MendelianRandomization v0.9.0: an R package for performing Mendelian randomization analyses using summarized data

Maintained by Stephen Burgess (sb452@medschl.cam.ac.uk)

MendelianRandomization is a package developed to carry out Mendelian randomization analyses using summarized genetic data in R. The package implements various methods to help assess whether a risk factor (also called an exposure) has a causal effect on an outcome.

Several additional references are available which describe the operation of this package, including:

- Yavorska and Burgess (2017) "MendelianRandomization: an R package for performing Mendelian randomization analyses using summarized data", doi: 10.1093/ije/dyx034. This describes the basic operation of the package and core functions.
- Broadbent et al (2020) "MendelianRandomization v0.5.0: updates to an R package for performing Mendelian randomization analyses using summarized data", doi: 10.12688/wellcomeopenres.16374.2. This describes updates to the package up to version 0.5.0.
- Patel et al (2023) "MendelianRandomization v0.9.0: updates to an R package for performing Mendelian randomization analyses using summarized data". This describes updates to the package up to version 0.9.0.

```
library(MendelianRandomization) # loading the package
```

The Input

The package uses a special class called *MRInput* within the analyses in order to pass in all necessary information through one simple structure rather than inserting the data in parts. In order to make an *MRInput* object, one can do the following:

- specify values for each slot separately, or
- extract values from the PhenoScanner web-based database

We focus initially on the first option.

The *MRInput* object has the following "slots" :

- *betaX* and *betaXse* are both numeric vectors describing the associations of the genetic variants with the exposure. *betaX* are the beta-coefficients from univariable regression analyses of the exposure on each genetic variant in turn, and *betaXse* are the standard errors.
- *betaY* and *betaYse* are both numeric vectors describing the associations of the genetic variants with the outcome. *betaY* are the beta-coefficients from regression analyses of the outcome on each genetic variant in turn, and *betaYse* are the standard errors.
- *correlation* is a matrix with the signed correlations between the variants. If a correlation matrix is not provided, it is assumed that the variants are uncorrelated.
- *exposure* is a character string giving the name of the risk factor, e.g. LDL-cholesterol.

- *outcome* is a character string giving the name of the outcome, e.g. coronary heart disease. These inputs are only used in the graphing functions.
- *snps* is a character vector of the names of the various genetic variants (SNPs) in the dataset, e.g. rs12785878. It is not necessary to name the exposure, outcome, or SNPs, but these names are used in the graphing functions and may be helpful for keeping track of various analyses.

To generate the *MRInput* object slot by slot, one can use the *mr_input()* function :

```
MRInputObject <- mr_input(bx = ldlc,
                          bxse = ldlcse,
                          by = chdlodds,
                          byse = chdloddsse)

MRInputObject  # example with uncorrelated variants
```

```
##       SNP exposure.beta exposure.se outcome.beta outcome.se
## 1   snp_1        0.0260       0.004       0.0677     0.0286
## 2   snp_2       -0.0440       0.004      -0.1625     0.0300
## 3   snp_3       -0.0380       0.004      -0.1054     0.0310
## 4   snp_4       -0.0230       0.003      -0.0619     0.0243
## 5   snp_5       -0.0170       0.003      -0.0834     0.0222
## 6   snp_6       -0.0310       0.006      -0.1278     0.0667
## 7   snp_7       -0.0180       0.004      -0.0408     0.0373
## 8   snp_8        0.0460       0.007       0.0770     0.0543
## 9   snp_9        0.0590       0.004       0.1570     0.0306
## 10 snp_10        0.0040       0.003      -0.0305     0.0236
## 11 snp_11        0.0110       0.004       0.0100     0.0277
## 12 snp_12       -0.0050       0.005       0.1823     0.0403
## 13 snp_13        0.0040       0.005      -0.0408     0.0344
## 14 snp_14        0.0220       0.005       0.1989     0.0335
## 15 snp_15       -0.0050       0.004       0.0100     0.0378
## 16 snp_16       -0.0020       0.004       0.0488     0.0292
## 17 snp_17       -0.0020       0.003       0.0100     0.0253
## 18 snp_18        0.0040       0.004      -0.0408     0.0319
## 19 snp_19        0.0110       0.004      -0.0305     0.0316
## 20 snp_20        0.0090       0.003      -0.0408     0.0241
## 21 snp_21       -0.0110       0.004      -0.0202     0.0285
## 22 snp_22       -0.0030       0.003      -0.0619     0.0217
## 23 snp_23       -0.0120       0.004       0.0296     0.0298
## 24 snp_24        0.0003       0.003       0.0677     0.0239
## 25 snp_25       -0.0150       0.003      -0.0726     0.0220
## 26 snp_26       -0.0080       0.004      -0.0726     0.0246
## 27 snp_27        0.0090       0.003       0.0000     0.0255
## 28 snp_28       -0.0360       0.007       0.0198     0.0647
```

```
MRInputObject.cor <- mr_input(bx = calcium,
                              bxse = calciumse,
                              by = fastgluc,
                              byse = fastglucse,
                              corr = calc.rho)

MRInputObject.cor  # example with correlated variants
```

```
##     SNP exposure.beta exposure.se outcome.beta outcome.se
## 1 snp_1       0.00625     0.00233      0.02805     0.0122
## 2 snp_2       0.00590     0.00338      0.00953     0.0198
```

```
## 3 snp_3       0.01822     0.00318      0.03646     0.0173
## 4 snp_4       0.00598     0.00233      0.01049     0.0119
## 5 snp_5       0.00825     0.00229      0.02357     0.0122
## 6 snp_6       0.00651     0.00352      0.00204     0.0179
```

It is not necessary for all the slots to be filled. For example, some of the methods do not require *bxse* to be specified; for example, the *mr_ivw* function will still run with *bxse* set to zeros. If the vectors *bx*, *bxse*, *by*, and *byse* are not of equal length, then an error will be reported. Note that the package does not implement any harmonization of associations to the same effect allele; this must be done by the user.

