A short manual for LFMM
(command-line version)

Eric Frichot (eric.frichot@imag.fr)
Olivier François (olivier.francois@imag.fr)

January 11, 2015

Please, print this reference manual only if it is necessary.

This short manual aims to help users to run LFMM command-line engine on Mac and Linux.

Contents

1 Description

We proposed an integrated framework based on population genetics, ecological modeling and machine learning techniques for screening genomes for signatures of local adaptation. We implemented fast algorithms using a hierarchical Bayesian mixed model based on a variant of principal component analysis in which residual population structure is introduced via unobserved factors. These algorithms can detect correlations between environmental and genetic variation at the same time as they infer the background levels of population structure. A description of the method is available in our paper:

Eric Frichot, Sean Schoville, Guillaume Bouchard, Olivier François, 2013. *Landscape genomic tests for associations between loci and environmental gradients* Molecular Biology and Evolution, 30 (7), 1687-1699.

2 Installation

We provide a set R scripts and C programs to perform LFMM analyses and to convert to LFMM format. To install LFMM CL version, you just have to execute the install script (install.sh) in LFMM main directory. To execute it in a terminal shell, go to LFMM main directory and write "./install.sh". If the script is not

executable, type "chmod +x install.sh" and then "./install.sh". A binary called `LFMM` should be created in the "./bin/" directory.

3 Data format

Input files are composed of two files: a genotype file in the lfmm format and a variable file in the env format.

The **lfmm** format for the **genotype file** has one row for each individual. Each row contains one value per SNP (separated by spaces or tabulations): the number of alleles. The number of alleles can be the number of reference alleles or the number of derived alleles as long as it is the same choice for an entire SNP. The missing genotypes are encoded with the value -9 or 9. Here is an example with 3 individuals and 4 SNPs:

```
1 0 0 1
1 1 9 2
2 0 1 1
```

The **lfmm** format can also be used for allele frequency data. In this case, each population is an individual and the number of allele is replaced by the allele frequency.
The **env** format for the **variable file** has one row for each individual. Each row contains one value per environmental variable (separated by spaces or tabulations). Below, an example of variable file for $n = 3$ individuals and $D = 1$ covariable. Warning: If you set several covariables, the program can be launched for each covariable sequentially or all variables together (see command-line options).

```
0.252477
0.216618
-0.47509
```

The **zscore** format for the **output file** has one row for each SNP. Each row contains three values: The first value is the zscore, the second value is the -log10(pvalue), the third value is the p-value (separated by spaces or tabulations). Below, an example of output file for $L = 4$ loci.

```
0.698819 0.314558 0.484665
1.35961 0.759568 0.173953
0.771135 0.355929 0.440627
0.879092 0.420959 0.379351
```

4 Run the programs

The program is executed by a command line. The format is:

```
./bin/LFMM -x genotype_file -v variable_file -K latent_factors_number
```

All the previous options are mandatory. There is no order for the options in the command line. Here is a more precise description of the options:

- `-x genotype_file` is the path for the genotype file (in **lfmm** format).
- `-v variable_file` is the path for the variable file (in **env** format).
- `-K latent_factor_number` is the number of latent factors. Several methods to choose K are described in the tutorial. Check the references for more informations.

Other options are not mandatory:

- `-d nd` fit `LFMM` with the nd-th variable only from the variable file. By default (if NULL and all is FALSE), fit `LFMM` with each variable from the variable file sequentially and independently.
- `-a` is a boolean option. If true, fit `LFMM` with all variables from the variable file at the same time. This option is not compatible with d option (by default: FALSE).

- `-o output_file` is the base of the path for the zscore output files (in **zscore** format). There is one output file per environmental variable. By default, the base of the output file is the base of the name of the input file. Then it is completed with the number of the covariable (with a "a" for all and a "s" for sequentially), the number of latent factors and .zscore extension. For example, for the 3rd variable sequentially and 3 latent factors, the output file is by default input_file_s3.3.zscore.
- `-m` is a boolean option. If true, the input file contains missing genotypes. By default the program assumes that there is no missing data. The program is a bit slower with missing data. Please, do not use this option if it is not necessary.
- `-p p` is the number of processes (CPU) that you choose to use if you run the algorithm in parrallel. Be careful, the number of processes has to be lower or equal than the number of physical processes available on your computer. By default, the number of processes is 1.
- `-i iteration_number` is the number of iterations in the Gibbs Sampling algorithm. This number should be large enough (by default: 10000).
- `-b burnin_number` is the number of burning iterations in the Gibbs Sampling algorithm (by default 5000).
- `-s seed` is the seed to initialize the randomization. By default, the seed is randomly chosen.
- `-C dev_file` is the path for the file containing the DIC and the deviance criterion. By default, the name of the DIC file is the name of the input file with .dic extension.

