mlgt

Dave T. Gerrard
david.gerrard@manchester.ac.uk

April 4, 2012

Abstract

Processing and analysis of high throughput (Roche 454) sequences generated from multiple loci and multiple biological samples. Sequences are assigned to their locus and sample of origin, aligned and trimmed. Where possible, genotypes are called and variants mapped to known alleles.

1 Introduction

The purpose of mlgt is to genotype multiple loci from multiple samples, where the sequence data is a single output file from a high throughput sequencing machine (e.g. the Roche 454 system). Of course, the samples, and markers need to have been properly prepared (amplified and tagged) prior to sequencing so that they can be sorted out again.

The data is expected to be full length sequences from mixed PCR amplicons. The amplicons from different samples will have been barcoded with unique end sequences (barcodes, MIDs) and these sequences need to be provided by the user. A set of reference amplicon sequences (markers) are also required, one for each amplicon.

Definitions

allele Top variants called by mlgt OR alleles from an external source.

amplicon The sequence segment amplified in PCR.

barode/MID Short sequence tags ligated to ends of amplicons.

marker A reference sequence against which all variants are aligned.

sample The biological samples.

A note on markers. You should use a sequence that is contained within the expected amplicon. I recommend NOT including the primer sequence as this is constrained by the primers used and may not reflect the true sequence of the samples. If you wish to compare your variants with alleles from an external source (e.g. the HLA database) then your marker sequence should be precisely bounded by the sequence that is in BOTH your amplicon and the allele sequences (e.g. the exon only part is good).

Currently, mlgt makes no checks for similarity across markers, but it does assign all sequences to the best possible marker, even if that marker was never amplified in your dataset. This is particularly an issue with HLA datasets where supposedly distinct loci have very similar sequences.

2 Installation

mlgt runs in R version 2.13 or greater [1]. mlgt depends on another R package (*seqinr*) and several external applications. These must all be installed and working for mlgt to work.

To assign sequences and retrieve specific sequences, mlgt makes use of the NCBI programs *formatdb*, *blastall* and *fastacommand*, which are available here: `ftp://ftp.ncbi.nih.gov/blast/executables/release/2.2.24/`

To align sequence variants, mlgt uses *MUSCLE*, which is available here: `http://www.drive5.com/muscle/downloads.htm`

Download the correct versions for your system and follow the installation instructions (if any). Make a note of the installation directories or, even better, specify where you want the programs to be installed. **N.B.** mlgt does not cope well with whitespace (gaps) when passing path names to the auxillary programs - please install the programs in a location that does not feature whitespace, if you can.

On my machine the formatdb program is in "C:/Users/Public/Apps/Blast/bin/formatdb.exe" and MUSCLE is in "C:/Users/Public/Apps/Muscle/muscle3.8.31_i86win32.exe"

The R packages *mlgt* and *seqinr* are available on the public R repository CRAN and can be installed from within R. You can do this from the packages menu or using this command, which also installs *seqinr*:-

```
> install.packages("mlgt")
```

To install a different version of mlgt from within R you can do this from the packages menu (Install packages from local zip files) or using this command giving the full path to the package zip archive.:-

```
> install.packages("mlgt_0.15.zip", repos=NULL)
```

3 Using mlgt

Once installation is complete, you can begin to use mlgt. Load the library.

```
> library(mlgt)
```

You will also need to specify the locations of the auxillary programs. They can be set as environment variables.

```
> Sys.setenv(BLASTALL_PATH="C:/Users/Public/Apps/Blast/bin/blastall.exe",
+                 FORMATDB_PATH="C:/Users/Public/Apps/Blast/bin/formatdb.exe",
+                 FASTACMD_PATH="C:/Users/Public/Apps/Blast/bin/fastacmd.exe",
+                 MUSCLE_PATH="C:/Users/Public/Apps/Muscle/muscle3.8.31_i86win32.exe")
```

3.1 Prepare the analysis

Start each analysis in a clean directory, perhaps named for the sequencing run and nested within a folder of all runs for the project.

[1] It uses long variable names (>256 bytes) only implemented since R 2.13.

```
> analysisDir <-  "C:/Users/me/genoProject1/run1/analysis/"
> setwd(analysisDir)
```

You will need to create some variables to describe your sequencing run. You need a named list of the MIDs/barcodes used to mark each end of the amplicons and a list of samples. The easiest way to get this is from a fasta file containing the barcode sequences with each sequence annotated with the sample name you will use. In the example below, I load the barcodes from a common file and use a table to change the names to match the samples in this run. Example data are in the /data sub-directory of the mlgt package installation directory and can be found using `system.file()`. Finally, specify the location of the raw sequence file (fasta format) you want to analyse (**N.B.** The path to this file MUST NOT contain whitespace.).

```
> system.file("namedBarcodes.fasta", package="mlgt")

[1] "C:/Users/Dave/Documents/R/win-library/2.14/mlgt/namedBarcodes.fasta"

> # Load MIDs used to mark samples
> fTagList <- read.fasta(system.file("namedBarcodes.fasta", package="mlgt"),
+                           as.string=T)
> # Optionally, rename the barcodes to the samples used in this run
> sampleBarcodeTable <- read.delim(system.file("tableOfSampleBarcodeMapping.tab",
+                   package="mlgt"), header=T)
> names(fTagList) <- sampleBarcodeTable$sample[
+                           match(names(fTagList), sampleBarcodeTable$barcode)]
> # here we're using the same tags at both ends of the amplicons.
> rTagList <- fTagList
> #The names of the samples
> sampleList <- names(fTagList)
> # Load the marker sequences.
> myMarkerList <- read.fasta(system.file("HLA_namedMarkers.fasta", package="mlgt"),
+                           as.string=T)
> # The fasta file of sequence reads
> inputDataFile <- system.file("sampleSequences.fasta", package="mlgt")
```

Inspect what is stored in each variable by typing its name.

