Package 'LEA'

October 9, 2015

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

Version 1.0.0

Date 2014-09-17

Author Eric Frichot <`eric.frichot@gmail.com`>, Olivier Francois <`olivier.francois@imag.fr`>

Maintainer Eric Frichot <`eric.frichot@gmail.com`>

Depends R (>= 3.1.0), methods

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), and controlling the false discovery rate. LEA is mainly based on optimized C programs that can scale with the dimension of very large data sets.

biocViews Software, StatisticalMethod, Clustering, Regression

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

NeedsCompilation yes

R topics documented:

`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:	1.0
Date:	2013-12-16
License:	GPL-3

Author(s)

Eric Frichot Maintainer: Eric Frichot <eric.frichot@imag.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)
```

`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 from snmf runs*

Description

Return the cross-entropy criterion for the chosen runs with K ancestral populations. For an example, see `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 criterion can be automatically calculated by the `snmf` function with the `entropy` option.

Usage

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

Arguments

`object`	A snmfProject object.
`K`	The number of ancestral populations.
`run`	A list of chosen run number.

Value

`res`	A list containing the cross-entropy criterion for the chosen runs with K ancestral populations.

Author(s)

Eric Frichot

See Also

`geno` `snmf` `G` `Q`

Examples

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

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

#################
# runs of snmf #
#################

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

# Runs with K = 3 with cross-entropy and 2 repetitions.
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.snmf/K3/run1/genotypes_r1.3.Q",
    "./genotypes.snmf/K3/run1/genotypes_r1.3.G")

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

---

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 variable:

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

Author(s)

Eric Frichot

See Also

`lfmm` `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 the chosen run with K ancestral populations.

Author(s)

Eric Frichot

See Also

`geno` `snmf` `Q` `cross.entropy`

Examples

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

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

################
```

```
# runs of snmf #
################

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

# Runs with K between 1 and 5 with cross-entropy and 2 repetitions.
project = NULL
project = snmf("genotypes.geno", K = 3, repetitions = 2, project = "new")

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

---

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)
```

`lfmm` *Fitting Latent Factor Mixed Models*

Description

`lfmm` is used to fit Latent Factor Mixed Models. The goal of `lfmm` is to identify genetic polymorphisms that exhibit high correlation with some environmental gradient or with the variables used as proxies for ecological pressures.

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_fomat`} format.
`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 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.
`d`	An integer corresponding to the fit of `lfmm` model with the `d`-th variable only from `environment.file`. By default (if NULL and all is FALSE), fit `lfmm` with each variable from `environment.file` sequentially and independently.

`all`	A boolean option. If true, fit `lfmm` with all variables from `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.
`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 iterations in the Gibbs Sampling algorithm.
`burnin`	The burnin number of iterations in the Gibbs Sampling algorithm.
`seed`	A seed to initialize the random number generator. By default, the seed is randomly chosen. The seed is initialized at each repetition. If you want to set a seed, please provide a seed per repetition.
`repetitions`	The number of repetitions of each run.
`epsilon.noise`	Prior on the different variances.
`epsilon.b`	Prior on the variance of the correlation coefficients.
`random.init`	A boolean option. If true, the Gibbs Sampler is initiliazed randomly. Otherwise, it is initialized with zeros.

Value

`lfmm` returns an object of class `lfmmProject`.

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

`show`	Display information about the analyses.
`summary`	Summarize the analyses.
`z.scores`	Return the `lfmm` output vector of zscores for the chosen runs with K latent factors, the d-th variable and the all option.
`p.values`	Return the `lfmm` output vector of p-values for the chosen runs with K latent factors, the d-th variable and the all option.
`mlog10p.values`	Return the `lfmm` output vector of -log10(p-values) for the chosen runs with K latent factors, the d-th variable and the all option.
`load.lfmmProject (file = "character")`	Load the file containing an lfmmProject objet and return the lfmmProject object.
`remove.lfmmProject (file = "character")`	Erase a `lfmmProject` object. Caution: All the files associated with the object will be removed.

Author(s)

Eric Frichot

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` `p.values` `mlog10p.values` `pca` `snmf` tutorial

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.

# Runs with K = 9 and 5 repetitions.
# The runs are composed of 6000 iterations including 3000 iterations
# for burnin.
# around 30 seconds per run.
project = NULL
project = lfmm("genotypes.lfmm", "gradients.env", K = 6, repetitions = 5,
        project = "new")

# get the zscores of each run for K = 6
zs = z.scores(project, K = 6)

# Combine the z-scores using the Stouffer method
zs.stouffer = apply(zs, MARGIN = 1, median)

# calculate the inflation factor
lambda = median(zs.stouffer^2)/.456

# calculate adjusted p-values
cp.values = pchisq(zs.stouffer^2/lambda, df = 1, lower = FALSE)

for (alpha in c(.05,.1,.15,.2)) {
    # expected FDR
    print(paste("expected FDR:", alpha))
    L = length(cp.values)
    # return a list of candidates with an expected FDR of alpha.
    w = which(sort(cp.values) < alpha * (1:L) / L)
    candidates = order(cp.values)[w]

    # estimated FDR and True Positif
    estimated.FDR = length(which(candidates <= 350))/length(candidates)
```

```
    estimated.TP = length(which(candidates > 350))/50
    print(paste("FDR:", estimated.FDR, "True Positive:", estimated.TP))
}

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

# show the project
show(project)

# summary of the project
summary(project)

# get the z-scores for the 2nd run for K = 6
z = z.scores(project, K = 6, run = 2)

# get the p-values for the 2nd run for K = 6
p = p.values(project, K = 6, run = 2)

# get the -log10(p-values) for the 2nd run for K = 6
mp = mlog10p.values(project, K = 6, run = 2)

##########################
# Manage an lfmm project #
##########################

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

# An lfmmProject can be load in a different session.
project = load.lfmmProject("genotypes_gradients.lfmmProject")

# An lfmmProject can be erased.
# Caution: All the files associated with the project will be removed.
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`, 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 9.

