LEA: An R Package for Landscape and Ecological Association Studies

Eric Frichot and Olivier François
Université Grenoble-Alpes,
Centre National de la Recherche Scientifique,
TIMC-IMAG UMR 5525, Grenoble, 38042, France.

Contents

1 Overview **1**

2 Introduction **2**
2.1 Input files 3

3 Analysis of population structure and imputation of missing data **4**
3.1 Principal Component Analysis 4
3.2 Inference of individual admixture coefficients using `snmf` . . . 4
3.3 Population differentation tests using `snmf()` 7
3.4 Missing genotype imputation using `snmf` 8

4 Ecological association tests using lfmm **10**

5 Ecological association tests using lfmm2 **13**

6 Predictive Ecological Genomics: genetic gap **18**

1 Overview

`LEA` is an `R` package dedicated to population genomics, landscape genomics and genotype-environment association tests (Frichot and François, 2015; Gain and François, 2021). `LEA` can run analyses of population structure and genome-wide tests for local adaptation. Italso performs imputation of missing genotypes. The package contains statistical methods for estimating ancestry coefficients from large genotypic matrices and for evaluating the number of ancestral populations (`snmf`). It performs statistical tests for identifying genetic polymorphisms that exhibit association with environmental gradients or phenotypic traits using latent factor mixed models

(`lfmm2`). In addition, `LEA` computes values of genetic offset statistics based on current and new environments (`genetic.gap`). `LEA` is mainly based on optimized statistical procedure that can scale with the dimensions of large data sets.

2 Introduction

The goal of this tutorial is to give an overview of the main functionalities of the `R` package `LEA`. It will show the main steps of analysis, including 1) analysing population structure and preparing a genotypic matrix for genomewide association studies, 2) fitting latent factor mixed models to the data and extracting regions of interest, 3) computing genomic offset statistics.

As some functions may take a few hours to analyse very large data sets, output files are written into text files that can be read by `LEA` after each batch of runs (called a 'project'). We advise creating working directories containing genotypic data and covariables when starting `LEA`. Note that two files with the same names but a different extension are assumed to contain the same data in distinct formats.

```
# creation of a directory for LEA analyses
dir.create("LEA_analyses")
# set the created directory as the working directory
setwd("LEA_analyses")
```

The first example below is based on a small dataset consisting of 400 SNPs genotyped for 50 diploid individuals. The last 50 SNPs are correlated with an environmental variable, and represent the target loci for a genotype-environment association (GEA) analysis. Similar artificial data were analyzed in the computer note introducing the R package `LEA` (Frichot and François, 2015). More realistic data are also included as examples in the package, and they will be analyzed later.

```
library(LEA)
# Creation of a genotype matrix data file: "genotypes.lfmm"
# The data include 400 SNPs for 50 individuals.
data("tutorial")
# Write genotypes in the lfmm format
write.lfmm(tutorial.R, "genotypes.lfmm")
# Write genotypes in the geno format
write.geno(tutorial.R, "genotypes.geno")
# creation of an environment gradient file: gradient.env.
# The .env file contains a single ecological variable
```

```
# for each individual.
write.env(tutorial.C, "gradients.env")
```

Note that the LEA package is to be able to handle very large genotype matrices. Genomic data are processed using fast C codes wrapped into the R code. Most LEA functions use character strings containing paths to input files as arguments.

2.1 Input files

The R package LEA can handle several input file formats for genotypic matrices. More specifically, the package uses the lfmm and geno formats, and provides functions to convert from other formats such as ped, vcf, and ancestrymap formats. The program VCFTOOLS can be very useful in providing one of those formats. The ped format is the closest to an lfmm matrix.

The lfmm and geno formats can also be used for coding multiallelic marker data (eg, microsatellites). For multiallelic marker data, the conversion function struct2geno() converts files in the STRUCTURE format in the geno or lfmm formats. LEA can process any type of allele frequency data if they are encoded in the lfmm format. In that case, the lfmm() and lfmm2() functions will use allele frequencies for populations in their statistical models.

Ecological predictors (or phenotypic traits) must be formatted in the env format. This format corresponds to a matrix in which each variable is represented as a column and each sample is represented as a row (Frichot et al., 2013). An external env file requires a .env extension.

When using ecological data, we often need to decide which variables should be used among a large number of predictors (e.g., bioclimatic variables). We suggest that users summarize their data using linear combinations of those predictors. Considering principal component analysis and using the first principal components as proxies for ecological gradients linked to selective forces can be useful in this context. Of course, using principal components of categories of variables, like precipitations, temperatures, soil characteristics, etc, could be a valuable alternative that also reduce dimension and collinearity issues.

The LEA package can handle missing data in population structure analyses. In association analyses, missing genotypes must be replaced by imputed values using a missing data imputation method. We encourage users to remove their missing data by using the function impute(), which is based on population structure analysis and nonnegative matrix factorization (see next section). Note that specialized genotype imputation programs such as BEAGLE, IMPUTE2 or MENDEL-IMPUTE could provide better imputation results than LEA, in particular for model species with a published

reference genome. Filtering out rare variants – retaining minor allele frequency greater than 5 percent –, and pruning regions in strong LD may also result in better analyses with LEA.

