CoMM: a collaborative mixed model to dissecting genetic contributions to complex traits by leveraging regulatory information

Jin Liu

2019-04-13

Introduction

This vignette provides an introduction to the `CoMM` package. R package `CoMM` implements CoMM, a collaborative mixed model to dissecting genetic contributions to complex traits by leveraging regulatory information. The package can be installed with the command:

```
library(devtools)
```

```
install_github("gordonliu810822/CoMM")
```

The package can be loaded with the command:

```
library("CoMM")
```

Fit CoMM using simulated data

We first generate genotype data using function *genRawGeno*:

```
library(mvtnorm)
#> Warning: package 'mvtnorm' was built under R version 3.4.4
L = 1; M = 100; rho =0.5
n1 = 350; n2 = 5000;
maf = runif(M,0.05,0.5)
X = genRawGeno(maf, L, M, rho, n1 + n2);
```

Then, effect sizes are generated from standard Gaussian distribution with sparse structure:

```
beta_prop = 0.2;
b = numeric(M);
m = M * beta_prop;
b[sample(M,m)] = rnorm(m);
```

Subsequently, the gene expression `y` is generated by controlling cellular heritability at prespecified level (`h2y`):

```
h2y = 0.05;
b0 = 6;
y0 <- X%*%b + b0;
y  <- y0 + (as.vector(var(y0)*(1-h2y)/h2y))^0.5*rnorm(n1+n2);
```

Finally, the phenotype data is generated as the generative model of CoMM with a prespecified trait heritability (`h2`) as:

```
h2 = 0.001;
y1 <- y[1:n1]
X1 <- X[1:n1,]
y2 <- y0[(n1+1):(n1+n2)]
X2 <- X[(n1+1):(n1+n2),]
```

```r
alpha0 <- 3
alpha <- 0.3
sz2 <- var(y2*alpha) * ((1-h2)/h2)
z <- alpha0 + y2*alpha + rnorm(n2,0,sqrt(sz2))
```

The genotype data `X1` and `X2` are normalized as

```r
y = y1;
mean.x1 = apply(X1,2,mean);
x1m = sweep(X1,2,mean.x1);
std.x1 = apply(x1m,2,sd)
x1p = sweep(x1m,2,std.x1,"/");
x1p = x1p/sqrt(dim(x1p)[2])

mean.x2 = apply(X2,2,mean);
x2m = sweep(X2,2,mean.x2);
std.x2 = apply(x2m,2,sd)
x2p = sweep(x2m,2,std.x2,"/");
x2p = x2p/sqrt(dim(x2p)[2])

w2 = matrix(rep(1,n2),ncol=1);
w1 = matrix(rep(1,n1),ncol=1);
```

Initilize the parameters by using linear mixed model (function *lmm_pxem*, LMM implemented ($n < p$) using PX-EM algorithm, function *lmm_pxem2*, LMM implemented ($n > p$)):

```r
fm0 = lmm_pxem2(y, w1,x1p, 100)
sigma2beta =fm0$sigma2beta;
sigma2y =fm0$sigma2y;
beta0 = fm0$beta0;
```

Fit CoMM w/ and w/o constraint that alpha $= 0$ as

```r
fmHa = CoMM_covar_pxem(y, z, x1p, x2p, w1, w2,constr = 0);
fmH0 = CoMM_covar_pxem(y, z, x1p, x2p, w1, w2,constr = 1);
loglikHa = max(fmHa$loglik,na.rm=T)
loglikH0 = max(fmH0$loglik,na.rm=T)
tstat = 2 * (loglikHa - loglikH0);
pval = pchisq(tstat,1,lower.tail=F)
alpha_hat = fmHa$alpha
```

Fit CoMM using GWAS and eQTL data

The example of running CoMM using GWAS and eQTL data in plink binary format

```r
file1 = "1000G.EUR.QC.1";
file2 = "NFBC_filter_mph10";
file3 = "Geuvadis_gene_expression_qn.txt";
file4 = "";
file5 = "pc5_NFBC_filter_mph10.txt";
whichPheno = 1;
bw = 500000;
```

