CHESS: Causal Heterogeneity using Summary Statistics

Yadong Yang, Minxi Bai, Jiacheng Miao, Jin Liu, Qiongshi Lv and Xingjie Shi

August 25, 2025

Introduction

This vignette provides an introduction to the *CHESS* package. R package *CHESS* implements CHESS for causal heterogeneity using summary statistics.

Install the development version of *CHESS* by use of the ‘devtools’ package. Note that *CHESS* depends on the ‘Rcpp’ and ‘RcppArmadillo’ package, which also requires appropriate setting of Rtools and Xcode for Windows and Mac OS/X, respectively.

To install this package, run the following command in R.

```
library(devtools)
install_github("shilab-ecnu/MR-CHESS")
```

Load the package using the following command:

```r
library(CHESS)
```

Fit CHESS using simulated data

We first generate the genotype data and environmental variable:

```r
library(mvtnorm)
library(MASS)
set.seed(2025)
```

```r
n_exp <- 80000;
n_out <- 80000;
m <- 1000;
h_g1 <- 0.3;
h_g3 <- 0.1;
b1 <- 0;
b4 <- 0.3;
cor_g1g3 <- 0.4;
rhoxy <- 0.6;

maf <- runif(m, 0.05, 0.5);
G <- matrix(rbinom((n_exp + n_out) * m, 2, rep(maf, each = n_exp + n_out)),
            nrow = n_exp + n_out, ncol = m);
G <- scale(G, center = TRUE, scale = FALSE);
E_x <- rnorm(n_exp + n_out)
```

Now simulate the genetic effect sizes. The main genetic effects (γ_1) and G×E interaction effects (γ_3) are generated as correlated multivariate normal variables with specified heritabilities.

```
sigma2g1 <- h_g1 / m;
sigma2g3 <- h_g3 / m;

cov_matrix <- matrix(c(sigma2g1,
                      cor_g1g3 * sqrt(sigma2g1 * sigma2g3),
                      cor_g1g3 * sqrt(sigma2g1 * sigma2g3),
                      sigma2g3), ncol = 2);

gamma1_3 <- rmvnorm(m, mean = c(0, 0), sigma = cov_matrix)
gamma_1x <- gamma1_3[, 1];
gamma_3x <- gamma1_3[, 2]
```

Generate the exposure (X) and outcome (Y) variables With the genetic effects defined.

```
GE <- G * E_x;

noise_x <- rnorm(n_exp + n_out, sd = sqrt(1 - h_g1 - h_g3));
X <- G %*% gamma_1x + GE %*% gamma_3x + noise_x;

noise_y <- rnorm(n_exp + n_out, sd = sqrt(1 - b1^2 - b4^2));
Y <- X * b1 + GE %*% gamma_3x * b4 + noise_y;

exp_gwas <- X[1:n_exp];
exp_gwis <- X[1:n_exp];
exp_E <- E_x[1:n_exp];
out_gwas <- Y[(n_exp + 1):(n_exp + n_out)];
out_gwis <- Y[(n_exp + 1):(n_exp + n_out)];
out_E <- E_x[(n_exp + 1):(n_exp + n_out)]
```

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

```
get_sumstats <- function(G, pheno, interaction = FALSE, E = NULL) {
  betas <- numeric(ncol(G));
  ses <- numeric(ncol(G));

  for(i in 1:ncol(G)) {
    if(interaction) {
      # Model with G×E interaction term
      model <- lm(pheno ~ G[, i] + E + G[, i]:E);
      betas[i] <- coef(model)[4];
      ses[i] <- summary(model)$coefficients[4, 2];
    } else {
      # Standard additive genetic model
      model <- lm(pheno ~ G[, i]);
      betas[i] <- coef(model)[2];
      ses[i] <- summary(model)$coefficients[2, 2]
    }
  }
  return(list(beta = betas, se = ses))
}
```

```r
exp_gwas_sum <- get_sumstats(G[1:n_exp, ], exp_gwas);
exp_gwis_sum <- get_sumstats(G[1:n_exp, ], exp_gwis, interaction = TRUE, E = exp_E);

out_gwas_sum <- get_sumstats(G[(n_exp + 1):(n_exp + n_out), ], out_gwas);
out_gwis_sum <- get_sumstats(G[(n_exp + 1):(n_exp + n_out), ], out_gwis,
                             interaction = TRUE, E = out_E)
```