Now you can create an object of class *mlgtDesign* to hold all this information. Give the names of the variables you have just created for the marker list, the sample list and the MIDs. Also give a project name and a name for this run; this will help to identify the source of this object later on.

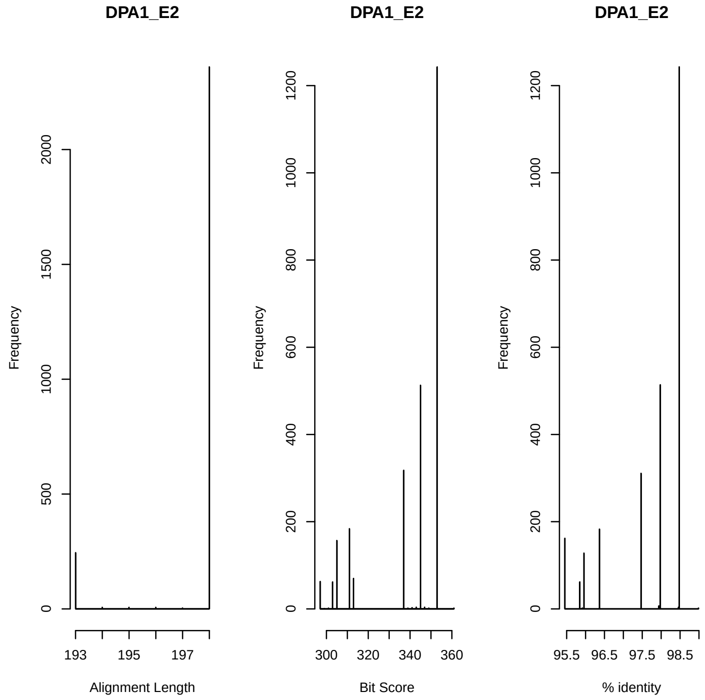
```
> # Creates object to store run settings
> my.mlgt.Design <- prepareMlgtRun(projectName="myProject",
+                                   runName="myRun", samples=sampleList,
+                                   markers=myMarkerList, fTags=fTagList,
+                                   rTags=rTagList, inputFastaFile=inputDataFile,
+                                   overwrite="yes")

myProject
Checking parameters...
Setting up BLAST DBs...
Running BLAST searches...
```

As this object is created, multiple BLAST databases are also created in the working directory and all the input sequences are BLASTed against the databases. These BLAST results are used to assign sequences to markers and samples.

It might be instructive to see how many sequences in your dataset are being assigned to each marker, especially if the marker list includes sequences which were meant NOT to be targetted by your primers.

```
> # inspect BLAST results for a specific marker
> thisMarker <- "DPA1_E2"
> topHits <- getTopBlastHits(my.mlgt.Design@markerBlastResults)
> #inspectBlastResults(topHits, thisMarker)
```

```
> inspectBlastResults(topHits, thisMarker)
```



Alternatively, print these plots for a set of markers to file. The function *printBlastResultGraphs* knows how to find the BLAST results from an object of class *mlgtDesign*.

```
> # automatic output to pdf of blast result graphs for a list of markers.
> printBlastResultGraphs(my.mlgt.Design)
```

3.2 Run mlgt

You can now proceed to extacting the sequences of the most common variants assigned to each marker/sample pair.

Run *mlgt* and save the results to file.

```
> my.mlgt.Result <- mlgt(my.mlgt.Design)

A_E3
Sample-1 : Using 45 variants, accounting for 154 of 154 reads
Sample-3 : Using 18 variants, accounting for 33 of 33 reads
Sample-4 : Using 132 variants, accounting for 418 of 418 reads
Sample-5 : Using 47 variants, accounting for 141 of 141 reads
Sample-6 : Using 49 variants, accounting for 142 of 142 reads
Sample-7 : Using 3 variants, accounting for 4 of 4 reads
Sample-8 : Using 40 variants, accounting for 128 of 128 reads
Sample-9 : Using 4 variants, accounting for 4 of 4 reads
Sample-10 : Using 55 variants, accounting for 149 of 149 reads
B_E2
Sample-1 : Using 57 variants, accounting for 141 of 141 reads
Sample-2 : Using 24 variants, accounting for 64 of 64 reads
Sample-3 : Using 23 variants, accounting for 54 of 54 reads
Sample-4 : Using 24 variants, accounting for 55 of 55 reads
Sample-5 : Using 37 variants, accounting for 57 of 57 reads
Sample-6 : Using 29 variants, accounting for 74 of 74 reads
Sample-7 : Using 1 variants, accounting for 1 of 1 reads
Sample-8 : Using 18 variants, accounting for 42 of 42 reads
Sample-10 : Using 38 variants, accounting for 89 of 89 reads
DPA1_E2
Sample-1 : Using 34 variants, accounting for 97 of 97 reads
Sample-2 : Using 94 variants, accounting for 254 of 254 reads
Sample-3 : Using 39 variants, accounting for 143 of 143 reads
Sample-4 : Using 30 variants, accounting for 430 of 526 reads
Sample-5 : Using 48 variants, accounting for 199 of 199 reads
Sample-6 : Using 85 variants, accounting for 339 of 339 reads
Sample-7 : Using 2 variants, accounting for 9 of 9 reads
Sample-8 : Using 30 variants, accounting for 486 of 609 reads
Sample-10 : Using 119 variants, accounting for 418 of 418 reads
DQA1_E2
Sample-1 : Using 27 variants, accounting for 104 of 104 reads
Sample-2 : Using 4 variants, accounting for 22 of 22 reads
Sample-3 : Using 6 variants, accounting for 8 of 8 reads
Sample-4 : Using 9 variants, accounting for 26 of 26 reads
Sample-5 : Using 11 variants, accounting for 34 of 34 reads
Sample-6 : Using 42 variants, accounting for 138 of 138 reads
Sample-7 : Using 1 variants, accounting for 1 of 1 reads
Sample-8 : Using 47 variants, accounting for 146 of 146 reads
Sample-9 : Using 1 variants, accounting for 1 of 1 reads
Sample-10 : Using 39 variants, accounting for 122 of 122 reads

> save(my.mlgt.Result, file="thisRun.mlgtResult.Rdata")
```