If you need a summary of the options, you can use the `-h` option by typing the command line

```
./bin/LFMM -h
```

A full example is available at the end of this note.

5 Using LFMM in practice

In this section, we give practical recommendations for analyzing real data sets using the `LFMM` computer program. These recommendations should help users avoiding important mistakes when using the `LFMM` algorithm. Note that the following comments should not be taken too literally. Several alternative options might work equally well.

Preparing the data. Genotypic data must be prepared using the `lfmm` format (any type of allelic data is allowed). The `LFMM` program can handle missing data, but the algorithm used for genotype imputation is basic. We encourage users having more than 10% missing genotypes in their data to fix the missing data issue by using *matrix completion* or *genotype imputation* programs such as `IMPUTE2` or `MENDEL-IMPUTE` before starting their analyses with `LFMM`. Ecological data must be prepared using the `env` format. To decide which variables should be used among a large number of ecological indicators (eg, climatic variables), we suggest that users summarize their data using linear combinations of those indicators. Considering principal component analysis (or similar approaches) and using the first components as new ecological variables is one of these approaches.

Setting run parameters. The `LFMM` program is based on a stochastic algorithm (MCMC) which cannot provide exact results. We recommend using large number of cycles (e.g., `-i 10000`) and the burn-in period should set at least to one-half of the total number of cycles (`-b 5000`). We have noticed that the program results are sensitive to the run-length parameter when data sets have relatively small sizes (eg, a few hundreds of individuals, a few thousands of loci). We recommend increasing the burn-in period and the total number of cycles in this situation.

Deciding the number of latent factors. Deciding an appropriate value for the number of latent factors in `LFMM` can be based on the analysis of histograms of test p-values. Here, the objective is to control the false discovery rate while keeping reasonable power to reject the null hypothesis. To choose the number of factors, we suggest using the genomic inflation factor. According to Devlin and Roeder (1999), this quantity is defined as

$$\lambda = \text{median}(z^2)/0.456\,.$$

The inflation factor usually decreases with increasing values of K. To compute the genomic inflation factor, we recommend using several runs for each value of K and taking the median or the mean of the λ values obtained from the above formula (use 5 to 10 runs, see our script below). Choosing values of K for which the estimate

of λ is close to (or slightly below) 1.0 warrants that the FDR can be controlled efficiently.

Testing all K values in a large range, from 1 to 20 for example, is generally useless. A careful analysis of population structure and estimates of the number of ancestral populations contributing to the genetic data indicates the range of values to be explored. If for example the `sNMF` or `ADMIXTURE` programs estimate 5 ancestral populations, then running `LFMM` $K = 4 - 7$ often provides inflation factors close to 1.0.

Combining z-scores obtained from multiple runs. We suggest using the Fisher-Stouffer or a similar method to combine z-scores from multiple runs. In practice, we found that using the median z-scores of 5-10 runs and re-adjusting the p-values afterwards increase the power of `LFMM` tests. This can be done by using the following sequence of `R` commands. Assuming that results from 5 runs with a particular value of K are recorded in external files named `res1_s1.9.zscore`, `res2_s1.9.zscore`, etc, we can compute ajusted p-values using the the following commands (see the tutorial for an example).

```
z.table = NULL
for (i in 1:5){
        file.name = paste("res",i, "_s1.9.zscore", sep="")
        z.table = cbind(z.table, read.table(file.name)[,1])
}
z.score = apply(z.table, MARGIN = 1, median)
lambda = median(z.score^2) / 0.456
ap.values = pchisq(z.score^2 / lambda, df = 1, lower = F))
```

For an expected value of the FDR equal to q (set $q = 10\%$ or $q = 15\%$), a list of candidate loci stored in the object `candidates` can be obtained by using the Benjamini-Hochberg procedure as follows

```
q = 0.1
L = length(ap.values)
w = which(sort(ap.values) < q * (1:L) / L)
candidates = order(ap.values)[w]
```

6 Tutorial

6.1 Simulated data set

The data set is composed of 165 individuals for 1000 SNPs simulated with SPLACHE (see our note for LEA package for more informations). In this data set, the last 100 SNPs are simulated as associated with the environnemental gradient. The data set is in the directory "examples/simulated_example/".

