microTyper

genotyping microhaplotypes

Genotyping using reads from amplicon sequencing.

Reads bam files using the bamtools API (Barnett et al. 2011, https://doi.org/10.1093/bioinformatics/btr174), which is bundled with the microTyper code for ease of install.

Installation and example

First, we unpack and compile

```
tar -xzvf microTyper.tar.gz
cd microTyper/
cmake .
make
```

Note that cmake 3.9 or higher is required. On some machines, the `cmake` command may need to be replaced with `cmake3`.

Now, navigate to the example folder

```
cd example/
```

Here, there are three samples, a position file, and a reference file. We then count reads of each possible allele for each sample and generate a list of all possible genotypes

```
../mtype -f *.bam -p examplePositionFile.txt -r exampleReference.fasta --count
```

Then, we call genotypes using a minimum posterior probability of .99 and a minimum number of reads that perfectly match one of the alleles in the genotype of 10

```
../genoCaller -f microtyper_llh.mhgenos -c .99 -m 10 --count
```

If we don't want to save the intermediate output file, we can pipe the output from one to the other (`-o -` specifies writing output to stdout, and `-f -` specifies reading from stdin):

```
../mtype -f *.bam -p examplePositionFile.txt -r exampleReference.fasta -o - --count |
 ../genoCaller -f - -c .99 -m 10 --count
```

An example of using the "original" method which has a different error model. This is not recommended unless you know what you are doing and why you are using it.

```
../mtype -f *.bam -p examplePositionFile.txt -r exampleReference.fasta -o - |
 ../genoCaller -f - -c .99 -m 10
```

Manual

microTyper is intended to genotype individual samples for microhaplotypes using aligned reads from amplicon sequencing. The goal is to provide a straightforward program that takes reasonble inputs and outputs genotypes in a format that is easy to use as input for downstream analyses and is amenable to the level of automation required by high-throughput genotyping projects.

microTyper optionally uses the openMP library for parallel processing. If the openMP library is NOT detected during compilation, microTyper will still compile and function, but will be limited to one thread.

microTyper has two options for analysis, which mainly differ in their error model. The "count" method assumes that there is a fixed probability that a read is an error (ie, different from the allele that was sequenced), and if it is an error, it is equally likely to be any other allele. The "original" method assumes that there is a fixed probability that each base is an error plus the error rate calculated from the PHRED score. If the true allele being sequenced is ATA, the "original" method considers a read for ATG more likely than a read for AAG, whereas the "count" method considers both errors equally likely. The "count" method seems to be more robust, particularly in the presence of low levels of index hopping (or contamination), and so I generally recommend using it.

All phase information comes from reads that span the entire locus. No phasing algorithm is used.

The required inputs are a file giving information about known variable positions and one or more bam files. Each bam file should represent one diploid individual. Only the known variable positions will be considered, all other positions are ignored.

The first step:

```
mtype -f sample.bam -p positionFile.txt -r reference.fasta --count
```

Required arguments

- `-f` space separated list of input bam files. For example:
  - `-f sample1.bam`
  - `-f sample1.bam sample2.bam`
  - `-f ./bamfiles/*.bam`
- `-p` the "position file" giving information about the microhaplotypes to genotype (see below for details)
- `-r` a fasta file with the reference sequences that reads were mapped to. Each reference sequence is assumed to be one locus (one microhaplotype).

Optional arguments

- `--count` to skip calculation of log-ilkelihoods for the "original" method. This speeds up computation considerably, but is only applicable if you also specify the `--count` option in the second step.
- `-o` the name to give the output file (default: microtyper_llh.mhgenos).
  - To write output to stdout, use `-o -`
- `-eps_S` the probability of an incorrect base being present in a read prior to sequencing error (error due to PCR error during amplification, index hopping, etc.) (default: 0.01)
- `-eps_I` the probability of an indel error in a read (default: 0.01)
- `-b` the "batch size" to use when multithreading. in order to control memory usage, individuals are genotyped in batches and then results written out. Larger batches make processing quicker but use more memory. (default: 100)
- `-t` the number of threads to use. Ignored if openMP was not found during compilation. (default: 1)
- `--manySNPs` If your loci have a very, very large number of SNPs in them, it is imaginable that without this option, log-likelihoods will not all be able to be calculated by the normal routine. This will result in "nan" being present in the output file where the log-likelihood should be. If that is the case, using this option will calculate the log-likelihoods, but will be slower than otherwise.
- `--version` print the version of microTyper being used and exit