Have a look at the summary table for the run, this is located in the slot ‘runSummaryTable’

```
> my.mlgt.Result@runSummaryTable
```

```
   marker assignedSeqs assignedVariants minVariantLength maxVariantLength minAlleleLength maxAlleleLength
1    A_E3         1173              315                5              265             263             265
2    B_E2          577              188                5              250             247             250
3 DPA1_E2         2375              330                5              198             197             198
4 DQA1_E2          602              151                5              283             184             283
```

The results for each marker are stored in a list and can be accessed indidually using the marker name.

```
> thisMarker <- "DPA1_E2"
> my.mlgt.Result@markerSampleList[[thisMarker]]
```

```
    marker    sample rawTotal rawVars usedTotal usedVars numbSeqs numbVars varName.1 varFreq.1  varName.2 varFreq.2  varName.3
1  DPA1_E2  Sample-1       97      34        97       34       97       21 DPA1_E2.1        32  DPA1_E2.2        26  DPA1_E2.3
2  DPA1_E2  Sample-2      254      94       254       94      254       70 DPA1_E2.4        70  DPA1_E2.5        62  DPA1_E2.6
3  DPA1_E2  Sample-3      143      39       143       39      143       33 DPA1_E2.1        89  DPA1_E2.7        22  DPA1_E2.8
4  DPA1_E2  Sample-4      526     126       430       30      430       17 DPA1_E2.1       330  DPA1_E2.9        66 DPA1_E2.10
5  DPA1_E2  Sample-5      199      48       199       48      199       31 DPA1_E2.1       131 DPA1_E2.11        34 DPA1_E2.12
6  DPA1_E2  Sample-6      339      85       339       85      339       64 DPA1_E2.1       117 DPA1_E2.13        99 DPA1_E2.14
7  DPA1_E2  Sample-7        9       2         9        2        9        2 DPA1_E2.1         7 DPA1_E2.15         2         NA
8  DPA1_E2  Sample-8      609     153       486       30      486       12 DPA1_E2.1       187 DPA1_E2.16       183 DPA1_E2.17
9  DPA1_E2  Sample-9        0       0         0        0        0        0        NA         0         NA         0         NA
10 DPA1_E2 Sample-10      418     119       418      119      418       80 DPA1_E2.1       240 DPA1_E2.18        93 DPA1_E2.19
   varFreq.3
1         21
2         50
3          2
4          5
5          3
6         59
7          0
8         99
9          0
10         2
```

3.3 Call genotypes

The new *mlgtResult* object contains a table for each marker giving counts of unique variants for each sample including the counts of the most common 3 variants. The 'genotyping' has not yet been done. Genotyping is done by a separate function *callGenotypes* so that users can run different genotyping methods on the same *mlgtResult* object.

A simple example of calling genotypes is

```
> my.genotypes <- callGenotypes(my.mlgt.Result)
```

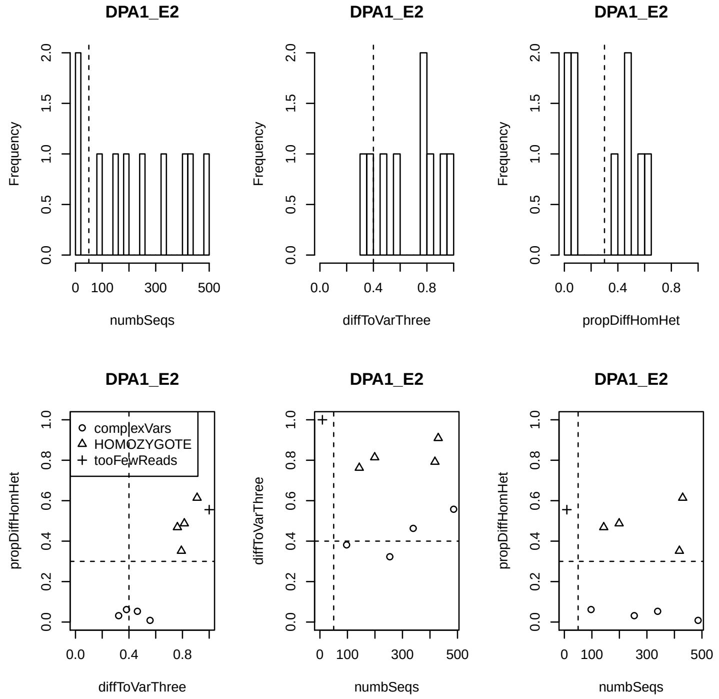
the result is the same table of variant counts with new columns to represent the genotype calls. N.B. currently, only one method is implemented but users can supply their own - use `?callGenotypes` to see details.

Once you have some genoytpes, you may want to export them to files. You can do this for individual markers or for all markers in the genotypeCall object.

```
> writeGenotypeCallsToFile(my.genotypes)
```

As with the BLAST results, it is also instructive to look at the distribution of statistics used in genotype calls. There is another function to plot the statistics.

```
> plotGenotypeEvidence(genotypeCall=my.genotypes[["DPA1_E2"]])
```



Again, these plots can be output to file, use `?plotGenotypeEvidence` to find out how.

3.4 Map to known allleles

The previous genotype calling was done without reference to previously known alleles. To map the newly tabulated variants to known alleles, a local variantMap of known alleles must be loaded or created. *mlgt* contains a function to build a list of alleles bounded by the marker sequence. **N.B.** the names of otherwise distinct alleles that are identical within the region overlapping the marker sequence are condensed to one sequence and names are concatenated.