It is also possible to run the analysis using the syntax:

```
MRInputObject <- mr_input(ldlc, ldlcse, chdlodds, chdloddsse)
```

However, care must be taken in this case to give the vectors in the correct order (that is: *bx, bxse, by, byse*).

The data

Two sets of data are provided as part of this package:

- *ldlc, ldlcse, hdlc, hdlcse, trig, trigse, chdlodds, chdloddsse*: these are the associations (beta-coefficients and standard errors) of 28 genetic variants with LDL-cholesterol, HDL-cholesterol, triglycerides, and coronary heart disease (CHD) risk (associations with CHD risk are log odds ratios) taken from Waterworth et al (2011) "Genetic variants influencing circulating lipid levels and risk of coronary artery disease", doi: 10.1161/atvbaha.109.201020.
- *calcium, calciumse, fastgluc, fastglucse*: these are the associations (beta-coefficients and standard errors) of 7 genetic variants in the *CASR* gene region. These 7 variants are all correlated, and the correlation matrix is provided as *calc.rho*. These data were analysed in Burgess et al (2015) "Using published data in Mendelian randomization: a blueprint for efficient identification of causal risk factors", doi: 10.1007/s10654-015-0011-z.

Univariable estimation methods

The MendelianRandomization package supports several univariable estimation methods (that is, methods for a single exposure variable): the inverse-variance weighted method, the median-based method, the MR-Egger method, the mode-based method, the maximum likelihood method, the heterogeneity penalized method, the contamination mixture method, the lasso method, the debiased inverse-variance weighted method, the penalized inverse-variance weighted method, the constrained maximum likelihood method, and the principal components generalized method of moments (GMM) method. We describe these methods in turn.

Inverse-variance weighted method

The inverse-variance method is equivalent to the standard instrumental variable method using individual-level data, the two-stage least squares method. Either a fixed- or a random-effects analysis can be performed; the *"default"* option is a fixed-effect analysis when there are three variants or fewer, and a random-effects analysis otherwise. The *robust* option uses robust regression rather than standard regression in the analysis, and the *penalized* option downweights the contribution to the analysis of genetic variants with outlying (heterogeneous) variant-specific estimates. If a correlation matrix is provided in the *MRInput* object, then the correlated method is used by default (*correl = TRUE*), and the *robust* and *penalized* arguments are ignored.

The default options for constructing confidence intervals are based on a normal distribution and a 95% confidence level, however one can use the t-distribution (*distribution = "t-dist"*) and alternative significance level if desired.

The default option for the weights is *"simple"*, which corresponds to an inverse-variance weighted meta-analysis of the ratio estimates *by/bx* using first-order standard errors *byse/bx*. Alternatively, weights can be set to *"delta"*, to include second-order terms of the delta expansion. This makes use of *psi*, which is the correlation for each variant between its association with the exposure and with the outcome.

The *mr_ivw* function automatically calculates an approximation to the first-stage F statistic from regression of the exposure on the genetic variants. This is an important measure of instrument strength. When the correlation matrix is specified, calculation of the estimated F statistic accounts for the correlation between variants. When a correlation matrix is specified, a warning message appears to remind the user of the importance of correct harmonization of the correlation matrix to the same effect and other alleles as the genetic associations with the exposure and outcome.

```
IVWObject <- mr_ivw(MRInputObject,
                    model = "default",
                    robust = FALSE,
                    penalized = FALSE,
                    correl = FALSE,
                    weights = "simple",
                    psi = 0,
                    distribution = "normal",
                    alpha = 0.05)

IVWObject <- mr_ivw(mr_input(bx = ldlc, bxse = ldlcse,
   by = chdlodds, byse = chdloddsse))

IVWObject
```

```
##
## Inverse-variance weighted method
## (variants uncorrelated, random-effect model)
##
## Number of Variants : 28
##
## ------------------------------------------------------------------
##  Method Estimate Std Error 95% CI       p-value
##     IVW    2.834     0.530 1.796, 3.873   0.000
## ------------------------------------------------------------------
## Residual standard error =  1.920
## Heterogeneity test statistic (Cochran's Q) = 99.5304 on 27 degrees of freedom,
##  (p-value = 0.0000). I^2 = 72.9%.
## F statistic = 28.0.
```

```
IVWObject.correl <- mr_ivw(MRInputObject.cor,
                    model = "default",
                    correl = TRUE,
                    distribution = "normal",
                    alpha = 0.05)

IVWObject.correl <- mr_ivw(mr_input(bx = calcium, bxse = calciumse,
   by = fastgluc, byse = fastglucse, corr = calc.rho))

IVWObject.correl
```

```
##
## Inverse-variance weighted method
## (variants correlated, random-effect model)
##
## Number of Variants : 6
##
## ------------------------------------------------------------------
##  Method Estimate Std Error 95% CI       p-value
```

```
##     IVW    2.245     0.643 0.984, 3.505   0.000
## ------------------------------------------------------------------
## Residual standard error =  0.641
## Residual standard error is set to 1 in calculation of confidence interval
##  when its estimate is less than 1.
## Heterogeneity test statistic (Cochran's Q) = 2.0530 on 5 degrees of freedom,
##  (p-value = 0.8418). I^2 = 0.0%.
## F statistic = 11.5.
##
## (Estimates with correlated variants are sensitive to the signs in the correlation matrix
##  - please ensure that your correlations are expressed with respect to the same
##    effect alleles as your summarized association estimates.)
```

