# Package 'LEA'

February 2, 2026

**Title** LEA: an R package for Landscape and Ecological Association
Studies

**Version** 3.23.0

**Date** 2025-11-3

**Author** Eric Frichot <eric.frichot@gmail.com>, Olivier Francois

<olivier.francois@grenoble-inp.fr>, Clement Gain

<clement.gain@univ-grenoble-alpes.fr>

**Maintainer** Olivier Francois <olivier.francois@grenoble-inp.fr>

**Depends** R (>= 3.3.0), methods, stats, utils, graphics

**Suggests** knitr

**Description** LEA is an R package dedicated to population genomics, landscape genomics and
genotype-environment association tests. LEA can run analyses of
population structure and genome-wide tests for local adaptation,
and also performs imputation of missing genotypes.
The package includes statistical methods for estimating ancestry
coefficients from large genotypic matrices and for evaluating the
number of ancestral populations (snmf). It performs statistical
tests using latent factor mixed models for identifying
genetic polymorphisms that exhibit association with
environmental gradients or phenotypic traits (lfmm2).
In addition, LEA computes values of genetic offset statistics based
on new or predicted environments (genetic.gap, genetic.offset).
LEA is mainly based on optimized programs
that can scale with the dimensions of large data sets.

**License** GPL-3

**biocViews** Software, Statistical Method, Clustering, Regression

**URL** http://membres-timc.imag.fr/Olivier.Francois/lea.html

**NeedsCompilation** yes

**VignetteBuilder** knitr

**RoxygenNote** 6.0.1

**git_url** https://git.bioconductor.org/packages/LEA

**git_branch** devel

**git_last_commit** f812da2

**git_last_commit_date** 2025-10-29

**Repository** Bioconductor 3.23

**Date/Publication** 2026-02-01

# Contents

---

| | |
|---|---|
| `LEA-package` | *LEA: an R package for Landscape and Ecological Associations studies.* |

---

## Description

LEA is an R package dedicated to landscape genomics and ecological association tests. LEA can run analyses of population structure and genome scans for local adaptation. It includes statistical methods for estimating ancestry coefficients from large genotypic matrices and evaluating the number of ancestral populations (snmf, pca) and identifying genetic polymorphisms that exhibit high correlation with some environmental gradient or with the variables used as proxies for ecological pressures (lfmm). LEA is mainly based on optimized C programs that can scale with the dimension of very large data sets.

## Details

| | |
|---|---|
| Package: | LEA |
| Type: | Package |
| Version: | 2.0 |
| Date: | 2017-07-16 |
| License: | GPL-3 |

## Author(s)

Eric Frichot Olivier Francois Maintainer: Olivier Francois <olivier.francois@grenoble-inp.fr>

ancestrymap                ancestrymap *format description*

## Description

Description of the `ancestrymap` format. The `ancestrymap` format can be used as an input format for genotypic matrices in the functions `pca`, `lfmm` and `snmf`.

## Details

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. Genotypes for a given SNP name are written in consecutive lines. The number of alleles can be the number of reference alleles or the number of derived alleles. Missing genotypes are encoded by the value 9.

Here is an example of a genotypic matrix using the ancestrymap format with 3 individuals and 4 SNPs:

```
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
```

## Author(s)

Eric Frichot

## See Also

ancestrymap2lfmm ancestrymap2geno geno lfmm.data ped vcf

---

ancestrymap2geno                 *Convert from* ancestrymap *to* geno *format*

---

### Description

A function that converts from the ancestrymap format to the geno format.

### Usage

```
ancestrymap2geno(input.file, output.file = NULL, force = TRUE)
```

### Arguments

| | |
|---|---|
| input.file | A character string containing a path to the input file, a genotypic matrix in the ancestrymap format. |
| output.file | A character string containing a path to the output file, a genotypic matrix in the geno format. By default, the name of the output file is the same name as the input file with a .geno extension. |
| force | A boolean option. If FALSE, the input file is converted only if the output file does not exist. If TRUE, convert the file anyway. |

### Value

| | |
|---|---|
| output.file | A character string containing a path to the output file, a genotypic matrix in the geno format. |

### Author(s)

Eric Frichot

### See Also

ancestrymap geno read.geno ancestrymap2lfmm geno2lfmm ped2lfmm ped2geno vcf2geno lfmm2geno

### Examples

```
# Creation of of file called "example.ancestrymap"
# a file containing 4 SNPs for 3 individuals.
data("example_ancestrymap")
write.table(example_ancestrymap,"example.ancestrymap",
col.names = FALSE, row.names = FALSE, quote = FALSE)

# Conversion    from the ancestrymap format ("example.ancestrymap")
#               to the geno format ("example.geno").
# By default,   the name of the output file is the same name
#               as the input file with a .geno extension.
# Create file:  "example.geno".
```

```
output = ancestrymap2geno("example.ancestrymap")

# Conversion    from the ancestrymap format (example.ancestrymap)
#               to the geno format with the output file called plop.geno.
# Create file:  "plop.geno".
output = ancestrymap2geno("example.ancestrymap", "plop.geno")

# As force = false and the file "example.geno" already exists,
# nothing happens.
output = ancestrymap2geno("example.ancestrymap", force = FALSE)
```

---

ancestrymap2lfmm              *Convert from* ancestrymap *to* lfmm *format*

---

### Description

A function that converts from the ancestrymap format to the lfmm format.

### Usage

```
ancestrymap2lfmm(input.file, output.file = NULL, force = TRUE)
```

### Arguments

input.file      A character string containing a path to the input file, a genotypic matrix in the
                ancestrymap format.

output.file     A character string containing a path to the output file, a genotypic matric in the
                lfmm format. By default, the name of the output file is the same name as the
                input file with a .lfmm extension.

force           A boolean option. If FALSE, the input file is converted only if the output file
                does not exist. If TRUE, convert the file anyway.

### Value

output.file     A character string containing a path to the output file, a genotypic matric in the
                lfmm format.

### Author(s)

Eric Frichot

### See Also

ancestrymap lfmm.data ancestrymap2geno geno2lfmm ped2lfmm ped2geno vcf2geno lfmm2geno

## Examples

```
# Creation of a file called "example.ancestrymap"
# containing 4 SNPs for 3 individuals.
data("example_ancestrymap")
write.table(example_ancestrymap,"example.ancestrymap",
col.names = FALSE, row.names = FALSE, quote = FALSE)

# Conversion    from the ancestrymap format ("example.ancestrymap")
#               to the lfmm format ("example.lfmm").
# By default,   the name of the output file is the same name
#               as the input file with a .lfmm extension.
# Create file:  "example.lfmm".
output = ancestrymap2lfmm("example.ancestrymap")

# Conversion    from the ancestrymap format (example.ancestrymap)
#               to the geno format with the output file called plop.lfmm.
# Create file:  "plop.lfmm".
output = ancestrymap2lfmm("example.ancestrymap", "plop.lfmm")

# As force = false and the file "example.lfmm" already exists,
# nothing happens.
output = ancestrymap2lfmm("example.ancestrymap", force = FALSE)
```

| barchart | *Bar plot representation of an snmf Q-matrix* |
|----------|-----------------------------------------------|

## Description

This function displays a bar plot/bar chart representation of the Q-matrix computed from an snmf run. The function can use a sort by Q option. See snmf.

## Usage

```
barchart (object, K, run, sort.by.Q = TRUE, lab = FALSE, ...)
```

## Arguments

| | |
|----------|---|
| object | an snmfProject object. |
| K | an integer value corresponding to number of ancestral populations. |
| run | an integer value. Usually the run number that minimizes the cross-entropy criterion. |
| sort.by.Q | a Boolean value indicating whether individuals should be sorted by their ancestry or not. |
| lab | a list of individual labels. |
| ... | other parameters of the function barplot.default. |

**Value**

A permutation of individual labels used in the sort.by.Q option (order). Displays the Q matrix.

**Author(s)**

Olivier Francois

**See Also**

[snmf](#)

**Examples**

```
# creation of a genotype file: genotypes.geno.
# 400 SNPs for 50 individuals.

data("tutorial")
write.geno(tutorial.R, "genotypes.geno")


################
# running snmf #
################

project.snmf <- snmf("genotypes.geno",
                    K = 4, entropy = TRUE,
                    repetitions = 10,
                    project = "new")

# get the cross-entropy value for each run
ce <- cross.entropy(project.snmf, K = 4)

# select the run with the lowest cross-entropy value
best <- which.min(ce)

# plot the ancestry coefficients for the best run and K = 4

my.colors <- c("tomato", "lightblue", "olivedrab", "gold")

barchart(project.snmf, 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 = 3,
      cex.axis = .4)
```

---

create.dataset                    *create a data set with masked data*

---

### Description

[create.dataset](#) creates a data set with a given percentage of masked data from the original data set. It is used to calculate the [cross.entropy](#) criterion.

### Usage

```
create.dataset (input.file, output.file, seed = -1, percentage = 0.05)
```

### Arguments

input.file      A character string containing a path to the input file, a genotypic matrix in the [geno](#) format.

output.file     A character string containing a path to the output file, a genotypic matrix in the [geno](#) format. The output file is the input file with masked genotypes. By default, the name of the output file is the same name as the input file with a _I.geno extension.

seed            A seed to initialize the random number generator. By default, the seed is randomly chosen.

percentage      A numeric value between 0 and 1 containing the percentage of masked genotypes.

### Details

This is an internal function, automatically called by [snmf](#) with the entropy option.

### Value

output.file     A character string containing a path to the output file, a genotypic matrix in the [geno](#) format.

### Author(s)

Eric Frichot

### See Also

[geno](#) [snmf](#) [cross.entropy](#)

## Examples

```
# Creation of tuto.geno
# A file containing 400 SNPs for 50 individuals.
data("tutorial")
write.geno(tutorial.R,"genotypes.geno")

# Creation     of the masked data file
# Create file:  "genotypes_I.geno"
output = create.dataset("genotypes.geno")
```

| cross.entropy | *Cross-entropy criterion for snmf runs* |
|---------------|------------------------------------------|

## Description

Return the cross-entropy criterion for runs of snmfcwith K ancestral populations. The cross-entropy criterion is based on the prediction of masked genotypes to evaluate the fit of a model with K populations. The cross-entropy criterion helps choosing the number of ancestral populations or a best run for a fixed value of K. A smaller value of cross-entropy means a better run in terms of prediction capability. The cross-entropy criterion is computed by the [snmf](#) function when the entropy Boolean option is TRUE.