Here is a small script to run LFMM with 5 repetitions.

```
mkdir res;
for i in 1 2 3 4 5; do
        ./bin/LFMM -x examples/simulated_example/genotypes.lfmm \
        -v examples/simulated_example/gradients.env -K 9 -o res/res$i
done
```

Then, apply the following R script:

```
z.table = NULL
for (i in 1:5){
        file.name = paste("res/res",i, "_s1.9.zscore", sep="")
        z.table = cbind(z.table, read.table(file.name)[,1])
}
z.score = apply(z.table, MARGIN = 1, median)
lambda = median(z.score^2) / 0.456
ap.values = pchisq(z.score^2 / lambda, df = 1, lower = F)

q = 0.1
L = length(ap.values)
w = which(sort(ap.values) < q * (1:L) / L)
candidates = order(ap.values)[w]
```

```
# estimated FDR and True Positif
estimated.FDR = length(which(candidates <= 900))/length(candidates)
estimated.TP = length(which(candidates > 900))/100
print(paste("expected FDR:", q))
print(paste("FDR:", estimated.FDR, "True Positive:", estimated.TP))
```

6.2 Real data set

6.2.1 Description

The data set that we analyze in this tutorial is an Asian human data set of SNPs data. This data is a worldwide sample of genomic DNA (10757 SNPs) from 388 individuals, taken from the Harvard Human Genome Diversity Project - Centre Etude Polymorphism Humain (Harvard HGDP-CEPH)2 . In those data, each marker has been ascertained in samples of Mongolian ancestry (referenced population HGDP01224) [4]. We selected all samples from Asia. Using Tracy-Widom tests implemented in `SmartPCA` [3], we found that the number of principal components with P-values smaller than 0.01 was around $KTW = 10$. Using the Bayesian clustering programs `STRUCTURE` [5] and `TESS` [1, 2], we found that $K = 7$ components could better describe our simulated data. We extracted climatic data population samples using the WorldClim data set at 30 arcsecond (1km2) resolution (Hijmans, Cameron, Parra, Jones, and Jarvis (2005)). We summarized the climatic variables by using the first axis of a principal component analysis for temperature variables and for precipitation variables. The data set is in directory `examples/human_example/`. The genotypic information are in `panel11_Asia.lfmm`. the SNPs information is in `panel11.pedsnp`. The environmental file is `cov_panel11_Asia.env`. There are 2 variables, one proxy for temperature and one for precipitation.

6.2.2 Run LFMM

Here is an example of command line to analyse our dataset

```
./bin/LFMM -x examples/human_example/panel11_Asia.lfmm \
-v examples/human_example/cov_panel11_Asia.env  -K 7 -d 1
```

Output for LFMM

```
LFMM  Copyright (C) 2012 Eric Frichot
This program comes with ABSOLUTELY NO WARRANTY; for details type './LFMM -l'.
This is free software, and you are welcome to redistribute it
under certain conditions; type './LFMM -l' for details.

****                          LFMM Version 1.3                                  *****
****           E. Frichot, S. Schoville, G. Bouchard, O. Francois               *****
****                          Please cite our paper !                           *****
****   Information at http://membres-timc.imag.fr/Olivier.Francois/lfmm/        *****

./bin/LFMM -x examples/human_example/panel11_Asia.lfmm -v examples/human_example/cov_panel11_Asia.env -
Summary of the options:

        -n (number of individuals)      388
        -L (number of loci)             10757
        -K (number of latent factors)   7
        -o (output file)                /home/frichote/replop/CL_code/examples/LFMM/human_example/panel
        -i (number of iterations)       10000
        -b (burnin)                     5000
        -s (seed random init)           4769701177742658440
        -p (number of processes (CPU))  1
        -x (genotype file)              examples/human_example/panel11_Asia.lfmm
        -v (variable file)              examples/human_example/cov_panel11_Asia.env
        -D (number of covariables)      2
        -d (the dth covariable)         1

Read variable file:
```

```
         examples/human_example/cov_panel11_Asia.env                OK.

Read genotype file:
         examples/human_example/panel11_Asia.lfmm                OK.

<<<<
         Analyse for variable 1

                Start of the Gibbs Sampler algorithm.

        [                                                                              ]
        [==============================================================================]

                End of the Gibbs Sampler algorithm.

        ED: 4173815.5         DIC: 4173900.09

        The statistics for the run are registered in:
                 examples/LFMM/human_example/panel11_Asia_s1.7.dic.

        The zscores for variable 1 are registered in:
                 examples/LFMM/human_example/panel11_Asia_s1.7.zscore.
        The columns are: zscores, -log10(p-values), p-values.

        -------------------------
        The execution for variable 1 worked without error.
>>>>
```

