

# Package ‘lfa’

December 20, 2024

**Title** Logistic Factor Analysis for Categorical Data

**Version** 2.7.0

**Encoding** UTF-8

**LazyData** true

**Description** Logistic Factor Analysis is a method for a PCA analogue on Binomial data via estimation of latent structure in the natural parameter. The main method estimates genetic population structure from genotype data. There are also methods for estimating individual-specific allele frequencies using the population structure. Lastly, a structured Hardy-Weinberg equilibrium (HWE) test is developed, which quantifies the goodness of fit of the genotype data to the estimated population structure, via the estimated individual-specific allele frequencies (all of which generalizes traditional HWE tests).

**Imports** utils, methods, corpcor, RSpectra

**Depends** R (>= 4.0)

**Suggests** knitr, ggplot2, testthat, BEDMatrix, genio

**VignetteBuilder** knitr

**License** GPL (>= 3)

**biocViews** SNP, DimensionReduction, PrincipalComponent, Regression

**BugReports** <https://github.com/StoreyLab/lfa/issues>

**URL** <https://github.com/StoreyLab/lfa>

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**git\_url** <https://git.bioconductor.org/packages/lfa>

**git\_branch** devel

**git\_last\_commit** bf41026

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2024-12-20

**Author** Wei Hao [aut],  
 Minsun Song [aut],  
 Alejandro Ochoa [aut, cre] (ORCID:  
<https://orcid.org/0000-0003-4928-3403>),  
 John D. Storey [aut] (ORCID: <https://orcid.org/0000-0001-5992-402X>)  
**Maintainer** Alejandro Ochoa <[alejandro.ochoa@duke.edu](mailto:alejandro.ochoa@duke.edu)>

## Contents

af . . . . .	2
af_snp . . . . .	3
center . . . . .	4
center-deprecated . . . . .	5
centerscale . . . . .	5
hgdp_subset . . . . .	6
lfa . . . . .	6
model.gof-deprecated . . . . .	8
pca_af . . . . .	9
read.bed-deprecated . . . . .	10
read.tped.recode-deprecated . . . . .	10
sHWE . . . . .	11
trunc_svd . . . . .	12

<b>Index</b>	<b>14</b>
--------------	-----------

---

af	<i>Allele frequencies</i>
----	---------------------------

---

### Description

Compute matrix of individual-specific allele frequencies

### Usage

```
af(X, LF, safety = FALSE, max_iter = 100, tol = 1e-10)
```

### Arguments

X	A matrix of SNP genotypes, i.e. an integer matrix of 0's, 1's, 2's and NAs. BEDMatrix is supported. Sparse matrices of class Matrix are not supported (yet).
LF	Matrix of logistic factors, with intercept. Pass in the return value from <code>lfa()</code> !
safety	Optional boolean to bypass checks on the genotype matrices, which require a non-trivial amount of computation. Ignored if X is a BEDMatrix object.
max_iter	Maximum number of iterations for logistic regression
tol	Numerical tolerance for convergence of logistic regression

## Details

Computes the matrix of individual-specific allele frequencies, which has the same dimensions of the genotype matrix. Be warned that this function could use a ton of memory, as the return value is all doubles. It could be wise to pass only a selection of the SNPs in your genotype matrix to get an idea for memory usage. Use `gc()` to check memory usage!

## Value

Matrix of individual-specific allele frequencies.

## Examples

```
LF <- lfa( hgdp_subset, 4 )
allele_freqs <- af( hgdp_subset, LF )
```

---

af_snp	<i>Allele frequencies for SNP</i>
--------	-----------------------------------

---

## Description

Computes individual-specific allele frequencies for a single SNP.