## Usage

```
cross.entropy(object, K, run)
```

## Arguments

| | |
|---------|-----------------------------------------|
| object  | A snmfProject object. |
| K       | The number of ancestral populations. |
| run     | A vector of run labels. |

## Value

| | |
|-----|-----------------------------------------|
| res | A matrix containing the cross-entropy criterion for runs with K ancestral populations. |

## Author(s)

Eric Frichot

## See Also

[geno snmf G Q](#)

**Examples**

```
### Example of analyses using snmf ###

# creation of a genotype file: genotypes.geno.
# The data contains 400 SNPs for 50 individuals.
data("tutorial")
write.geno(tutorial.R, "genotypes.geno")

################
# running snmf #
###############

# Runs with K = 3 populations
# cross-entropy is computed for 2 runs.
project = NULL
project = snmf("genotypes.geno",
               K = 3,
               entropy = TRUE,
               repetitions = 2,
               project = "new")

# get the cross-entropy for all runs for K = 3
ce = cross.entropy(project, K = 3)

# get the cross-entropy for the 2nd run for K = 3
ce = cross.entropy(project, K = 3, run = 2)
```

---

cross.entropy.estimation

*compute the cross-entropy criterion*

---

**Description**

Calculate the cross-entropy criterion. This is an internal function, automatically called by snmf. The cross-entropy criterion is a value based on the prediction of masked genotypes to evaluate the error of ancestry estimation. The criterion will help to choose the best number of ancestral population (K) and the best run among a set of runs in snmf. A smaller value of cross-entropy means a better run in terms of prediction capacity. The cross.entropy.estimation function displays the cross-entropy criterion estimated on all data and on masked data based on the input file, the masked data file (created by create.dataset, the estimation of the ancestry coefficients Q and the estimation of ancestral genotypic frequencies, G (calculated by snmf). The cross-entropy estimation for all data is always lower than the cross-entropy estimation for masked data. The cross-entropy estimation useful to compare runs is the cross-entropy estimation for masked data. The cross-entropy criterion can also be automatically calculated by the snmf function with the entropy option.

**Usage**

```
cross.entropy.estimation (input.file, K, masked.file, Q.file, G.file,
    ploidy = 2)
```

## Arguments

| | |
|---|---|
| `input.file` | A character string containing a path to the input file without masked genotypes, a genotypic matrix in the [geno] format. |
| `K` | An integer corresponding to the number of ancestral populations. |
| `masked.file` | A character string containing a path to the input file with masked genotypes, a genotypic matrix in the [geno] format. This file can be generated with the function, [create.dataset]). By default, the name of the masked data file is the same name as the input file with a `_I.geno` extension. |
| `Q.file` | A character string containing a path to the input ancestry coefficient matrix Q. By default, the name of this file is the same name as the input file with a `K.Q` extension. |
| `G.file` | A character string containing a path to the input ancestral genotype frequency matrix G. By default, the name of this file is the same name as the input file with a `K.G` extension (`input_file.K.G`). |
| `ploidy` | 1 if haploid, 2 if diploid, n if n-ploid. |

## Value

`cross.entropy.estimation` returns a list containing the following components:

| | |
|---|---|
| `masked.ce` | The value of the cross-entropy criterion of the masked genotypes. |
| `all.ce` | The value of the cross-entropy criterion of all the genotypes. |

## Author(s)

Eric Frichot

## References

Frichot E, Mathieu F, Trouillon T, Bouchard G, Francois O. (2014). *Fast and Efficient Estimation of Individual Ancestry Coefficients*. Genetics, 194(4) : 973–983.

## See Also

[geno] [create.dataset] [snmf]

## Examples

```
# Creation of tuto.geno
# A file containing 400 SNPs for 50 individuals.
data("tutorial")
write.geno(tutorial.R,"genotypes.geno")

# The following command are equivalent with
# project = snmf("genotypes.geno", entropy = TRUE, K = 3)
# cross.entropy(project)

# Creation      of the masked data file
```

```
# Create file:  "genotypes_I.geno"
output = create.dataset("genotypes.geno")

# run of snmf with genotypes_I.geno and K = 3
project = snmf("genotypes_I.geno", K = 3, project = "new")

# calculate the cross-entropy
res = cross.entropy.estimation("genotypes.geno", K = 3, "genotypes_I.geno",
    "./genotypes_I.snmf/K3/run1/genotypes_I_r1.3.Q",
    "./genotypes_I.snmf/K3/run1/genotypes_I_r1.3.G")

# get the result
res$masked.ce
res$all.ce

#remove project
remove.snmfProject("genotypes_I.snmfProject")
```

---

env                    *Environmental input file format for* lfmm

---

### Description

Description of the env format. The env format can be used as an input format for the environmental variables in the lfmm function.

### Details

The env format has one row for each individual. Each row contains one value for each environmental variable (separated by spaces or tabulations).

Here is an example of an environmental file using the env format with 3 individuals and 2 variables:

```
0.252477 0.95250639
0.216618 0.10902647
-0.47509 0.07626694
```

### Author(s)

Eric Frichot

### See Also

lfmm lfmm2 read.env write.env

G *Ancestral allele frequencies from a snmf run*

**Description**

Return the snmf output matrix of ancestral allele frequency matrix for the chosen run with K ancestral populations. For an example, see snmf.

**Usage**

```
G(object, K, run)
```

**Arguments**

| | |
|---|---|
| object | A snmfProject object. |
| K | The number of ancestral populations. |
| run | A chosen run. |

**Value**

| | |
|---|---|
| res | A matrix containing the ancestral allele frequencies for a run with K ancestral populations. |

**Author(s)**

Eric Frichot

**See Also**

geno snmf Q cross.entropy

**Examples**

```
### Example of analyses using snmf ###

# creation of a genotype file: genotypes.geno.
# The data contain 400 SNPs for 50 individuals.
data("tutorial")
write.geno(tutorial.R, "genotypes.geno")

################
# running snmf #
################


# Two runs for K = 1 to 5
project.snmf = snmf("genotypes.geno",
                K = 3,
```

```
                    repetitions = 2,
                    project = "new")

# get the ancestral genotype frequency matrix, G, for the 2nd run for K = 3.
freq = G(project.snmf, K = 3, run = 2)
```

---

genetic.gap                    *Genetic gap:  genetic offset and genetic distance between environ-*
                               *ments.*

---

### Description

The function returns estimates of the geometric genetic offset (genetic gap) computed from grids
of new and predicted environments. The estimates are based on the covariance matrix of effect
sizes obtained from an lfmm2 model. The function takes as input the data that are used to adjust
the LFMM, a matrix of environmental variables measured at new locations (new.env) or at the
same locations as in the LFMM estimates (new.env = env is accepted), and a matrix of predicted
environmental variables for the new locations (pred.env) in the same format as the new.env ones.

### Usage

```
genetic.gap (input, env, new.env, pred.env, K, scale, candidate.loci)
```

### Arguments

| | |
|---|---|
| input | A genotypic matrix or a character string containing a path to the input file. The genotypic matrix must be in the [lfmm](#) format without missing values (9 or NA). See [impute](#) for completion based on nonnegative matrix factorization. Also consider R packages for reading large matrices. |
| env | A matrix of environmental covariates or a character string containing a path to the environmental file. The environmental matrix must be in the [env](#) format without missing values. The variables must be encoded as numeric and sampled at the same locations as for the genotype matrix. |
| new.env | A matrix of new environmental covariates or a character string containing a path to the new environmental data. The data are environmental covariates sampled at locations that can differ from those used in the estimation of the LFMM (env). By default, the matrix provided as the env argument is used. The new environmental matrix must be in the [env](#) format without missing values. The variables must be encoded as numeric |
| pred.env | A matrix of predicted (new) environmental covariates or a character string containing a path to the predicted environmental data file. The predicted environmental matrix must be in the [env](#) format without missing values, and of same dimension as the new.env matrix. All variables must be encoded as numeric and sampled at the same locations as for the new.env matrix. Predicted environmental covariates typically result from bioclimatic models (eg, worldclim). |

| K | An integer or a sequence of integers corresponding to the number of latent factors in the LFMM. The number of latent factors could be estimated from the elbow point in the PCA screeplot for the genotype matrix. For a sequence of values, an average prediction will be returned. |
|---|---|
| scale | A logical value indicating whether the environmental data are scaled or not. If scale == TRUE, then all environmental matrices are centered and scaled from the columwise mean and standard deviations of the env matrix. This option should be used only to evaluate the relative importance of environmental variables with the eigenvalues of the covariance matrix of effect sizes when the environmental data have different scales. |
| candidate.loci | A vector specifying which loci (column label) in the genotype matrix are included in the computation of the genetic offset. The default value includes all loci. |

## Value

| offset | A vector of genomic offset values computed for every sample location in new.env and pred.env. The genomic offset is the genetic gap defined in (Gain et al. 2023). |
|---|---|
| distance | A vector of environmental distance values computed for every sample location in new.env and pred.env. The distances to an estimate of the risk of nonadaptedness that includes correction for confounding factors and analyzes multiple predictors simultaneously (modified version of RONA). |
| eigenvalues | Eigenvalues of the covariance matrix of LFMM effect sizes. They represent the relative importance of combinations of environmental variables described in vectors when the environmental data have similar scales. To be used with scale == TRUE. |
| vectors | Eigenvectors of the covariance matrix of LFMM effect sizes representing combinations of environmental variablessorted by importance (eigenvalues). |

## Author(s)

Olivier Francois, Clement Gain

## References

Gain, C., et al. (2023). A quantitative theory for genomic offset statistics. bioRxiv, 10.1101/2023.01.02.522469.

Gain C, Francois O. (2021). LEA 3: Factor models in population genetics and ecological genomics with R. Molecular Ecology Resources. Molecular Ecology Resources 21 (8), 2738-2748. doi.org/10.1111/1755-0998.13366.

## See Also

lfmm.data lfmm2

**Examples**

```
### Example of genetic offset computation using lfmm2 ###

data("offset_example")

Y <- offset_example$geno
X <- offset_example$env
X.pred <- offset_example$env.pred

#PCA of the genotype data suggests k = 2 factors
plot(prcomp(Y), col = "blue")

## genetic gap

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

# return the values of the offset (genetic gap) for each sample location
round(g.gap$offset, digit = 3)

# plot the squared root of the genetic gap vs Euclidean environmental distance
Delta = X - X.pred
dist.env = sqrt( rowSums(Delta^2) )
plot(dist.env, sqrt(g.gap$offset),  cex = .6)


# plot RONA vs the genetic gap
plot(g.gap$offset, g.gap$distance,  cex = .6)

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

# Scaling does not change genetic gaps
plot(g.gap$offset, g.gap.scaled$offset,  cex = .6)

# But scaling is useful for evaluating the relative importance of environmental variables
# Only two dimensions of the environmental space influence the genetic gap
barplot(g.gap.scaled$eigenvalues, col = "orange", xlab = "Axes", ylab = "Eigenvalues")

# The loadings for the first two variables indicate their relative contribution to local adaptation
g.gap.scaled$vectors[,1:2]

#rm(list = ls())
```

---

genetic.offset              *Genetic offset and genetic distance between environments.*

---

**Description**

The function returns estimates of the geometric genetic offset computed from grids of new and predicted environments. The function takes as input the data that are used to adjust the LFMM, a matrix of environmental variables measured at new locations (new.env) or at the same locations as in the LFMM estimates (new.env = env is accepted), and a matrix of predicted environmental variables for the new locations (pred.env) in the same format as the new.env ones. It is equivalent to `genetic.gap` function.

