VIMCO: Variational Inference for Multiple Correlated Outcomes in Genome-wide Association Studies

Xingjie Shi

2019-11-21

vimco package proves a Bayesian variable selection methods for GWAS data with multiple traits. Unlike in BVSR where each trait is analyzed seperately, **vimco** performs a joint analysis for the multiple traits, while accounting for correlation among the multiple traits.

Installation

To install the development version of **vimco**, it's easiest to use the 'devtools' package. Note that **vimco** depends on the 'Rcpp' package, which also requires appropriate setting of Rtools and Xcode for Windows and Mac OS/X, respectively.

```
library(devtools)
install_github("XingjieShi/VIMCO")
```

Fit VIMCO for simulated data

Simulate data

We first generate genotypes and multiple phenotypes based on the linear modle $\mathbf{Y} = \mathbf{XB} + \mathbf{E}$ as follows:

```
library(mvtnorm)
#> Warning: package 'mvtnorm' was built under R version 3.4.4
n = 300
p = 200
K = 5
set.seed(20132014)
X   = rmvnorm(n, mean=rep(0, p))
sigma.beta = rep(1, K)
bet = matrix(0, nrow = p, ncol = K)
lambda = 0.05
eta = rbinom(p, 1, lambda)
alpha = 1
gam = matrix(rbinom(p*K, 1, alpha), ncol=K)
for (k in 1:K){
  bet[, k] = rnorm(p, mean = 0, sd = sigma.beta[k]) * gam[,k] * eta
}

sigma = diag(rep(1, K))
lp  = X %*% bet
sigma.e = diag(sqrt(diag(var(lp)))) %*% sigma %*% diag(sqrt(diag(var(lp))))
err = rmvnorm(n, rep(0, K), sigma.e)
Y   = lp + err
```

Fit BVSR

```
library(vimco)
fit_Ind = emInd(X, Y)
str(fit_Ind)
#> List of 13
#>  $ N         : int 300
#>  $ mu        : num [1:200, 1:5] -0.2894 0.2219 -0.0696 -0.1424 0.1411 ...
#>  $ s         : num [1:200, 1:5] 0.0482 0.0401 0.0458 0.0422 0.0478 ...
#>  $ Beta0     : num [1, 1:5] 0.0118 0.2778 -0.276 -0.0383 -0.2078
#>  $ Beta      : num [1:200, 1:5] -0.01958 0.01083 -0.00211 -0.00498 0.00508 ...
#>  $ Alpha     : num [1:200, 1:5] 0.0677 0.0488 0.0303 0.0349 0.036 ...
#>  $ fdr       : num [1:200, 1:5] 0.932 0.951 0.97 0.965 0.964 ...
#>  $ sigb      : num [1:5, 1] 0.701 2.746 0.526 1.917 1.225
#>  $ sige      : num [1:5, 1] 14.4 19.6 11.9 22.5 16.3
#>  $ alpha     : num [1:5, 1] 0.1041 0.0399 0.0998 0.0487 0.0733
#>  $ iter      : int 22
#>  $ eps       : num 8.28e-06
#>  $ LowerBound: num [1:22, 1] -3129 -3098 -3093 -3091 -3090 ...
```

Fit VIMCO

```
fit_Mul = emMultiple(X, Y)
str(fit_Mul)
#> List of 16
#>  $ N         : int 300
#>  $ mu        : num [1:200, 1:5] -0.305 0.1963 -0.0981 -0.1667 0.1623 ...
#>  $ s         : num [1:200, 1:5] 0.0479 0.0398 0.0455 0.0419 0.0475 ...
#>  $ Beta0     : num [1, 1:5] 0.0157 0.2808 -0.2758 -0.0635 -0.2094
#>  $ Beta      : num [1:200, 1:5] -0.02057 0.00765 -0.00283 -0.00575 0.00564 ...
#>  $ Alpha     : num [1:200, 1:5] 0.0675 0.039 0.0288 0.0345 0.0348 ...
#>  $ alpha     : num [1:5, 1] 0.0984 0.0386 0.0975 0.0477 0.0722
#>  $ Lambda    : num [1:200, 1] 1 1 1 1 1 1 1 1 1 1 ...
#>  $ lambda    : num 1
#>  $ fdr       : num [1:200, 1:5] 0.933 0.961 0.971 0.966 0.965 ...
#>  $ sigb      : num [1:5, 1] 0.76 2.867 0.551 1.898 1.212
#>  $ Sigma     : num [1:5, 1:5] 14.429 0.822 -0.873 0.933 0.951 ...
#>  $ Theta     : num [1:5, 1:5] 0.07043 -0.00379 0.00637 -0.00324 -0.00472 ...
#>  $ iter      : int 18
#>  $ eps       : num 7.16e-06
#>  $ LowerBound: num [1:18, 1] -3163 -3087 -3082 -3081 -3080 ...
```