3 Analysis of population structure and imputation of missing data

The R package LEA implements two classical approaches for the estimation of population genetic structure: principal component analysis (pca) and admixture analysis using sparse nonnegative matrix factorization (snmf) (Frichot et al., 2014; Patterson et al., 2006; Pritchard et al., 2000). The algorithms programmed in LEA are improved versions of admixture analysis, that are able to process large genotypic matrices efficiently.

3.1 Principal Component Analysis

The LEA function pca() computes the scores of a PCA for a genotypic matrix, and returns a screeplot for the eigenvalues of the sample covariance matrix. Using the function pca(), an object of class pcaProject is created. This object contains a path to the files storing eigenvectors, eigenvalues and projections.

```
# run of pca
# Available options, K (the number of PCs),
#                    center and scale.
# Create files: genotypes.eigenvalues - eigenvalues,
#               genotypes.eigenvectors - eigenvectors,
#               genotypes.sdev - standard deviations,
#               genotypes.projections - projections,
# Create a pcaProject object: pc.
pc = pca("genotypes.lfmm", scale = TRUE)
tw = tracy.widom(pc)
```

The number of "significant" components can be evaluated using a graphical method based on the screeplot (Figure 1). According to Cattell's rule, the elbow in the screeplot indicates that there are around $K = 4$ major components in the data. This corresponds to ≈ 5 genetic clusters.

3.2 Inference of individual admixture coefficients using snmf

The package LEA includes the R function snmf() that estimates individual admixture coefficients from the genotypic matrix (Frichot et al., 2014). The function provides results very close to Bayesian clustering programs such as

```
# plot the percentage of variance explained by each component
plot(tw$percentage, pch = 19, col = "darkblue", cex = .8)
```

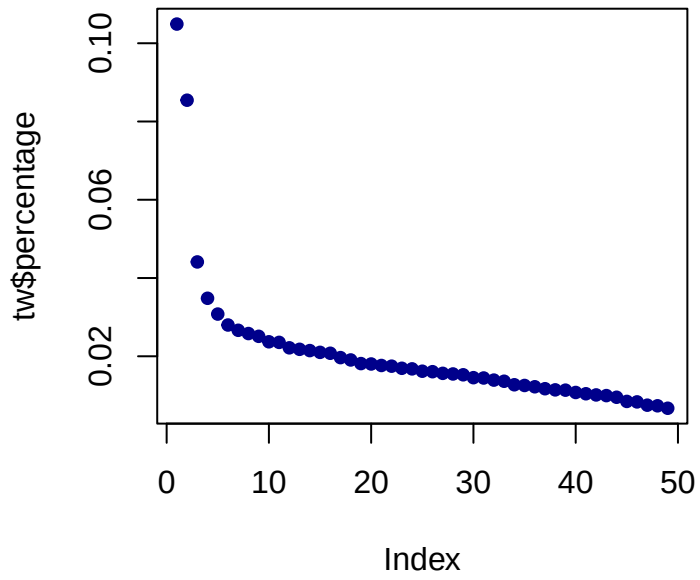


Figure 1: Screeplot for the percentage of variance explained by each component in a PCA of the genetic data. The elbow at $K = 4$ indicates that there are 5 major genetic clusters in the data.

STRUCTURE (Pritchard et al., 2000; François and Durand, 2010). Assuming K ancestral populations, the `R` function `snmf()` computes least-squares estimates of ancestry proportions and ancestral allelic frequencies (Frichot et al., 2014).

```
# main options
# K = number of ancestral populations
# entropy = TRUE computes the cross-entropy criterion,
# CPU = 4 is the number of CPU used (hidden input)
project = NULL
project = snmf("genotypes.geno",
               K = 1:10,
               entropy = TRUE,
               repetitions = 10,
```

```
                project = "new")
```

The `snmf()` function computes an entropy criterion that evaluates the quality of fit of the statistical model to the data by using a cross-validation technique (Figure 2). The entropy criterion can help choosing the number of ancestral populations that best explains the genotypic data (Alexander and Lange, 2011; Frichot et al., 2014). Here we have a clear minimum at $K = 4$, suggesting 4 genetic clusters in the data. Often, the plot shows a less clear pattern, and choosing the "elbow" point can be a good approach. The number of ancestral populations is closely linked to the number of principal components of the genomic data. Both numbers can help determining the number of latent factors when correcting for confounding effects due to population structure in GEA tests with `lfmm()` and with `lfmm2()`.

```
# plot cross-entropy criterion for all runs in the snmf project
plot(project, col = "blue", pch = 19, cex = 1.2)
```

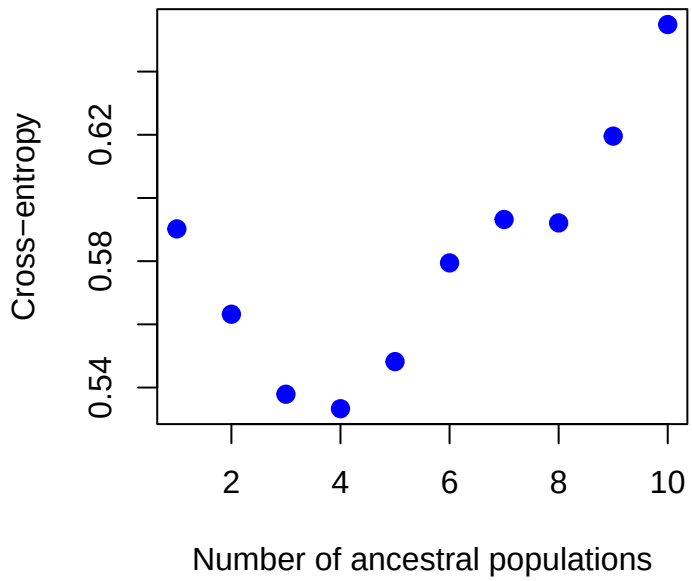


Figure 2: Value of the cross-entropy criterion as a function of the number of populations in `snmf`.