## Usage

```
af_snp(snp, LF, max_iter = 100, tol = 1e-10)
```

## Arguments

snp	vector of 0's, 1's, and 2's
LF	Matrix of logistic factors, with intercept. Pass in the return value from <code>lfa()</code> !
max_iter	Maximum number of iterations for logistic regression
tol	Numerical tolerance for convergence of logistic regression

## Value

vector of allele frequencies

## See Also

[af\(\)](#)

## Examples

```
LF <- lfa(hgdp_subset, 4)
# pick one SNP only
snp <- hgdp_subset[ 1, ]
# allele frequency vector for that SNP only
allele_freqs_snp <- af_snp(snp, LF)
```

---

center

*Deprecated functions in package lfa.*

---

## Description

The functions listed below are deprecated and will be defunct in the near future. When possible, alternative functions with similar functionality are also mentioned. Help pages for deprecated functions are available at `help("<function>-deprecated")`.

## Usage

```
center(A)
```

```
model.gof(X, LF, B)
```

```
read.bed(bed.prefix)
```

```
read.tped.recode(tped.filename, buffer.size = 5e+08)
```

## Value

Function-dependent

center

For center, use `function(x) x - rowMeans(x)`.

model.gof

For model.gof, use [SHWE\(\)](#).

read.bed

For read.bed, use [genio::read\\_plink\(\)](#).

read.tped.recode

For read.tped.recode, use `plink` (external binary) to convert to BED/BIM/FAM, then parse with [genio::read\\_plink\(\)](#).

---

center-deprecated      *Matrix centering*

---

**Description**

C routine to row-center a matrix

**Usage**

```
center(A)
```

**Arguments**

A                      matrix

**Value**

A but row centered

**See Also**

[lfa-deprecated\(\)](#)

---

centerscale              *Matrix centering and scaling*

---

**Description**

C routine to row-center and scale a matrix. Doesn't work with missing data.

**Usage**

```
centerscale(A)
```

**Arguments**

A                      matrix

**Value**

matrix same dimensions A but row centered and scaled

**Examples**

```
Xc <- centerscale(hgdp_subset)
```

---

hgdp_subset	<i>HGDP subset</i>
-------------	--------------------

---

**Description**

Subset of the HGDP dataset.

**Usage**

```
hgdp_subset
```

**Format**

a matrix of 0's, 1's and 2's.

**Value**

genotype matrix

**Source**

Stanford HGDP <http://www.hagsc.org/hgdp/files.html>

---

lfa	<i>Logistic factor analysis</i>
-----	---------------------------------

---

**Description**

Fit logistic factor model of dimension  $d$  to binomial data. Computes  $d - 1$  singular vectors followed by intercept.

**Usage**

```
lfa(  
  X,  
  d,  
  adjustments = NULL,  
  override = FALSE,  
  safety = FALSE,  
  rspectra = FALSE,  
  ploidy = 2,  
  tol = .Machine$double.eps,  
  m_chunk = 1000  
)
```

## Arguments

<code>X</code>	A matrix of SNP genotypes, i.e. an integer matrix of 0's, 1's, 2's and NAs. BEDMatrix is supported. Sparse matrices of class Matrix are not supported (yet).
<code>d</code>	Number of logistic factors, including the intercept
<code>adjustments</code>	A matrix of adjustment variables to hold fixed during estimation. Number of rows must equal number of individuals in X. These adjustments take the place of LFs in the output, so the number of columns must not exceed d-2 to allow for the intercept and at least one proper LF to be included. When present, these adjustment variables appear in the first columns of the output. Not supported when X is a BEDMatrix object.
<code>override</code>	Optional boolean passed to <code>trunc_svd()</code> to bypass its Lanczos bidiagonalization SVD, instead using <code>corpacor::fast.svd()</code> . Usually not advised unless encountering a bug in the SVD code. Ignored if X is a BEDMatrix object.
<code>safety</code>	Optional boolean to bypass checks on the genotype matrices, which require a non-trivial amount of computation. Ignored if X is a BEDMatrix object.
<code>rspectra</code>	If TRUE, use <code>RSpectra::svds()</code> instead of default <code>trunc_svd()</code> or <code>corpacor::fast.svd()</code> options. Ignored if X is a BEDMatrix object.
<code>ploidy</code>	Ploidy of data, defaults to 2 for bi-allelic unphased SNPs
<code>tol</code>	Tolerance value passed to <code>trunc_svd()</code> Ignored if X is a BEDMatrix object.
<code>m_chunk</code>	If X is a BEDMatrix object, number of loci to read per chunk (to control memory usage).