**Usage**

```
genetic.offset (input, env, new.env, pred.env, K, scale, candidate.loci)
```

**Arguments**

input           A genotypic matrix or a character string containing a path to the input file. The genotypic matrix must be in the `lfmm` format without missing values (9 or NA). See `impute` for completion based on nonnegative matrix factorization. Also consider R packages for reading large matrices.

env             A matrix of environmental covariates or a character string containing a path to the environmental file. The environmental matrix must be in the `env` format without missing values. The variables must be encoded as `numeric` and sampled at the same locations as for the genotype matrix.

new.env         A matrix of new environmental covariates or a character string containing a path to the new environmental data. The data are environmental covariates sampled at locations that can differ from those used in the estimation of the LFMM (env). By default, the matrix provided as the env argument is used. The new environmental matrix must be in the `env` format without missing values. The variables must be encoded as `numeric`

pred.env        A matrix of predicted (new) environmental covariates or a character string containing a path to the predicted environmental data file. The predicted environmental matrix must be in the `env` format without missing values, and of same dimension as the new.env matrix. All variables must be encoded as `numeric` and sampled at the same locations as for the new.env matrix. Predicted environmental covariates typically result from bioclimatic models (eg, worldclim).

K               An integer or a sequence of integers corresponding to the number of latent factors in the LFMM. The number of latent factors could be estimated from the elbow point in the PCA screeplot for the genotype matrix. For a sequence of values, an average prediction will be returned.

scale           A logical value indicating whether the environmental data are scaled or not. If scale == TRUE, then all environmental matrices are centered and scaled from the columwise mean and standard deviations of the env matrix. This option should

be used only to evaluate the relative importance of environmental variables with the eigenvalues of the covariance matrix of effect sizes when the environmental data have different scales.

candidate.loci   A vector specifying which loci (column label) in the genotype matrix are included in the computation of the genetic offset. The default value includes all loci.

## Value

offset        A vector of genomic offset values computed for every sample location in new.env and pred.env. The genomic offset is the genetic gap defined in (Gain et al. 2023).

distance      A vector of environmental distance values computed for every sample location in new.env and pred.env. The distances to an estimate of the risk of nonadaptedness that includes correction for confounding factors and analyzes multiple predictors simultaneously (modified version of RONA).

eigenvalues   Eigenvalues of the covariance matrix of LFMM effect sizes. They represent the relative importance of combinations of environmental variables described in vectors when the environmental data have similar scales. To be used with scale == TRUE.

vectors       Eigenvectors of the covariance matrix of LFMM effect sizes representing combinations of environmental variablessorted by importance (eigenvalues).

## Author(s)

Olivier Francois, Clement Gain

## References

Gain, C., et al. (2023). A quantitative theory for genomic offset statistics. bioRxiv, 10.1101/2023.01.02.522469.

Gain C, Francois O. (2021). LEA 3: Factor models in population genetics and ecological genomics with R. Molecular Ecology Resources. Molecular Ecology Resources 21 (8), 2738-2748. doi.org/10.1111/1755-0998.13366.

## See Also

[lfmm.data](lfmm.data) [lfmm2](lfmm2)

## Examples

```
### Example of genetic offset computation using lfmm2 ###

data("offset_example")

Y <- offset_example$geno
X <- offset_example$env
X.pred <- offset_example$env.pred

#PCA of the genotype data suggests k = 2 factors
```

```
plot(prcomp(Y), col = "blue")

## genetic offset

g.gap <- genetic.offset(input = Y,
                        env = X,
                        pred.env = X.pred,
                        K = 2)

# return the values of the offset (genetic gap) for each sample location
round(g.gap$offset, digit = 3)

# plot the squared root of the genetic gap vs Euclidean environmental distance
Delta = X - X.pred
dist.env = sqrt( rowSums(Delta^2) )
plot(dist.env, sqrt(g.gap$offset),  cex = .6)


# plot RONA vs the genetic gap
plot(g.gap$offset, g.gap$distance,  cex = .6)

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

# Scaling does not change genetic offsets
plot(g.gap$offset, g.gap.scaled$offset,  cex = .6)

# But scaling is useful for evaluating the relative importance of environmental variables
# Two dimensions in environmental space have influence on the genetic offset

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

# The loadings for the first two variables indicate their relative contribution to local adaptation
g.gap.scaled$vectors[,1:2]

#rm(list = ls())
```

---

geno                     *Input file for* snmf

---

## Description

Description of the geno format. The geno format can be used as an input format for genotypic
matrices in the functions snmf, lfmm, and pca.

## Details

The geno format has one row for each SNP. Each row contains 1 character for each 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.

Here is an example of a genotypic matrix using the geno format with 3 individuals and 4 loci:

```
112
010
091
121
```

## Author(s)

Eric Frichot

## See Also

geno2lfmm lfmm2geno ancestrymap2geno ped2geno vcf2geno read.geno write.geno

---

geno2lfmm                    *Convert from* geno *to* lfmm *format*

---

## Description

A function that converts from the geno format to the lfmm format.

## Usage

```
geno2lfmm(input.file, output.file = NULL, force = TRUE)
```

## Arguments

input.file       A character string containing a path to the input file, a genotypic matrix in the geno format.

output.file      A character string containing a path to the output file, a genotypic matrix in the lfmm format. By default, the name of the output file is the same name as the input file with a .lfmm extension.

force            A boolean option. If FALSE, the input file is converted only if the output file does not exist. If TRUE, convert the file anyway.

## Value

output.file      A character string containing a path to the output file, a genotypic matrix in the lfmm format.

**Author(s)**

Eric Frichot

**See Also**

[lfmm.data](#) [geno](#) [ancestrymap2lfmm](#) [ancestrymap2geno](#) [ped2lfmm](#) [ped2geno](#) [vcf2geno](#) [lfmm2geno](#)
[read.geno](#) [write.geno](#)

**Examples**

```
# Creation of a file called "genotypes.geno" in the working directory
# with 400 SNPs for 50 individuals.
data("tutorial")
write.geno(tutorial.R, "genotypes.geno")

# Conversion    from the geno format ("genotypes.geno")
#               to the lfmm format ("genotypes.lfmm").
# By default,   the name of the output file is the same name
#               as the input file with a .lfmm extension.
# Create file:  "genotypes.lfmm".
output = geno2lfmm("genotypes.geno")

# Conversion    from the geno format ("genotypes.geno")
#               to the lfmm format with the output file called "plop.lfmm".
# Create file:  "plop.lfmm".
output = geno2lfmm("genotypes.geno", "plop.lfmm")

# As force = false and the file "genotypes.lfmm" already exists,
# nothing happens.
output = geno2lfmm("genotypes.geno", force = FALSE)
```

---

impute                          *Impute missing genotypes using an snmf object*

---

**Description**

Impute missing genotypes in a genotype file (.lfmm) by using ancestry and genotype frequency
estimates from an snmf run. The function generates a new lfmm file. See [lfmm](#) and [lfmm2](#).

**Usage**

```
impute (object, input.file, method, K, run)
```

**Arguments**

| | |
|---|---|
| object | An snmfProject object. |
| input.file | A path (character string) to an input file in lfmm format with missing genotypes. The same input data must be used when generating the snmf object. |

| method | A character string: "random" or "mode". With "random", imputation is performed by using the genotype probabilities. With "mode", the most likely genotype is used for matrix completion. |
| --- | --- |
| K | An integer value. The number of ancestral populations. |
| run | An integer value. A particular run used for imputation (usually the run number that minimizes the cross entropy criterion). |

## Value

| NULL | The function writes the imputed genotypes in an output file having the "_imputed.lfmm" suffix. |
| --- | --- |

## Author(s)

Olivier Francois

## References

Gain C, Francois O. (2021). LEA 3: Factor models in population genetics and ecological genomics with R. Molecular Ecology Resources, doi.org/10.1111/1755-0998.13366.

## See Also

snmf lfmm lfmm2

## Examples

```
### Example of analysis ###

data("tutorial")
# creation of a genotype file with missing genotypes
# The data contain 400 SNPs for 50 individuals.

dat = as.numeric(tutorial.R)
dat[sample(1:length(dat), 100)] <-  9
dat <- matrix(dat, nrow = 50, ncol = 400 )
write.lfmm(dat, "genotypes.lfmm")

################
# running snmf #
################

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

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

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

```
# Compare with truth
# Proportion of correct imputation results:
mean( tutorial.R[dat == 9] == read.lfmm("genotypes.lfmm_imputed.lfmm")[dat == 9] )
```

---

lfmm                    *Fitting Latent Factor Mixed Models (MCMC algorithm)*

---

### Description

Latent Factor Mixed Models (LFMMs) are factor regression models in which the response vari-
able is a genotypic matrix, and the explanatory variables are environmental measures of ecological
interest or trait values. The [lfmm](#) function estimates latent factors and effect sizes based on an
MCMC algorithm. The resulting object can be used in the function [lfmm.pvalues](#) to identify
genetic polymorphisms exhibiting association with ecological gradients or phenotypes, while cor-
recting for unobserved confounders. An exact and computationally efficient least-squares method
is implemented in the function [lfmm2](#) which should be the prefered option.

### Usage

```
lfmm(input.file, environment.file, K,
    project = "continue",
    d = 0, all = FALSE,
    missing.data = FALSE, CPU = 1,
    iterations = 10000, burnin = 5000,
    seed = -1, repetitions = 1,
    epsilon.noise = 1e-3, epsilon.b = 1000,
    random.init = TRUE)
```

### Arguments

| | |
|---|---|
| input.file | A character string containing a path to the input file, a genotypic matrix in the [lfmm](#){lfmm_format} format. The matrix must not contain missing values. See [impute](#) for completion based on nonnegative matrix factorization. |
| environment.file | |
| | A character string containing a path to the environmental file, an environmental data matrix in the [env](#) format. |
| K | An integer corresponding to the number of latent factors. |
| project | A character string among "continue", "new", and "force". If "continue", the results are stored in the current project. If "new", the current project is removed and a new project is created. If "force", the results are stored in the current project even if the input file has been modified since the creation of the project. |
| d | An integer corresponding to the fit of an lfmm model with the d-th variable only from environment.file. By default (if NULL and all are FALSE), lfmm fits each variable from environment.file sequentially and independently. |

| | |
|---|---|
| all | A Boolean option. If TRUE, lfmm fits all variables from the environment.file at the same time. This option is not compatible with the d option. |
| missing.data | A Boolean option. If TRUE, the input.file contains missing genotypes. Caution: lfmm requires imputed genotype matrices. See [impute](). |
| CPU | A number of CPUs to run the parallel version of the algorithm. By default, the number of CPUs is 1. |
| iterations | The total number of cycles for the Gibbs Sampling algorithm. |
| burnin | The burnin number of cycles for the Gibbs Sampling algorithm. |
| seed | A seed to initialize the random number generator. By default, the seed is randomly chosen. The seed is initialized in each run of the program. For modifying the default setting, provide one seed per run. |
| repetitions | A number of replicate runs for the Gibbs Sampler algorithm. |
| epsilon.noise | A prior parameter for variances. |
| epsilon.b | A prior parameter for the variance of correlation coefficients. |
| random.init | A Boolean option. If TRUE, the Gibbs Sampler is initiliazed randomly. Otherwise, it is initialized with zero values. |

## Value

lfmm returns an object of class lfmmProject.

The following methods can be applied to an object of class lfmmProject:

| | |
|---|---|
| show | Display information about all analyses. |
| summary | Summarize analyses. |
| \link{z.scores} | |
| | Return the lfmm output vector of z.scores for some runs. |
| \link{lfmm.pvalues} | |
| | Return the vector of adjusted p-values for a combination of runs with K latent factors, and for the d-th predictor. |
| load.lfmmProject (file = "character") | |
| | Load the file containing an lfmmProject objet and show the object. |
| remove.lfmmProject (file = "character") | |
| | Erase a lfmmProject object. Caution: All the files associated with the object will be removed. |
| export.lfmmProject(file.lfmmProject) | |
| | Create a zip file containing the full lfmmProject object. It allows users to move the project to a new directory or a new computer (using import). If you want to overwrite an existing export, use the option force == TRUE. |
| import.lfmmProject(file.lfmmProject) | |
| | Import and load an lfmmProject object from a zip file (made with the export function) into the chosen directory. If you want to overwrite an existing project, use the option force == TRUE. |
| combine.lfmmProject(file.lfmmProject, toCombine.lfmmProject) | |
| | Combine to.Combine.lfmmProject into file.lfmmProject. Caution: Only projects with runs coming from the same input file can be combined. If the same input file has different names in the two projects, use the option force == TRUE. |

**Author(s)**

Eric Frichot Olivier Francois

**References**

Frichot E, Schoville SD, Bouchard G, Francois O. (2013). *Testing for associations between loci and environmental gradients using latent factor mixed models.* Molecular biology and evolution, 30(7), 1687-1699.

**See Also**

lfmm.data z.scores lfmm.pvalues pca lfmm tutorial

**Examples**