Creating the allele map is a little more involved than other aspects but not too difficult with a little set up. You need to provide an alignment (msf or fasta format) of known alleles for each marker. The corresponding file names for each marker could be provided in table format in a file. The allele map should be a list with one *variantMap* element per marker. Each variantMap is created by running *createKnownAlleleList* to align each marker against the respective allele alignment file. Here, I download the alignment files direct from the HLA/IMGT ftp site into the current working directory. You might want to run this section separately in a different location because it only needs to be done once per set of markers or version of IMGT/HLA.

```
> markerImgtFileTable <- read.delim(system.file("marker.imgt.msf.list.tab", package="mlgt"),
+                                   header=T)
> alignFilesSource <- 'ftp://ftp.ebi.ac.uk/pub/databases/imgt/mhc/hla/'
> # select a folder to store the alignments in. Here using current working directory.
> alignFilesDir <- getwd()
> ## Download the allele alignments and create a 'variantMap' object for each marker and store them all in a list.
> knownAlleleDb <- list()
> for(thisMarker in names(myMarkerList)) {
+         fileName <-  markerImgtFileTable$imgtAlignFile[match(thisMarker, markerImgtFileTable$marker)]
+         alleleAlignUrl  <- paste(alignFilesSource , fileName , sep="/")
+         alleleAlignFile <- paste(alignFilesDir , fileName , sep="/")
+         download.file(alleleAlignUrl,alleleAlignFile)
+         knownAlleleDb[[thisMarker]] <- createKnownAlleleList(thisMarker,
+                 myMarkerList[[thisMarker]][1], alleleAlignFile)
+ }
```

Once you have the allele map, it's a good idea to save it as an 'RData' file for future use (`?save`, `?load`). Give it a name that describes the source of the known alleles and the marker used.

Now you can map variants to the new allele map. Run *callGenotypes* again with the option `mapAlleles=TRUE` and giving the name of the allele map.

```
> my.genotypes <- callGenotypes(my.mlgt.Result,  mapAlleles=TRUE,
+                  alleleDb=knownAlleleDb)
```

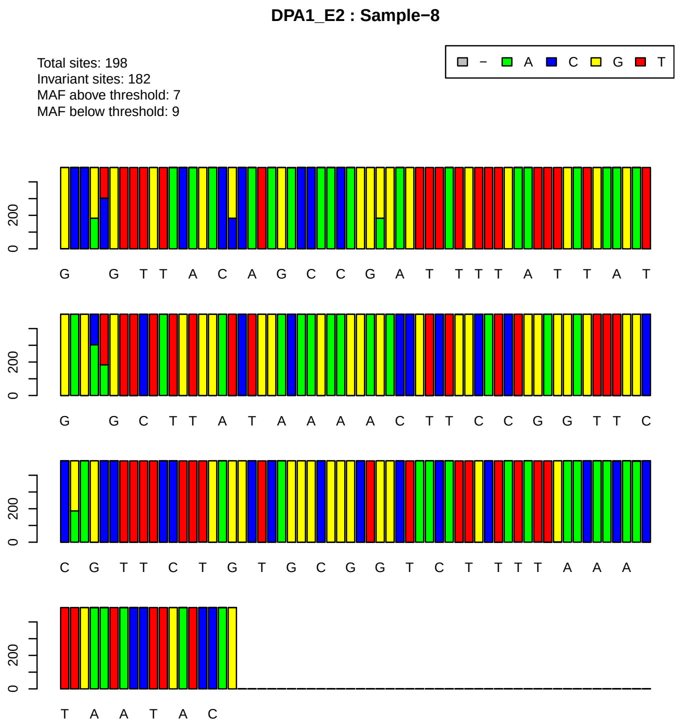
The result is the same table of genotypes as before but with additional columns giving the names of alleles mapped to the variants.

3.5 Error correction

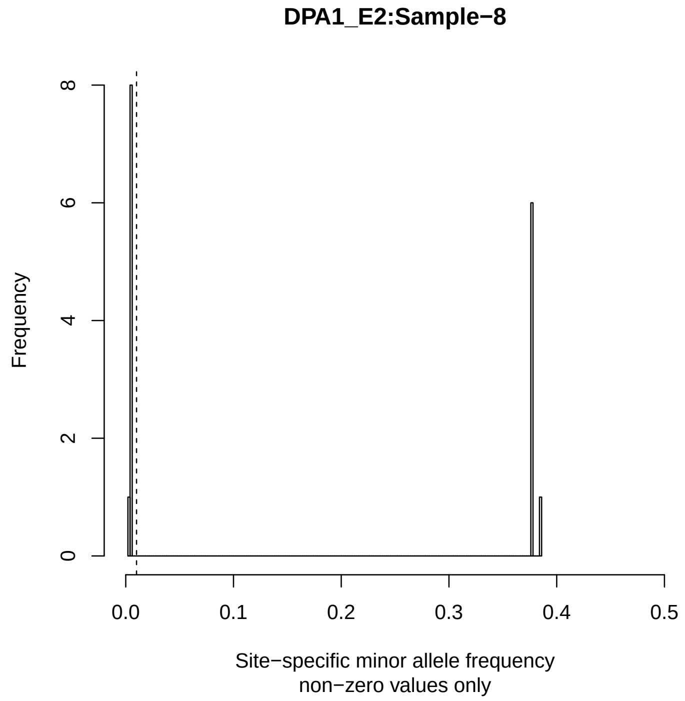
In some test data sets, we found very high numbers of unique variants for many marker-sample pairs. Most variants differed by only one or two positions from the most commonly found variants. The situation was worst for long sequences. Believing that this was due to errors introduced during amplification and/or sequencing, we sought to 'correct' some of the sequences. N.B. the following is conducted for sets of sequences from the same marker sample pair - information is not shared across samples or markers.