Median-based method

The median-based method calculates a median of the variant-specific estimates from the ratio method for each genetic variant individually. The default option is to calculate a weighted median using the inverse-variance weights. Alternatively, one can calculate a simple (unweighted) median, or a weighted median using penalization of weights for heterogeneous variants. Since the calculation of standard error requires bootstrapping, the number of bootstrap iterations can be varied. The random seed is set automatically so that results are reproducible; however, the value of the seed can be changed if required. Once the function has been completed, the random seed is reset to its previous value (this is true for all functions in the package).

The median-based method requires data on at least 3 genetic variants. Variants must be uncorrelated.

```
WeightedMedianObject <- mr_median(MRInputObject,
                                  weighting = "weighted",
                                  distribution = "normal",
                                  alpha = 0.05,
                                  iterations = 10000,
                                  seed = 314159265)
```

```
WeightedMedianObject <- mr_median(mr_input(bx = ldlc, bxse = ldlcse,
  by = chdlodds, byse = chdloddsse))
```

```
WeightedMedianObject
```

```
##
##  Weighted median method
##
## Number of Variants : 28
## ------------------------------------------------------------------
##                   Method Estimate Std Error 95% CI       p-value
##  Weighted median method    2.683     0.419 1.862, 3.504   0.000
## ------------------------------------------------------------------
```

```
SimpleMedianObject <- mr_median(mr_input(bx = ldlc, bxse = ldlcse,
  by = chdlodds, byse = chdloddsse), weighting = "simple")
```

```
SimpleMedianObject
```

```
##
##  Simple median method
##
## Number of Variants : 28
## ------------------------------------------------------------------
##                 Method Estimate Std Error 95% CI       p-value
```

```
##  Simple median method    1.755     0.740 0.305, 3.205   0.018
## ------------------------------------------------------------------
```

MR-Egger method

The MR-Egger method is implemented here using a random-effects model only. The *robust* and *penalized* options are the same as for the inverse-variance weighted method. The method can be used for both correlated and uncorrelated sets of variants. Confidence intervals can be constructed either using a normal distribution (*distribution = "normal"*, the default option), or a t-distribution (*distribution = "t-dist"*).

With a t-distribution, in case of under-dispersion (the estimated residual standard error in the regression model is less than 1), confidence intervals and p-values use either a t-distribution with no correction for under-dispersion, or a normal distribution with the residual standard error set to 1 – whichever is wider. This means that under-dispersion is not doubly penalized by setting the residual standard error to 1 and using a t-distribution, but also that the confidence intervals are not narrower (p-values not more extreme) than those using a fixed-effect model.

The MR-Egger method requires data on at least 3 genetic variants. Variants are permitted to be correlated.

```
EggerObject <- mr_egger(MRInputObject,
                        robust = FALSE,
                        penalized = FALSE,
                        correl = FALSE,
                        distribution = "normal",
                        alpha = 0.05)

EggerObject <- mr_egger(mr_input(bx = ldlc, bxse = ldlcse,
  by = chdlodds, byse = chdloddsse))

EggerObject
```

```
##
## MR-Egger method
## (variants uncorrelated, random-effect model)
##
## Number of Variants =  28
##
## ------------------------------------------------------------------
##       Method Estimate Std Error  95% CI       p-value
##     MR-Egger    3.253     0.770  1.743, 4.762   0.000
##  (intercept)   -0.011     0.015 -0.041, 0.018   0.451
## ------------------------------------------------------------------
## Residual Standard Error :  1.935
## Heterogeneity test statistic = 97.3975 on 26 degrees of freedom, (p-value = 0.0000)
## I^2_GX statistic: 91.9%
```

```
EggerObject.corr <- mr_egger(MRInputObject.cor,
                        correl = TRUE,
                        distribution = "normal",
                        alpha = 0.05)

EggerObject.corr <- mr_egger(mr_input(bx = calcium, bxse = calciumse,
  by = fastgluc, byse = fastglucse, corr = calc.rho))

EggerObject.corr
```

```
##
```

```
## MR-Egger method
## (variants correlated, random-effect model)
##
## Number of Variants =  6
##
## ------------------------------------------------------------------
##       Method Estimate Std Error  95% CI       p-value
##     MR-Egger    1.302     1.518 -1.672, 4.276   0.391
##  (intercept)    0.009     0.013 -0.017, 0.035   0.493
## ------------------------------------------------------------------
## Residual Standard Error :  0.629
## Residual standard error is set to 1 in calculation of confidence interval
##  when its estimate is less than 1.
## Heterogeneity test statistic = 1.5825 on 4 degrees of freedom, (p-value = 0.8119)
```

Maximum likelihood method

An alternative estimation method is the maximum likelihood method, introduced in Burgess et al (2013) "Mendelian randomization analysis with multiple genetic variants using summarized data", doi: 10.1002/gepi.21758.