```
### Example of analysis using lfmm ###

data("tutorial")
# creation of a genotype file: genotypes.lfmm.
# The file contains 400 SNPs for 50 individuals.
write.lfmm(tutorial.R, "genotypes.lfmm")

# Creation of a phenotype/environment file: gradient.env.
# One environmental predictor for 40 individuals.
write.env(tutorial.C, "gradients.env")

################
# running lfmm #
################

# main options, K: (the number of latent factors),
#            CPU: the number of CPUs.

# Runs with K = 6 and 5 repetitions.
# runs with 6000 iterations
# including 3000 iterations for burnin.
# Around 30 seconds per run.
project = lfmm( "genotypes.lfmm",
                "gradients.env",
                 K = 6,
                 repetitions = 5,
                 project = "new")

# get adjusted p-values using all runs
pv = lfmm.pvalues(project, K = 6)

# Evaluate FDR and POWER (TPR)
for (alpha in c(.05,.1,.15,.2)) {
    # expected FDR
    print(paste("expected FDR:", alpha))
    L = length(pv$pvalues)
    # Benjamini-Hochberg's mehod for an expected FDR = alpha.
```

```
    w = which(sort(pv$pvalues) < alpha * (1:L)/L)
    candidates = order(pv$pvalues)[w]

    # estimated FDR and True Positive Rate
    # The targets SNPs are loci 351 to 400
    Lc = length(candidates)
    estimated.FDR = length(which(candidates <= 350))/Lc
    estimated.TPR = length(which(candidates > 350))/50
    print(paste("FDR:", estimated.FDR, "True Positive Rate:", estimated.TPR))
}


# remove project
remove.lfmmProject("genotypes_gradients.lfmmProject")
```

---

lfmm.data                        *Input file for* lfmm

---

### Description

Description of the lfmm format. The lfmm format can be used as an input format for genotypic matrices in the functions snmf, lfmm, lfmm2, and pca.

### Details

The lfmm format has one row for each individual. Each row contains one value at each loci (separated by spaces or tabulations) corresponding to the number of alleles. The number of alleles corresponds to the number of reference alleles or the number of derived alleles. Missing genotypes are encoded by the value -9 or the value 9.

For the use of functions lfmm and lfmm2 missing genotypes must be removed or imputed with the function impute.

Here is an example of a genotypic matrix using the lfmm format with 3 individuals and 4 loci:

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

### Author(s)

Eric Frichot

### See Also

lfmm lfmm2 geno2lfmm lfmm2geno ancestrymap2lfmm ped2lfmm read.lfmm write.lfmm

lfmm.pvalues                    *P-values from lfmm runs*

### Description

Returns a vector of p-values computed from a combination of lfmm runs. For an example, see lfmm.

### Usage

```
lfmm.pvalues (object, genomic.control, lambda, K, d, all, run)
```

### Arguments

object              An lfmmProject object.

genomic.control

                    A Boolean value. If TRUE, the p-values are automatically calibrated using ge-
                    nomic control. If FALSE, the p-values are calculated by rescaling the chi-squared
                    test statistics using the lambda parameter.

lambda              A numeric value. The lambda value is used as inflation factor to rescale the
                    chi-squared statistics in the computation of p-values. This option requires that
                    genomic.control = FALSE. The default value of lambda is equal to 1.0 (no
                    rescaling).

K                   An integer value. The number of latent factors used in the model.

d                   An integer value. Computes the p-values for the d-th covariable in the model.

all                 A Boolean value. Each variable is considered separately (Obscure parameter).

run                 An integer vector representing a list of runs to be combined in the computation
                    of p-values (by default, all runs).

### Value

pvalues             A vector of combined p-values for each locus.

GIF                 The inflation factor value used for correcting the test statistics.

### Author(s)

Eric Frichot Olivier Francois

### See Also

lfmm.data lfmm

## Examples

```
### Example of analyses using lfmm ###

data("tutorial")
# creation of a genotype file, "genotypes.lfmm".
# The data contain 400 SNPs for 50 individuals.
write.lfmm(tutorial.R, "genotypes.lfmm")
# creation of an environmental variable file, "gradient.env".
# The data contain one environmental variable measured for 50 individuals.
write.env(tutorial.C, "gradients.env")


################
# lfmm runs    #
################

# main options, K: (the number of latent factors),
#          CPU: the number of CPUs.

# runs with K = 3 and 2 repetitions.
# around 15 seconds per run.
project = NULL
project = lfmm("genotypes.lfmm", "gradients.env", K = 3, repetitions = 2,
    iterations = 6000, burnin = 3000, project = "new")

# get adjusted p-values using the genomic control method
p = lfmm.pvalues(project, K = 3)

hist(p$pvalues, col = "yellow3")

# get adjusted p-values using lambda = 0.6
p = lfmm.pvalues(project, genomic.control = FALSE,
    lambda = 0.6, K = 3)

hist(p$pvalues, col = "yellow3")
```

---

lfmm2                          *Fitting Latent Factor Mixed Models (Least squares algorithm)*

---

## Description

Latent Factor Mixed Models (LFMMs) are factor regression models in which the response variable
is a genotypic matrix, and the explanatory variables are environmental measures of ecological in-
terest or trait values. The lfmm2 function estimates latent factors based on an exact least-squares
approach. The resulting object can be used by the function lfmm2.test to identify genetic poly-
morphisms exhibiting association with ecological gradients or phenotypes, while correcting for
unobserved confounders. An MCMC estimation algorithm is implemented in the function lfmm,
but this version should be prefered.

## Usage

```
lfmm2 (input, env, K, lambda, effect.sizes)
```

## Arguments

input         A genotypic matrix or a character string containing a path to the input file. The
              genotypic matrix must be in the [lfmm](){lfmm_format} format without missing
              values (9 or NA). See [impute]() for completion based on nonnegative matrix fac-
              torization and consider R packages for reading large matrices.

env           A matrix of environmental covariates or a character string containing a path
              to the environmental file. The environment matrix must be in the [env]() format
              without missing values. Response variables must be encoded as numeric.

K             An integer corresponding to the number of latent factors. The number of latent
              factors could be estimated from the elbow point in the PCA screeplot for the
              genotype matrix.

lambda        A positive numeric value for a ridge regularization parameter. The default value
              is set to 1e-5.

effect.sizes  A logical value that indicates whether the matrix of effect sizes should be re-
              turned or not. The default value is set to FALSE for saving memory space.

## Value

lfmm2 returns an object of class lfmm2Class that contains $K$ estimated latent factors @U and their
loadings @V.

The following method can be applied to an object of class lfmm2Class:

\link{lfmm2.test}
                P-values adjusted for the $K$ latent factors computed by lfmm2.

## Author(s)

Olivier Francois

## References

Caye K, Jumentier B, Lepeule J, Francois O. (2019). LFMM 2: fast and accurate inference of
gene-environment associations in genome-wide studies. Molecular biology and evolution, 36(4),
852-860.

Gain C, Francois O. (2021). LEA 3: Factor models in population genetics and ecological genomics
with R. Molecular Ecology Resources. doi: 10.1111/1755-0998.13366

## See Also

[lfmm.data]() [impute]() [lfmm2.test]() [pca]() [lfmm]() [tutorial]()

**Examples**

```
### Example of analysis using lfmm2 ###


# Simulation with 10 target loci, with effect sizes ranging between -10 an 10
# n = 100 individuals and L = 1000 loci

X <- as.matrix(rnorm(100)) # causal environmental variable
B <- rep(0, 1000)
target <- sample(1:1000, 10) # target loci
B[target] <- runif(10, -10, +10) # effect sizes


# Creating hidden factors and loadings

U <- t(tcrossprod(as.matrix(c(-1,0.5,1.5)), X))+ matrix(rnorm(300), ncol = 3)
V <- matrix(rnorm(3000), ncol = 3)


# Simulating a binarized matrix containing haploid genotypes
# Simulation performed with the generative LFMM

Y <- tcrossprod(as.matrix(X), B) + tcrossprod(U, V) + matrix(rnorm(100000, sd = .5), nrow = 100)
Y <- matrix(as.numeric(Y > 0), ncol = 1000)

#######################################
# Fitting an LFMM with K = 3 factors #
#######################################

mod2 <- lfmm2(input = Y, env = X, K = 3)

# Computing P-values and plotting their minus log10 values
# Target loci are highlighted

pv <- lfmm2.test(object = mod2, input = Y, env = X, linear = TRUE)
plot(-log10(pv$pvalues), col = "grey", cex = .4, pch = 19)
points(target, -log10(pv$pvalues[target]), col = "red")

#rm(list = ls())
```

---

lfmm2.test                    *P-values adjusted for latent factors computed by* lfmm2.

---

**Description**

The function returns a vector of p-values for association between loci and environmental variables adjusted for latent factors computed by lfmm2. As input, it takes an object of class lfmm2Class with the data that were used to adjust the LFMM. If full is set to FALSE, the function computes significance values (p-values) for each environmental variable, otherwise it returns p-values for the full set of environmental variables.

## Usage