The next step is to display a barplot for the ancestry proportions, recorded in the Q-matrix. In Figure 3, the `Q()` function of `LEA` is called and the Q-

matrix output is converted into a `Qmatrix` object. The conversion of the Q-matrix as a `Qmatrix` object can be useful for running improved graphical functions from other packages such as `tess3r` (Caye et al., 2016, 2018).

```
# select the best run for K = 4 clusters
best = which.min(cross.entropy(project, K = 4))
my.colors <- c("tomato", "lightblue",
               "olivedrab", "gold")
barchart(project, K = 4, run = best,
         border = NA, space = 0,
         col = my.colors,
         xlab = "Individuals",
         ylab = "Ancestry proportions",
         main = "Ancestry matrix") -> bp
axis(1, at = 1:length(bp$order),
     labels = bp$order, las=1,
     cex.axis = .4)
```

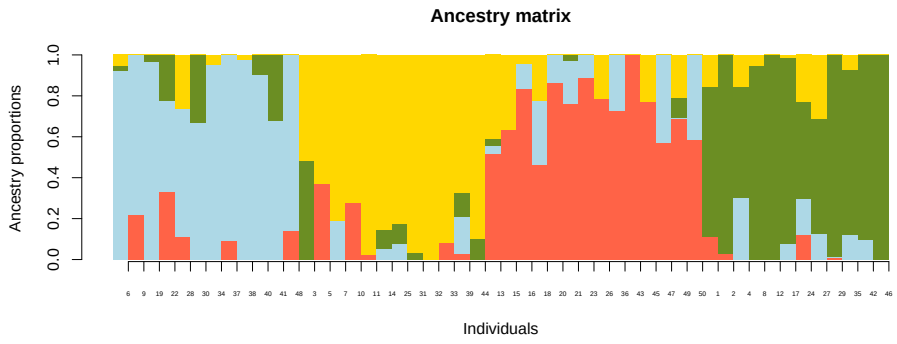


Figure 3: Ancestry coefficients obtained from `snmf()`.

3.3 Population differentation tests using `snmf()`

common approaches to detecting outlier loci from a genomic background focus on extreme values of the fixation index, F_{ST}, across loci. The `snmf()` function can compute fixation indices when the population is genetically continuous, when predefining subpopulations is difficult, and in the presence of admixed individuals in the sample (Martins et al., 2016). In the snmf approach, population differentiation statistics are computed from ancestry coefficients obtained from an `snmf` object, and p-values are returned for all loci. Figure 4 is an example of outlier analysis with `snmf()`.

```
# Genome scan for selection: opulation differentiation tests
p = snmf.pvalues(project,
                 entropy = TRUE,
                 ploidy = 2,
                 K = 4)
pvalues = p$pvalues
par(mfrow = c(2,1))
hist(pvalues, col = "orange")
plot(-log10(pvalues), pch = 19, col = "blue", cex = .5)
```

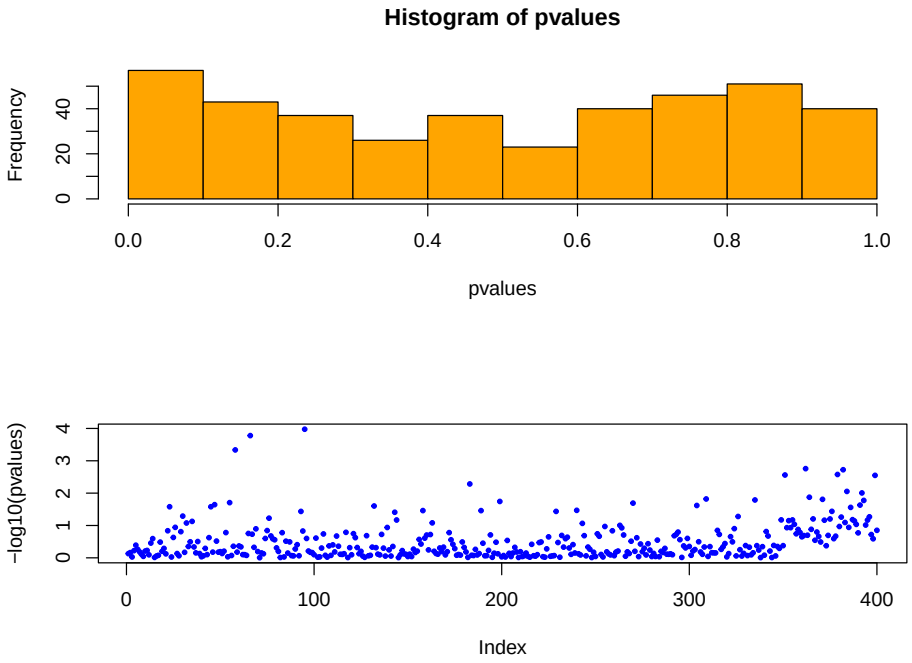


Figure 4: *P*-values for population differentiation tests with `snmf()`.