The method has two main advantages over the IVW method: it allows for uncertainty in the genetic associations with the exposure (which is ignored in the IVW method using simple weights), and it allows for genetic associations with the exposure and with the outcome for each variant to be correlated. This correlation arises if the samples for the associations with the exposure and the outcome overlap. In a strict two-sample Mendelian randomization setting, the correlation parameter *psi* is set to zero (the default option). If the associations are estimated in the same individuals (complete sample overlap), then the observed correlation between the exposure and the outcome is a reasonable proposal for the value of *psi*.

```
MaxLikObject <- mr_maxlik(MRInputObject,
                          model = "default",
                          correl = FALSE,
                          psi = 0,
                          distribution = "normal",
                          alpha = 0.05)

MaxLikObject <- mr_maxlik(mr_input(bx = ldlc, bxse = ldlcse,
  by = chdlodds, byse = chdloddsse))

MaxLikObject
```

```
##
## Maximum-likelihood method
## (variants uncorrelated, random-effect model)
##
## Number of Variants : 28
## ------------------------------------------------------------------
##  Method Estimate Std Error 95% CI       p-value
##  MaxLik    3.225     0.569 2.110, 4.340   0.000
## ------------------------------------------------------------------
## Residual standard error =  1.793
## Heterogeneity test statistic = 86.8145 on 27 degrees of freedom, (p-value = 0.0000)
```

```
MaxLikObject.corr <- mr_maxlik(mr_input(bx = calcium, bxse = calciumse,
  by = fastgluc, byse = fastglucse, corr = calc.rho))
```

```
MaxLikObject.corr
```

```
##
## Maximum-likelihood method
## (variants correlated, random-effect model)
##
## Number of Variants : 6
## ------------------------------------------------------------------
##  Method Estimate Std Error 95% CI       p-value
##  MaxLik    2.303     0.709 0.914, 3.692   0.001
## ------------------------------------------------------------------
## Residual standard error =  0.588
## Residual standard error is set to 1 in calculation of confidence interval
##  when its estimate is less than 1.
## Heterogeneity test statistic = 1.7277 on 5 degrees of freedom, (p-value = 0.8854)
```

Mode-based estimation method

An additional method for robust estimation that gives consistent when a plurality of genetic variants is valid is the mode-based estimation method, introduced in Hartwig, Davey Smith and Bowden (2017) "Robust inference in summary data Mendelian randomization via the zero modal pleiotropy assumption", doi: 10.1093/ije/dyx102.

While the median-based method calculates the variant-specific estimates and then obtains the median of these estimates, and the inverse-variance weighted estimate is a weighted mean of the variant-specific estimates, the mode-based method obtains the "mode" of these estimates. However, in finite samples, no two estimates will be exactly the same, so a mode does not exist. The method proceeds by constructing a kernel-weighted density of the variant-specific estimates, and taking the maximum point of this density as the point estimate. A confidence interval is obtained by bootstrapping.

Several options can be specified: whether the mode should be *"weighted"* or *"unweighted"* (default is weighted), whether the standard errors should be calculated using the *"simple"* or *"delta"* formula (default is *delta*), the value of the bandwidth parameter multiplication factor (default is 1), the random seed, and number of iterations in the bootstrap procedure.

```
MBEObject <- mr_mbe(MRInputObject,
                    weighting = "weighted",
                    stderror = "delta",
                    phi = 1,
                    seed = 314159265,
                    iterations = 10000,
                    distribution = "normal",
                    alpha = 0.05)

MBEObject <- mr_mbe(mr_input(bx = ldlc, bxse = ldlcse,
  by = chdlodds, byse = chdloddsse))

MBEObject
```

```
##
## Mode-based method of Hartwig et al
## (weighted, delta standard errors [not assuming NOME], bandwidth factor = 1)
##
## Number of Variants : 28
## ------------------------------------------------------------------
##  Method Estimate Std Error 95% CI       p-value
##     MBE    2.946     1.005 0.977, 4.916   0.003
```

```
## ------------------------------------------------------------------
```

Heterogeneity-penalized method

Another method for robust estimation that also gives consistent when a plurality of genetic variants is valid is the heterogeneity-penalized model-averaging method, introduced in Burgess et al (2018) "Modal-based estimation via heterogeneity-penalized weighting: model averaging for consistent and efficient estimation in Mendelian randomization when a plurality of candidate instruments are valid", doi: 10.1093/ije/dyy08010.1101/175372.

The heterogeneity-penalized method uses the same consistency criterion as the mode-based estimation method, but evaluates the modal estimate in a different way. It proceeds by evaluating weights for all subsets of genetic variants (excluding the null set and singletons). Subsets receive greater weight if they include more variants, but are severely downweighted if the variants in the subset have heterogeneous variant-specific estimates. As such, the method will identify the subset with the largest number (by weight) of variants having similar variant-specific estimates.