To help decide if error correction would be worthwhile, several alignment reports can be produced. The default is a table giving the alignment length, the number of invariant sites and the numbers of sites where the Minor Allele Frequency (MAF) is above or below a threshold. The MAF is the proportion of sequences with the SECOND most common variant across a set of sequences and is calculated site-by-site. The default threshold is 0.01. The function `alignReport()` can produce two graphics: 1) a profile of the alignment showing frequencies at every site, and 2) a histogram of site frequencies (a site frequency spectrum) with the default correction threshold shown as a dotted line.

```
> alignReport(my.mlgt.Result,markers="DPA1_E2", samples="Sample-8", method="profile")

DPA1_E2
$DPA1_E2
         numbSeqs numbVars alignLength invar.sites mafAboveThreshold mafBelowThreshold
Sample-8      486       12         198         182                 7                 9
```



```
> alignReport(my.mlgt.Result,markers="DPA1_E2", samples="Sample-8", method="hist")

DPA1_E2
$DPA1_E2
         numbSeqs numbVars alignLength invar.sites mafAboveThreshold mafBelowThreshold
Sample-8      486       12         198         182                 7                 9
```



The graphs for all markers and samples can be output to files using `alignReport` with the `fileName` option specified.

If the alignments have many sites below the correction threshold (i.e. very low frequency or unique variants) then `errorCorrect` can be used to change the bases at that position to the majority base. Any site with MAF above the threshold will be unchanged. If an alignment has fewer than 1/correctThreshold sequences, then errorCorrect will not attempt to make a correction (as none of the variants will have less than the threshold frequency).