3.4 Missing genotype imputation using `snmf`

Missing genotypes are critical to genome-wide association and GEA studies. Before running an association study, an important step is to replace the missing data, represented as '9' in the `geno` and `lfmm`) files, by imputed values. To provide an example of missing data imputation, let us start by removing 100 genotypes from the original data.The resulting matrix is saved in the file `genoM.lfmm`.

```
# creation of a genotype matrix  with missing genotypes
dat = as.numeric(tutorial.R)
dat[sample(1:length(dat), 100)] <-  9
dat <- matrix(dat, nrow = 50, ncol = 400)
write.lfmm(dat, "genoM.lfmm")

## [1] "genoM.lfmm"
```

Next, the function **snmf()** can be run on the data with missing genotypes as follows.

```
project.missing = snmf("genoM.lfmm", K = 4,
        entropy = TRUE, repetitions = 10,
        project = "new")
```

Imputation of missing genoypes is based on estimated ancestry coefficients and on ancestral genotype frequencies. The **snmf** project data can be used to impute the missing data as follows.

```
# select the run with the lowest cross-entropy value
best = which.min(cross.entropy(project.missing, K = 4))

# Impute the missing genotypes
impute(project.missing, "genoM.lfmm",
       method = 'mode', K = 4, run = best)

## Missing genotype imputation for K = 4
## Missing genotype imputation for run = 3
## Results are written in the file:  genoM.lfmm_imputed.lfmm

# Proportion of correct imputation results
dat.imp = read.lfmm("genoM.lfmm_imputed.lfmm")
mean( tutorial.R[dat == 9] == dat.imp[dat == 9] )

## [1] 0.77
```

The results are saved in an output file with the string **"imputed"** in its suffix name. Since we removed genotypes from the original matrix, the accuracy of results can be evaluated.The last **R** command above shows the proportion of missing genotypes that were correctly imputed by the **impute()** function.

4 Ecological association tests using lfmm

The `R` package `LEA` performs genome-wide association analysis based on latent factor mixed models using the `lfmm()` and `lfmm2()` functions (Frichot et al., 2013; Caye et al., 2019). The two functions are based on the same statistical model, but their estimation algorithms are different. `lfmm()` is Bayesian method that uses a Monte-Carlo Markov Chain algoritm, and `lfmm2()` is a frequentist approach that uses least-squares estimates. **For large genotype matrices (e.g. more than 1,000-10,000 genetic loci), the best is to use `lfmm2()`.**

To recall the statistical model, let G denote the genotype matrix, storing allele frequencies for each individual at each locus, and let X denote a set of d ecological predictors (or phenotypic traits). LFMMs consider the genotype matrix entries as response variables in a latent factor regression model

$$G_{i\ell} = \mu_\ell + \beta_\ell^T X_i + U_i^T V_\ell + \epsilon_{i\ell}\,, \tag{1}$$

where μ_ℓ is a locus specific effect, β_ℓ is a d-dimensional vector of regression coefficients, U_i contains K latent factors, and V_ℓ contains their corresponding loadings (i stands for an individual and ℓ for a locus). The residual terms, $\epsilon_{i\ell}$, are statistically independent Gaussian variables with mean zero and variance σ^2.

In latent factor models, effect size of environmental predictors on allelic frequencies are estimated simultaneously with the K unobserved latent factors that model confounding effects. In principle, the latent factors include levels of population structure due to shared demographic history or background genetic variation, but they are generally different of population structure estimates (such as principal components). After correction for confounding effects, the level of association between a particular allelic frequency and an ecological predictor is often interpreted as a signature of natural selection at the corresponding locus.

Running LFMM. The `lfmm()` program is based on a stochastic algorithm (MCMC). See the section on `lfmm2()` for an alternative method which provides exact results under simplified assumptions. We recommend using large number of cycles (e.g., `-i 6000`) and the burnin period should set at least to one-half of the total number of cycles (`-b 3000`). We have noticed that results are sensitive to the run-length parameter when data sets have relatively small sizes (e.g., a few hundreds of individuals, a few thousands of loci). We recommend increasing the burnin period and the total number of cycles in this situation.

```
# main options:
# K = the number of latent factors
```

```
# Runs with K = 6 using 5 repetitions.
project = NULL
project = lfmm("genotypes.lfmm",
               "gradients.env",
                K = 6,
                repetitions = 5,
                project = "new")

## lfmm uses a very naive imputation method which has low power
when genotypes are missing:  See impute() for a better imputation
method.
## Note that lfmm has an improved estimation algorithm implemented
in lfmm2, which should be the prefered option.
```

Deciding the number of latent factors. Deciding an appropriate value for the number of latent factors in the `lfmm()` call can be based on the analysis of histograms of test significance values. Ideally, histograms should be flat, with a peak close to zero. A good hint for the number of factors is the elbow value in the PCA screeplot obtained from the genotype matrix.

Since the objective is to control the false discovery rate (FDR) while keeping reasonable power to reject the null hypothesis, we recommend using several runs for each value of K and to combine p-values (use 5 to 10 runs, see our script below). Choosing values of K for which the histograms show their correct shape warrants that the FDR can be controlled efficiently.