The second step (calculating posterior probabilities and calling genotypes):

```
genoCaller -f microtyper_llh.mhgenos --count
```

Required arguments

- `-f` the file containing log-likelihoods that was output during step 1. The order of the lines in this file is important, so do not reorder any lines after running step 1. To read from stdin, use `-f -`

Optional arguments

- `--count` to use the "count" method for calling genotypes. Omit to use the "original" method.
- `-o` the name to give the output file (default: mh_genotypes.txt)
- `-p` a file specifying the priors to use, see below for format. (default: uniform priors)
- `-c` the minimum posterior probability the most probable genotype must have (for a given individual and locus) to be accepted (default: 0.95)
- `-m` the minimum number of reads that match one of the alleles in the most probable genotype at all SNPs for that genotype to be accepted (default: 0)
- `--version` print the version of microTyper being used and exit

The position file

The position file gives information about the SNPs within each locus. It is fairly straightforward to create from a VCF file containing your target SNPs. It is a tab-delimited file with a REQUIRED header row (actual text in header row is not important). Each line defines a SNP. The columns are, in order,

Locus RefPos Type ValidAlt

- Locus: the name of the reference sequence for the locus containing this SNP
- RefPos: the position of the SNP in the reference sequence
- Type: the type of SNP, relative to the reference. One of "S", "I", or "D" for substitution, insertion, or deletion, respectively
- ValidAlt:
  - for substitution SNPs, a comma separated list of alternative bases to look for in the read (A,C,T, or G). Bases must be capitalized. The reference base is pulled automatically from the reference fasta file.
  - for insertion SNPs, a character (not "D") to use in the output when an insertion is found. The reference allele is automatically designated as "D".
  - for deletion SNPs, a character (not "I") to use in the output when a deletion is found. The reference allele is automatically designated as "I".

The prior file

The prior gives the prior probability for each genotype of each locus. It is a tab-delimited file with a REQUIRED header row (actual text in header row is not important). Each line describes a genotype. The columns are, in order,

Locus Allele1 Allele2 Prior

- Locus: the name of the reference sequence for the locus
- Allele1: the first allele in this genotype
- Allele2: the second allele in this genotype
- Prior: The prior probability

After being read, the priors are normalized within each locus, and so these may be relative probabilities. Priors must not be zero, but can be very small. Priors can be expressed using exponential notation (i.e. 1e-20).

The entries for Allele1 and Allelle2 must match exactly the entries present in the output file created by step 1 (with the log-likelihoods for each genotype). **The easiest way to make this file is to run the first step with one input bam file, then delete the "Indiv", "A1_perfect_count", and "A2_perfect_count" columns, and then replace the LLH values with your prior values.** There is an example prior input file in the "examples" directory.

Output file formats

The output of the first step is a file containing the log-likelihood of each genotype for each locus in each individual calculated using the "original" method. It also has two columns containing the counts of reads that match each allele of each genotype at every SNP in the locus. The "A1_perfect_count" column has the number of reads that match Allele1 at all SNPs in the locus. If the genotype is a homozygous genotype, the "A2_perfect_count" column is 0. Otherwise, the "A2_perfect_count"

column has the number of reads that match Allele2 at all SNPs in the locus. This file is mainly used as the input for the second step, but is output for users who may want to use the likelihoods or perfect match read counts directly. If a user wants to examine the number of reads that are perfect matches for each allele, this can be extracted by using the values in "A1_perfect_count" for all homozygous genotypes. If `--count` is used, only the line with homozygous genotypes are given allele counts, and the log-likelihoods are not calculated during this step. This option reduces compuation time, but can be only be used if `--count` is also used for the second step.

The output of the second step contains one line for each individual and each locus. If the most probable genotype for a locus in a given individual passes the filters (-c and -m), it is included in the output in the "Allele1" and "Allele2" columns. If it does not pass the filters, those two columns are blank. Regardless of whether it passed the filters or not, the posterior probability of the most probable genotype is in the "Pr_geno" column. The "A1_perfect" column has the number of reads that match Allele1 at all SNPs in the locus. If the genotype is a homozygous genotype, the "A2_perfect" column is 0. Otherwise, the "A2_perfect" column has the number of reads that match Allele2 at all SNPs in the locus.