Here, file1 is the prefix for eQTL genotype data in plink binary format, file2 is the GWAS data in plink binary format, file3 is the gene expression file with extended name, file4 and file5 are covariates file for eQTL and GWAS data, respectively. Then run `fm = CoMM_testing_run(file1,file2,file3, file4,file5,` `whichPheno, bw);`. For gene expresion file, it must have the following format (rows for genes and columns for individuais and note that it must be tab delimited):

lower	up	genetype1	genetype2	TargetID	Chr	HG00105	HG00115
59783540	59843484	lincRNA	PART1	ENSG00000152931.6	5	0.5126086	0.7089508
48128225	48148330	protein_coding	UPP1	ENSG00000183696.9	7	1.4118007	-0.0135644
57846106	57853063	protein_coding	INHBE	ENSG00000139269.2	12	0.5755268	-1.0162217
116054583	116164515	protein_coding	AFAP1L2	ENSG00000169129.8	10	1.1117776	0.0407033
22157909	22396763	protein_coding	RAPGEF5	ENSG00000136237.12	7	0.2831573	-0.1772559
11700964	11743303	lincRNA	RP11-434C1.1	ENSG00000247157.2	12	0.2550282	-0.2831573

To make 'CoMM' further speeding, we implement multiple thread version of 'CoMM' by just run `fm = CoMM_testing_run_mt(file1,file2,file3, file4,file5, whichPheno, bw, coreNum);` where `coreNum = 24` is the number of cores in your CPU.

Figures

The following data and codes are used to produce one of the figures in the Yang et al. (2018).

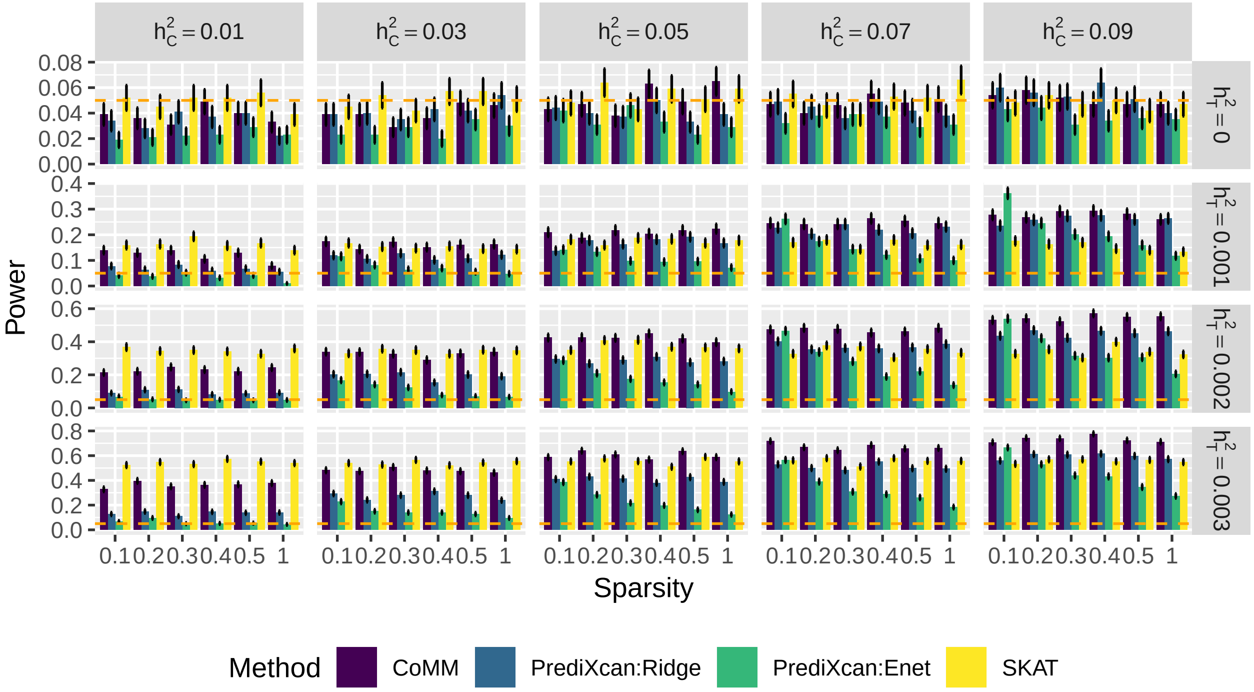
```
dat_rej = dat[[3]];
dat_rej$h2z=paste("",dat_rej$h2,sep="")
dat_rej$Power = dat_rej$rej_prop
dat_rej$Sparsity = dat_rej$beta_prop
dat_rej$sd_rej = as.numeric(as.character(dat_rej$sd_rej))
dat_rej = dat_rej[dat_rej$Method!="2-stage:AUDI",]
library(plyr)
dat_rej$Method=revalue(dat_rej$Method, c("AUDI"="CoMM"))
dat_rej$Method=revalue(dat_rej$Method, c("2-stage:Ridge"="PrediXcan:Ridge"))
dat_rej$Method=revalue(dat_rej$Method, c("2-stage:Enet"="PrediXcan:Enet"))
dat_rej$Method=droplevels(dat_rej$Method)

rho = 0.5; n2 = 8000;
t1e_rej = dat_rej[dat_rej$RhoX==rho&dat_rej$n2==n2,]

t1e_rej$h2z = factor(t1e_rej$h2z)
t1e_rej$h2y = factor(t1e_rej$h2y)
t1e_rej$Sparsity = factor(t1e_rej$Sparsity)
t1e_rej$n2 = factor(t1e_rej$n2)
t1e_rej$Method <- ordered(t1e_rej$Method, levels = c("CoMM","PrediXcan:Ridge","PrediXcan:Enet","SKAT"))
t1e_rej$Power = as.numeric(as.character((t1e_rej$Power)))

t1e_rej$h2y2 <- factor(t1e_rej$h2y, labels = c("h[C]^2==0.01", "h[C]^2==0.03",
                        "h[C]^2==0.05", "h[C]^2==0.07", "h[C]^2==0.09"))
t1e_rej$h2z2 <- factor(t1e_rej$h2z, labels = c("h[T]^2==0", "h[T]^2==0.001",
                        "h[T]^2==0.002", "h[T]^2==0.003"))

library(ggplot2)
#> Warning: package 'ggplot2' was built under R version 3.4.4
ggplot(t1e_rej, aes(x = Sparsity, y = Power,fill = Method))+
  geom_bar(stat="identity", position=position_dodge())+
  geom_errorbar(aes(ymin=Power-sd_rej, ymax=Power+sd_rej), width=.2,
                position=position_dodge(.9)) +
  facet_grid(h2z2~h2y2,labeller = label_parsed,scales = "free_y") +
  geom_hline(yintercept=0.05,colour="orange",linetype="dashed")+
  theme(legend.position="bottom")
```