We select genetic instruments using a p-value threshold. SNPs in either the GWAS or GWIS analysis are included in the union set of instruments.

```r
p_threshold <- 0.01;

pvals_gwas <- 2 * pnorm(-abs(exp_gwas_sum$beta / exp_gwas_sum$se));
iv_gwas <- which(pvals_gwas < p_threshold);

pvals_gwis <- 2 * pnorm(-abs(exp_gwis_sum$beta / exp_gwis_sum$se));
iv_gwis <- which(pvals_gwis < p_threshold);

iv_union <- union(iv_gwas, iv_gwas);
R <- diag(length(iv_union))
```

Finally, we apply the CHESS methods.

```r
gamma_hat <- exp_gwas_sum$beta[iv_union];
gamma3_hat <- exp_gwis_sum$beta[iv_union];
Gamma_hat <- out_gwas_sum$beta[iv_union];
Gamma3_hat <- out_gwis_sum$beta[iv_union];

se_gamma <- exp_gwas_sum$se[iv_union];
se_gamma3 <- exp_gwis_sum$se[iv_union];
se_Gamma <- out_gwas_sum$se[iv_union];
se_Gamma3 <- out_gwis_sum$se[iv_union];

rho_1 <- 0;
rho_2 <- 0;

res <- CHESS(gamma_hat, gamma3_hat, Gamma_hat, Gamma3_hat,
             se_gamma, se_gamma3, se_Gamma, se_Gamma3, R, rho_1, rho_2);
str(res);

beta1_hat <- res$Beta1.hat;
se1_hat <- res$Beta1.se;
pval1 <- res$Beta1.pval;
beta4_hat <- res$Beta4.hat;
se4_hat <- res$Beta4.se;
pval4 <- res$Beta4.pval
```

beta1__hat, se1__hat, pval1 are estimated average causal effect, corresponding standard error and p-value of beta1__hat. beta4__hat, se4__hat, pval4 are estimated heterogeneity causal effect, corresponding standard error and p-value of beta4__hat.

Fit CHESS using Testosterone-BD study with environmental factor sex

Furthermore, we give an example to illustrate the implements of CHESS for real data analysis. The following datasets ('Testosterone.GWAS.txt.gz', 'Testosterone.GWIS.txt.gz', 'BD.GWAS.txt.gz', 'BD.GWIS.txt.gz', 'g1000_eur.bed','g1000_eur.fam', 'g1000_eur.bim', 'all.bed') should be prepared. Download here: $https://figshare.com/articles/dataset/Data_forC_HESS/29910116$.

```
expgwas <- "Testosterone.GWAS.txt.gz";
expgwis <- "Testosterone.GWIS.txt.gz";
outgwas <- "BD.GWAS.txt.gz";
outgwis <- "BD.GWIS.txt.gz";
stringname3 <- "g1000_eur";
block_file <- "all.bed";
```

'expgwas', 'expgwis', 'outgwas', 'outgwis' are the datasets names for exposure GWAS, exposure GWIS, outcome GWAS and outcome GWIS, respectively. Here the environment variable is sex.

These four datasets must have the following format (note that it must be tab delimited): including columns as SNP,CHR,BP,A1,A2,BETA,SE,P.

Table 1: Data format used for exposure and outcome data.

SNP	CHR	BP	A1	A2	BETA	SE	P
rs1000149	14	64024032	A	G	-0.007053	0.002150	0.001034
rs10001493	4	106818592	G	C	-0.001727	0.003293	0.600015
rs10001494	4	132996059	T	C	0.001469	0.001455	0.312651
rs10001495	4	149936868	A	G	0.003784	0.001681	0.024369
rs10001497	4	4733919	A	G	-0.000557	0.001890	0.768299

If GWAS and GWIS data cannot be directly obtained, and the environmental factor is a binary variable (e.g., sex), one can generate the required GWAS and GWIS inputs for CHESS by converting the sex-stratified summary statistics (e.g., Testosterone.male.txt and Testosterone.female.txt) as follows.

The GWAS summary statistics can be generated by meta-analyzing the male and female data using inverse-variance weighting, as implemented in METAL ($https://github.com/statgen/METAL$). After installing the software, the analysis can be executed via the command line (e.g., in Linux or other shell environments) using a configuration file. A sample configuration file 'metal.config.Testosterone.txt' is available for download: $https://figshare.com/articles/dataset/Data_forC_HESS/29910116$.

```
metal metal.config.Testosterone.txt
```

The SNP effects and standard errors for GWIS summary statistics were derived based on the following formula, assuming sex coded as Male=1, Female=-1. Allele direction must be aligned prior to analyzing sex-stratified data.

$$\hat{b}_{gwis,j} = \frac{1}{2}(\hat{b}_{male,j} - \hat{b}_{female,j})$$

$$se(\hat{b}_{gwis,j}) = \frac{1}{2}\sqrt{(se(\hat{b}_{male,j})^2 + se(\hat{b}_{female,j})^2}$$