Fit VIMCO using initial values from BVSR

```
p = ncol(X)
mu0     = fit_Ind$mu
sigb0   = fit_Ind$sigb
Theta0  = matrix(0, nrow=ncol(Y), ncol=ncol(Y))
diag(Theta0)  =   1/c(fit_Ind$sige)
Lambda0 = rep(1, p)
Alpha0  = fit_Ind$Alpha
```

```
lambda0 = 1
fit_Mul_Ini = emMultiple(X, Y, mu0, sigb0, Theta0, Lambda0, Alpha0, TRUE)
str(fit_Mul_Ini)
#> List of 16
#>  $ N         : int 300
#>  $ mu        : num [1:200, 1:5] -0.305 0.196 -0.098 -0.167 0.162 ...
#>  $ s         : num [1:200, 1:5] 0.0479 0.0398 0.0455 0.0419 0.0475 ...
#>  $ Beta0     : num [1, 1:5] 0.0157 0.2808 -0.2758 -0.0635 -0.2094
#>  $ Beta      : num [1:200, 1:5] -0.0206 0.00766 -0.00283 -0.00576 0.00565 ...
#>  $ Alpha     : num [1:200, 1:5] 0.0676 0.039 0.0289 0.0346 0.0348 ...
#>  $ alpha     : num [1:5, 1] 0.0985 0.0386 0.0976 0.0477 0.0722
#>  $ Lambda    : num [1:200, 1] 1 1 1 1 1 1 1 1 1 1 ...
#>  $ lambda    : num 1
#>  $ fdr       : num [1:200, 1:5] 0.932 0.961 0.971 0.965 0.965 ...
#>  $ sigb      : num [1:5, 1] 0.759 2.867 0.551 1.899 1.212
#>  $ Sigma     : num [1:5, 1:5] 14.427 0.822 -0.873 0.933 0.951 ...
#>  $ Theta     : num [1:5, 1:5] 0.07044 -0.00379 0.00637 -0.00324 -0.00472 ...
#>  $ iter      : int 13
#>  $ eps       : num 6.97e-06
#>  $ LowerBound: num [1:13, 1] -3081 -3080 -3080 -3080 -3080 ...
```

Fit VIMCO for GWAS data with multiple traits

To handle GWAS data that is in a *PLINK* format (incudes sim.bed, sim.bim, sim.fam) directly, **vimco** package provides a function `vimco::vimco()`, check its help file for more details. Here we provide a simple examples. The following 3 *PLINK* files are already provided in the packages:

- sim.bed
- sim.bim
- sim.fam

```
path <- system.file(package = "vimco")
setwd(path)
stringname <- "sim"
tmp <- vimco(stringname, nPheno = 4, fit = TRUE)
```

Replicate simulation results in Shi et al. (2018)

All the simulation results can be reproduced by using the code at simulation. Before running simulation to reproduce the results, please familarize yourself with **vimco** using 'VIMCO' vignette. Simulation results can be reproduced by following steps:

- simulation.r This function can be run in a PC, it will output files, named h_0.1_g_0_rhoE_0.2_rhoX_0.8, which contain inference results of each replicate, for all methods: VIMCO, BVSR, sLMM and mvLMM.
- summary.r This function read all outputs from simulation.r and calculate Power, FDR and AUC. It will output all summary results in *rds* format, e.g. h_0.1_g_0_rhoE_0.2_rhoX_0.8.rds.
- barplot.r This function produce all figures in Shi et al. (2018).

Reference

VIMCO: Variational Inference for Multiple Correlated Outcomes in Genome-wide Association Studies