Corrections for CoMMs (Yang et al.)

In Algorithm 1 (in the supplementary document), the Reduction-step should be $\left(\sigma_u^{(t+1)}\right)^2 = \left(\gamma^{(t+1)}\right)^2 \left(\sigma_u^{(t+1)}\right)^2$.

Fit CoMM_S2 using simulated data

We first generate genotype data using function *genRawGeno*:

```
library(mvtnorm)
set.seed(1000)
L = 1; M = 100; rho =0.5
n1 = 400; n2 = 5000; n3 = 400;
maf = runif(M, min = 0.05, max = 0.5);
X = genRawGeno(maf, L, M, rho, n1 + n2);
X3 = genRawGeno(maf, L, M, rho, n3)
```

Then, effect sizes are generated from standard Gaussian distribution with sparse structure:

```
beta_prop = 0.2;
b = numeric(M);
m = M * beta_prop;
b[sample(M,m)] = rnorm(m);
```

Subsequently, the gene expression `y` is generated by controlling cellular heritability at prespecified level (`h2y`):

```r
h2y = 0.05;
b0 = 6;
y0 <- X%*%b + b0;
y  <- y0 + (as.vector(var(y0)*(1-h2y)/h2y))^0.5*rnorm(n1+n2);
```

Finally, the phenotype data is generated as the generative model of CoMM with a prespecified trait heritability (h2) as:

```r
h2 = 0.001;
y1 <- y[1:n1]
X1 <- X[1:n1,]
y2 <- y0[(n1+1):(n1+n2)]
X2 <- X[(n1+1):(n1+n2),]
alpha0 <- 3
alpha <- 0.3
sz2 <- var(y2*alpha) * ((1-h2)/h2)
z <- alpha0 + y2*alpha + rnorm(n2,0,sqrt(sz2))
```

The genotype data `X1`, `X2` and `X3` are centered as

```r
y = y1;
mean.x1 = apply(X1,2,mean);
x1p = sweep(X1,2,mean.x1);

mean.x2 = apply(X2,2,mean);
x2p = sweep(X2,2,mean.x2);

mean.x3 = apply(X3,2,mean);
x3p = sweep(X3,2,mean.x3);

w = matrix(rep(1,n1),ncol=1);
```

The summary statistics are generated from GWAS individual data

```r
hatmu = matrix(0, M, 1)
hats = matrix(0, M, 1)

for (m in 1:M){
  fm = lm(z~1+x2p[,m]);
  hatmu[m] = summary(fm)$coefficients[2,1]
  hats[m] = summary(fm)$coefficients[2,2];
}
```

