Logistic Factor Analysis Vignette

Wei Hao, Minsun Song, John D. Storey

October 17, 2016

1 Introduction

Logistic Factor Analysis (LFA) [**?**]. Briefly, LFA fits a latent variable model on categorical (i.e. SNP genotypes coded as 0, 1, and 2) data by modelling the logit transformed binomial parameters in terms of latent variables. The resulting "logistic factors" are analagous to principal components, but fit into a convenient likelihood based model. As a result, the logistic factors can power a number of other analyses.

2 Sample usage

We include a sample real dataset with the package as the variable `hgdp_subset`—a small subset of the HGDP genotypes. The row names are the rsids for the SNPs and the column names are coarse geographical labels for the individuals.

```
library(lfa)
dim(hgdp_subset)

## [1] 5000  159
```

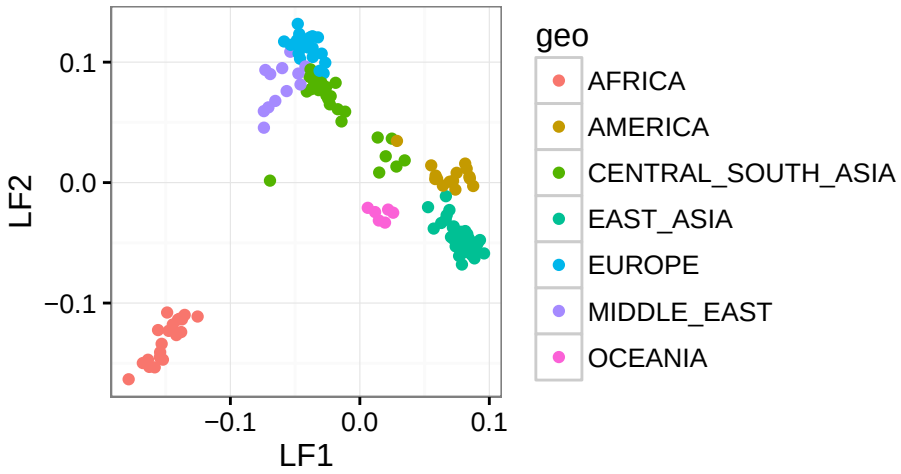
2.1 `lfa`

The `lfa` function has two required arguments. The first is the genotype matrix, and the second is the number of logistic factors including the intercept.

```
LF = lfa(hgdp_subset, 4)
dim(LF)

## [1] 159   4

head(LF)

##              [,1]       [,2]         [,3] [,4]
## [1,] -0.03635636 0.10431465 -0.026157330    1
## [2,] -0.04618695 0.10283828 -0.014107285    1
## [3,] -0.02800495 0.09071295 -0.030261474    1
## [4,] -0.03513497 0.08332765 -0.002122198    1
## [5,] -0.03064114 0.07938743 -0.021696325    1
## [6,] -0.02442314 0.06953981 -0.005312549    1
```

We can plot the first two logistic factors and color by geographical information:

```
dat = data.frame(LF[,1], LF[,2], colnames(hgdp_subset))
colnames(dat) = c("LF1", "LF2", "geo")
library(ggplot2)
ggplot(dat, aes(LF1, LF2, color=geo)) + geom_point() + theme_bw() +
  coord_fixed(ratio=(max(dat[,1])-min(dat[,1]))/(max(dat[,2])-min(dat[,2])))
```



One aspect of `lfa` is that the return value is a matrix of logistic factors, thus, an important part of subsequent analysis is to keep your matrix of logistic factors to pass as an argument.

2.2 af

Given a genotype matrix and logistic factors, the `af` function computes the individual-specific allele frequencies

```
allele_freqs = af(hgdp_subset, LF)
allele_freqs[1:5, 1:5]

##                [,1]      [,2]      [,3]      [,4]      [,5]
## rs4050954  0.6799538 0.7061143 0.6424166 0.6716247 0.6400715
## rs302665   0.2560284 0.2569022 0.2406448 0.2457005 0.2309623
## rs2899446  0.3431056 0.3539195 0.3759817 0.4283358 0.4124516
## rs10069940 0.5453350 0.5380593 0.5287222 0.5023167 0.5093611
## rs6114921  0.3944875 0.4466875 0.3337968 0.3834435 0.3344592
```

Since the calculation is independent at each locus, you can pass a subset of the genotype matrix as an argument if you aren't interested in all the SNPs.

```
subset = af(hgdp_subset[15:25,], LF)
subset[1:5,1:5]

##                  [,1]       [,2]       [,3]      [,4]       [,5]
## rs409169   0.17045449 0.14658345 0.19092932 0.1303421 0.17978531
## rs1530933  0.03441206 0.03465226 0.03969294 0.0412471 0.04427358
## rs1007833  0.53811318 0.55250219 0.53989070 0.5674498 0.55458834
## rs2328383  0.46839267 0.48418147 0.42004022 0.4419977 0.40006848
## rs12637969 0.74369515 0.75969674 0.71407465 0.7317521 0.70772691
```

Given the allele frequencies, you can do some other interesting calculations—for example, compute the log-likelihood for each SNP.

```
ll = function(snp, af){
    -sum(snp*log(af) + (2-snp)*log(1-af))
}
log_lik = sapply(1:nrow(hgdp_subset), function(i) {ll(hgdp_subset[i,], allele_freqs[i,])})
which(max(log_lik) == log_lik)

## [1] 2514
```

3 Data Input

The best way to load genotypes is by using the function `read.bed`, which assumes that you have binary PLINK formatted genotypes. The binary PLINK format uses files: a `.bed` for the genotypes, a `.bim` for the genotype information, and a `.fam` for the individuals information. `read.bed` takes as an argument the prefix for your three files.