MR-LDP: a two-sample Mendelian randomization for GWAS summary statistics accounting linkage disequilibrium and horizontal pleiotropy

Qing Cheng and Jin Liu

2023-11-22

Introduction

This vignette provides an introduction to the `MR.LDP` package. R package `MR.LDP` implements MR-LDP, a two-sample Mendelian randomization for GWAS summary statistics accounting linkage disequilibrium and horizontal pleiotropy. The package can be installed with the following commands:

```
library(devtools);
install_github("QingCheng0218/MR.LDP");
```

Load the package using the following command:

```
library(MR.LDP);
```

Fit MR-LDP using simulated data

We first generate genotype data using function *genRawGeno*:

```
library("mvtnorm");
library("PDSCE");

set.seed(2019);

rho = 0.4;L = 1; M = 50; p = M*L; m = p; Alrate = 1;
n1 = 20000; n2 = 20000; n3 = 2500; lam = 0.055;
maf = runif(p, 0.05, 0.5);
G = genRawGeno(maf, L, M, rho, n1 + n2 + n3);

G1 = G[1:n1,];
G2 = G[(n1+1):(n1+n2),];
G12 = G[1:(n1+n2),];
G3 = G[(n1+n2+1):(n1+n2+n3),];
```

Estimate the covariance matrix using function *pdsoft*:

```
R0 = cor(G3);
R = pdsoft(R0, lam)$theta;
diag(R) = rep(1, p);
mask = kronecker(diag(L),matrix(1,M,M));
R = R*mask;
```

Generate the exposure data(y) and outcome data(z) with prespecified indirect(h_y^2) and direct(h_z^2) heritability

based on

$$\mathbf{y} = \mathbf{G}_1\gamma + \mathbf{U}_x\eta_x + \mathbf{e}_1, \quad \mathbf{z} = \beta_0\mathbf{x} + \mathbf{G}_2\alpha + \mathbf{U}_y\eta_y + \mathbf{e}_2,$$

```
h2z <- 0.05; h2y <- 0.1; b0 <- 0.1; q <- 50;
u = matrix(rnorm( (n1+n2) * q),ncol=q);

sigma2g <- 0.005;
gamma.nz = rnorm(m)*sqrt(sigma2g);
indx = sample(1:p,m);
gamma = numeric(p);
gamma[indx] = gamma.nz;

Su = matrix(c(1,0.8,0.8,1),nrow=2)
bu = rmvnorm(q,mean=rep(0,2), sigma = Su,method="chol")
by = bu[,1]; bz = bu[,2];
uby = u%*%by; ubz = u%*%bz;
uby = uby/sqrt(as.numeric(var(uby)/0.6));
ubz = ubz/sqrt(as.numeric(var(ubz)/0.2));

G12g = G12%*%gamma;

if(b0!=0){
  h2ga = (h2y *( 1 + b0^2))/(b0^2 * (1 - h2y));
  gamma0 = gamma/sqrt(as.numeric(var(G12g)/h2ga));
  G12g = G12%*%gamma0;
}

yall = G12g + uby + rnorm(n1+n2)*as.numeric(sqrt(1-var(uby)));

# The direct effects on Z
h2yb = var(b0*yall);
h2al = (h2z + h2z*h2yb)/(1 - h2z)

sigma2a <- 0.005;
if(h2z==0){
  alpha0 = rep(0, m);
  G12a = G12%*%alpha0;
}else{
  alno = floor(p*Alrate);
  alpha.nz <- rnorm(alno)*sqrt(sigma2a);
  # sparse setting for pleiotropy
  indxAL = sample(1:p,alno);
  alpha = numeric(p);
  alpha[indxAL] = alpha.nz;

  G12a = G12%*%alpha;
  alpha0 = alpha/sqrt(as.numeric(var(G12a)/(h2al)));
  G12a = G12%*%alpha0;
}

resz = ubz + rnorm(n1+n2)*as.numeric(sqrt(1-var(ubz)));
zall = b0*yall  + G12a +  resz;
H2a.res <- var(G12a)/var(zall);
```

```
H2g.res <- var(b0*G12g)/var(zall);

y = yall[1:n1];
z = zall[(n1+1):(n1+n2)];
```