The correlation matrix reflecting LD information is estimated using reference panel

```r
lam = 0.8
sumx3p = apply(x3p*x3p, 2, sum)
R = matrix(0, M, M);
for (i1 in 1:M){
  for (j1 in 1:M){
    R[i1,j1] = t(x3p[,i1])%*%x3p[,j1]/sqrt(sumx3p[i1]*sumx3p[j1])
  }
}
R = R*lam + (1 - lam)*diag(M)
```

The likelihood ratio test is implemented

```
px = 1
opts = list(max_iter = 10000, dispF = 1, display_gap = 10, epsStopLogLik = 1e-5, fix_alphag = 0);
opts1 = list(max_iter = 10000, dispF = 1, display_gap = 10, epsStopLogLik = 1e-5, fix_alphag = 1);

fmHa = CoMM_S2(x1p, y, w, hatmu, hats, R, opts, px);
#> ***Iteration*******Fnew*******Fold*********Diff***
#>    1.0000e+01  -1.1934e+03  -1.1934e+03   1.3005e-02
fmH0 = CoMM_S2(x1p, y, w, hatmu, hats, R, opts1, px);
#> ***Iteration*******Fnew*******Fold*********Diff***
#>    1.0000e+01  -1.1989e+03  -1.1989e+03   1.6075e-03

stat = 2*(fmHa$LRLB - fmH0$LRLB)
pval = pchisq(stat, 1, lower.tail = F)
str(fmHa)
#> List of 7
#>  $ vardist_mu: num [1:100, 1] -0.1061 -0.1586 -0.0563 -0.08 -0.2769 ...
#>  $ sigma2mu  : num 0.2
#>  $ alphag    : num 0.749
#>  $ sigma2beta: num 0.329
#>  $ sigma2y   : num 105
#>  $ LRLB      : num -1276
#>  $ Lq        : num [1, 1:19] -1375 -1203 -1197 -1195 -1194 ...
str(fmH0)
#> List of 7
#>  $ vardist_mu: num [1:100, 1] -0.6199 0.0629 -0.0129 0.1128 -0.2567 ...
#>  $ sigma2mu  : num 0.221
#>  $ alphag    : num 0
#>  $ sigma2beta: num 0.329
#>  $ sigma2y   : num 105
#>  $ LRLB      : num -1282
#>  $ Lq        : num [1, 1:16] -1377 -1206 -1201 -1199 -1199 ...
print(stat)
#> [1] 11.9037
print(pval)
#> [1] 0.0005602251
```

The output of CoMM_S2 is a list with 7 variables, mean of variational distribution `vardist_mu`, variance component `sigma2mu`, gene effect size `alphag`, variance component `sigma2y`, calibrated ELBO `LRLB`, original ELBO `Lq`.

Fit CoMM_S2 using GWAS and eQTL data

The example of running CoMM_S2 using GWAS summary statistics and eQTL data in plink binary format

```
file1 = "1000G.EUR.QC.1";
file2 = "NFBC_beta_se_TG.txt"
file3 = "1000G_chr_all";
file4 = "Geuvadis_gene_expression_qn.txt";
file5 = "";
bw = 500000;
lam = 0.95;
coreNum = 24;
```

Here, file1 is the prefix for eQTL genotype data in plink binary format, file2 is the GWAS summary data, file3 is the prefix for reference panel data in plink binary format, file4 is the gene expression file with extended name, file5 are covariates file for eQTL data. bw is the number of downstream and upstream SNPs that are considered as cis-SNP within a gene. lam is the shirnkage intensify for reference panel. coreNum is the number of cores in parallel. Then run `fm = CoMM_S2_testing(file1, file2, file3, file4, file5, bw, lam);`. For GWAS summary data file, it must have the following format (note that it must be tab delimited):