It creates a new mlgtResult object but doesn't take very long:-

```
> my.mlgt.Result.Corrected <- errorCorrect(my.mlgt.Result)
```

```
A_E3
Sample-1  36 / 154 unique seqs in original alignment, 6 / 154 unique seqs in new alignment,
Sample-2  Sample-3  16 / 33 unique seqs in original alignment, 16 / 33 unique seqs in new alignment,
Sample-4  107 / 418 unique seqs in original alignment, 4 / 418 unique seqs in new alignment,
Sample-5  37 / 141 unique seqs in original alignment, 7 / 141 unique seqs in new alignment,
Sample-6  37 / 142 unique seqs in original alignment, 5 / 142 unique seqs in new alignment,
Sample-7  3 / 4 unique seqs in original alignment, 3 / 4 unique seqs in new alignment,
Sample-8  30 / 128 unique seqs in original alignment, 3 / 128 unique seqs in new alignment,
Sample-9  4 / 4 unique seqs in original alignment, 4 / 4 unique seqs in new alignment,
Sample-10  45 / 149 unique seqs in original alignment, 5 / 149 unique seqs in new alignment,
B_E2
Sample-1  44 / 141 unique seqs in original alignment, 6 / 141 unique seqs in new alignment,
Sample-2  21 / 64 unique seqs in original alignment, 21 / 64 unique seqs in new alignment,
Sample-3  17 / 54 unique seqs in original alignment, 17 / 54 unique seqs in new alignment,
Sample-4  19 / 55 unique seqs in original alignment, 19 / 55 unique seqs in new alignment,
Sample-5  23 / 57 unique seqs in original alignment, 23 / 57 unique seqs in new alignment,
Sample-6  23 / 74 unique seqs in original alignment, 23 / 74 unique seqs in new alignment,
Sample-7  1 / 1 unique seqs in original alignment, 1 / 1 unique seqs in new alignment,
Sample-8  16 / 42 unique seqs in original alignment, 16 / 42 unique seqs in new alignment,
Sample-9  Sample-10  24 / 89 unique seqs in original alignment, 24 / 89 unique seqs in new alignment,
DPA1_E2
Sample-1  21 / 97 unique seqs in original alignment, 21 / 97 unique seqs in new alignment,
Sample-2  70 / 254 unique seqs in original alignment, 3 / 254 unique seqs in new alignment,
Sample-3  33 / 143 unique seqs in original alignment, 3 / 143 unique seqs in new alignment,
Sample-4  17 / 430 unique seqs in original alignment, 3 / 430 unique seqs in new alignment,
Sample-5  31 / 199 unique seqs in original alignment, 6 / 199 unique seqs in new alignment,
Sample-6  64 / 339 unique seqs in original alignment, 5 / 339 unique seqs in new alignment,
Sample-7  2 / 9 unique seqs in original alignment, 2 / 9 unique seqs in new alignment,
Sample-8  12 / 486 unique seqs in original alignment, 3 / 486 unique seqs in new alignment,
Sample-9  Sample-10  80 / 418 unique seqs in original alignment, 2 / 418 unique seqs in new alignment,
DQA1_E2
Sample-1  23 / 104 unique seqs in original alignment, 7 / 104 unique seqs in new alignment,
Sample-2  4 / 22 unique seqs in original alignment, 4 / 22 unique seqs in new alignment,
Sample-3  5 / 8 unique seqs in original alignment, 5 / 8 unique seqs in new alignment,
Sample-4  9 / 26 unique seqs in original alignment, 9 / 26 unique seqs in new alignment,
Sample-5  11 / 34 unique seqs in original alignment, 11 / 34 unique seqs in new alignment,
Sample-6  30 / 138 unique seqs in original alignment, 8 / 138 unique seqs in new alignment,
Sample-7  1 / 1 unique seqs in original alignment, 1 / 1 unique seqs in new alignment,
Sample-8  32 / 146 unique seqs in original alignment, 8 / 146 unique seqs in new alignment,
Sample-9  1 / 1 unique seqs in original alignment, 1 / 1 unique seqs in new alignment,
Sample-10  35 / 122 unique seqs in original alignment, 8 / 122 unique seqs in new alignment,
```

```
> # Produce an alignment report for the un-corrected and corrected results.
> alignReport(my.mlgt.Result, method="profile", fileName="alignReport_my.mlgt.Result")
```

```
A_E3
B_E2
DPA1_E2
DQA1_E2
Alignment figures(s) plotted to alignReport_my.mlgt.Result.pdf
$A_E3
          numbSeqs numbVars alignLength invar.sites mafAboveThreshold mafBelowThreshold
Sample-1       154       36         264         214                21                29
Sample-2         0        0          NA          NA                NA                NA
Sample-3        33       16         264         236                28                 0
Sample-4       418      107         264         170                 2                92
Sample-5       141       37         264         212                22                30
Sample-6       142       37         264         221                12                31
Sample-7         4        3         264         249                15                 0
Sample-8       128       30         264         236                 2                26
Sample-9         4        4         265         249                16                 0
Sample-10      149       45         264         217                 8                39

$B_E2
          numbSeqs numbVars alignLength invar.sites mafAboveThreshold mafBelowThreshold
Sample-1       141       44         248         198                14                36
Sample-2        64       21         248         212                36                 0
Sample-3        54       17         248         223                25                 0
Sample-4        55       19         248         224                24                 0
Sample-5        57       23         248         217                31                 0
Sample-6        74       23         248         212                36                 0
Sample-7         1        1         250          NA                NA                NA
Sample-8        42       16         248         213                35                 0
Sample-9         0        0          NA          NA                NA                NA
Sample-10       89       24         249         215                34                 0

$DPA1_E2
          numbSeqs numbVars alignLength invar.sites mafAboveThreshold mafBelowThreshold
Sample-1        97       21         198         178                20                 0
Sample-2       254       70         198         135                 6                57
Sample-3       143       33         198         166                 2                30
Sample-4       430       17         198         182                 2                14
Sample-5       199       31         198         168                 5                25
Sample-6       339       64         198         138                 6                54
Sample-7         9        2         198         197                 1                 0
Sample-8       486       12         198         182                 7                 9
Sample-9         0        0          NA          NA                NA                NA
Sample-10      418       80         198         130                 1                67

$DQA1_E2
          numbSeqs numbVars alignLength invar.sites mafAboveThreshold mafBelowThreshold
Sample-1       104       23         187         141                30                16
Sample-2        22        4         237         212                25                 0
Sample-3         8        5         237         210                27                 0
Sample-4        26        9         237         229                 8                 0
Sample-5        34       11         286         277                 9                 0
```

```
Sample-6       138       30         187         137                29                21
Sample-7         1        1         187          NA                NA                NA
Sample-8       146       32         187         135                28                24
Sample-9         1        1         188          NA                NA                NA
Sample-10      122       35         254         201                27                26
```

```
> alignReport(my.mlgt.Result.Corrected, method="profile", fileName="alignReport_my.mlgt.Result.Corrected")
```

```
A_E3
B_E2
DPA1_E2
DQA1_E2
Alignment figures(s) plotted to alignReport_my.mlgt.Result.Corrected.pdf
$A_E3
          numbSeqs numbVars alignLength invar.sites mafAboveThreshold mafBelowThreshold
Sample-1       154        6         264         243                21                 0
Sample-2         0        0          NA          NA                NA                NA
Sample-3        33       16         264         236                28                 0
Sample-4       418        4         264         262                 2                 0
Sample-5       141        7         264         242                22                 0
Sample-6       142        5         264         252                12                 0
Sample-7         4        3         264         249                15                 0
Sample-8       128        3         264         262                 2                 0
Sample-9         4        4         265         249                16                 0
Sample-10      149        5         264         256                 8                 0

$B_E2
          numbSeqs numbVars alignLength invar.sites mafAboveThreshold mafBelowThreshold
Sample-1       141        6         248         234                14                 0
Sample-2        64       21         248         212                36                 0
Sample-3        54       17         248         223                25                 0
Sample-4        55       19         248         224                24                 0
Sample-5        57       23         248         217                31                 0
Sample-6        74       23         248         212                36                 0
Sample-7         1        1         250          NA                NA                NA
Sample-8        42       16         248         213                35                 0
Sample-9         0        0          NA          NA                NA                NA
Sample-10       89       24         249         215                34                 0

$DPA1_E2
          numbSeqs numbVars alignLength invar.sites mafAboveThreshold mafBelowThreshold
Sample-1        97       21         198         178                20                 0
Sample-2       254        3         198         192                 6                 0
Sample-3       143        3         198         196                 2                 0
Sample-4       430        3         198         196                 2                 0
Sample-5       199        6         198         193                 5                 0
Sample-6       339        5         198         192                 6                 0
Sample-7         9        2         198         197                 1                 0
Sample-8       486        3         198         191                 7                 0
Sample-9         0        0          NA          NA                NA                NA
Sample-10      418        2         198         197                 1                 0

$DQA1_E2
          numbSeqs numbVars alignLength invar.sites mafAboveThreshold mafBelowThreshold
Sample-1       104        7         187         157                30                 0
Sample-2        22        4         237         212                25                 0
Sample-3         8        5         237         210                27                 0
Sample-4        26        9         237         229                 8                 0
Sample-5        34       11         286         277                 9                 0
Sample-6       138        8         187         158                29                 0
Sample-7         1        1         187          NA                NA                NA
Sample-8       146        8         187         159                28                 0
Sample-9         1        1         188          NA                NA                NA
Sample-10      122        8         254         227                27                 0
```

If it has worked well, you may find that running callGenotypes on the corrected results gives more HOMOZYGOTE and HETEROZYGOTE calls.

Running errorCorrect was originally implemented as an additional step to mlgt. Once you are happy using it, it is much better to run the error correction as part of mlgt itself using the errorCorrect parameter.

3.6 Combining result sets

Once you have a genotyping system up and running you may want to compare results from one run to another. The easiest way is probably to run each dataset through a common workflow and compare results after output of genotypes. However there are several instances in which you may want to combine results into a single mlgtResult object (e.g. to use a common set of allele names, or see recurrence of the same variants across samples).