Confidence intervals are evaluated by calculating a log-likelihood function, and finding all points within a given vertical distance of the maximum of the log-likelihood function (which is the overall estimate). As such, if the log-likelihood function is multimodal, then the confidence interval may include multiple disjoint ranges. This may indicate the presence of multiple causal mechanisms by which the exposure may influence the outcome with different magnitudes of causal effect. As the confidence interval is determined by a grid search, care must be taken when chosing the minimum (*CIMin*) and maximum (*CIMax*) values in the search, as well as the step size (*CIStep*). The default values will not be suitable for all applications.

```
HetPenObject <- mr_hetpen(MRInputObject,
                          prior = 0.5,
                          CIMin = -1,
                          CIMax = 1,
                          CIStep = 0.001,
                          alpha = 0.05)
```

As the method evaluates estimates and weights for each subset of variants, the method is quite computationally expensive to run, as the complexity doubles with each additional variant. We run the method for a subset of 10 genetic variants:

```
HetPenObject <- mr_hetpen(mr_input(bx = ldlc[1:10], bxse = ldlcse[1:10],
  by = chdlodds[1:10], byse = chdloddsse[1:10]), CIMin = -1, CIMax = 5, CIStep = 0.01)

HetPenObject
```

```
##
## Heterogeneity-penalized method
## (Prior probability of instrument validity = 0.5)
##
## Number of Variants : 10
## ------------------------------------------------------------------
##  Method Estimate 95% CI
##  HetPen     2.85  1.87, 3.90
## ------------------------------------------------------------------
## Note: confidence interval is a single range of values.
```

As an example of a multimodal confidence interval:

```
bcrp    =c(0.160, 0.236, 0.149, 0.09, 0.079, 0.072, 0.047, 0.069)
bcrpse  =c(0.006, 0.009, 0.006, 0.005, 0.005, 0.005, 0.006, 0.011)
bchd    =c(0.0237903, -0.1121942, -0.0711906, -0.030848, 0.0479207, 0.0238895,
  0.005528, 0.0214852)
bchdse  =c(0.0149064, 0.0303084, 0.0150552, 0.0148339, 0.0143077, 0.0145478,
```

```
  0.0160765, 0.0255237)

HetPenObject.multimode <- mr_hetpen(mr_input(bx = bcrp, bxse = bcrpse,
  by = bchd, byse = bchdse))

HetPenObject.multimode
```

```
##
## Heterogeneity-penalized method
## (Prior probability of instrument validity = 0.5)
##
## Number of Variants : 8
## ------------------------------------------------------------------
##  Method Estimate  95% CI
##  HetPen   -0.437 -0.599, -0.256
##                   0.014,  0.449
## ------------------------------------------------------------------
## Note: confidence interval contains multiple ranges of values.
```

Other univariable estimation methods

Several other estimation methods are available, which we do not describe in detail, as their operation is similar to methods already described above. We would encourage interested analysts to read the help files for each method, which describe the inputs and outputs of each method.

The contamination mixture method is introduced in Burgess et al (2020) "A robust and efficient method for Mendelian randomization with hundreds of genetic variants", doi: 10.1038/s41467-019-14156-4. It is similar in spirit to the heterogeneity-penalized method, but much more computationally efficient.

```
ConMixObject <- mr_conmix(MRInputObject,
                          psi = 0,
                          CIMin = NA,
                          CIMax = NA,
                          CIStep = 0.01,
                          alpha = 0.05)

ConMixObject <- mr_conmix(mr_input(bx = ldlc, bxse = ldlcse,
  by = chdlodds, byse = chdloddsse))

ConMixObject
```

```
##
## Contamination mixture method
## (Standard deviation of invalid estimands = 66.17905)
##
## Number of Variants : 28
## ------------------------------------------------------------------
##  Method Estimate 95% CI       p-value
##  ConMix     2.73  2.00, 3.43 4.79e-15
## ------------------------------------------------------------------
## Note: confidence interval is a single range of values.
```

The lasso method is introduced in Rees et al (2019) "Robust methods in Mendelian randomization via penalization of heterogeneous causal estimates", doi: 10.1371/journal.pone.0222362. It is an outlier-robust method, which excludes variants with heterogeneous variant-specific estimates using lasso penalization.

```
LassoObject <- mr_lasso(MRInputObject,
```

```
                        distribution = "normal",
                        alpha = 0.05,
                        lambda = numeric(0))

LassoObject <- mr_lasso(mr_input(bx = ldlc, bxse = ldlcse,
  by = chdlodds, byse = chdloddsse))

LassoObject
```

```
##
## MR-Lasso method
##
## Number of variants : 28
## Number of valid instruments : 26
## Tuning parameter : 0.7757552
## ------------------------------------------------------------------
##  Exposure Estimate Std Error 95% CI       p-value
##  exposure    2.671     0.431 1.827, 3.515   0.000
## ------------------------------------------------------------------
```

The debiased inverse-variance weighted method is a variation on the inverse-variance weighted method that is more robust to bias due to weak instruments and winner's curse. It is described in Ting, Shao, Kang (2021) "Debiased inverse-variance weighted estimator in two-sample summary-data Mendelian randomization", doi: 10.1214/20-aos2027.

```
DIVWObject <- mr_divw(MRInputObject,
                      over.dispersion = TRUE,
                      alpha = 0.05,
                      diagnostics = FALSE)

DIVWObject <- mr_divw(mr_input(bx = ldlc, bxse = ldlcse,
  by = chdlodds, byse = chdloddsse))

DIVWObject
```

```
##
## Debiased inverse-variance weighted method
## (Over.dispersion:TRUE)
##
## Number of Variants : 28
## ------------------------------------------------------------------
##  Method Estimate Std Error 95% CI       p-value Condition
##    dIVW    2.940     0.531 1.900, 3.980   0.000   142.920
## ------------------------------------------------------------------
```