We then conduct single-variant analysis to obtain the summary statistics.

```
gammah = numeric(p); Gammah = numeric(p);
segamma = numeric(p); seGamma = numeric(p);
pval = numeric(p);
for (i in 1:p){
  fm = lm(y~1+G1[,i]);
  gammah[i] = summary(fm)$coefficients[2,1];
  segamma[i] = summary(fm)$coefficients[2,2];
  pval[i] = summary(fm)$coefficients[2,4];

  fm = lm(z~1+G2[,i]);
  Gammah[i] = summary(fm)$coefficients[2,1];
  seGamma[i] = summary(fm)$coefficients[2,2];
}
```

Until now, we obtain the summary statistics: `gammah` and `segamma` for esposure data, `Gammah` and `seGamma` for outcome data.

Initilize the parameters for MR-LDP algorithm. `epsStopLogLik` is the convergence tolerance, `maxIter` is the iteration number. `beta0, gamma, alpha, sgga2, sgal2` are the initial values for the PX-VBEM algorithm.

```
epsStopLogLik <- 1e-7; maxIter <- 10000;
beta0 <- 0;
gamma <- rep(0, p);
alpha <- rep(0, p);
sgga2 <- 0.01;
sgal2 <- 0.01;
```

We conduct the simulation study using `MRLDP_SimPXvb`, `model = 1` and `model = 2` represent MR-LD and MR-LDP, respectively.

Fit MR-LD w/ (`constr = 1`) and w/o (`constr = 0`) constraint that $\beta = 0$ as:

```
SimMRLD_Hb = MRLDP_SimPXvb(gammah, Gammah, segamma, seGamma,
                      gamma, alpha,  beta0, sgga2, sgal2, R,
                      constr = 0, epsStopLogLik, maxIter, model = 1);

SimMRLD_H0 = MRLDP_SimPXvb(gammah, Gammah, segamma, seGamma,
                      gamma, alpha,  beta0, sgga2, sgal2, R,
                      constr = 1, epsStopLogLik, maxIter, model = 1);

tstat = 2*(SimMRLD_Hb$tstat - SimMRLD_H0$tstat);
pval = pchisq(tstat, 1, lower.tail = F);
beta_hat = SimMRLD_Hb$beta0;
se_hat = abs(beta_hat/sqrt(tstat));
```

Fit MR-LDP w/ (`constr = 1`) and w/o (`constr = 0`) constraint that $\beta = 0$ as:

```
SimMRLDP_Hb = MRLDP_SimPXvb(gammah, Gammah, segamma, seGamma,
                      gamma, alpha,  beta0, sgga2, sgal2, R,
                      constr = 0, epsStopLogLik, maxIter, model = 2);
```

```
SimMRLDP_H0 = MRLDP_SimPXvb(gammah, Gammah, segamma, seGamma,
                        gamma, alpha,  beta0, sgga2, sgal2, R,
                        constr = 1, epsStopLogLik, maxIter, model = 2);

tstat = 2*(SimMRLDP_Hb$tstat - SimMRLDP_H0$tstat);
pval = pchisq(tstat, 1, lower.tail = F);
beta_hat = SimMRLDP_Hb$beta0;
se_hat = abs(beta_hat/sqrt(tstat));
```

beta_hat, se_hat, pval are estimated causal effect, corresponding standard error and p-value of beta_hat.

Fit MR-LDP using CAD-CAD study.

Furthermore, we give an example to illustrate the implements of MR.LDP for real data analysis. The following datasets('heart attack_myocardial infarction.txt', 'c4d.txt', 'cardiogram.txt', 'all_chr_1000G.bed', 'all_chr_1000G.fam', 'all_chr_1000G.bim', 'fourier_ls-all.bed') should be prepared. Download here https://drive.google.com/drive/folders/1IAs3daG9TIvjnneR32j1pfV0niz8PHyu.

```
filescreen= "heart attack_myocardial infarction.txt";
fileexposure = "c4d.txt";
fileoutcome = "cardiogram.txt";
stringname3 = "all_chr_1000G";
block_file = "fourier_ls-all.bed"
```

'filescreen', 'fileexposure', 'fileoutcome' are the datasets names for screen, exposure and outcome, respectively. These three datasets must have the following format(note that it must be tab delimited):