7 Additional programs

We provide C programs to perform a PCA and tracy-widom tests and to convert to lfmm format.

7.1 Principal Component Analysis

The program is executed by a command line. The format is:

```
./bin/pca -x genotype_file
```

All the previous options are mandatory. There is no order for the options in the command line. Here is a more precise description of the options:

- `-x genotype_file` is the path for the genotype file (in **lfmm** format).

Other options are not mandatory:

- `-K K` compute only K principal components (default: n, the number of individuals).
- `-a eigenvalue_file` output eigenvalues file (default: genotype_file.eigenvalues).
- `-a eigenvector_file` output eigenvectors file (default: genotype_file.eigenvectors).
- `-a sdev_file` output standard deviation file (default: genotype_file.sdev).
- `-a projection_file` output projections file (default: genotype_file.projections).
- `-c` boolean option, datat centered (default: FALSE)
- `-s` boolean option, datat centered and scaled(default: FALSE)

If you need a summary of the options, you can use the `-h` option by typing the command line

```
./bin/pca -h
```

Here is a small example:

```
./bin/pca -x examples/simulated_example/genotypes.lfmm
```

A program is also available to perform Tracy-Widom tests on a set of eigenvalues. The program is executed by a command line. The format is:

```
./bin/tracyWidom -i input_file.eigenvalues [output_file.tracywidom]
```

where

- `input_file.eigenvalues` is the path for the input file containing the eigenvalues.
- `output_file.tracywidom` is the path for the output_file. By default, the name of the output file is the name of the input_file with a .tracywidom extension.

Here is a small example:

```
./bin/tracyWidom -i examples/simulated_example/genotypes.eigenvalues
```

7.2 Data Format

Input files are composed of two mandatory files (a genotype file and an environmental variable file). It is not necessary to provide information about individuals. All data formats are described with the same example. These files are available in `examples/format_example/`.

- lfmm (example.lfmm)
  The **lfmm** format has one row for each individual. Each row contains one value per SNP (separated by spaces or tabulations): the number of alleles. The number of alleles can be the number of reference alleles or the number of derived alleles as long as it is the same choice for an entire SNP. The missing genotypes are encoded with the value -9 or 9.

  ```
  1 0 0 1
  1 1 9 2
  2 0 1 1
  ```

- ped (example.ped)
  The **ped** format has one row for each individual. Each row contains 6 columns of information for each individual, plus two genotype columns for each SNP. Each column must be separated by spaces or tabulations. The genotype format must be either 0ACGT or 01234, where 0 means missing genotype. The first 6 columns of the genotype file are: the 1st column is the family ID, the 2nd column is the sample ID, the 3rd and 4th columns are the sample IDs of parents, the 5th column is the gender (male is 1, female is 2), the 6th column is the case/control status (1 is control, 2 is case), the quantitative trait value or the population group label.

  The ped format is described here: http://pngu.mgh.harvard.edu/ purcell/plink/data.shtml.

  ```
  1       SAMPLE0 0 0 2 2 1 2 3 3 1 1 2 1
  2       SAMPLE1 0 0 1 2 2 1 1 3 0 4 1 1
  3       SAMPLE2 0 0 2 1 2 2 3 3 1 4 1 2
  ```

- ancestrymap (example.ancestrymap)
  The **ancestrymap** format has one row for each genotype. Each row has 3 columns: the 1st column is the SNP name, the 2nd column is the sample ID, the 3rd column is th number of alleles. It is assumed that the genotypes for a given SNP name are written in consecutive lines. It is also assumed that the genotypes for a set of individuals are given in the same order as lines. The number of alleles can be the number of reference alleles or the number of derived alleles as long as it is the same choice for an entire SNP. It is assumed that the missing genotypes are encoded with the value 9.

```
rs0000          SAMPLE0   1
rs0000          SAMPLE1   1
rs0000          SAMPLE2   2
rs1111          SAMPLE0   0
rs1111          SAMPLE1   1
rs1111          SAMPLE2   0
rs2222          SAMPLE0   0
rs2222          SAMPLE1   9
rs2222          SAMPLE2   1
rs3333          SAMPLE0   1
rs3333          SAMPLE1   2
rs3333          SAMPLE2   1
```