The penalized inverse-variance weighted method is a further extension of the inverse-variance weighted and debiased inverse-variance weighted methods to deal with selection of variants more explicitly. It is described in Wang et al (2022) "A novel penalized inverse-variance weighted estimator for Mendelian randomization with applications to COVID-19 outcomes", doi: 10.1111/biom.13732.

```
PIVWObject <- mr_pivw(MRInputObject,
                      over.dispersion = TRUE,
                      delta = 0,
                      sel.pval = NULL,
                      Boot.Fieller = TRUE,
                      alpha = 0.05)
```

```
PIVWObject <- mr_pivw(mr_input(bx = ldlc, bxse = ldlcse,
  by = chdlodds, byse = chdloddsse))

PIVWObject
```

```
##
## Penalized inverse-variance weighted method
##
## Over dispersion: TRUE
## Bootstrapping Fieller: TRUE
## Penalty parameter (lambda): 1
## IV selection threshold (delta): 0
## Number of variants : 28
## ------------------------------------------------------------------
##  Method Estimate Std Error 95% CI       p-value Condition
##    pIVW    2.932     0.528 1.922, 4.001   0.000   142.920
## ------------------------------------------------------------------
```

The constrained maximum likelihood method is a robust method that can account for violation of any of the three instrumental variable assumptions under mild assumptions. In a maximum likelihood framework, this method constrains the number of invalid instruments with horizontal pleiotropy. The number of invalid instruments is asymptotically consistently selected by the Bayesian information criterion. It is described in Xue, Shen, Pan (2021) "Constrained maximum likelihood-based Mendelian randomization robust to both correlated and uncorrelated pleiotropic effects", doi: 10.1016/j.ajhg.2021.05.014.

Note that this method requires the sample size used to calculate the genetic associations with the exposure and/or outcome to be specified (if these differ, then it is recommended to use the lower value; see reference for details).

```
cMLObject <- mr_cML(MRInputObject,
                    MA = TRUE,
                    DP = TRUE,
                    K_vec = 0:(length(object@betaX)-2),
                    random_start = 0,
                    num_pert = 200,
                    random_start_pert = 0,
                    maxit = 100,
                    random_seed = 314,
                    n,
                    Alpha = 0.05)
```

```
cMLObject <- mr_cML(mr_input(bx = ldlc, bxse = ldlcse,
  by = chdlodds, byse = chdloddsse), n = 17723)

cMLObject
```

```
##
## Constrained maximum likelihood method (MRcML)
## Number of Variants:  28
## Results for:  cML-MA-BIC-DP
## ------------------------------------------------------------------
##          Method Estimate    SE Pvalue        95% CI
##  cML-MA-BIC-DP    2.821 0.388  0.000 [2.060,3.582]
## ------------------------------------------------------------------
```

The principal components GMM method is an implementation of the GMM method with summarized data that is designed for use when performing Mendelian randomization using genetic variants from a single gene

region. As an alternative to pruning and clumping approaches, which take a large number of variants from a gene region (potentially hundreds or thousands) and select a small number of uncorrelated (or weakly correlated) variants, the principal components approach performs dimension reduction on the full set of variant associations. The principal components are then used as instrument variables, rather than the individual genetic variants. The method is described in Patel et al (2023) "Robust use of phenotypic heterogeneity at drug target genes for mechanistic insights: application of cis-multivariable Mendelian randomization to GLP1R gene region", doi: 10.1101/2023.07.20.23292958.

Note that this method requires the sample size used to calculate the genetic associations with the exposure (nx) and outcome (ny) to be specified. The number of principal components includes as instruments can either be set directly, by setting r, or indirectly, by setting *thres*. The default option *thres = 0.999* includes enough principal components to explain 99.9% of the variance in the weighted variant correlation matrix.

```
pcGMMObject <- mr_pcgmm(MRInputObject.cor,
                        nx,
                        ny,
                        r = NULL,
                        thres = 0.999,
                        robust = TRUE,
                        alpha = 0.05)
```

```
pcGMMObject <- mr_pcgmm(mr_input(bx = calcium, bxse = calciumse,
   by = fastgluc, byse = fastglucse, corr = calc.rho),
   nx=6351, ny=133010)

pcGMMObject
```

```
##
## Univariable principal components generalized method of moments (PC-GMM) method
##
## Number of principal components used : 5
##
## Robust model with overdispersion heterogeneity.
##
## ------------------------------------------------------------------
##  Method Estimate Std Error 95% CI       p-value F-stat
##  PC-GMM    2.435     0.786 0.895, 3.975   0.002   11.2
## ------------------------------------------------------------------
##
## Overdispersion heterogeneity parameter estimate = -12.65081
##
## Heterogeneity test statistic = 1.6414
```

Multivariable Mendelian randomization

An analytic approach when genetic variants are associated with multiple risk factors is multivariable Mendelian randomization, introduced in Burgess and Thompson (2015) "Multivariable Mendelian randomization: the use of pleiotropic genetic variants to estimate causal effects", doi: 10.1093/aje/kwu283.

There are two contexts in which we envision the method being used. First, when there is a set of related risk factors, such as lipid fractions. It is difficult to find genetic predictors of HDL-cholesterol that do not also predict LDL-cholesterol and/or triglycerides. Multivariable Mendelian randomization allows genetic variants to be associated with all the risk factors in the model provided that they do not influence the outcome via other pathways. Secondly, when there is a network of risk factors, typically a primary risk factor and a secondary risk factor or mediator. In both cases, multivariable estimates reflect direct causal effects - the result of intervening on the risk factor under analysis while keeping other risk factors in the model fixed.