## Details

Genotype matrix should have values in 0, 1, 2, or NA. The coding of the SNPs (which case is 0 vs 2) does not change the output.

## Value

The matrix of logistic factors, with individuals along rows and factors along columns. The intercept appears at the end of the columns, and adjustments in the beginning if present.

## Examples

```
LF <- lfa(hgdp_subset, 4)
dim(LF)
head(LF)
```

---

model.gof-deprecated *LFA model goodness of fit*

---

### Description

Compute SNP-by-SNP goodness-of-fit when compared to population structure. This can be aggregated to determine genome-wide goodness-of-fit for a particular value of  $d$ .

### Usage

```
model.gof(X, LF, B)
```

### Arguments

X	A matrix of SNP genotypes, i.e. an integer matrix of 0's, 1's, 2's and NAs. BEDMatrix is supported.
LF	matrix of logistic factors
B	number of null datasets to generate, $B = 1$ is usually sufficient. If computational time/power allows, a few extra B could be helpful

### Details

This function returns p-values for LFA model goodness of fit based on a simulated null.

### Value

vector of p-values for each SNP.

### Note

Genotype matrix is expected to be a matrix of integers with values 0, 1, and 2. Currently no support for missing values. Note that the coding of the SNPs does not affect the algorithm.

### See Also

[lfa-deprecated\(\)](#)



---

pca\_af *PCA Allele frequencies*

---

**Description**

Compute matrix of individual-specific allele frequencies via PCA

**Usage**

```
pca_af(X, d, override = FALSE, ploidy = 2, tol = 1e-13, m_chunk = 1000)
```

**Arguments**

X	A matrix of SNP genotypes, i.e. an integer matrix of 0's, 1's, 2's and NAs. BEDMatrix is supported. Sparse matrices of class Matrix are not supported (yet).
d	Number of logistic factors, including the intercept
override	Optional boolean passed to <code>trunc_svd()</code> to bypass its Lanczos bidiagonalization SVD, instead using <code>corpcor::fast.svd()</code> . Usually not advised unless encountering a bug in the SVD code. Ignored if X is a BEDMatrix object.
ploidy	Ploidy of data, defaults to 2 for bi-allelic unphased SNPs
tol	Tolerance value passed to <code>trunc_svd()</code> Ignored if X is a BEDMatrix object.
m_chunk	If X is a BEDMatrix object, number of loci to read per chunk (to control memory usage).

**Details**

This corresponds to algorithm 1 in the paper. Only used for comparison purposes.

**Value**

Matrix of individual-specific allele frequencies.

**Examples**

```
LF <- lfa(hgdp_subset, 4)
allele_freqs_lfa <- af(hgdp_subset, LF)
allele_freqs_pca <- pca_af(hgdp_subset, 4)
summary(abs(allele_freqs_lfa-allele_freqs_pca))
```

read.bed-deprecated    *File input: .bed*

---

**Description**

Reads in genotypes in .bed format with corresponding bim and fam files

**Usage**

```
read.bed(prefix)
```

**Arguments**

bed.prefix    Path leading to the bed, bim, and fam files.

**Details**

Use plink with `-make-bed`

**Value**

Genotype matrix

**See Also**

[lfa-deprecated\(\)](#)

---

read.tped.recode-deprecated  
*Read .tped*

---

**Description**

Reads a .tped format genotype matrix and returns the R object needed by [lfa](#).

**Usage**

```
read.tped.recode(tped.filename, buffer.size=5e8)
```

**Arguments**

tped.filename    Path to your .tped file after transposing and recoding.  
buffer.size    Number of characters to keep in the buffer

**Details**

Use `–transpose` and `–recode12` on your plink formatted genotypes to generate the proper tped file. This is a pretty terrible function that uses a growing matrix for the genotypes so it is to your benefit to have as large a `buffer.size` as possible.

**Value**

genotype matrix with elements 0, 1, 2, and NA.