- geno (example.geno)
  The **geno** format has one row for each SNP. Each row contains 1 character per individual: 0 means zero copy of the reference allele. 1 means one copy of the reference allele. 2 means two copies of the reference allele. 9 means missing data.

```
112
010
091
121
```

- vcf (example.vcf)
  The **vcf** The vcf format is described here
  http://www.1000genomes.org/wiki/Analysis/Variant%20Call%20Format/vcf-variant-call-format-version-41

```
##fileformat=VCFv4.1
##FORMAT=<ID=GM,Number=1,Type=Integer,Description="Genotype meta">
##INFO=<ID=VM,Number=1,Type=Integer,Description="Variant meta">
##INFO=<ID=SM,Number=1,Type=Integer,Description="SampleVariant meta">
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT SAMPLE0 SAMPLE1 SAMPLE2
1 1001 rs0000 T C 999 . VM=1;SM=100 GT:GM 1/0:1 0/1:2 1/1:3
1 1002 rs1111 G A 999 . VM=2;SM=101 GT:GM 0/0:6 0/1:7 0/0:8
1 1003 notres G AA 999 . VM=3;SM=102 GT:GM 0/0:11 ./.:12 0/1:13
1 1004 rs2222 G A 999 . VM=3;SM=102 GT:GM 0/0:11 . 1/0:13
1 1003 notres GA A 999 . VM=3;SM=102 GT:GM 0/0:11 ./.:12 0/1:13
1 1005 rs3333 G A 999 . VM=3;SM=102 GT:GM 1/0:11 1/1:12 0/1:13
```

Tips: As LFMM does not model allele frequencies, genotype file can be the number of copy of either the reference allele or the derived allele.

7.3 Data conversion

The LFMM command-line engine allows data in lfmm format. You can convert from ped, eigenstratgeno, ancestrymap to lfmm format using C progams.

The format of the command-line is (replacing ¡format¿ by ped, ancestrymap, or geno):

```
./bin/<format>2lfmm input_file [output_file]
```

where

- `input_file` is the path for the input file.
- `output_file` is the path for the output_file (in lfmm format). By default, the name of the output file is the name of the input_file with .lfmm extension.

For examples,

- example.ped

```
./bin/ped2lfmm examples/format_example/example.ped
```

- example.ancestrymap

```
./bin/ancestrymap2lfmm examples/format_example/example.ancestrymap
```

- example.geno

```
./bin/geno2lfmm examples/format_example/example.geno
```

- example.vcf

```
./bin/vcf2geno examples/format_example/example.vcf
```

7.4 Manhattan plot

To display a manhattan plot of the results, you may consider the package `qqman`.

8 Contact

A FAQ (Frequently Asked Questions) section is available on our webpage (ttp://membres-timc.imag.fr/Olivier.Francois/lfmm. LFMM software is still under development. All your comments and feedbacks are more than welcome.

References

[1] Chibiao Chen, Eric Durand, Florence Forbes, and Olivier François. Bayesian clustering algorithms ascertaining spatial population structure: a new computer program and a comparison study. *Molecular Ecology Notes*, 7(5):747–756, 2007.

[2] E Durand, F Jay, O E Gaggiotti, and O François. Spatial inference of admixture proportions and secondary contact zones. *Molecular Biology and Evolution*, 26(9):1963–1973, 2009.

[3] N Patterson, A L Price, and D Reich. Population structure and eigenanalysis. *PLoS Genetics*, 2:20, 2006.

[4] Nick J. Patterson, Priya Moorjani, Yontao Luo, Swapan Mallick, Nadin Rohland, Yiping Zhan, Teri Genschoreck, Teresa Webster, and David Reich. Ancient admixture in human history. *Genetics, doi:10.1534/genetics.112.145037*, 2012.

[5] J K Pritchard, M Stephens, and P Donnelly. Inference of population structure using multilocus genotype data. *Genetics*, 155(2):945–959, 2000.