```
lfmm2.test (object, input, env, full, genomic.control, linear, family)
```

## Arguments

| | |
|---|---|
| object | An object of class `lfmm2Class`. |
| input | A genotypic matrix or a character string containing a path to the input file. The genotypic matrix must be in the [lfmm](){lfmm_format} format without missing values (9 or NA). See [impute]() for completion based on nonnegative matrix factorization and consider R packages for reading large matrices. |
| env | A matrix of environmental covariates or a character string containing a path to the environmental file. The environment matrix must be in the [env]() format without missing values. Variables must be encoded as `numeric`. |
| full | A logical value. If TRUE, p-values are computed for the full set of environmental variables (a single value at each locus). If FALSE, p-values are computed for each environmental variable (as many values as environmental variable at each locus). |
| genomic.control | |
| | A logical value. If TRUE, the p-values are recalibrated by using genomic control after correction for confounding. |
| linear | A logical value indicating whether linear or generalized linear models should be used to perform the association tests. If FALSE, `family` should be provided in the next argument. |
| family | a `family` for generalized linear models used in the association tests. The default is binomial(link = "logit")), which requires that y is between 0 and 1. |

## Value

| | |
|---|---|
| pvalues | If `full` is set to FALSE, a matrix of p-values for all loci and for each environmental variable. Otherwise a vector of p-values for all loci (all environmental variables are included in the model). |
| zscores | If `full` is set to FALSE, a matrix of z-scores for each locus and each environmental variable. |
| fscores | If `full` is set to TRUE, a vector of f-scores for each locus. |
| adj.r.squared | If `full` is set to TRUE, a vector of R squared values or variances explained by all environmental variables for all loci. The values are uncalibrated. |
| gif | If `full` is set to FALSE, a vector of genomic inflation factors computed for each environmental variable. A single genomic inflation factor otherwise. |

## Author(s)

Olivier Francois

## References

Caye K, Jumentier B, Lepeule J, Francois O. (2019). LFMM 2: fast and accurate inference of gene-environment associations in genome-wide studies. Molecular biology and evolution, 36(4), 852-860.

## See Also

`lfmm.data` `lfmm2`

## Examples

```
### Example of analysis using lfmm2 ###

# Simulation with 10 target loci, with effect sizes ranging between -10 an 10
# n = 100 individuals and L = 1000 loci

X <- as.matrix(rnorm(100)) # environmental variable
B <- rep(0, 1000)
target <- sample(1:1000, 10) # target loci
B[target] <- runif(10, -10, +10) # effect sizes

# Creating hidden factors and loadings

U <- t(tcrossprod(as.matrix(c(-1,0.5,1.5)), X)) + matrix(rnorm(300), ncol = 3)
V <- matrix(rnorm(3000), ncol = 3)

# Simulating a binarized matrix containing haploid genotypes
# Simulation performed with the generative LFMM

Y <- tcrossprod(as.matrix(X), B) + tcrossprod(U, V) + matrix(rnorm(100000, sd = .5), nrow = 100)
Y <- matrix(as.numeric(Y > 0), ncol = 1000)

######################################
# Fitting an LFMM with K = 3 factors #
######################################

mod2 <- lfmm2(input = Y, env = X, K = 3)

# Computing P-values and plotting their minus log10 values
# Target loci are highlighted

pv <- lfmm2.test(object = mod2, input = Y, env = X, linear = TRUE)
plot(-log10(pv$pvalues), col = "grey", cex = .4, pch = 19)
points(target, -log10(pv$pvalues[target]), col = "red")
```

---

lfmm2geno                     *Convert from* `lfmm` *to* `geno` *format*

---

**Description**

A function that converts from the lfmm format to the geno format.

**Usage**

```
lfmm2geno(input.file, output.file = NULL, force = TRUE)
```

**Arguments**

| | |
|---|---|
| input.file | A character string containing a path to the input file, a genotypic matrix in the lfmm format. |
| output.file | A character string containing a path to the output file, a genotypic matrix in the geno format. By default, the name of the output file is the same name of the input file with a .geno extension. |
| force | A boolean option. If FALSE, the input file is converted only if the output file does not exist. If TRUE, convert the file anyway. |

**Value**

| | |
|---|---|
| output.file | A character string containing a path to the output file, a genotypic matrix in the geno format. |

**Author(s)**

Eric Frichot

**See Also**

lfmm.data geno ancestrymap2lfmm ancestrymap2geno geno2lfmm ped2lfmm ped2geno vcf2geno

**Examples**

```
# Creation of a file called "genotypes.lfmm" in the working directory,
# with 400 SNPs for 50 individuals.
data("tutorial")
write.lfmm(tutorial.R, "genotypes.lfmm")

# Conversion    from the lfmm format ("genotypes.lfmm")
#               to the geno format ("genotypes.geno").
# By default,   the name of the output file is the same name
#               as the input file with a .geno extension.
# Create file:  "genotypes.geno".
output = lfmm2geno("genotypes.lfmm")

# Conversion    from the lfmm format ("genotypes.lfmm")
#               to the geno format with the output file called "plop.geno".
# Create file:  "plop.geno".
output = lfmm2geno("genotypes.lfmm", "plop.geno")

# As force = false and the file "genotypes.geno" already exists,
```

```
# nothing happens.
output = lfmm2geno("genotypes.lfmm", force = FALSE)
```

---

offset_example          *Example data for genetic offset analysis*

---

### Description

The data set is composed of a genotypic matrix stored in a lfmm format (geno) containing 200 individuals genotyped at 510 SNPs, a matrix with 4 correlated environmental variables measured for each individual in the env format, and a matrix with the same 4 variables after environmental change (env.pred).

### Value

| | |
|---|---|
| geno | A genotypic matrix that contains haploid genotypes for 200 individuals at 510 SNPs (lfmm format). |
| env | A matrix with 4 correlated environmental variables measured for 200 genotyped individuals. |
| env.pred | A matrix with the same 4 variables predicted for the 200 individuals after environmental change. |

---

pca          *Principal Component Analysis*

---

### Description

The pca function performs a principal component analysis of a genotypic matrix encoded in one of the following formats: [lfmm](), [geno](), [ancestrymap](), [ped]() or [vcf](). The pca function computes eigenvalues, eigenvectors, and standard deviations for all principal components and the projections of individuals on each component. Thepca function returns an object of class "pcaProject" containing the output data and the input parameters.

### Usage

```
pca (input.file, K, center = TRUE, scale = FALSE)
```

### Arguments

| | |
|---|---|
| input.file | A character string containing the path to the genotype input file, a genotypic matrix in the [lfmm]() format. |
| K | An integer corresponding to the number of principal components calculated. By default, all principal components are calculated. |
| center | A boolean option. If TRUE, the data matrix is centered (default: TRUE). |
| scale | A boolean option. If TRUE, the data matrix is centered and scaled (default: FALSE). |

## Value

pca returns an object of class `pcaProject` containing the following components:

eigenvalues    The vector of eigenvalues.

eigenvectors   The matrix of eigenvectors (one column for each eigenvector).

sdev           The vector of standard deviations.

projections    The matrix of projections (one column for each projection).

The following methods can be applied to the object of class pcaProject returned by `pca`:

plot           Plot the eigenvalues.

show           Display information on analysis.

summary        Summarize analysis.

tracy.widom    Perform Tracy-Widom tests for eigenvalues.

load.pcaProject(file.pcaProject)
               Load the file containing a pcaProject object and return the pcaProject object.

remove.pcaProject(file.pcaProject)
               Erase a `pcaProject` object. Caution: All the files associated with the `pcaProject` object will be removed except the genotype file.

export.pcaProject(file.pcaProject)
               Create a zip file containing the full `pcaProject` object. It allows users to move the project to a new directory or a new computer (using import). If you want to overwrite an existing export, use the option `force == TRUE`.

import.pcaProject(file.pcaProject)
               Import and load an `pcaProject` object from a zip file (made with the export function) into the chosen directory. If you want to overwrite an existing project, use the option `force == TRUE`.

## Author(s)

Eric Frichot

## See Also

[lfmm.data](#) [snmf](#) [lfmm](#) [lfmm2](#) [tutorial](#)

## Examples

```
# Create a genotype file "genotypes.lfmm"
# with 1000 SNPs for 165 individuals.
data("tutorial")
write.lfmm(tutorial.R,"genotypes.lfmm")


#################
# Perform PCA   #
#################


# run PCA
```

```
# Available options: K (the number of PCs),
#                     center and scale.
# Creation of  genotypes.pcaProject - the pcaProject object.
#              a directory genotypes.pca containing:
#  genotypes.eigenvalues - eigenvalues,
#  genotypes.eigenvectors - eigenvectors,
#  genotypes.sdev - standard deviations,
#  genotypes.projections - projections,

# Create a pcaProject object: pc.
pc <- pca("genotypes.lfmm", scale = TRUE)


#######################
# Display information #
#######################

# Display information on analysis.
show(pc)

# Summarize analysis.
summary(pc)


#####################
# Graphical outputs #
#####################

par(mfrow=c(2,2))

# Plot eigenvalues.
plot(pc, lwd=5, col="blue", cex = .7, xlab=("Factors"), ylab="Eigenvalues")

# PC1-PC2 plot.
plot(pc$projections)
# PC3-PC4 plot.
plot(pc$projections[,3:4])

# Plot standard deviations.
plot(pc$sdev)


#############################
# Perform Tracy-Widom tests #
#############################

# Perfom Tracy-Widom tests for all eigenvalues.
# Create file: genotypes.tracyWidom - tracy-widom test information,
#          in the directory genotypes.pca/.
tw <- tracy.widom(pc)

# Plot the percentage of variance explained by each component.
plot(tw$percentage)

# Show p-values for the Tracy-Widom tests.
tw$pvalues
```

```
###########################
# Manage a pca project    #
###########################

# All the project files for a given input matrix are
# automatically saved into a pca project directory.
# The name of the pcaProject file is the same name as
# the name of the input file with a .pcaProject extension
# ("genotypes.pcaProject").
# The name of the pcaProject directory is the same name as
# the name of the input file with .pca extension ("genotypes.pca/")
# There is only one pca Project for each input file including all the runs.