Testing all K values in a large range, say from 1 to 20, 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. For example, if the `snmf()` command estimates 4 ancestral populations, then running `lfmm()` for $K = 3 - 6$ often provides good results.

Combining z-scores obtained from multiple runs. We use the Fisher-Stouffer 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 can increase the power of `lfmm` tests. This procedure is implemented in `LEA` function `lfmm.pvalues()`.

```
# compute adjusted p-values
p = lfmm.pvalues(project, K = 6)
pvalues = p$pvalues
```

The results displayed in Figure 5 show that the null-hypothesis is correctly calibrated. The loci exhibiting significant associations are found at the right on the Manhattan plot.

```
# GEA significance test
par(mfrow = c(2,1))
hist(pvalues, col = "lightblue")
plot(-log10(pvalues), pch = 19, col = "blue", cex = .7)
```

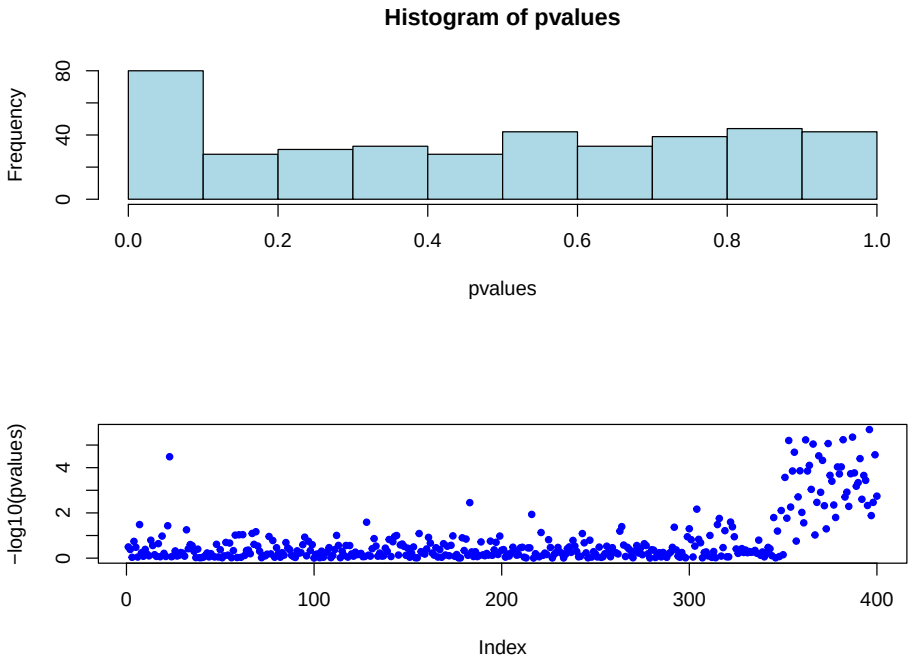


Figure 5: P-values for LFMM tests. The loci showing significant associations are at the right on the Manhattan plot.

To adjust p-values for multiple testing issues, we use the Benjamini-Hochberg procedure (Benjamini and Hochberg, 1995). We set the expected levels of FDR to $q = 5\%$, 10%, 15% and 20% respectively . The lists of candidate loci are given by the following script. Since we the ground truth is known for the simulated data, we can compare the expected FDR levels to their observed levels, and compute the power (TPR, true positive rate) of the test.

```
for (alpha in c(.05,.1,.15,.2)) {
    # expected FDR
```

```
    print(paste("Expected FDR:", alpha))
    L = length(pvalues)

    # return a list of candidates with expected FDR alpha.
    # Benjamini-Hochberg's algorithm:
    w = which(sort(pvalues) < alpha * (1:L) / L)
    candidates = order(pvalues)[w]

    # estimated FDR and True Positive Rate
    Lc = length(candidates)
    estimated.FDR = sum(candidates <= 350)/Lc
    print(paste("Observed FDR:",
                round(estimated.FDR, digits = 2)))
    estimated.TPR = sum(candidates > 350)/50
    print(paste("Estimated TPR:",
                round(estimated.TPR, digits = 2)))
}
```

```
## [1] "Expected FDR: 0.05"
## [1] "Observed FDR: 0.05"
## [1] "Estimated TPR: 0.82"
## [1] "Expected FDR: 0.1"
## [1] "Observed FDR: 0.1"
## [1] "Estimated TPR: 0.86"
## [1] "Expected FDR: 0.15"
## [1] "Observed FDR: 0.13"
## [1] "Estimated TPR: 0.92"
## [1] "Expected FDR: 0.2"
## [1] "Observed FDR: 0.16"
## [1] "Estimated TPR: 0.94"
```

5 Ecological association tests using lfmm2

As an efficient alternative to the MCMC algorithm implemented in the `lfmm()`, genome-wide association and GEA analysis can be performed by using the `lfmm2()` command. This function allows estimating K latent factors and the effect sizes corresponding to environmental variables in the same statistical model as `lfmm()`. For `lfmm2()`, the estimation algorithm is based on exact solutions of a least-squares minimization problem (Caye et al., 2019). For large data sets, `lfmm2()` is much faster than the MCMC version (Gain and François, 2021).

Using `lfmm2()` decouples latent factor estimation from association tests.

The decoupling allows implementing various types of tests including linear or generalized linear models, and also to test genotypes not used in the estimation procedures (e.g., unpruned genotypes).