Here I outline a case where samples have been split across several runs and you want to combine the results before genotyping.

```
> my.design.list <- list()
> my.design.list[['A']] <- my.mlgt.Design
> my.design.list[['A']]@samples <- sampleList[1:5]
> my.design.list[['B']] <- my.mlgt.Design
> my.design.list[['B']]@samples <- sampleList[6:10]
> my.result.list <- lapply(my.design.list, FUN=function(x) mlgt(x))

A_E3
Sample-1 : Using 45 variants, accounting for 154 of 154 reads
Sample-3 : Using 18 variants, accounting for 33 of 33 reads
Sample-4 : Using 132 variants, accounting for 418 of 418 reads
Sample-5 : Using 47 variants, accounting for 141 of 141 reads
B_E2
Sample-1 : Using 57 variants, accounting for 141 of 141 reads
Sample-2 : Using 24 variants, accounting for 64 of 64 reads
Sample-3 : Using 23 variants, accounting for 54 of 54 reads
Sample-4 : Using 24 variants, accounting for 55 of 55 reads
Sample-5 : Using 37 variants, accounting for 57 of 57 reads
DPA1_E2
Sample-1 : Using 34 variants, accounting for 97 of 97 reads
Sample-2 : Using 94 variants, accounting for 254 of 254 reads
Sample-3 : Using 39 variants, accounting for 143 of 143 reads
Sample-4 : Using 30 variants, accounting for 430 of 526 reads
Sample-5 : Using 48 variants, accounting for 199 of 199 reads
DQA1_E2
Sample-1 : Using 27 variants, accounting for 104 of 104 reads
Sample-2 : Using 4 variants, accounting for 22 of 22 reads
Sample-3 : Using 6 variants, accounting for 8 of 8 reads
Sample-4 : Using 9 variants, accounting for 26 of 26 reads
Sample-5 : Using 11 variants, accounting for 34 of 34 reads
A_E3
Sample-6 : Using 49 variants, accounting for 142 of 142 reads
Sample-7 : Using 3 variants, accounting for 4 of 4 reads
Sample-8 : Using 40 variants, accounting for 128 of 128 reads
Sample-9 : Using 4 variants, accounting for 4 of 4 reads
Sample-10 : Using 55 variants, accounting for 149 of 149 reads
B_E2
Sample-6 : Using 29 variants, accounting for 74 of 74 reads
Sample-7 : Using 1 variants, accounting for 1 of 1 reads
Sample-8 : Using 18 variants, accounting for 42 of 42 reads
Sample-10 : Using 38 variants, accounting for 89 of 89 reads
DPA1_E2
Sample-6 : Using 85 variants, accounting for 339 of 339 reads
Sample-7 : Using 2 variants, accounting for 9 of 9 reads
Sample-8 : Using 30 variants, accounting for 486 of 609 reads
Sample-10 : Using 119 variants, accounting for 418 of 418 reads
DQA1_E2
Sample-6 : Using 42 variants, accounting for 138 of 138 reads
Sample-7 : Using 1 variants, accounting for 1 of 1 reads
Sample-8 : Using 47 variants, accounting for 146 of 146 reads
Sample-9 : Using 1 variants, accounting for 1 of 1 reads
Sample-10 : Using 39 variants, accounting for 122 of 122 reads

> my.result.list

$A
Results for mlgt run:
Project: myProject
Run: myRun
```

```
Samples: 5
fTags: 10
rTags: 10
Markers: 4
   marker assignedSeqs assignedVariants minVariantLength maxVariantLength minAlleleLength maxAlleleLength
1    A_E3          746              196                5              264             264             264
2    B_E2          371              124                5              248             247             248
3 DPA1_E2         1123              172                5              198             197             198
4 DQA1_E2          194               52                5              283             184             283

$B
Results for mlgt run:
Project: myProject
Run: myRun
Samples: 5
fTags: 10
rTags: 10
Markers: 4
   marker assignedSeqs assignedVariants minVariantLength maxVariantLength minAlleleLength maxAlleleLength
1    A_E3          427              119                5              265             263             265
2    B_E2          206               64                5              250             248             250
3 DPA1_E2         1252              158                5              198             198             198
4 DQA1_E2          408               99                5              254             184             254

> combined.result <- combineMlgtResults(my.result.list)

Complex join

> combined.result

Results for mlgt run:
Project: myProject
Run: myRun
Samples: 10
fTags: 10
rTags: 10
Markers: 4
   marker assignedSeqs assignedVariants minVariantLength maxVariantLength minAlleleLength maxAlleleLength
1    A_E3         1173              315                5              265             263             265
2    B_E2          577              188                5              250             247             250
3 DPA1_E2         2375              330                5              198             197             198
4 DQA1_E2          602              151                5              283             184             283
```

Another opportunity to combine results comes during a parallelised mlgt run - see below.

3.7 Parallelization

The slowest part of mlgt is the mlgt() function itself. As each marker is analysed separately, the function is 'embarrassingly parallel' and easy to speed up if you have access to more than one processor. N.B. this section is about multi-threading, not about running on a compute cluster, though mlgt could be adapted to do that. The procedure is to create a list of mlgtDesign objects, pertaining to a discrete subset of the markers and then use a separate processor to run mlgt on each member of the list. After this has finished, there is a function to recombine the separate mlgtResult objects into a single result.