# An pcaProject can be load in a different session.
project = load.pcaProject("genotypes.pcaProject")

# An pcaProject can be exported to be imported in another directory
# or in another computer
export.pcaProject("genotypes.pcaProject")

dir.create("test", showWarnings = TRUE)
#import
newProject = import.pcaProject("genotypes_pcaProject.zip", "test")
# remove
remove.pcaProject("test/genotypes.pcaProject")


# A pcaProject can be erased.
# Caution: All the files associated with the project will be removed.
remove.pcaProject("genotypes.pcaProject")
```

---

ped                              ped *format description*

---

### Description

Description of the ped format. The ped format can be used as an input format for genotypic matrices in the functions snmf, lfmm, and pca.

### Details

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.

Here is an example with 3 individuals and 4 SNPs:

```
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
```

## Author(s)

Eric Frichot

## See Also

ped2lfmm ped2geno geno lfmm.data ancestrymap vcf

---

ped2geno *Convert from ped to geno format*

---

## Description

A function that converts from the ped format to the geno format.

## Usage

```
ped2geno(input.file, output.file = NULL, force = TRUE)
```

## Arguments

| | |
|---|---|
| input.file | A character string containing a path to the input file, a genotypic matrix in the ped format. |
| output.file | A character string containing a path to the output file, a genotypic matrix in the geno format. By default, the name of the output file is the same name as the input file with a .geno extension. |
| force | A boolean option. If FALSE, the input file is converted only if the output file does not exist. If TRUE, convert the file anyway. |

## Value

| | |
|---|---|
| output.file | A character string containing a path to the output file, a genotypic matrix in the geno format. |

## Author(s)

Eric Frichot

## See Also

ped geno ancestrymap2lfmm ancestrymap2geno geno2lfmm ped2lfmm vcf2geno lfmm2geno

## Examples

```
# Creation of a file called "example.ped"
# with 4 SNPs for 3 individuals.
data("example_ped")
write.table(example_ped,"example.ped",
    col.names = FALSE, row.names = FALSE, quote = FALSE)

# Conversion    from the ped format ("example.ped")
#               to the geno format ("example.geno").
# By default,   the name of the output file is the same name
#               as the input file with a .geno extension.
# Create file:  "example.geno".
output = ped2geno("example.ped")

# Conversion    from the ped format ("example.ped")
#               to the geno format with the output file called "plop.geno".
# Create file:  "plop.geno".
output = ped2geno("example.ped", "plop.geno")

# As force = false and the file "example.geno" already exists,
# nothing happens.
output = ped2geno("example.ped", force = FALSE)
```

---

ped2lfmm                        *Convert from* ped *to* lfmm *format*

---

## Description

A function that converts from the ped format to the lfmm format.

## Usage

```
ped2lfmm(input.file, output.file = NULL, force = TRUE)
```

## Arguments

input.file      A character string containing a path to the input file, a genotypic matrix in the
                ped format.

output.file     A character string containing a path for the output file, a genotypic matricx in
                the lfmm format. By default, the name of the output file is the same name as the
                input file with a .lfmm extension.

force           A boolean option. If FALSE, the input file is converted only if the output file
                does not exist. If TRUE, convert the file anyway.

## Value

output.file     A character string containing a path for the output file, a genotypic matricx in
                the lfmm format.

**Author(s)**

Eric Frichot

**See Also**

[ped](#) [lfmm.data](#) [ancestrymap2lfmm](#) [ancestrymap2geno](#) [geno2lfmm](#) [ped2geno](#) [vcf2geno](#) [lfmm2geno](#)

**Examples**

```
# Creation of a file called "example.ped"
# with 4 SNPs for 3 individuals.
data("example_ped")
write.table(example_ped,"example.ped",
    col.names = FALSE, row.names = FALSE, quote = FALSE)

# Conversion    from the ped format ("example.ped")
#               to the lfmm format ("example.lfmm").
# By default,   the name of the output file is the same name
#               as the input file with a .lfmm extension.
# Create file:  "example.lfmm".
output = ped2lfmm("example.ped")

# Conversion    from the ped format ("example.ped")
#               to the geno format with the output file called "plop.lfmm".
# Create file:  "plop.lfmm".
output = ped2lfmm("example.ped", "plop.lfmm")

# As force = false and the file "example.lfmm" already exists,
# nothing happens.
output = ped2lfmm("example.ped", force = FALSE)
```

---

Q                          *Admixture coefficients from a snmf run*

---

**Description**

Return the snmf output matrix of admixture coefficients for the chosen run with K ancestral populations. For an example, see [snmf](#).

**Usage**

```
Q(object, K, run)
```

**Arguments**

| | |
|---|---|
| object | A snmfProject object. |
| K | The number of ancestral populations. |
| run | A chosen run. |

**Value**

res          A matrix containing the admixture coefficients for the chosen run with K ances-
             tral populations.

**Author(s)**

Eric Frichot

**See Also**

geno snmf G cross.entropy

**Examples**

```
### Example of analysis using snmf ###

# Creation of the genotype file: genotypes.geno.
# The data contain 400 SNPs for 50 individuals.
data("tutorial")
write.geno(tutorial.R, "genotypes.geno")

################
# running snmf #
###############

project.snmf <- snmf("genotypes.geno",
                K = 3,
                repetitions = 2,
                project = "new")

# get the ancestry coefficients for the 2nd run for K = 3.
Q.3 <- Q( project.snmf, K = 3, run = 2)

# cluster assignment for each individual
cluster <- apply( Q.3, 1, which.max)
table(cluster)
```

---

read.env                     *Read environmental file in the* env*format*

---

**Description**

Read a file in the env format.

**Usage**

```
read.env(input.file)
```

## Arguments

input.file    A character string containing a path to the input file, an environmental data matrix in the env format.

## Value

R    A matrix containing the environmental variables with one line for each individual and one column for each environmental variable.

## Author(s)

Eric Frichot

## See Also

env write.env lfmm

## Examples

```
# Creation of an environmental matrix, C
# containing 2 environmental variables for 3 individuals.
# C contains one line for each individual and one column for each variable.
C = matrix(runif(6), ncol=2, nrow=3)

# Write C in a file called "example.env".
# Create file:    "example.env".
write.env(C,"example.env")

# Read the file "example.env".
C = read.env("example.env")
```

---

read.geno    *read a file in the* geno *format*

---

## Description

Read a file in the geno format.

## Usage

```
read.geno(input.file)
```

## Arguments

input.file    A character string containing a path to the input file, a genotypic matrix in the geno format.

## Value

R               A matrix containing the genotypes with one line for each individual and one
                column for each SNP.

## Author(s)

Eric Frichot

## See Also

[write.geno](#) [geno](#) [snmf](#) [geno2lfmm](#) [lfmm2geno](#) [ancestrymap2geno](#) [ped2geno](#) [vcf2geno](#)

## Examples

```
# tutorial contains a matrix of genotypes R with 1000 SNPs for 165 individuals.
# and a matrix with an environmental variable C.
data("tutorial")

# Write R in a file called "genotypes.geno".
# Create file:    "genotypes.geno".
write.geno(tutorial.R,"genotypes.geno")

# Read the file "genotypes.geno".
R = read.geno("genotypes.geno")
```

---

read.lfmm                    *Read files in the* lfmm *format*

---

## Description

Read a file in the lfmm format.

## Usage

```
read.lfmm(input.file)
```

## Arguments

input.file      A character string containing a path to the input file, a genotypic matrix in the
                lfmm format.

## Value

R               A matrix containing the genotypes with one line per individual and one column
                per SNP.

## Author(s)

Eric Frichot

### See Also

write.lfmm lfmm.data lfmm geno2lfmm lfmm2geno ancestrymap2lfmm ped2lfmm

### Examples

```
# tutorial contains a matrix of genotypes R with 1000 SNPs for 165 individuals.
# and a matrix with an environmental variable C.
data("tutorial")

# write R in a file called "genotypes.lfmm"
# Create file:    "genotypes.lfmm".
write.lfmm(tutorial.R,"genotypes.lfmm")

# read the file "genotypes.lfmm".
R = read.lfmm("genotypes.lfmm")
```

---

read.zscore                 *Read the output files of* lfmm

---

### Description

Read the output file from lfmm. This is an internal function. Zscores of a run can be accessed using the function z.scores.

### Usage

```
read.zscore(input.file)
```

### Arguments

input.file      a character string containing a path to the output of lfmm.

### Value

R               A matrix containing the lfmm results with one line per SNP. The first column is the zscore. The second column is the -log10(p-value). The third column is the p-value.

### Author(s)

Eric Frichot

### See Also

zscore.format lfmm

## Examples

```
### Example of analyses using lfmm ###

data("tutorial")
# creation of the genotype file, genotypes.lfmm.
# It contains 400 SNPs for 50 individuals.
write.lfmm(tutorial.R, "genotypes.lfmm")
# creation of the environment file, gradient.env.
# It contains 1 environmental variable for 40 individuals.
write.env(tutorial.C, "gradients.env")

################
# runs of lfmm #
################

# main options, K: (the number of latent factors),
#             CPU: the number of CPUs.

# Toy runs with K = 3 and 2 repetitions.
# around 15 seconds per run.
project = NULL
project = lfmm("genotypes.lfmm", "gradients.env", K = 3,
        iterations = 6000, burnin = 3000, project = "new")

res = read.zscore("./genotypes_gradients.lfmm/K3/run1/genotypes_r1_s1.3.zscore")
```

---

| snmf | *Estimates individual ancestry coefficients and ancestral allele frequencies.* |
|------|----------------------------------------------------------------------|

---

## Description

[snmf](snmf) estimates admixture coefficients using sparse Non-Negative Matrix Factorization algorithms, and provides STRUCTURE-like outputs.

## Usage

```
snmf (input.file, K,
    project = "continue",
    repetitions = 1, CPU = 1,
    alpha = 10, tolerance = 0.00001, entropy = FALSE, percentage = 0.05,
    I, iterations = 200, ploidy = 2, seed = -1, Q.input.file)
```

## Arguments

input.file    A character string containing a the path to the input file, a genotypic matrix in the [geno](geno) format.

K             An integer vector corresponding to the number of ancestral populations for which the snmf algorithm estimates have to be calculated.

| | |
|---|---|
| project | A character string among "continue", "new", and "force". If "continue", the results are stored in the current project. If "new", the current project is removed and a new one is created to store the result. If "force", the results are stored in the current project even if the input file has been modified since the creation of the project. |
| repetitions | An integer corresponding with the number of repetitions for each value of K. |
| CPU | A number of CPUs to run the parallel version of the algorithm. By default, the number of CPUs is 1. |
| alpha | A numeric value corresponding to the snmf regularization parameter. The results can depend on the value of this parameter, especially for small data sets. |
| tolerance | A numeric value for the tolerance error. |
| entropy | A boolean value. If true, the cross-entropy criterion is calculated (see `create.dataset` and `cross.entropy.estimation`). |
| percentage | A numeric value between 0 and 1 containing the percentage of masked genotypes when computing the cross-entropy criterion. This option applies only if entropy == TRUE (see `cross.entropy`). |
| I | The number of SNPs to initialize the algorithm. It starts the algorithm with a run of snmf using a subset of nb.SNPs random SNPs. If this option is set with nb.SNPs, the number of randomly chosen SNPs is the minimum between 10000 and 10 % of all SNPs. This option can considerably speeds up snmf estimation for very large data sets. |
| iterations | An integer for the maximum number of iterations in algorithm. |
| ploidy | 1 if haploid, 2 if diploid, n if n-ploid. |
| seed | A seed to initialize the random number generator. By default, the seed is randomly chosen. |
| Q.input.file | A character string containing a path to an initialization file for Q, the individual admixture coefficient matrix. |

## Value

snmf returns an object of class snmfProject.

The following methods can be applied to the object of class snmfProject:

| | |
|---|---|
| plot | Plot the minimal cross-entropy in function of K. |
| show | Display information about the analyses. |
| summary | Summarize the analyses. |
| \link{Q} | Return the admixture coefficient matrix for the chosen run with K ancestral populations. |
| \link{G} | Return the ancestral allele frequency matrix for the chosen run with K ancestral populations. |
| \link{cross.entropy} | Return the cross-entropy criterion for the chosen runs with K ancestral populations. |

\link{snmf.pvalues}

> Return the vector of adjusted p-values for a run with K ancestral populations.

\link{impute}     Return a geno or lfmm file with missing data imputation.

\link{barchart}

> Return a bar plot representation of the Q matrix from a run with K ancestral populations .

load.snmfProject(file.snmfProject)

> Load the file containing an snmfProject objet and return the snmfProject object.

remove.snmfProject(file.snmfProject)

> Erase a snmfProject object. Caution: All the files associated with the object will be removed.

export.snmfProject(file.snmfProject)

> Create a zip file containing the full snmfProject object. It allows to move the project to a new directory or a new computer (using import). If you want to overwrite an existing export, use the option force == TRUE.

import.snmfProject(file.snmfProject)

> Import and load an snmfProject object from a zip file (made with the export function) into the chosen directory. If you want to overwrite an existing project, use the option force == TRUE.

combine.snmfProject(file.snmfProject, toCombine.snmfProject)

> Combine to.Combine.snmfProject into file.snmfProject. Caution: Only projects with runs coming from the same input file can be combined. If the same input file has different names in the two projects, use the option force == TRUE.

### Author(s)

Eric Frichot

### References

Frichot E, Mathieu F, Trouillon T, Bouchard G, Francois O. (2014). *Fast and Efficient Estimation of Individual Ancestry Coefficients*. Genetics, 194(4): 973–983.

### See Also

geno pca lfmm Q barchart tutorial

### Examples