Let us consider an example with 200 diploid individuals genotyped at 510 SNP loci, and four ecological predictors. The principal component analysis of the genotype matrix suggests that there are two main axes of variation ($K = 2$). The latent factors are estimated as follows.

```
# load simulated data
data("offset_example")
# 200 diploid individuals genotyped at 510 SNP
Y <- offset_example$geno
# 4 environmental variables
X <- offset_example$env

mod.lfmm2 <- lfmm2(input = Y, env = X, K = 2)
```

The `lfmm2()` command generates an object of class `lfmm2Class` which contains estimated factors (`mod2@U`) and loadings (`mod2@V`) for being included as correction factors in genome-wide association tests. Note that `LEA` is using S4 objects rather than S3 objects. Getting the factors could be useful for implementing customized tests. For example, they could be used for computing a covariance matrix for random effects in a mixed linear model. The `lfmm2.test()` function implements simpler tests such as linear or generalized linear model tests.

Although there are four environmental predictors, we want a single test significance value for association with environment at each locus. For this, we test the fit of the model using Fisher's tests. The test significance values are computed using the option `full = TRUE`.

Let us consider a second example simulated from the LFMM generative model. The simulated genotype matrix contains $n = 100$ individuals genotyped for $L = 1,000$ loci, ten of which are truly associated with an artificial environmental variable, X.

```
# Simulate non-null effect sizes for 10 target loci
#individuals
n = 100
#loci
L = 1000
# Environmental variable
X = as.matrix(rnorm(n))
# effect sizes
B = rep(0, L)
target = sample(1:L, 10)
```

```
# GEA significance test
# showing the K = 2 estimated factors
plot(mod.lfmm2@U, col = "grey", pch = 19,
     xlab = "Factor 1",
     ylab = "Factor 2")
```

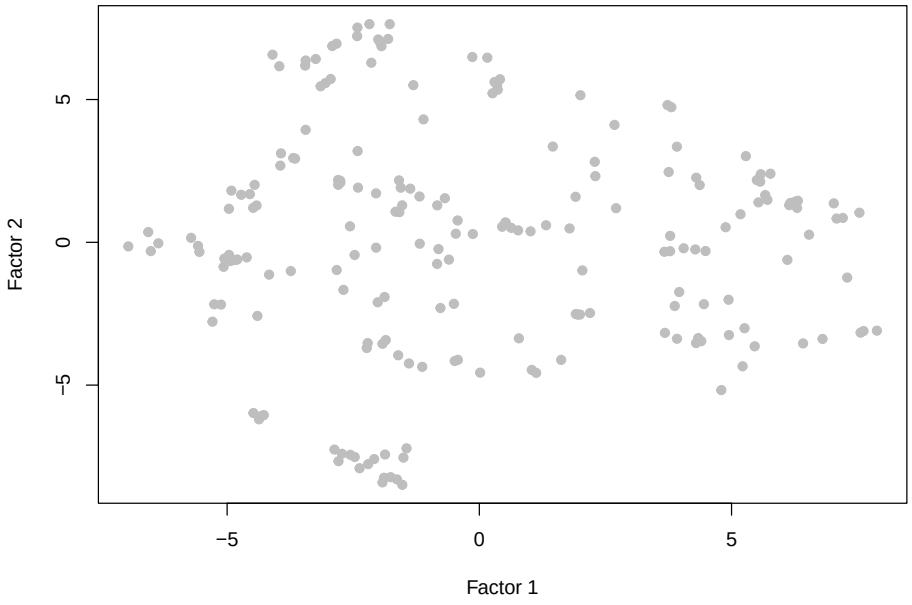


Figure 6: Latent factors estimated in the offset example data analysis.

```
B[target] = runif(10, -10, 10)
```

The latent factors, $\mathbf{U}$, contained in an $n \times 3$ matrix, are random vectors created as follows. Correlations between the factors and environmental predictor, X, are introduced in the simulation model.

```
# Create 3 hidden factors and their loadings
U = t(tcrossprod(as.matrix(c(-1,0.5,1.5)), X)) +
    matrix(rnorm(3*n), ncol = 3)

V <- matrix(rnorm(3*L), ncol = 3)
```

The genotypic matrix, $\mathbf{Y}$, is simulated according to an approximation of the LFMM generative model. This matrix has dimension $n \times L$.

```
pv <- lfmm2.test(object = mod.lfmm2,
                 input = Y,
                 env = X,
                 full = TRUE)
plot(-log10(pv$pvalues), col = "grey", cex = .5, pch = 19)
abline(h = -log10(0.1/510), lty = 2, col = "orange")
```

