	MSH2	Time (in sec)
SRR3883464	4,758,482	WT	WT	WT	WT	WT	15.06
SRR3883465	6,030,042	WT	WT	WT	WT	WT	18.62
SRR3883466	5,941,964	WT	WT	WT	WT	WT	18.49
SRR3883467	5,294,772	WT	WT	WT	WT	WT	16.89
SRR3883468	7,370,926	WT	WT	WT	WT	WT	23.25
SRR3883470	5,968,482	WT	WT	WT	WT	WT	18.91
SRR3883471	5,907,784	R	WT	WT	WT	WT	18.92
SRR3883472	6,008,014	WT	WT	WT	WT	WT	19.75
SRR3883473	353,863	R	WT	WT	WT	WT	1.2
SRR3883474	2,031,618	WT	WT	WT	WT	WT	9.7

Limitation of the approach

A known limitation of this approach, and many sequence based approaches, is a high false positive rate when paralogs of the gene of interest are present within the genome. E.g., if the user is interested in detecting ERG2 and a paralog of ERG2 exist within the same genome (demonstrating considerable sequence similarity), **`abil-genecaller`** will pick up noticeable amount of k-mers matching a fragment of sequence and may incorrectly infer the presence of the gene. Another related instance where this problem may be observed is when a domain is shared between the gene of interest and another gene within the genome. This is in part resolved by ensuring that each base of the gene is covered by some k-mers.

Instructions for Running `abil-genecaller`

There are two steps for running **`abil-genecaller`**: 1) database building and 2) gene prediction. As the name suggests, the database building step creates the database that will be used for predicting genes in the next step. The database building step is only required once for a given set of gene sequences (*i.e.*, gens of interest). This step is not to be repeated unless there is an update/addition of newer gene sequences. The gene prediction step utilizes the database generated in the previous step and will be executed once per isolate. This step can be easily parallelized by using basic BASH based utilities.

Database Building

In order to run **`abil-genecaller`**, two types of input files are required from the user: 1) FASTQ reads in which the genes are to be detected, 2) a sequence database stored in a series of FASTA files and described in a text file (see below). The FASTQ files are expected to be in standard FASTQ format as produced by the Illumina machine.

Format Description: FASTQ

- FASTQ is a text based format for storing sequence reads
- Each read is stored as four lines as shown below:

Sequence ID	`@SEQ_ID`
Sequence Read	`GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT`
Description Line	`+`
Encoded Quality	`!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65`

- The first line of each sequence read should always starts with @ and the third line should always starts with +
- The first line contains the mandatory sequence identifier and the third line can optionally contain the sequence description
- The fourth line contains the sequence quality encoded into American Standard Code for Information Interchange (ASCII) with some offset

The user needs to select the genes of interest and create a database for it. For creating the database, each gene is required to be stored in a separate FASTA file and described in a text file. All the available variants for a gene are required to be placed within the same FASTA file. E.g., if the user was interested in ERG11, ERG2 and FKS1, then the user is required to create three files: ERG11.fasta, ERG2.fasta and FKS1.fasta. The ERG11.fasta will contain all available/known ERG11 sequences, ERG2.fasta will contain all ERG2 variants and FKS1 will contain all the FKS1 variants. Two important points to note here:

1) An exhaustive listing of variants is not required till a sufficient space of the variants is covered. I.e., if all the known sequences that are at least 95% identical are identified and included, **`abil-genecaller`** should be able to detect the gene accurately. The variant can be identified using the assembled fragment.
2) It is not necessary to set the filenames as *genename.fasta* format. However, this notation is recommended to avoid confusion at a later stage.

These files are briefly described/mapped in a **configuration file**. A configuration file is a text file that looks as following:

Format Description: Configuration file and Profile file

- Configuration file has two sections: loci and profile
- Loci section contains the genes that are to be detected and the file they are placed in
- The profile section similarly maps the profile file
- The profile file is required even when it is not utilized
- Proceeding with the previous example of ERG11, ERG2 and FKS1 genes, the config file should look like following:

```
[loci]
ERG11   /path/to/erg11.fasta
ERG2    /path/to/erg2.fasta
FKS1    /path/to/fks1.fasta

[profile]
profile /path/to/profile.txt
```

- **Please use full paths to ensure that the software is able to find the correct files**

- The information present in the profile file will not be used in gene detection exercise but it is essential if a typing scheme is devised based on the variants present at each of the loci
- The format for the profile file is as follows:

```
ST      ERG11    ERG2    FKS1
0       1        1       1
```

- If the user doesn't plan on using the profile file, a minimum of two lines are required in the file as shown above
- If the user plans on using the profile file, he/she can assign each variant allele combination an ST value and the **`abil-genecaller`** will produce them as part of the output

Regarding the profile file, one of the basic ability of stringMLST is the detection of sequence type which is the combination of the allelic variants present within an isolate. Since this is not the focus here, this file can be kept as minimal.

Once these files have been, the database can be generated by executing the following command:

```
abil-genecaller.py --buildDB -c config.txt –P database
```

The --buildDB specifies that a database is to be build and -c argument specifies the location of the config file.