The list approach can be demonstrated on a single processor (where each mlgt run happens in turn) using the lapply command.

```
> # Create a list of mlgtDesign objects, each with only one marker.
> my.design.list <- list()
> for(thisMarker in names(myMarkerList))  {
+         my.design.list[[thisMarker]] <- my.mlgt.Design
+         my.design.list[[thisMarker]]@markers <- myMarkerList[thisMarker]
+
+ }
> # Use lapply to run mlgt() on each member of the list.
> # N.B. we are using errorCorrection within mlgt(), which slows it down a bit.
> system.time(
+         my.result.list <- lapply(my.design.list,
+                        FUN=function(x) mlgt(x, errorCorrect=TRUE))
+ )

Using error correction at 0.01
A_E3
Sample-1 : Using 45 variants, accounting for 154 of 154 reads
Sample-3 : Using 18 variants, accounting for 33 of 33 reads
Sample-4 : Using 132 variants, accounting for 418 of 418 reads
Sample-5 : Using 47 variants, accounting for 141 of 141 reads
Sample-6 : Using 49 variants, accounting for 142 of 142 reads
Sample-7 : Using 3 variants, accounting for 4 of 4 reads
Sample-8 : Using 40 variants, accounting for 128 of 128 reads
Sample-9 : Using 4 variants, accounting for 4 of 4 reads
Sample-10 : Using 55 variants, accounting for 149 of 149 reads
Using error correction at 0.01
B_E2
Sample-1 : Using 57 variants, accounting for 141 of 141 reads
Sample-2 : Using 24 variants, accounting for 64 of 64 reads
Sample-3 : Using 23 variants, accounting for 54 of 54 reads
Sample-4 : Using 24 variants, accounting for 55 of 55 reads
Sample-5 : Using 37 variants, accounting for 57 of 57 reads
Sample-6 : Using 29 variants, accounting for 74 of 74 reads
Sample-7 : Using 1 variants, accounting for 1 of 1 reads
Sample-8 : Using 18 variants, accounting for 42 of 42 reads
Sample-10 : Using 38 variants, accounting for 89 of 89 reads
Using error correction at 0.01
DPA1_E2
Sample-1 : Using 34 variants, accounting for 97 of 97 reads
Sample-2 : Using 94 variants, accounting for 254 of 254 reads
Sample-3 : Using 39 variants, accounting for 143 of 143 reads
Sample-4 : Using 30 variants, accounting for 430 of 526 reads
Sample-5 : Using 48 variants, accounting for 199 of 199 reads
Sample-6 : Using 85 variants, accounting for 339 of 339 reads
Sample-7 : Using 2 variants, accounting for 9 of 9 reads
Sample-8 : Using 30 variants, accounting for 486 of 609 reads
Sample-10 : Using 119 variants, accounting for 418 of 418 reads
Using error correction at 0.01
DQA1_E2
Sample-1 : Using 27 variants, accounting for 104 of 104 reads
Sample-2 : Using 4 variants, accounting for 22 of 22 reads
Sample-3 : Using 6 variants, accounting for 8 of 8 reads
Sample-4 : Using 9 variants, accounting for 26 of 26 reads
Sample-5 : Using 11 variants, accounting for 34 of 34 reads
Sample-6 : Using 42 variants, accounting for 138 of 138 reads
```

```
Sample-7 : Using 1 variants, accounting for 1 of 1 reads
Sample-8 : Using 47 variants, accounting for 146 of 146 reads
Sample-9 : Using 1 variants, accounting for 1 of 1 reads
Sample-10 : Using 39 variants, accounting for 122 of 122 reads
   user  system elapsed
  45.08    0.39   79.36
```

Alone, this isn't much use, but when combined with multi-threading, things get much faster. The easiest way I have found to do this in R is with the aid of the snowfall package. You will need to install the package and set a few environment variables.

```
> #install.packages('snowfall')
> library(snowfall)
> sfInit(parallel=TRUE, cpus=4, type="SOCK")        # set your number of processors here.
```

```
R Version:  R version 2.14.1 (2011-12-22)
```

```
> sfExport(list=ls())        # is this necessary?
> sfLibrary(mlgt)                # the 'nodes' need to load a copy of the relevant librari
```

```
Library mlgt loaded.
```

```
> sfLibrary(seqinr)        # is this one necessary?
```

```
Library seqinr loaded.
```

```
> # Then we run mlgt over the list of mlgtDesign objects.
> # Note that extra parameters can be passed to sfLapply().
> system.time(
+         sf.result.list <- sfLapply(my.design.list, mlgt, errorCorrect=TRUE)
+ )
```

```
   user  system elapsed
    0.0     0.0    30.3
```

That should have been substantially faster. Now you need to combine the results into a single mlgtResult object. This is perhaps the most useful function of combineMlgtResults().

```
> project.mlgt.Results <- combineMlgtResults(sf.result.list)
```

```
Simple join
Simple join
Simple join
```

3.8 Custom genotype call methods

I made up the method to call genotypes based on the relative frequencies of certain variants. If users would like to use an alternative method within mlgt, they can specify a new function and pass this by name to callGenotypes(). The function must accept a data frame as argument 'table' and return the same table after modification. Give default values to any additional parameters that you want to include. The example below creates a column 'status' and records genotypes ”good” or ”bad” depending on the proportion of unique variants.

```
> callGenotypes.custom <- function(table, maxPropUniqueVars=0.5) {
+         table$status <- "notCalled"
+         table$propUniqueVars <- table$numbVar/table$numbSeq
+         table$status <- ifelse(table$propUniqueVars <= maxPropUniqueVars,"good", "bad")
+         return(table)
+ }
> my.custom.Genotypes <- callGenotypes(my.mlgt.Result, method="callGenotypes.custom")
```

3.9 Miscellaneous

I've not yet included good provision for exporting all the sequence variants. The `mlgtResult` objects do currently store the DNA sequences and their export should probably be linked to `callGenotypes`. In the meantime, unique variants that have been assigned allele names can be output as fasta with this function:-

```
> dumpVariantMap.mlgtResult(my.mlgt.Result)
```

In addition another function can be used to export all sequences found for a marker-sample pair into a separate fasta alignment for each pair. The output is in fasta format with the name of the sequence set to the sequence itself (?!). The 'unique' flag can be used to limit the output to unique variants, in which case a count for that sequence is appended at the end of the sequence name line.

```
> dumpVariants(my.mlgt.Result)
```

4 TO-DO

Other filters. More/better genotyping methods. Run comparison.

5 R Session

```
> sessionInfo()

R version 2.14.1 (2011-12-22)
Platform: i386-pc-mingw32/i386 (32-bit)

locale:
[1] LC_COLLATE=English_United Kingdom.1252  LC_CTYPE=English_United Kingdom.1252    LC_MONETARY=English_United Kingdom.1252
[4] LC_NUMERIC=C                            LC_TIME=English_United Kingdom.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] snowfall_1.84 snow_0.3-8    mlgt_0.17     seqinr_3.0-6

loaded via a namespace (and not attached):
[1] tools_2.14.1
```