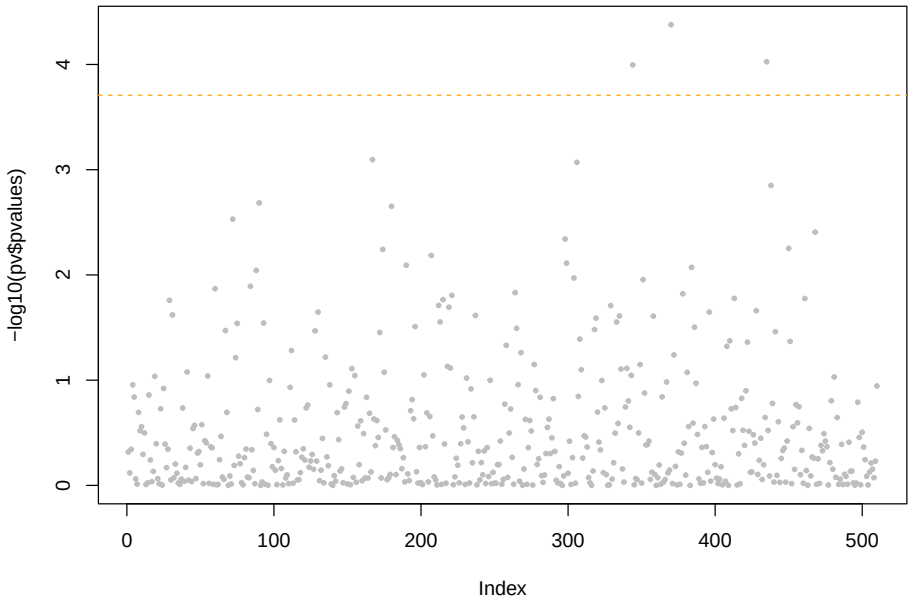


Figure 7: Offset example: p-values for LFMM2 tests.

```
# Simulate a matrix containing haploid genotypes
Y <-  tcrossprod(as.matrix(X), B) +
      tcrossprod(U, V) +
      matrix(rnorm(n*L, sd = .5), nrow = n)

Y <- matrix(as.numeric(Y > 0), ncol = L)
```

We fit an LFMM by using $K = 3$ latent factors. This value corresponds to the true value in the model (note that $K = 3$ could be easily recovered from a PCA screeplot).

```
# Fitting an LFMM with K = 3 factors
mod <- lfmm2(input = Y, env = X, K = 3)
```

To adjust p-values for multiple testing issues, we can use the Benjamini-Hochberg procedure as with the `lfmm()` tests. The tests and a Manhattan plot can be performed as follows.

```
# Computing P-values and plotting their minus log10 values
pv <- lfmm2.test(object = mod,
                 input = Y,
                 env = X,
                 linear = TRUE)

plot(-log10(pv$pvalues), col = "grey", cex = .6, pch = 19)
points(target, -log10(pv$pvalues[target]), col = "red")
```

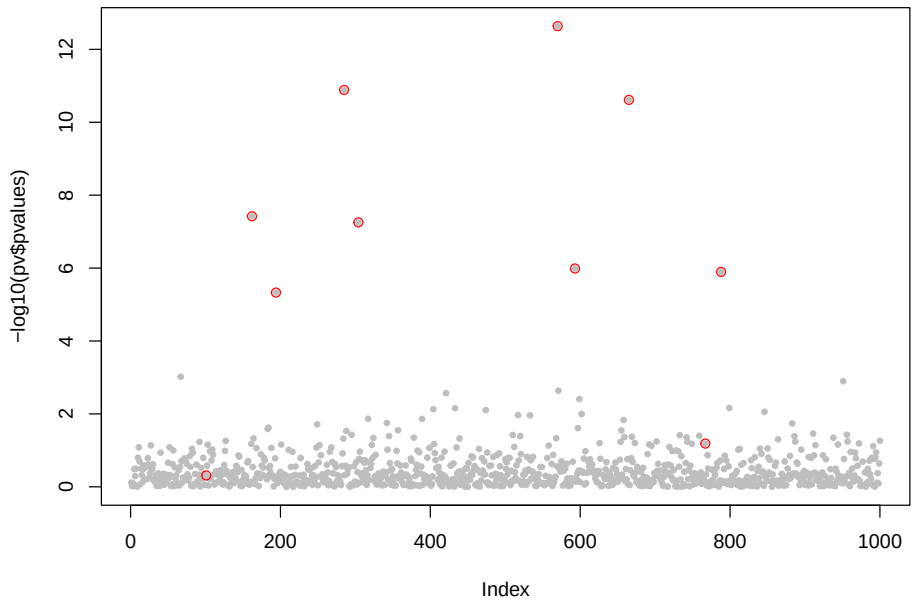


Figure 8: Manhattan plot of $\log_{10}$ p-values for LFMM2 tests. The loci showing real associations are circled in red.

6 Predictive Ecological Genomics: genetic gap

Genomic offset is a genome-based approach to assess the maladaptation of populations to abrupt alteration of their habitat. Genomic offset statistics use environmental and genomic data to predict shifts in adaptive traits without direct observations of those traits. By relying on GEA models, genomic offset statistics are based on differences in allelic frequencies for two sets of conditions in an ecological niche.

The `R` package `LEA` implements a geometric statistic to estimate genomic offset (genetic gap), based on the covariance matrix of effect sizes obtained from an LFMM (Gain, 2022). The same function can also compute a modified version of the risk of nonadaptedness (RONA) that includes correction for confounding factors and performs a multivariate analysis of environmental effects.

The `genetic.gap()` function takes as input the environmental and genomic data that are used to adjust the LFMM. It also requires a matrix of environmental predictors measured at new locations (new.env), and a matrix of predicted environmental variables for the new locations (pred.env) in the same format as the new.env ones. New locations at not necessarily new, as they could be taken as sample locations (new.env = env).