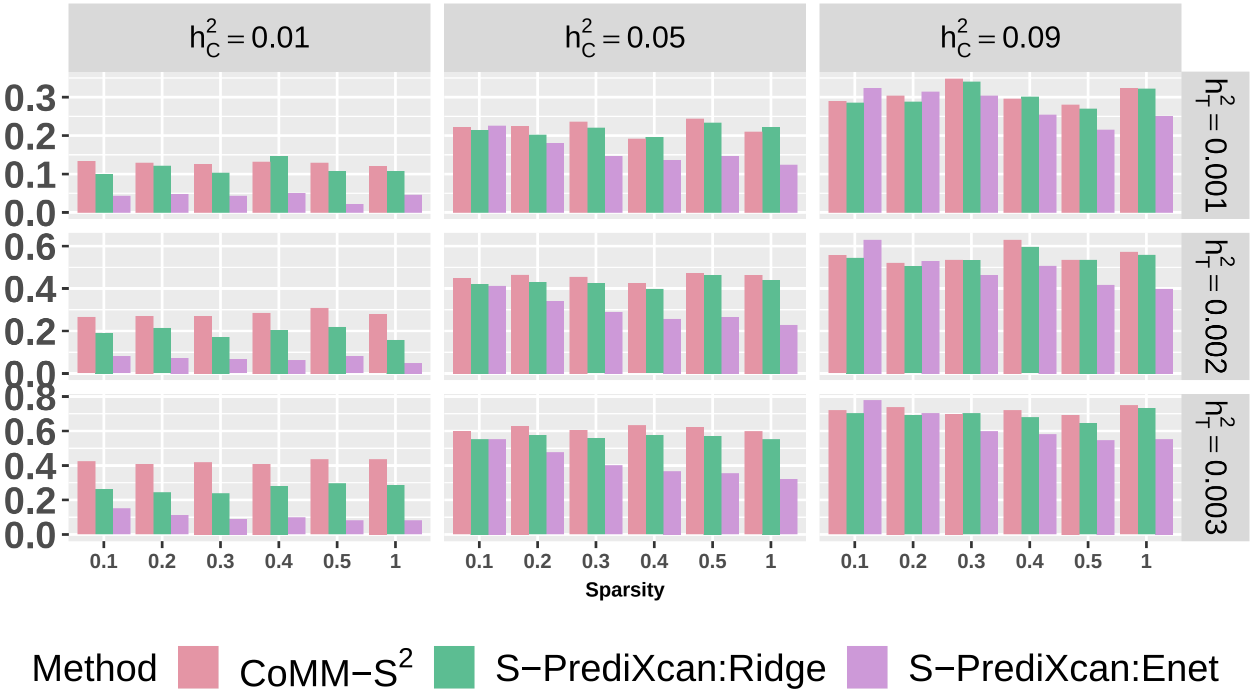
SNP	chr	BP	A1	A2	beta	se
rs3094315	1	752566	G	A	-0.0122	0.0294
rs3128117	1	944564	C	T	-0.0208	0.0278
rs1891906	1	950243	C	A	-0.0264	0.0260
rs2710888	1	959842	T	C	-0.0439	0.0297
rs4970393	1	962606	G	A	-0.0252	0.0233
rs7526076	1	998395	A	G	-0.0512	0.0229
rs4075116	1	1003629	C	T	-0.0497	0.0220
rs3934834	1	1005806	T	C	0.0364	0.0256
rs3766192	1	1017197	C	T	-0.0116	0.0178
rs3766191	1	1017587	T	C	0.0318	0.0262

To make 'CoMM_S2' further speeding, we implement multiple thread version of 'CoMM_S2' by just run `fm = CoMM_S2_paral_testing(file1, file2, file3, file4, file5, bw, lam, coreNum);`

Figures

The following data and codes are used to produce the barplot of power

```
library(ggplot2)
library(colorspace)
bp2 <- ggplot(pval2, aes(x=Sparsity, y=Power, fill=Method)) +
    geom_bar(stat="identity", position=position_dodge()) +
    facet_grid(h2~hc, scales = "free", labeller = label_parsed)  +
    theme(strip.text.x = element_text(size=12, color="black",
                                      face="bold"),
          strip.text.y = element_text(size=12, color="black",
                                      face="bold"),
          plot.title = element_text(size=20,face = "bold",hjust=0.5),
          axis.title.x = element_text(size=8,face = "bold"),
          axis.text.x = element_text(size=8,face = "bold"),
          axis.title.y = element_blank(),
          axis.text.y = element_text(size=15,face = "bold"),
          legend.position="bottom",
          legend.title=element_text(size=15),
          legend.text=element_text(size=15))
  colours<-rainbow_hcl(3, start = 0, end = 300)
  bp2 = bp2 + scale_fill_manual(values=colours, labels=expression("CoMM-S"^2,"S-PrediXcan:Ridge","S-Pre
bp2
```


$h_C^2 = 0.01$
$h_C^2 = 0.05$
$h_C^2 = 0.09$
$h_T^2 = 0.001$
$h_T^2 = 0.002$
$h_T^2 = 0.003$
0.0
0.1
0.2
0.3
0.4
0.6
0.8
0.1
0.2
0.3
0.4
0.5
1
Sparsity
Method
CoMM−S^2
S−PrediXcan:Ridge
S−PrediXcan:Enet