SNP	chr	BP	A1	A2	beta	se	pvalue
rs3094315	1	752566	A	C	0.00012546	0.00042437	0.76750
rs3131969	1	754182	G	A	0.00033099	0.00045415	0.46611
rs3131972	1	752721	G	A	0.00010445	0.00042414	0.80548
rs1048488	1	760912	T	C	0.00017441	0.00042195	0.67935
rs12562034	1	768448	A	C	-0.00003632	0.00049399	0.94138

'stringname3' is the name of reference panel data. Here we use samples from '1000 Genome Project Phase 1' which is in plink binary format. 'block_file' is used to partition the whole genome into blocks.

matchscreen function is used to match the three datasets with a cutoff named 'pva_cutoff'. 'matchExp = TRUE' means we fix the direction of exposure data. Since MR-LDP is invariant to the orientation of genetic variants, 'matchExp' does not affect the results. Default is FALSE.

```
pva_cutoff = 1e-4;
scrres = matchscreen(filescreen, fileexposure, fileoutcome, stringname3,
                     pva_cutoff, matchExp = FALSE)
bh1 = as.numeric(scrres$bh1);
bh2 = as.numeric(scrres$bh2);
s12 = as.numeric(scrres$s12);
s22 = as.numeric(scrres$s22);
chr = as.numeric(scrres$chr);
bp = scrres$bp;
rsname = scrres$rsname
avbIndex = scrres$idxin;
idx4panel = scrres$idx4panel;
```

`bh1`and `s12` are the SNP effects and corresponding standard errors on the exposure variable, `bh2`and `s22` are the SNP effects and corresponding standard errors on the outcome variable. After matching the three datasets, we obtain 'chr'(chromosome number) , 'bp'(base position),'rsname'(rs number).'avbIndex'(location) and 'idx4panel'(Indicators to be adjusted in reference panel data).

One can using the following function *summaryQC* to remove the MHC region(QCindex = 1), or skip the procedure(QCindex = 0).

```
QCindex = 1;
if(QCindex){
  QCresult = summaryQC(mhcstart, mhcend, bh1, bh2, s12, s22, bp,
                       chr, rsname, avbIndex, idx4panel, Inf, Inf)
  bh1new = QCresult$bh1new;
  bh2new = QCresult$bh2new;
  s12new = QCresult$s12new;
  s22new = QCresult$s22new;
  bpnew = QCresult$bpnew;
  chrnew = QCresult$chrnew;
  avbIndexnew = QCresult$avbIndexnew;
  idx4panelnew = QCresult$idx4panel
  rsnamenew = QCresult$rsnamenew;
}else{
  bh1new = bh1;
  bh2new = bh2;
  s12new = s12;
  s22new = s22;
  bpnew = bp;
  chrnew = chr;
  rsnamenew = rsname;
  idx4panelnew = idx4panel;
  avbIndexnew = avbIndex;
}

p = length(avbIndexnew);
```

Initilize the parameters for MR-LDP algorithm. CoreNum is the number of cores in your CPU.

```
gamma = rep(0.01, p);
alpha = rep(0.01, p);
sgga2 =  0.01;
sgal2 =  0.01;
beta0 = 0;
maxIter = 10000
coreNum = 24;
lam = 0.1;
epsStopLogLik = 1e-7;
```

Fit MR-LD w/ (`constr = 1`) and w/o (`constr = 0`) constraint that $\beta = 0$ as:

```
RealMRLD_Hb = MRLDP_RealPXvb_block(bpnew, chrnew, avbIndexnew-1, idx4panelnew, block_file,
                                   stringname3, bh1new, bh2new, s12new, s22new,
                                   gamma, alpha, beta0, sgga2, sgal2, coreNum,
                                   lam, 0, epsStopLogLik, maxIter, model = 1);
RealMRLD_H0 = MRLDP_RealPXvb_block(bpnew, chrnew, avbIndex-1, idx4panelnew, block_file,
                                   stringname3, bh1new, bh2new, s12new, s22new,
                                   gamma, alpha, beta0, sgga2, sgal2, coreNum,
```

```
                                    lam, 1, epsStopLogLik, maxIter, model = 1);

beta0_MRLD = RealMRLD_Hb$beta0;
Tstat_LD <- 2*(RealMRLD_Hb$tstat - RealMRLD_H0$tstat);
MRLD_se = abs(RealMRLD_Hb$beta0/sqrt(Tstat_LD));
```

beta0_MRLD is the estimated effect of exposure on outcome and MRLD_se is corresponding standard error using MRLD model.