Loading the offset example again, the genotype matrix is of dimension 200×510, corresponding to 200 sampled individuals genotyped at 510 loci. The data contains four environmental predictors, and predicted values of (future) change.

```
data("offset_example")
Y <- offset_example$geno
X <- offset_example$env
X.pred <- offset_example$env.pred
```

Below, we compute the geometric offset (genetic gap) at each sample locations, using all genomic loci in the genotype matrix. Note that the `candidate.loci = TRUE` option allows us to use any subset of loci, in particular those being above the significance level in the GEA analysis.

```
g.gap.scaled <- genetic.gap(input = Y,
                            env = X,
                            pred.env = X.pred,
                            scale = TRUE,
                            K = 2)
```

The ecological predictors were correlated to each other, and they contained redundant information. The analysis of the covariance matrix and the screeplot of eigenvalues showed that the dimension of the environmental space was equal to two.

```
g.gap.scaled$vectors[,1:2]^2

##              [,1]         [,2]
## [1,] 0.513933196 0.2127500317
## [2,] 0.179886480 0.6453681390
## [3,] 0.302617770 0.0005067079
## [4,] 0.003562555 0.1413751215
```

The loadings for the first two variables indicated the relative contributions of each predictor to local adaptation. Axis 1 mainly corresponded to variable 1 (51%), and axis 2 mainly corresponded to variable 2 (64%). The genetic gap correlated with the squared ecological distance computed from variables 1 and 2, showing that larger shifts in those variables are likely to create larger fitness loss.

```
par(mfrow = c(1,2))

barplot(g.gap.scaled$eigenvalues,
        col = "orange",
        xlab = "Axes",
        ylab = "Eigenvalues")

Delta = X[,1:2] - X.pred[,1:2]
squared.env.dist = rowSums(Delta^2)
plot(squared.env.dist, g.gap.scaled$offset,  cex = .6)
```

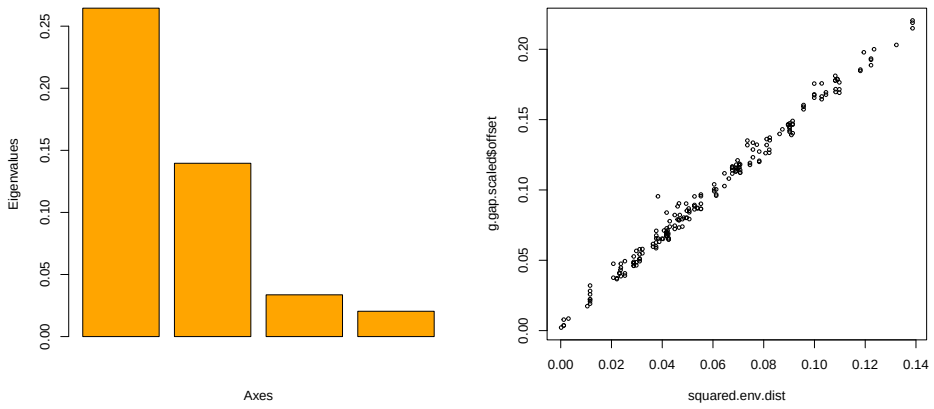


Figure 9: Left: Barplot of covariance eigenvalues showing that the dimension of environmental space is two. Right: The genetic gap correlated with squared ecological distance, showing that larger shifts corresponded with larger offsets.

References

Alexander DH and Lange K. 2011. Enhancements to the ADMIXTURE algorithm for individual ancestry estimation. BMC Bioinformatics 12:246.

Benjamini Y and Hochberg Y. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. J R Stat Soc B Met. pp. 289–300.

Caye K, Jay F, Michel O and Francois O. 2016. Fast inference of individual admixture coefficients using geographic data. bioRxiv p. 080291.

Caye K, Jay F, Michel O and Francois O. 2018. Fast inference of individ-

ual admixture coefficients using geographic data. The Annals of Applied Statistics 12:586–608.

Caye K, Jumentier B, Lepeule J and François O. 2019. Lfmm 2: Fast and accurate inference of gene-environment associations in genome-wide studies. Molecular Biology and Evolution 36:852–860.

François O and Durand E. 2010. Spatially explicit bayesian clustering models in population genetics. Mol Ecol Resour. 10:773–784.

Frichot E and François O. 2015. LEA: an R package for Landscape and Ecological Association studies. Methods in Ecology and Evolution 6:925–929.

Frichot E, Mathieu F, Trouillon T, Bouchard G and François O. 2014. Fast and efficient estimation of individual ancestry coefficients. Genetics 196:973–983.

Frichot E, Schoville SD, Bouchard G and François O. 2013. Testing for associations between loci and environmental gradients using latent factor mixed models. Mol Biol Evol. 30:1687–1699.

Gain C and François O. 2021. LEA 3: Factor models in population genetics and ecological genomics with R. Molecular Ecology Ressources 21:2738–2748.

Gain C et al. 2022. A quantitative theory for genomic offset statistics. BioRxiv .

Martins H, Caye K, Luu K, Blum MGB and Francois O. 2016. Identifying outlier loci in admixed and in continuous populations using ancestral population differentiation statistics. Molecular Ecology 25:5029–5042.

Patterson N, Price AL and Reich D 2006. Population structure and eigenanalysis. PLoS Genet. 2:20.

Pritchard JK, Stephens M and Donnelly P. 2000. Inference of population structure using multilocus genotype data. Genetics 155:945–959.