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

`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)
```

`mlog10p.values` *-log10(p-values) from a lfmm run*

Description

Return the `lfmm` output matrix of -log10(p-values) for the chosen runs with K latent factors, the d-th variable and the all option. For an example, see `lfmm`.

Usage

```
mlog10p.values (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 -log10(p-values) for the chosen runs per column.

Author(s)

Eric Frichot

See Also

`lfmm.data` `lfmm` `p.values` `z.scores`

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 -log10(p-values) for all runs for K = 3
mp = mlog10p.values(project, K = 3)

# get the -log10(p-values) for the 2nd run for K =3
mp = mlog10p.values(project, K = 3, run = 2)
```

`p.values` *p-values from a lfmm run*

Description

Return the `lfmm` output matrix of p-values for the chosen runs with K latent factors, the d-th variable and the all option. For an example, see `lfmm`.

Usage

```
p.values (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 p.values for the chosen runs per column.

Author(s)

Eric Frichot

See Also

`lfmm.data` `lfmm` `mlog10p.values` `z.scores`

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 p-values for all runs for K = 3
p = p.values(project, K = 3)

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

`pca` *Principal Component Analysis*

Description

The function pca performs a Principal Component Analysis of a genotypic matrix using the `lfmm`, `geno`, `ancestrymap`, `ped` or `vcf` format. The function computes eigenvalue, eigenvector, and standard deviation for each principal component and the projection of each individual on each component. The function pca 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 about the analysis.
`summary`	Summarize the analysis.
`tracy.widom`	Perform Tracy-Widom tests on the 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 object will be removed.

Author(s)

Eric Frichot

See Also

`lfmm.data` `snmf` `lfmm` `tutorial`

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)

#######################
# Display Information #
#######################
```

```
# Display information about the analysis.
show(pc)

# Summarize the analysis.
summary(pc)

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

par(mfrow=c(2,2))

# Plot eigenvalues.
plot(pc, lwd=5, col="red",xlab=("PCs"),ylab="eigen")

# 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 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
```

---

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 ancestral populations.

Author(s)

Eric Frichot

See Also

`geno` `snmf` `G` `cross.entropy`

Examples

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

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

################
# runs of snmf #
################

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

# Runs with K between 1 and 5 with cross-entropy and 2 repetitions.
project = NULL
project = snmf("genotypes.geno", K = 3, repetitions = 2, project = "new")

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

# plot the 2nd run for K = 3 (ancestry coefficients).
barplot(t(Q(project, K = 3, run = 2)))
```

`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` estimates admixture coefficients using sparse Non-Negative Matrix Factorization algorithms, and provide 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` 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.
`Q`	Return the admixture coefficient matrix for the chosen run with K ancestral populations.
`G`	Return the ancestral allele frequency matrix for the chosen run with K ancestral populations.
`cross.entropy`	Return the cross-entropy criterion for the chosen runs 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.

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` `tutorial`

Examples

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

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

################
# runs of snmf #
################

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

# Runs with K between 1 and 5 with cross-entropy and 2 repetitions.
project = NULL
project = snmf("genotypes.geno", K=1:10, entropy = TRUE, repetitions = 10,
    project = "new")

# plot cross-entropy criterion of all runs of the project
plot(project, lwd = 5, col = "red", pch=1)

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

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

# plot the best run for K = 4 (ancestry coefficients).
barplot(t(Q(project, K = 4, run = best)))

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

# show the project
show(project)

# summary of the project
summary(project)

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

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

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

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

# Q.input.file: init a run with a given ancestry coefficient matrix Q.
# Here, it is initialized with the Q matrix from the first run with K=4
project = snmf("genotypes.geno", K = 4,
    Q.input.file = "./genotypes.snmf/K4/run1/genotypes_r1.4.Q")

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

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

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

# seed: choose the seed to init the randomization.
project = snmf("genotypes.geno", K = 4, seed=42)

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

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

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

# All the runs of snmf for a given file are
# automatically saved into a 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 = load.snmfProject("genotypes.snmfProject")

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

`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`.

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
```

`tutorial` *Example tutorial data sets*

Description

This dataset is composed of a genotypic matrix called tutorial.R with 50 individuals for 400 SNPs. The last 50 SNPs are correlated with an environmental variable called tutorial.C. This dataset is a subset of the dataset displayed in the note associated with the package.

Usage

```
tutorial
```

Value

`tutorial.R`	A genotypic matrix with 50 individuals for 400 SNPs. The last 50 SNPs are correlated with an environmental variable called tutorial.C.
`tutorial.C`	An environmental variable for the 50 invdividuals.

`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.

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.

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 individual 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` `lfmm.data` `lfmm` `geno2lfmm` `lfmm2geno` `ancestrymap2lfmm` `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 a 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`.

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)
```

`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