```
### Example of analysis using snmf ###

# Creation of the genotype file: genotypes.geno.
# The data contain 400 SNPs for 50 individuals.
data("tutorial")
write.geno(tutorial.R, "genotypes.geno")


################
# running snmf #
################
```

```
project.snmf = snmf("genotypes.geno",
                    K = 1:10,
                    entropy = TRUE,
                    repetitions = 10,
                    project = "new")

# plot cross-entropy criterion of all runs of the project
plot(project.snmf, cex = 1.2, col = "lightblue", pch = 19)

# get the cross-entropy of the 10 runs for K = 4
ce = cross.entropy(project.snmf, K = 4)

# select the run with the lowest cross-entropy for K = 4
best = which.min(ce)

# display the Q-matrix

my.colors <- c("tomato", "lightblue",
               "olivedrab", "gold")

barchart(project.snmf, 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 = 3, cex.axis = .4)

###################
# Post-treatments #
###################

# show the project
show(project.snmf)

# summary of the project
summary(project.snmf)

# get the cross-entropy for all runs for K = 4
ce = cross.entropy(project.snmf, K = 4)

# get the cross-entropy for the 2nd run for K = 4
ce = cross.entropy(project.snmf, K = 4, run = 2)

# get the ancestral genotype frequency matrix, G, for the 2nd run for K = 4.
freq = G(project.snmf, K = 4, run = 2)

############################
# Advanced snmf run options #
############################
```

```
# Q.input.file: init a run with a given ancestry coefficient matrix Q.
# To run the example, remove the comment character

# Example where Q is initialized with the matrix resulting
# from a previous run with K = 4

# project.snmf = snmf("genotypes.geno", K = 4,
#       Q.input.file = "./genotypes.snmf/K4/run1/genotypes_r1.4.Q", project = "new")



# I: init the Q matrix of a run from a smaller run with 100 randomly chosen
# SNPs.
project.snmf = snmf("genotypes.geno", K = 4, I = 100, project = "new")

# CPU: run snmf with 2 CPUs.
project.snmf = snmf("genotypes.geno", K = 4, CPU = 2, project = "new")

# percentage: run snmf and calculate the cross-entropy criterion with 10% of
# masked genotypes, instead of 5% of masked genotypes.
project.snmf = snmf("genotypes.geno", K = 4, entropy = TRUE, percentage = 0.1, project = "new")

# seed: choose the seed for the random generator.
project.snmf = snmf("genotypes.geno", K = 4, seed = 42, project = "new")

# alpha: choose the regularization parameter.
project.snmf = snmf("genotypes.geno", K = 4, alpha = 100, project = "new")

# tolerance: choose the tolerance parameter.
project.snmf = snmf("genotypes.geno", K = 4, tolerance = 0.0001, project = "new")



###########################
# Manage an snmf project #
###########################

# All the runs of snmf for a given file are
# automatically saved into an snmf project directory and a file.
# The name of the snmfProject file is the same name as
# the name of the input file with a .snmfProject extension
# ("genotypes.snmfProject").
# The name of the snmfProject directory is the same name as
# the name of the input file with a .snmf extension ("genotypes.snmf/")
# There is only one snmf Project for each input file including all the runs.

# An snmfProject can be load in a different session.
project.snmf = load.snmfProject("genotypes.snmfProject")

# An snmfProject can be exported to be imported in another directory
# or in another computer
export.snmfProject("genotypes.snmfProject")
```

```
dir.create("test", showWarnings = TRUE)
#import
newProject = import.snmfProject("genotypes_snmfProject.zip", "test")
# combine projects
combinedProject = combine.snmfProject("genotypes.snmfProject", "test/genotypes.snmfProject")
# remove
remove.snmfProject("test/genotypes.snmfProject")


# An snmfProject can be erased.
# Caution: All the files associated with the project will be removed.
remove.snmfProject("genotypes.snmfProject")
```

---

snmf.pvalues                     *P-values for snmf population differentiation tests*

---

### Description

Returns a vector of p-values computed from an snmf run.

### Usage

```
snmf.pvalues (object, genomic.control, lambda, ploidy, entropy, fisher, K, run)
```

### Arguments

object            An snmfProject object.

genomic.control
                  A Boolean value. If TRUE, the p-values are automatically calibrated using ge-
                  nomic control. If FALSE, the p-values are calculated by rescaling the chi-squared
                  test statistics using the lambda parameter.

lambda            A numeric value. The lambda value is used as an inflation factor to rescale the
                  chi-squared statistics in the computation of p-values. This option requires that
                  genomic.control = FALSE. The default value of lambda is equal to 1.0 (no
                  rescaling).

ploidy            An integer value among 1 or 2. Tests are implemented for haploids and diploids
                  (to be extended to polypoids).

entropy           A Boolean value. If TRUE, the run of minimum entropy is used for computing
                  the p-values.

fisher            A Boolean value. If TRUE, F-distributions are used to test the null-hypothesis,
                  Chi-squared otherwise.

K                 An integer value. The number of genetic clusters.

run               An integer for the run number used the computation of p-values (by default, the
                  minimum entropy run).

## Value

| | |
|---|---|
| `p.values` | A vector of p-values for each locus for the population differentiation test. |
| `GIF` | The inflation factor value used in the test. |

## Author(s)

Olivier Francois

## References

Martins, H., Caye, K., Luu, K., Blum, M. G. B., Francois, O. (2016). Identifying outlier loci in ad-mixed and in continuous populations using ancestral population differentiation statistics. Molecular Ecology, 25(20), 5029-5042.

## See Also

[snmf](#)

## Examples

```
### Example of analyses using snmf ###

data("tutorial")
# creation of a genotype file, "genotypes.lfmm".
# The data contain 400 SNPs for 50 individuals.
write.geno(tutorial.R, "genotypes.geno")


################
# snmf runs    #
###############

# main options, K: the number of ancestral populations,
#        entropy: cross-entropy criterion,
#        CPU: the number of CPUs.

project.snmf = snmf("genotypes.geno",
                    K = 4,
                    entropy = TRUE,
                    ploidy = 2,
                    repetitions = 10,
                    project = "new")

# genome scan using adjusted p-values (genomic control method)

p = snmf.pvalues(project.snmf, entropy = TRUE, ploidy = 2, K = 4)
p$GIF

par(mfrow = c(2,1))
hist(p$pvalues, col = "orange")
```

```
plot(-log10(p$pvalues), pch = 19, col = "blue", cex = .7)
```

---

struct2geno                    *Conversion from the STRUCTURE format to the geno format.*

---

### Description

The function converts a multiallelic genotype file in the STRUCTURE format into a file in the 'geno' for [snmf](#) and the 'lfmm' format for [lfmm](#).

### Usage

```
struct2geno (input.file, ploidy, FORMAT, extra.row, extra.column)
```

### Arguments

| | |
|---|---|
| input.file | A character string. A path to a STRUCTURE or a TESS input file of multiallelic markers (eg, microsatellites) for haploid or diploid individuals. Missing data must be encoded as "-9" or as any negative value. Individual genotypes are encoded using either one or two rows of data. |
| ploidy | An integer value (1 or 2). Value 2 for diploids and 1 for haploids. |
| FORMAT | An integer value equal to 1 for markers encoded using one row of data for each individual, and 2 for markers encoded using two rows of data for each individual. |
| extra.row | An integer value indicating the number of extra rows in the header of the input file (eg, marker ids). |
| extra.column | an integer value indicating the number of extra columns in the input file. Extra columns can include individual ids, pop ids, geographic coordinates, etc. |

### Value

NULL. Output files in the 'geno' and the 'lfmm' format record individual genotypes for each allele at each marker.

### Author(s)

Olivier Francois

### See Also

[lfmm.data](#) [geno](#) [lfmm](#) [snmf](#)

## Examples

```
### Example of conversion from a STRUCTURE format ###
### Artificial data with 10 diploid individuals and 10 STR markers
### FORMAT = 1
### Input file: 'dat.str'

dat.str  <- matrix(sample(c(101:105,-9),
                   200, prob = c(rep(1,5), 0.1),
                   replace = TRUE),
                   nrow = 10, ncol = 20)
write.table(dat.str,
            file = "dat.str",
            col.names = FALSE,
            row.names = FALSE,
            quote = FALSE)

### Conversion
struct2geno("dat.str", ploidy = 2, FORMAT = 1)

### snmf run and barplot
s  <- snmf("dat.str.geno", K = 2, project = "new")
barchart(s, K = 2, run = 1, xlab = "Individuals")
```

---

tracy.widom                          *Tracy-Widom test for eigenvalues*

---

## Description

Perform tracy-widom tests on a set of eigenvalues to determine the number of significative eigenvalues and calculate the percentage of variance explained by each principal component. For an example, see [pca](pca).

## Usage

```
tracy.widom (object)
```

## Arguments

object              a pcaProject object.

## Value

tracy.widom returns a list containing the following components:

| | |
|---|---|
| eigenvalues | The sorted input vector of eigenvalues (by descreasing order). |
| twstats | The vector of tracy-widom statistics. |
| pvalues | The vector of p-values associated with each eigenvalue. |
| effecn | The vector of effective sizes. |
| percentage | The vector containing the percentage of variance explained by each principal component. |

## Author(s)

Eric Frichot

## References

Tracy CA and Widom H. (1994). *Level spacing distributions and the bessel kernel*. Commun Math Phys. 161 :289–309. Patterson N, Price AL and Reich D. (2006). *Population structure and eigenanalysis*. PLoS Genet. 2 :20.

## See Also

pca lfmm.data lfmm

## Examples

```
# Creation of the genotype file "genotypes.lfmm"
# with 1000 SNPs for 165 individuals.
data("tutorial")
write.lfmm(tutorial.R,"genotypes.lfmm")


#################
# Perform a PCA #
#################

# run of PCA
# Available    options, K (the number of PCs calculated),
#              center and scale.
# Creation of  genotypes.pcaProject - the pcaProject object.
#              a directory genotypes.pca containing:
# 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)