**See Also**

[lfa-deprecated\(\)](#)

**Examples**

```
#assuming you have a .tped file in the right directory
x = NULL
## Not run: x = read.tped.recode('file.tped')
```

---

sHWE

*Hardy-Weinberg Equilibrium in structure populations*


---

**Description**

Compute structural Hardy-Weinberg Equilibrium (sHWE) p-values on a SNP-by-SNP basis. These p-values can be aggregated to determine genome-wide goodness-of-fit for a particular value of `d`. See [doi:10.1101/240804](https://doi.org/10.1101/240804) for more details.

**Usage**

```
sHWE(X, LF, B, max_iter = 100, tol = 1e-10)
```

**Arguments**

<code>X</code>	A matrix of SNP genotypes, i.e. an integer matrix of 0's, 1's, 2's and NAs. BEDMatrix is supported. Sparse matrices of class Matrix are not supported (yet).
<code>LF</code>	matrix of logistic factors
<code>B</code>	number of null datasets to generate, <code>B = 1</code> is usually sufficient. If computational time/power allows, a few extra <code>B</code> could be helpful
<code>max_iter</code>	Maximum number of iterations for logistic regression
<code>tol</code>	Tolerance value passed to <code>trunc_svd()</code> Ignored if <code>X</code> is a BEDMatrix object.

**Value**

a vector of p-values for each SNP.

**Examples**

```

# get LFs
LF <- lfa(hgdp_subset, 4)
# look at a small (300) number of SNPs for rest of this example:
hgdp_subset_small <- hgdp_subset[ 1:300, ]
gof_4 <- sHWE(hgdp_subset_small, LF, 3)
LF <- lfa(hgdp_subset, 10)
gof_10 <- sHWE(hgdp_subset_small, LF, 3)
hist(gof_4)
hist(gof_10)

```

trunc\_svd

*Truncated singular value decomposition***Description**

Truncated SVD

**Usage**

```

trunc_svd(
  A,
  d,
  adjust = 3,
  tol = .Machine$double.eps,
  override = FALSE,
  force = FALSE,
  maxit = 1000
)

```

**Arguments**

A	matrix to decompose
d	number of singular vectors
adjust	extra singular vectors to calculate for accuracy
tol	convergence criterion
override	TRUE means we use <a href="#">corpcor::fast.svd()</a> instead of the iterative algorithm (useful for small data or very high d).
force	If TRUE, forces the Lanczos algorithm to be used on all datasets (usually <a href="#">corpcor::fast.svd()</a> is used on small datasets or large d)
maxit	Maximum number of iterations

**Details**

Performs singular value decomposition but only returns the first *d* singular vectors/values. The truncated SVD utilizes Lanczos bidiagonalization. See references.

This function was modified from the package *irlba* 1.0.1 under GPL. Replacing the `crossprod()` calls with the C wrapper to `dgemv` is a dramatic difference in larger datasets. Since the wrapper is technically not a matrix multiplication function, it seemed wise to make a copy of the function.

**Value**

list with singular value decomposition. Has elements 'd', 'u', 'v', and 'iter'

**Examples**

```
obj <- trunc_svd( hgdpc_subset, 4 )  
obj$d  
obj$u  
obj$v  
obj$iter
```

# Index

## \* internal

- center, 4
- center-deprecated, 5
- model.gof-deprecated, 8
- read.bed-deprecated, 10
- read.tped.recode-deprecated, 10

- af, 2
- af(), 3
- af\_snp, 3

- center, 4
- center-deprecated, 5
- centerscale, 5
- corpcor::fast.svd(), 7, 9, 12
- crossprod(), 13

- gc(), 3
- genio::read\_plink(), 4

- hgdp\_subset, 6

- lfa, 6, 10
- lfa(), 2, 3
- lfa-deprecated (center), 4

- model.gof (center), 4
- model.gof-deprecated, 8

- pca\_af, 9

- read.bed (center), 4
- read.bed-deprecated, 10
- read.tped.recode (center), 4
- read.tped.recode-deprecated, 10
- RSpectra::svds(), 7

- sHWE, 11
- sHWE(), 4

- trunc\_svd, 12
- trunc\_svd(), 7, 9, 11