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

`matchpanel` function is used to match a GWAS/GWIS dataset with the reference panel data, alongside initial data quality control. The output includes a data frame (`$data`) and the corresponding storage path (`$data_dir`).

```
expgwas.match <- matchpanel(expgwas,stringname3)$data_dir;
expgwis.match <- matchpanel(expgwis,stringname3)$data_dir;
outgwas.match <- matchpanel(outgwas,stringname3)$data_dir;
outgwis.match <- matchpanel(outgwis,stringname3)$data_dir;
```

Having given that we have the formatted data, we can use the `ivselect` function to screen the instrumental variables (IVs) and estimate the correlations among those IVs. `plink_dir` specifies the local path to the PLINK executable; if not provided, PLINK will be automatically downloaded. `pval_cutoff_gwas` and `pval_cutoff_gwis` define the P-value thresholds for the exposure GWAS and GWIS, respectively. `r2_cutoff` and `kb_cutoff` are used in LD clumping to specify the r^2 threshold and the physical distance (in kilobases) between SNPs. `maf_cutoff` sets the threshold for minor allele frequency. `lam` denotes the shrinkage parameter used in the regularization of the LD matrix. `CoreNum` indicates the number of CPU cores to be used for parallel computation. `intersect_mode` controls whether to merge GWAS and GWIS IVs using intersection (default: union).

```
plink_dir <- NULL;
pval_cutoff_gwas <- 5e-8;
pval_cutoff_gwis <- 5e-8;
r2_cutoff <- 0.5;
kb_cutoff <- 1024;
maf_cutoff <- 0.05;
lam <- 0.1;
coreNum <- 1;
intersect_mode <- FALSE;
```

```
ivselect.res <- ivselect(expgwas.match, expgwis.match, outgwas.match, outgwis.match,
                         stringname3, block_file, plink_dir,
                         pval_cutoff_gwas, pval_cutoff_gwis, r2_cutoff,
                         kb_cutoff, maf_cutoff, lam, coreNum, intersect_mode);
snp.causal <- ivselect.res$snp.causal;
gammah1 <- ivselect.res$gammah1;
gammah3 <- ivselect.res$gammah3;
Gammah1 <- ivselect.res$Gammah1;
Gammah3 <- ivselect.res$Gammah3;
se1 <- ivselect.res$se1;
se2 <- ivselect.res$se2;
se3 <- ivselect.res$se3;
se4 <- ivselect.res$se4;
R <- ivselect.res$R;
```

When the exposure and outcome samples are independent, the sample correlation parameters `rho1` (for GWAS) and `rho2` (for GWIS) are set to 0.

```
rho1 <- 0; rho2 <- 0;
```

For overlap samples, Since ρ_1 and ρ_2 are estimated using summary statistics among independent variants, we select independent SNPs using the clumping algorithm (r^2 threshold denoted by `ld_r2_thresh`). `pth` is the critical value adapted to the truncated normal distribution in the estimation procedure. `lambda` is the shrinkage turning parameter for LD estimator.

```
ld_r2_thresh <- 0.001;
lambad <- 0.85;
pth <- 1.96;
RhoEst1 <- EstRhofun(expgwas, outgwas, stringname3, ld_r2_thresh, lambad, pth);
rho1 <- mean(RhoEst1$Rhores);
RhoEst2 <- EstRhofun(expgwas, outgwas, stringname3, ld_r2_thresh, lambad, pth);
rho2 <- mean(RhoEst2$Rhores);
```

Now we can fit CHESS using the function *CHESS*.

```r
res <- CHESS(gammah1, gammah3, Gammah1, Gammah3,
             se1, se2, se3, se4, R, rho1, rho2)
str(res)
```

Check the convergence of Gibbs sampler using `traceplot`.

```r
traceplot(res$Beta1res);
traceplot(res$Beta4res);
```

```r
CHESSbeta1 <- res$Beta1.hat;
CHESSse1 <- res$Beta1.se;
CHESSpvalue1 <- res$Beta1.pval;
CHESSbeta4 <- res$Beta4.hat;
CHESSse4 <- res$Beta4.se;
CHESSpvalue4 <- res$Beta4.pval;
```

```r
cat("The estimated main effect of Testosterone on BD: ", CHESSbeta1,
    "\n with Standard error: ", CHESSse1, "and P-value: ", CHESSpvalue1);
```

```r
cat("The estimated interaction effect of Testosterone on BD: ", CHESSbeta4,
    "\n with Standard error: ", CHESSse4, "and P-value: ", CHESSpvalue4)
```