microTyper v2.0

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. If you want to call genotypes with a minimum depth of 10 and threshold posterior probability of .99 and save them to file "genos.txt"

```
../mtype2 -f *.bam -p examplePositionFile.txt -r exampleReference.fasta -d 10 -c .99 -o genos.txt
```

Let's say these were actually triploid samples and you wanted a minimum depth of 15

```
../mtype2 -f *.bam -p examplePositionFile.txt -r exampleReference.fasta -ploidy 3 -d 15 -c .99 -o genos_triploid.txt
```

And now let's say we wanted to see all posterior probabilities for all genotypes. Note that *only alleles detected in a given individual are considered for that individual*. This allows the program to scale efficiently with the number of SNPs in a microhap locus without requiring predefined valid allele combinations or combining information (alleles detected) across samples. This is also one reason the minimum depth parameter is important (see comments in "Method" below).

```
../mtype2 -f *.bam -p examplePositionFile.txt -r exampleReference.fasta -o post_probs.txt --all
```

And what if you just want read counts for the alleles in each individual? Maybe you want to call genotypes a different way that isn't supported here! Or maybe you want to infer ploidy from read counts!

```
../mtype2 -f *.bam -p examplePositionFile.txt -r exampleReference.fasta -o readCounts.txt --justCount
```

Please note that while the code is intended to be applicable to all plodies, testing has mostly been performed with diploid samples and, to a lesser extent, triploid samples. You may want to perform some validation on your own. Additionally, almost all of the testing has utilized microhaplotypes that are composed of substitution SNPs. While the code is intended to be able to genotype indels as well, performance with indels has not been exhaustively tested. You may want to perform some validation on your own. Please alert me of any errors or bugs that you find.

Manual

Goal

microTyper is intended to genotype individual samples for microhaplotypes (and SNPs, which are really just short microhaplotypes) using aligned reads from amplicon sequencing. The goal is to provide a straightforward program that

- takes reasonble inputs
- outputs genotypes in a format that is easy to use as input for downstream analyses
- is amenable to the level of automation required by high-throughput genotyping projects

It explicitly does not utilize algorithms that "share information" across samples as we want to avoid assuming that samples relevant to a project are all sequenced (and genotyped) on the same library or even that there are sufficient numbers of samples being genotyped at the same time for such algorithms to function well.

It also explicitly does not perform SNP discovery.

Parallel processing

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.

Method

microTyper uses aligned reads. It only uses single-end reads mapped to the forward strand. Reads should be aligned to "reference amplicons" with each amplicon representing one microhaplotype to call. The required inputs include a file giving information about known variable positions and, for substitution SNPs, the valid alternate alleles (reference alleles are pulled from the reference sequences). Only the known variable positions (and the known alleles at these positions) will be considered. All other positions are ignored.

For a given amplicon *within a given individual*, microTyper first counts reads for each allele. All phase information comes from reads that span the entire locus. No phasing algorithm is used. Reads that do not span the entire locus are not used. microTyper only considers the alleles detected within that individual. It does **not** consider the set of all alleles created by all possible combinations of the target SNPs (this would quickly become memory/computation intensive as the number of SNPs in a locus increases) and it does **not** first detect alleles across all individuals being genotyped (see comments above about avoiding sharing information across individuals).

After counting reads for each allele, if the depth of the locus is higher than the user input minimum depth, microTyper calculates the likelihood of each possible genotype by assuming read counts are a multinomial random variable. Probabilites for each allele are calcuated from the allele dosage in the given genotype, the user input error rate (`-eps`), and assuming that if a read is an error, it is equally likely to be any other allele. The set of all possible genotypes is created using the user input ploidy and the set of alleles detected in that individual. The posterior probability of each genotype is calcualted by applying a uniform prior (all genotypes equally likely). This is also just a scaled likelhood (scaled to sum to 1). If the largest posterior probability is greater than the user input value (`-c`), the genotype is output.

Considering only alleles detected in a given individual allows the program to execute much more efficiently than it otherwise would. The obvious downfall is: how do you know you've detected all alleles in that individual? The solution is to use a reasonable minimum depth. In amplicon sequencing, depth is usually fairly high, and so we can use that to our advantage. Considering a diploid, we may worry about not detecting allele "A" in an individual with genotype "AB" given a selected minimum depth of *d*. The probability of this can be modelled as the probability a binomial random variable with *d* trials and probability of success of 0.5 is 0. Considering *d* = 10, this probability is 0.0009765625, or about 0.1%. If we reframe the problem as "given a genotype of "AB", what is the probability I only detect one allele", we double this to 0.001953125, or about 0.2%. Depending on whether that is a low enough probability for you or not, you can adjust *d*. Similar logic applies to other ploidies, and will suggest larger values of *d* for larger ploidies. It is also worth noting that if you observed *d* counts of allele A and 0 counts of allele B, even if you knew both alleles A and B existed, you (your model) probably would not call a genotype of "AB", and if you (your model) are not confident calling a genotype of "AA" given that data, then you are sort of using a d value higher than what you set and you might as well increase d.

Evaluating multinomial likelihoods is very efficient, and for many loci it seems to be a good approximation of how they behave. This is why it is used here (and in many, many, many other genotyping methods). One drawback is that some loci may exhibit overdispersion or allelic bias. When designing an amplicon sequencing panel, these loci may need to be filtered out if the magnitudes are such that they cause problems for calling genotypes. Alternatively, you could use microTyper to obtain read counts (`--justCount`), and then write a method that handles overdispersion and allelic bias.

Detailed options:

```
mtype2 -f sample.bam -p positionFile.txt -r reference.fasta
```

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 (and one microhaplotype and one amplicon).

Optional arguments

- `-o` the name to give the output file (default: microtyper_llh.mhgenos).
  - To write output to stdout, use `-o -`
- `-eps` the probability of an read beign from the wrong allele. In other words, an error (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)
- `-ploidy` to specify the ploidy of individuals being genotyped (default: 2)
- `-c` the threshold posterior probability for calling genotypes (default: 0.99)
- `-d` the minimum depth to call a genotype. Depth is the sum of all reads matching any valid allele, whether or not that allele is in the called genotype or not. For example, if there are 120 A reads and 2 B reads, the genotype may be called AA, but the total depth is 122.
- `--all` report posterior probabilites for all genotypes. Implies ignoring `-c` and `-d` .
- `--justCount` to have the program just output read counts for observed alleles
- `--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. Note that *in the reference file the bases should be capitalized* (i.e., unmasked).
  - for insertion SNPs, this is ignored and the alternate allele is automatically designated as "I". The reference allele is automatically designated as "D".
  - for deletion SNPs, this is ignored and the alternate allele is automatically designated as "D". The reference allele is automatically designated as "I".

Output file formats

The output files have headers that are (hopefully) self explanatory. They give the input bam file name, the locus name, the alleles, the allele counts, and the posterior probability of a genotype. For genotypes with multiple copies of the same allele, the allele count for that allele is repeated multiple times in the corresponding columns.