This will generate a three flat files which together serve as the database. Each of these files will have the prefix kmer by default. For a given gene set, this step only needs to be executed once. Once the database has been created, it can be reused for all future searches of the same genes. We recommend updating the FASTA files used for database creation to add all the recently detected sequences to ensure accurate prediction. This should be done at least twice a year.

Gene Prediction

Once the database has been constructed, the genes can be detected using the predict step. A minimal predict command looks like following:

```
abil-genecaller.py --predict --config config.txt -1 reads_1.fq -2 reads_2.fq -P database
```

A video walkthrough of this process is available here: https://asciinema.org/a/WE8TA5m1CT3WEAValMWciDvLi

This command specifies that gene prediction is to be performed (predict), using the config file (--config; this enables the coverage calculation) and the database (-P database) on the files reads_1.fq and reads_2.fq (-1 and -2).

If all the files in a folder are to be processed in a serial manner, the following variant of the command can be used:

```
abil-genecaller.py --predict --config config.txt -d directory -p -P database
```

This command specifies that gene prediction is to be performed (predict), using the config file (--config; this enables the coverage calculation) and the database (-P database) on all the files in the directory (-d) and they are all paired (-p).

The predicted genes will be output in a tab separated format as follows:

```
Sample  ERG11   ERG2    FKS1
SampleName      WT      WT      WT
```

Detailed Usage Instructions

The detailed usage for the program are listed below and can also be obtained by running the script without any arguments. Being a variant of stringMLST, most of these arguments are similar to that of stringMLST.

```
[--buildDB]
[--predict]
[-1 filename_fastq1][--fastq1 filename_fastq1]
[-2 filename_fastq2][--fastq2 filename_fastq2]
[-d directory][--dir directory][--directory directory]
[-l list_file][--list list_file]
[-p][--paired]
[-s][--single]
[-c][--config]
[-P][--prefix]
[-z][--fuzzy]
[-a]
[-C][--coverage]
[-k]
[-o output_filename][--output output_filename]
[-x][--overwrite]
[-t]
[-r]
[-v]
[-h][--help]
===============================================================================
```

buildDB

```
Synopsis
abil-genecaller.py --buildDB -c <config file> -k <kmer length(optional)> -P <DB
prefix(optional)>

Required arguments
--buildDB
    Identifier for build db module
```

```
-c,--config = <configuration file>
    Config file in the format described above.
```

Optional arguments

```
-k = <kmer length>
    Kmer size for which the db has to be formed(Default k = 35). Note the tool works
best with kmer length in between 35 and 66 for read lengths of 55 to 150 bp. Kmer size
can be increased accordingly. It is advised to keep lower kmer sizes if the quality of
reads is not very good.

-P,--prefix = <prefix>
    Prefix for db and log files to be created(Default = kmer). Also you can specify
folder where you want the db to be created.

-h,--help
  Prints the help manual for this application
```

==

predict

Synopsis

```
abil-genecaller.py --predict -1 <fastq file> -2 <fastq file> -d <directory location> -
l <list file> -p -s -P <DB prefix(optional)> -k <kmer length(optional)> -o <output
file> -x
```

Required arguments

```
--predict
    Identifier for predict module
```

Optional arguments

```
-1,--fastq1 = <fastq1_filename>
  Path to first fastq file for paired end sample and path to the fastq file for single
end file.
  Should have extension fastq or fq.

-2,--fastq2 = <fastq2_filename>
  Path to second fastq file for paired end sample.
  Should have extension fastq or fq.

-d,--dir,--directory = <directory>
  BATCH MODE : Location of all the samples for batch mode.


-C,--coverage
    Calculate sequence coverage for each allele. Turns on read generation (-r) and
turns off fuzzy (-z 1)
    Requires bwa, bamtools and samtools be in your path

-k = <kmer_length>
  Kmer length for which the db was created(Default k = 35). Could be verified by
looking at the name of the db file.
  Could be used if the reads are of very bad quality or have a lot of N's.

-l,--list = <list_file>
  LIST MODE : Location of list file and flag for list mode.
  list file should have full file paths for all the samples/files.
  Each sample takes one line. For paired end samples the 2 files should be tab
separated on single line.

-o,--output = <output_filename>
  Prints the output to a file instead of stdio.

-p,--paired
  Flag for specifying paired end files. Default option so would work the same if you
do not specify for all modes.
  For batch mode the paired end samples should be differentiated by 1/2.fastq or
1/2.fq
```

```
-P,--prefix = <prefix>
    Prefix using which the db was created(Defaults = kmer). The location of the db
could also be provided.

-r
  A separate reads file is created which has all the reads covering all the locus.

-s,--single
  Flag for specifying single end files.

-t
  Time for each analysis will also be reported.

-v
  Prints the version of the software.

-x,--overwrite
  By default abil-genecaller appends the results to the output_filename if same name
is used. This argument overwrites the previously specified output file.

-z,--fuzzy = <fuzzy threshold int>
    Threshold for reporting a fuzzy match (Default=300). For higher coverage reads
this threshold should be set higher to avoid indicating fuzzy match when exact match
was more likely. For lower coverage reads, threshold of <100 is recommended

-h,--help
  Prints the help manual for this application

=======================================================================================
```