As the multivariable method requires genetic associations with multiple risk factors, a different input function is needed: *mr_mvinput*, which creates an *MRMVInput* object:

```
MVMRInputObject <- mr_mvinput(bx = cbind(ldlc, hdlc, trig),
                              bxse = cbind(ldlcse, hdlcse, trigse),
                              by = chdlodds,
                              byse = chdloddsse)
```

```
MVMRInputObject
```

```
##       SNP exposure_1.beta exposure_2.beta exposure_3.beta exposure_1.se
## 1   snp_1          0.0260          0.0020           0.016         0.004
## 2   snp_2         -0.0440          0.0050          -0.004         0.004
## 3   snp_3         -0.0380          0.0030          -0.009         0.004
## 4   snp_4         -0.0230          0.0010           0.003         0.003
## 5   snp_5         -0.0170          0.0110          -0.039         0.003
## 6   snp_6         -0.0310          0.0310          -0.142         0.006
## 7   snp_7         -0.0180         -0.0030           0.007         0.004
## 8   snp_8          0.0460         -0.0070           0.095         0.007
## 9   snp_9          0.0590         -0.0210           0.042         0.004
## 10 snp_10          0.0040          0.0180          -0.023         0.003
## 11 snp_11          0.0110         -0.0170           0.036         0.004
## 12 snp_12         -0.0050         -0.0470           0.097         0.005
## 13 snp_13          0.0040          0.0220           0.013         0.005
## 14 snp_14          0.0220         -0.0290           0.142         0.005
## 15 snp_15         -0.0050          0.0160          -0.005         0.004
## 16 snp_16         -0.0020          0.0340           0.019         0.004
## 17 snp_17         -0.0020          0.0350           0.003         0.003
## 18 snp_18          0.0040          0.0190          -0.018         0.004
## 19 snp_19          0.0110          0.0280          -0.020         0.004
## 20 snp_20          0.0090          0.0001           0.035         0.003
## 21 snp_21         -0.0110          0.0160          -0.036         0.004
## 22 snp_22         -0.0030          0.0050          -0.054         0.003
## 23 snp_23         -0.0120         -0.0100           0.054         0.004
## 24 snp_24          0.0003         -0.0230           0.067         0.003
## 25 snp_25         -0.0150          0.0120          -0.040         0.003
## 26 snp_26         -0.0080          0.0180          -0.067         0.004
## 27 snp_27          0.0090         -0.0060           0.028         0.003
## 28 snp_28         -0.0360          0.0040          -0.070         0.007
##    exposure_2.se exposure_3.se outcome.beta outcome.se
## 1          0.004         0.008       0.0677     0.0286
## 2          0.004         0.008      -0.1625     0.0300
## 3          0.004         0.008      -0.1054     0.0310
## 4          0.003         0.006      -0.0619     0.0243
## 5          0.003         0.006      -0.0834     0.0222
## 6          0.006         0.012      -0.1278     0.0667
## 7          0.004         0.007      -0.0408     0.0373
## 8          0.006         0.013       0.0770     0.0543
## 9          0.004         0.008       0.1570     0.0306
## 10         0.003         0.006      -0.0305     0.0236
## 11         0.003         0.007       0.0100     0.0277
## 12         0.005         0.010       0.1823     0.0403
## 13         0.004         0.009      -0.0408     0.0344
## 14         0.004         0.009       0.1989     0.0335
## 15         0.003         0.007       0.0100     0.0378
```

```
## 16         0.004         0.007       0.0488     0.0292
## 17         0.003         0.006       0.0100     0.0253
## 18         0.004         0.008      -0.0408     0.0319
## 19         0.004         0.008      -0.0305     0.0316
## 20         0.003         0.007      -0.0408     0.0241
## 21         0.003         0.007      -0.0202     0.0285
## 22         0.003         0.006      -0.0619     0.0217
## 23         0.004         0.008       0.0296     0.0298
## 24         0.003         0.006       0.0677     0.0239
## 25         0.003         0.006      -0.0726     0.0220
## 26         0.003         0.007      -0.0726     0.0246
## 27         0.003         0.006       0.0000     0.0255
## 28         0.006         0.013       0.0198     0.0647
```

```
MVMRInputObject.cor <- mr_mvinput(bx = cbind(ldlc, hdlc, trig),
                                  bxse = cbind(ldlcse, hdlcse, trigse),
                                  by = chdlodds,
                                  byse = chdloddsse,
                                  correlation = diag(length(ldlc)))
```

The *mr_mvivw* command performs an extension of the inverse-variance weighted method for multivariable Mendelian randomization, implementing multivariable weighted linear regression (the standard IVW method method performs univariable weighted linear regression) with the intercept term set to zero. As with the standard IVW method, a correlation matrix can be specified. Multivariable Mendelian randomization requires the number of genetic variants to exceed the number of risk factors; if this is not the case, the method will return an error.

```
MVIVWObject <- mr_mvivw(MVMRInputObject,
                        model = "default",
                        correl = FALSE,
                        correl.x = NULL,
                        nx = NA,
                        distribution = "normal",
                        alpha = 0.05)

MVIVWObject <- mr_mvivw(MVMRInputObject)

MVIVWObject
```

```
##
## Multivariable inverse-variance weighted method
## (variants uncorrelated, random-effect model)
##
## Number of Variants : 28
##
## ------------------------------------------------------------------
##    Exposure Estimate Std Error  95% CI       p-value
##  exposure_1    1.925     0.439  1.064, 2.786   0.000
##  exposure_2   -0.590     0.555 -1.677, 0.498   0.288
##  exposure_3    0.723     0.230  0.272, 1.174   0.002
## ------------------------------------------------------------------
## Residual standard error =  1.433
## Heterogeneity test statistic = 51.3599 on 25 degrees of freedom, (p-value = 0.0014)
```