Fit MR-LDP w/ (`constr = 1`) and w/o (`constr = 0`) constraint that $\beta = 0$ as:

```
RealMRLDP_Hb = MRLDP_RealPXvb_block(bp, chr, avbIndex-1, idx4panel, block_file,
                                    stringname3, bh1, bh2, s12, s22,
                                    gamma, alpha, beta0, sgga2, sgal2, coreNum,
                                    lam, 0, epsStopLogLik, maxIter, model = 2);
RealMRLDP_H0 = MRLDP_RealPXvb_block(bp, chr, avbIndex-1, idx4panel, block_file,
                                    stringname3, bh1, bh2, s12, s22,
                                    gamma, alpha, beta0, sgga2, sgal2, coreNum,
                                    lam, 1, epsStopLogLik, maxIter, model = 2);
beta0_MRLDP = RealMRLDP_Hb$beta0;
Tstat_MRLDP <- 2*(RealMRLDP_Hb$tstat - RealMRLDP_H0$tstat);
MRLDP_se = abs(RealMRLDP_Hb$beta0/sqrt(Tstat_MRLDP));
```

beta0_MRLDP is the estimated effect of exposure on outcome and MRLDP_se is the corresponding standard error using MRLD model.

Fit MR-LDP using available LD information.

In order to enhance the user-friendliness of our package, inspired by the MR.CUE package, we have strived to incorporate MR-LDP functionality by harnessing the available LD information sourced from the UK10K reference panel. This data can be conveniently obtained through the provided link: https://zenodo.org/records/7212938. Notably, in the current version, we could not use the third dataset (referred to as the screening dataset) but the exposure data to screen the significant IVs directly. Users are advised to conduct the initial step of matching the three datasets if utilization of the screening dataset is deemed necessary.

```
pva_cutoff = 5e-4; lambad = 0.85; ld_r2_thresh = 0.001
filepan <- vector("list", 22);
NumChr = 22;
for(i in 1:NumChr){
  filepan[[i]] <- paste0("UK10KCHR", i, "LDhm3.RDS");
}

fileexp = "BMIcauseSummaryStat.txt";
fileout = "T2DcauseSummaryStat.txt";
snpinfo = "UK10Ksnpinforhm3.RDS";

data <- ReadSummaryStat(fileexp, fileout, filepan, snpinfo, pva_cutoff, lambad);

F4bh1 <- data$ResF4gammah;
F4bh2 <- data$ResF4Gammah;
F4se1 <- data$ResF4se1;
F4se2 <- data$ResF4se2;
F4Rblock <- data$ResF4Rblock;
F4SNPs <- data$ResF4SNPchr;
```

```
# Fit MR-LD
RealMRLD_Hb = MRLDP_Real(F4bh1, F4bh2, F4se1, F4se2, F4Rblock, model = 1, constr = 0);
RealMRLD_H0 = MRLDP_Real(F4bh1, F4bh2, F4se1, F4se2, F4Rblock, model = 1, constr = 1);

beta0_MRLD1 = RealMRLD_Hb$beta0;
Tstat_MRLD1 <- 2*(RealMRLD_Hb$tstat - RealMRLD_H0$tstat);
MRLD_se1 = abs(RealMRLD_Hb$beta0/sqrt(Tstat_MRLD1));

# Fit MR-LDP
RealMRLDP_Hb = MRLDP_Real(F4bh1, F4bh2, F4se1, F4se2, F4Rblock, model = 2, constr = 0);
RealMRLDP_H0 = MRLDP_Real(F4bh1, F4bh2, F4se1, F4se2, F4Rblock, model = 2, constr = 1);

beta0_MRLDP1 = RealMRLDP_Hb$beta0;
Tstat_MRLDP1 <- 2*(RealMRLDP_Hb$tstat - RealMRLDP_H0$tstat);
MRLDP_se1 = abs(RealMRLDP_Hb$beta0/sqrt(Tstat_MRLDP1));
```