Package ‘lfa’

December 21, 2016

Title Logistic Factor Analysis for Categorical Data
Version 1.4.0
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
NeedsCompilation yes

R topics documented:

af ............................................................ 2
af.snp ..................................................... 2
center ..................................................... 3
centerscale .............................................. 3
hgdp_subset ............................................ 4
lfa .......................................................... 4
model.gof ................................................ 5
pca.af ..................................................... 6
read.bed .................................................. 6
read.tped.recode ...................................... 7
trunc.svd ............................................... 8

Index 9
### af

**Allele frequencies**

**Description**

Compute matrix of individual-specific allele frequencies

**Usage**

```r
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**

```r
LF = lfa(hgdpm_subset, 4)
allele_freqs = af(hgdpm_subset, LF)
```

### af_snp

**Allele frequencies for SNP**

**Description**

Computes individual-specific allele frequencies for a single SNP.

**Usage**

```r
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`!
### center

**Value**

vector of allele frequencies

<table>
<thead>
<tr>
<th>center</th>
<th>Matrix centering</th>
</tr>
</thead>
</table>

**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

**Description**

C routine to row-center and scale a matrix

**Usage**

`centerscale(A)`

**Arguments**

A matrix

**Value**

matrix same dimensions A but row centered and scaled

**Examples**

`centerscale(hgdp_subset)`
hgdpsubset  

**HGDP subset**

**Description**
Subset of the HGDP dataset.

**Usage**
hgdpsubset

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

**Value**
genotype matrix

**Source**
Stanford HGDP [http://www.hagsc.org/hgdpsubset.html](http://www.hagsc.org/hgdpsubset.html)

---

lfa  

**Logistic factor analysis.**

**Description**
Logistic factor analysis.

**Usage**
lfa(X, d, override = FALSE, safety = FALSE)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>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).</td>
</tr>
<tr>
<td>d</td>
<td>number of logistic factors, including the intercept</td>
</tr>
<tr>
<td>override</td>
<td>optional boolean to bypass Lanczos bidiagonalization SVD. Usually not advised unless encountering a bug in the SVD code.</td>
</tr>
<tr>
<td>safety</td>
<td>optional boolean to bypass checks on the genotype matrices, which require a non-trivial amount of computation.</td>
</tr>
</tbody>
</table>

**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.
model.gof

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. Currently no support for missing values. 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

LFA model goodness of fit

Usage

model.gof(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 usualy 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(bed.prefix)
```

**Arguments**

- `bed.prefix` Path leading to the bed, bim, and fam files.
### Details

Use plink with --make-bed

### Value

Genotype matrix

### Examples

```r
# 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**

```r
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**

```r
# assuming you have a .tped file in the right directory
x = NULL
## Not run: x = read.tped.recode("file.tped")
```
Truncated singular value decomposition

Usage

```r
## 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. The of 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, 2

center, 3
centerscale, 3

hgdp_subset, 4
lfa, 4, 7
model.gof, 5
pca_af, 6

read.bed, 6
read.tped.recode, 7

trunc.svd, 8