

# Package ‘lfa’

March 22, 2023

**Title** Logistic Factor Analysis for Categorical Data

**Version** 1.28.2

**Date** 2015-10-09

**Author** Wei Hao, Minsun Song, John D. Storey

**Maintainer** Wei Hao <whao@princeton.edu>, John D. Storey <jstorey@princeton.edu>

**LazyData** true

**Description** LFA is a method for a PCA analogue on Binomial data via estimation of latent structure in the natural parameter.

**Imports** corpcor

**Depends** R (>= 3.2)

**Suggests** knitr, ggplot2

**VignetteBuilder** knitr

**License** GPL-3

**biocViews** SNP, DimensionReduction, PrincipalComponent

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

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

**RoxygenNote** 6.0.1

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

**git\_branch** RELEASE\_3\_16

**git\_last\_commit** 7572f2f

**git\_last\_commit\_date** 2023-01-19

**Date/Publication** 2023-03-22

## R topics documented:

|        |   |
|--------|---|
| af     | 2 |
| af_snp | 3 |
| center | 3 |

|                            |   |
|----------------------------|---|
| centerscale . . . . .      | 4 |
| hgdp_subset . . . . .      | 4 |
| lfa . . . . .              | 5 |
| model.gof . . . . .        | 6 |
| pca_af . . . . .           | 7 |
| read.bed . . . . .         | 7 |
| read.tped.recode . . . . . | 8 |
| sHWE . . . . .             | 9 |
| trunc.svd . . . . .        | 9 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>11</b> |
|--------------|-----------|

---

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

---

## Description

Compute matrix of individual-specific allele frequencies

## Usage

```
af(X, LF, safety = FALSE)
```

## Arguments

|        |  |
|--------|--|
| X      | a matrix of SNP genotypes, i.e. an integer matrix of 0's, 1's, and 2's. Sparse matrices of class Matrix are not supported (yet). |
| LF     | Matrix of logistic factors, with intercept. Pass in the return value from lfa!   |
| safety | optional boolean to bypass checks on the genotype matrices, which require a non-trivial amount of computation.                   |

## 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)
```

**Arguments**

|     |  |
|-----|--|
| snp | vector of 0's, 1's, and 2's  |
| LF  | Matrix of logistic factors, with intercept. Pass in the return value from lfa! |

**Value**

vector of allele frequencies

---

|        |                         |
|--------|-------------------------|
| center | <i>Matrix centering</i> |
|--------|-------------------------|

---

**Description**

C routine to row-center a matrix

**Usage**

```
center(A)
```

**Arguments**

|   |        |
|---|--------|
| A | matrix |
|---|--------|

**Value**

matrix same dimensions A but row centered

**Examples**

```
center(hgdp_subset)
```

---

|             |                                     |
|-------------|-------------------------------------|
| centerscale | <i>Matrix centering and scaling</i> |
|-------------|-------------------------------------|

---

**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**

```
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 *Logistic factor analysis*

---

**Description**

Fit a factor model of dimension  $d$  for binomial data. Returns logistic factors.

**Usage**

```
lfa(X, d, adjustments = NULL, override = FALSE, safety = FALSE)
```

**Arguments**

|             |  |
|-------------|--|
| X           | a matrix of SNP genotypes, i.e. an integer matrix of 0's, 1's, and 2's. Sparse matrices of class Matrix are not supported (yet). |
| d           | number of logistic factors, including the intercept  |
| adjustments | a matrix of adjustment variables to hold fixed during estimation.  |
| override    | optional boolean to bypass Lanczos bidiagonalization SVD. Usually not advised unless encountering a bug in the SVD code.         |
| safety      | optional boolean to bypass checks on the genotype matrices, which require a non-trivial amount of computation.                   |

**Details**

This function performs logistic factor analysis on SNP data. As it stands, we follow the convention where  $d = 1$  is intercept only, and for  $d > 1$  we compute  $d - 1$  singular vectors and postpend the intercept.

**Value**

matrix of logistic factors, with the intercept at the end.

**Note**

Genotype matrix is expected to be a matrix of integers with values 0, 1, and 2. Note that the coding of the SNPs does not affect the algorithm.

**Examples**

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

---

`model.gof`*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**

|                 |   |
|-----------------|---|
| <code>X</code>  | a matrix of SNP genotypes, i.e. an integer matrix of 0's, 1's, and 2's. Sparse matrices of class <code>Matrix</code> are not supported (yet). |
| <code>LF</code> | matrix of logistic factors  |
| <code>B</code>  | 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.

**Examples**

```
LF <- lfa(hgdp_subset, 4)
gof_4 <- model.gof(hgdp_subset, LF, 3)
LF <- lfa(hgdp_subset, 10)
gof_10 <- model.gof(hgdp_subset, LF, 3)
hist(gof_4)
hist(gof_10)
```

---

pca\_af                      *PCA Allele frequencies*

---

**Description**

Compute matrix of individual-specific allele frequencies via PCA

**Usage**

```
pca_af(X, d, override = FALSE)
```

**Arguments**

X                      a matrix of SNP genotypes, i.e. an integer matrix of 0's, 1's, and 2's. Sparse matrices of class Matrix are not supported (yet).

d                      number of logistic factors, including the intercept

override              optional boolean to bypass Lanczos bidiagonalization SVD. Usually not advised unless encountering a bug in the SVD code.

**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, LF)
summary(abs(allele_freqs_lfa-allele_freqs_pca))
```

---

read.bed                      *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

**Examples**

```
# assuming you have PLINK format HapMap data from: http://pngu.mgh.harvard.edu/~purcell/plink/res.shtml
# run this in the unpacked folder
x = NULL
## Not run: x = read.bed("hapmap_r23a")
```

---

read.tped.recode      *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 = 5e+08)
```

**Arguments**

tped.filename    Path to your .tped file after tranposing 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.

**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 <https://doi.org/10.1101/240804> for more details.

### Usage

```
sHWE(X, LF, B)
```

### Arguments

**X** a matrix of SNP genotypes, i.e. an integer matrix of 0's, 1's, and 2's. Sparse matrices of class Matrix are not supported (yet).

**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

### Value

a vector of p-values for each SNP.

### Examples

```
LF <- lfa(hgdp_subset, 4)
gof_4 <- sHWE(hgdp_subset, LF, 3)
LF <- lfa(hgdp_subset, 10)
gof_10 <- sHWE(hgdp_subset, LF, 3)
hist(gof_4)
hist(gof_10)
```

---

trunc.svd

*Truncated singular value decomposition*

---

### Description

Truncated SVD

### Usage

```
## S3 method for class 'svd'
trunc(A, d, adjust = 3, tol = 1e-10, V = NULL,
      seed = NULL, ltrace = FALSE, override = FALSE)
```

**Arguments**

|          |   |
|----------|---|
| A        | matrix  |
| d        | number of singular vectors  |
| adjust   | extra singular vectors to calculate for accuracy  |
| tol      | convergence criterion   |
| V        | optional initial guess  |
| seed     | seed  |
| ltrace   | debugging output  |
| override | TRUE means we use fast.svd instead of the iterative algorithm (useful for small data or very high d). |

**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.

# Index

af, [2](#)  
af\_snp, [3](#)  
  
center, [3](#)  
centerscale, [4](#)  
  
hgdp\_subset, [4](#)  
  
lfa, [5](#), [8](#)  
  
model.gof, [6](#)  
  
pca\_af, [7](#)  
  
read.bed, [7](#)  
read.tped.recode, [8](#)  
  
sHWE, [9](#)  
  
trunc.svd, [9](#)