#############################
# Perform Tracy-Widom tests #
#############################

# Perfom Tracy-Widom tests on all eigenvalues.
# Create file:  genotypes.tracyWidom - tracy-widom test information,
#               in the directory genotypes.pca/.
tw = tracy.widom(pc)

# Plot the percentage of variance explained by each component.
plot(tw$percentage)

# Display the p-values for the Tracy-Widom tests.
tw$pvalues

# remove pca Project
```

```
remove.pcaProject("genotypes.pcaProject")
```

---

tutorial                        *Example tutorial data sets*

---

## Description

This data set is composed of a genotypic matrix stored in tutorial.R with 50 individuals genotyped at 400 SNPs. The last 50 SNPs are correlated with an environmental variable recorded in tutorial.C. The data are a subset of the data shown in the computer note associated with the package (Frichot and Francois 2015).

## Value

tutorial.R       A genotypic matrix for 50 individuals genotyped at 400 SNPs. The last 50 SNPs
                 are correlated with an environmental variable stored in tutorial.C.

tutorial.C       An environmental variable measured for 50 individuals.

---

vcf                             vcf *format description*

---

## Description

Description of the vcf format. The vcf format can be used as an input format for genotypic matrices in the functions [snmf](), [lfmm](), and [pca]().

## Details

The vcf format is described here.

Here is an example of a genotypic matrix using the vcf format with 3 individuals and 4 loci:

```
##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
```

## Author(s)

Eric Frichot

## See Also

vcf2geno vcf2lfmm geno lfmm ped ancestrymap

---

vcf2geno *Convert from* vcf *to* geno *format*

---

## Description

A function that converts from the vcf format to the geno format. Note: This function may be obsolete. Conversion in accepted format such as ped can be obtained with the program vcftools.

## Usage

```
vcf2geno(input.file, output.file = NULL, force = TRUE)
```

## Arguments

input.file       A character string containing a path to the input file, a genotypic matrix in the
                 vcf format.

output.file      A character string containing a path to the output file, a genotypic matrix in the
                 geno format. By default, the name of the output file is the same name as the
                 input file with a .geno extension.

force            A boolean option. If FALSE, the input file is converted only if the output file
                 does not exist. If TRUE, convert the file anyway.

## Value

output.file      A character string containing a path to the output file, a genotypic matrix in the
                 geno format.

## Author(s)

Eric Frichot

## See Also

vcf geno ancestrymap2lfmm ancestrymap2geno ped2lfmm ped2geno lfmm2geno geno2lfmm

## Examples

```
# Creation of a file called "example.vcf"
# with 4 SNPs for 3 individuals.
data("example_vcf")
write.table(example_vcf,"example.vcf",col.names =
    c("#CHROM", "POS", "ID", "REF", "ALT", "QUAL", "FILTER", "INFO",
    "FORMAT", "SAMPLE0", "SAMPLE1", "SAMPLE2"),
    row.names = FALSE, quote = FALSE)
```

```
# Conversion    from the vcf format ("example.vcf")
#               to the geno format ("example.geno").
# By default,   the name of the output file is the same name
#               as the input file with a .geno extension.
# Create files: "example.geno",
#               "example.vcfsnp" - SNP informations,
#               "example.removed" - removed lines.
output = vcf2geno("example.vcf")

# Conversion    from the vcf format ("example.vcf")
#               to the geno format with the output file called "plop.geno".
# Create files: "plop.geno",
#               "plop.vcfsnp" - SNP informations,
#               "plop.removed" - removed lines.
output = vcf2geno("example.vcf", "plop.geno")

# As force = false and the file "example.geno" already exists,
# nothing happens.
output = vcf2geno("example.vcf", force = FALSE)
```

---

vcf2lfmm                              *Convert from* vcf *to* lfmm *format*

---

### Description

A function that converts from the vcf format to the lfmm format. Note: This function may be
obsolete. Conversion in accepted format such as ped can be obtained with the program vcftools.

### Usage

```
vcf2lfmm(input.file, output.file = NULL, force = TRUE)
```

### Arguments

input.file      A character string containing a path to the input file, a genotypic matrix in the
                vcf format.

output.file     A character string containing a path to the output file, a genotypic matrix in the
                lfmm format. By default, the name of the output file is the same name as the
                input file with a .lfmm extension.

force           A boolean option. If FALSE, the input file is converted only if the output file
                does not exist. If TRUE, convert the file anyway.

### Value

output.file     A character string containing a path to the output file, a genotypic matrix in the
                lfmm format.

**Author(s)**

Eric Frichot

**See Also**

vcf lfmm.data ancestrymap2lfmm ancestrymap2geno ped2lfmm ped2geno vcf2geno

**Examples**

```
# Creation of a file called "example.vcf"
# with 4 SNPs for 3 individuals.
data("example_vcf")
write.table(example_vcf,"example.vcf",col.names =
    c("#CHROM", "POS", "ID", "REF", "ALT", "QUAL", "FILTER", "INFO",
    "FORMAT", "SAMPLE0", "SAMPLE1", "SAMPLE2"),
    row.names = FALSE, quote = FALSE)

# Conversion    from the vcf format ("example.vcf")
#               to the lfmm format ("example.lfmm").
# By default,   the name of the output file is the same name
#               as the input file with a .lfmm extension.
# Create files: "example.lfmm",
#               "example.vcfsnp" - SNP informations,
#               "example.removed" - removed lines.
output = vcf2lfmm("example.vcf")

# Conversion    from the vcf format ("example.vcf")
#               to the lfmm format with the output file called "plop.lfmm".
# Create files: "plop.lfmm",
#               "plop.vcfsnp" - SNP informations,
#               "plop.removed" - removed lines.
output = vcf2lfmm("example.vcf", "plop.lfmm")

# As force = false and the file "example.lfmm" already exists,
# nothing happens.
output = vcf2lfmm("example.vcf", force = FALSE)
```

---

write.env       *Write files in the* env *format*

---

**Description**

Write a file in the env format.

**Usage**

```
write.env(R, output.file)
```

## Arguments

R            A matrix containing the environmental variables with one line for each individ-
             ual and one column for each environmental variable. The missing genotypes
             have to be encoded with the value 9.

output.file  A character string containing a path to the output file, an environmental data
             matrix in the env formt.

## Value

output.file  A character string containing a path to the output file, an environmental data
             matrix in the env formt.

## Author(s)

Eric Frichot

## See Also

[read.env](#) [env](#) [lfmm](#)

## Examples

```
# Creation of an environmental matrix C
# containing 2 environmental variables for 3 individuals.
# C contains one line for each individual and one column for each variable.
C = matrix(runif(6), ncol=2, nrow=3)

# Write C in a file called "tuto.env".
# Create file:    "tuto.env".
write.env(C,"tuto.env")

# Read the file "tuto.env".
C = read.env("tuto.env")
```

---

write.geno                *Write files in the* geno *format*

---

## Description

Write a file in the [geno](#) format.

## Usage

```
write.geno(R, output.file)
```

## Arguments

| | |
|---|---|
| R | A matrix containing the genotypes with one line for each individual and one column for each SNP. The missing genotypes have to be encoded with the value 9. |
| output.file | A character string containing a path to the output file, a genotypic matrix in the geno format. |

## Value

| | |
|---|---|
| output.file | A character string containing a path to the output file, a genotypic matrix in the geno format. |

## Author(s)

Eric Frichot

## See Also

read.geno geno snmf geno2lfmm lfmm2geno ancestrymap2geno ped2geno vcf2geno

## Examples

```
# Creation of a file called "genotypes.geno" in the working directory,
# with 1000 SNPs for 165 individuals.
data("tutorial")

# Write R in a file called "genotypes.geno".
# Create file:    "genotypes.geno".
write.geno(tutorial.R,"genotypes.geno")

# Read the file "genotypes.geno".
R = read.geno("genotypes.geno")
```

---

write.lfmm                     *Write files in the* lfmm *format*

---

## Description

Write a file in the lfmm format.

## Usage

```
write.lfmm(R, output.file)
```

## Arguments

R                A matrix containing the genotypes with one line for each individual and one
                 column for each SNP. The missing genotypes have to be encoded with the value
                 9.

output.file      A character string containing a path to the output file, a genotypic matrix in the
                 lfmm format.

## Value

output.file      A character string containing a path to the output file, a genotypic matrix in the
                 geno format.

## Author(s)

Eric Frichot

## See Also

[read.lfmm](read.lfmm) [lfmm.data](lfmm.data) [lfmm](lfmm) [geno2lfmm](geno2lfmm) [lfmm2geno](lfmm2geno) [ancestrymap2lfmm](ancestrymap2lfmm) [ped2lfmm](ped2lfmm)

## Examples

```
# Creation of a file called "genotypes.geno" in the working directory,
# with 1000 SNPs for 165 individuals.
data("tutorial")

# write R in a file called "genotypes.lfmm"
# Create file:    "genotypes.lfmm".
write.lfmm(tutorial.R,"genotypes.lfmm")

# read the file "genotypes.lfmm".
R = read.lfmm("genotypes.lfmm")
```

---

z.scores                        *z-scores from an lfmm run*

---

## Description

Return the lfmm output matrix of zscores for the chosen runs with K latent factors, the d-th variable
and the all option. For an example, see [lfmm](lfmm).

## Usage

```
z.scores (object, K, d, all, run)
```

**Arguments**

| | |
|---|---|
| object | A lfmmProject object. |
| K | The number of latent factors. |
| d | The d-th variable. |
| all | A Boolean option. If true, the run with all variables at the same time. If false, the runs with each variable separately. |
| run | A list of chosen runs. |

**Value**

| | |
|---|---|
| res | A matrix containing a vector of z-scores for the chosen runs per column. |

**Author(s)**

Eric Frichot

**See Also**

[lfmm](#) [lfmm.data](#)

**Examples**

```
### Example of analyses using lfmm ###

data("tutorial")
# creation of the genotype file, genotypes.lfmm.
# It contains 400 SNPs for 50 individuals.
write.lfmm(tutorial.R, "genotypes.lfmm")
# creation of the environment file, gradient.env.
# It contains 1 environmental variable for 40 individuals.
write.env(tutorial.C, "gradients.env")

################
# runs of lfmm #
################

# main options, K: the number of latent factors,
# CPU: the number of CPUs.

# Toy runs with K = 3 and 2 repetitions.
# around 15 seconds per run.
project = NULL
project = lfmm("genotypes.lfmm", "gradients.env", K = 3, repetitions = 2,
    iterations = 6000, burnin = 3000, project = "new")

# get the z-scores for all runs for K = 3
z = z.scores(project, K = 3)

# get the z-scores for the 2nd run for K = 3
z = z.scores(project, K = 3, run = 2)
```

```
# remove
remove.lfmmProject("genotypes_gradients.lfmmProject")
```

zscore.format          *Output file format for* lfmm

### Description

Description of the zscore output format of lfmm.

### Details

The zscore format 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).

### Author(s)

Eric Frichot

### See Also

lfmm lfmm.data env

# Index