Conditional F statistics: The *nx* option specifies the sample size(s) for the genetic associations with the exposures (either a single value if these are all equal, or a vector if they differ). The *correl.x* option specifies

the correlation between the genetic associations with the exposures. If the genetic assocations with the exposures are estimates in separate datasets, then this correlation matrix should be set to the identity matrix (this is the default option). This matrix cannot be calculated from summarized data; if it is unknown, then a sensitivity analysis for its value is recommended. The *nx* and *correl.x* options are not used in the calculation of estimates from the multivariable inverse-variance weighted method method, but only in the calculation of conditional F statistics. Conditional F statistics are the relevant measure of instrument strength for multivariable Mendelian randomization. For more detail, see Sanderson, Spiller, Bowden (2021) "Testing and correcting for weak and pleiotropic instruments in two-sample multivariable Mendelian randomization", doi: 10.1002/sim.9133.

```
MVIVWObject.condF <- mr_mvivw(MVMRInputObject, nx = 17723)

MVIVWObject.condF
```

```
##
## Multivariable inverse-variance weighted method
## (variants uncorrelated, random-effect model)
##
## Number of Variants : 28
##
## ------------------------------------------------------------------
##    Exposure Estimate Std Error  95% CI       p-value Cond F-stat
##  exposure_1    1.925     0.439  1.064, 2.786   0.000        20.3
##  exposure_2   -0.590     0.555 -1.677, 0.498   0.288        12.9
##  exposure_3    0.723     0.230  0.272, 1.174   0.002        13.3
## ------------------------------------------------------------------
## Residual standard error =  1.433
## Heterogeneity test statistic = 51.3599 on 25 degrees of freedom, (p-value = 0.0014)
```

Other univariable estimation methods

In addition to the multivariable inverse-variance weighted method, several other multivariable estimation methods are available, which generally are multivariable extensions of univariable methods: multivariable MR-Egger, multivariable median-based method, multivariable lasso method, multivariable constrained maximum likelihood, multivariable GMM, and multivariable principal components GMM.

```
MVEggerObject <- mr_mvegger(MVMRInputObject)

MVEggerObject
```

```
##
## Multivariable MR-Egger method
## (variants uncorrelated, random-effect model)
##
## Orientated to exposure : 1
## Number of Variants : 28
## ------------------------------------------------------------------
##     Exposure Estimate Std Error  95% CI        p-value
##   exposure_1    2.829     0.515  1.819,  3.839   0.000
##   exposure_2   -0.663     0.496 -1.636,  0.310   0.182
##   exposure_3    0.827     0.209  0.417,  1.237   0.000
##  (intercept)   -0.028     0.010 -0.049, -0.008   0.007
## ------------------------------------------------------------------
## Residual standard error =  1.280
## Heterogeneity test statistic = 39.3421 on 24 degrees of freedom, (p-value = 0.0251)
```

```
MVMedianObject <- mr_mvmedian(MVMRInputObject)
```

```
MVMedianObject
```

```
##
## Multivariable median method
##
## Number of variants : 28
## ------------------------------------------------------------------
##    Exposure Estimate Std Error  95% CI       p-value
##  exposure_1    2.116     0.482  1.172, 3.061   0.000
##  exposure_2    0.111     0.656 -1.176, 1.397   0.866
##  exposure_3    1.039     0.261  0.528, 1.550   0.000
## ------------------------------------------------------------------
```

```
MVLassoObject <- mr_mvlasso(MVMRInputObject)

MVLassoObject
```

```
##
## Multivariable MR-Lasso method
##
## Orientated to exposure : 1
## Number of variants : 28
## Number of valid instruments : 25
## Tuning parameter : 0.4641104
## ------------------------------------------------------------------
##    Exposure Estimate Std Error  95% CI       p-value
##  exposure_1    1.726     0.383  0.976, 2.477   0.000
##  exposure_2   -0.126     0.441 -0.991, 0.738   0.775
##  exposure_3    0.858     0.185  0.496, 1.220   0.000
## ------------------------------------------------------------------
```

```
MVcMLObject <- mr_mvcML(MVMRInputObject, n = 17723)

MVcMLObject
```

```
##
## Multivariable MRcML method
##
## Number of variants : 28
## Results for:  MVMRcML-DP
## Number of data perturbations with successful convergence:  200
## ------------------------------------------------------------------
##    Exposure Estimate Std Error  95% CI       p-value
##  exposure_1    1.927     0.613  0.726, 3.127   0.002
##  exposure_2   -0.350     0.671 -1.666, 0.966   0.602
##  exposure_3    0.822     0.272  0.289, 1.354   0.002
## ------------------------------------------------------------------
```

```
MVGMMObject <- mr_mvgmm(MVMRInputObject, nx=rep(17723,3), ny=17723)

MVGMMObject
```

```
##
## Multivariable generalized method of moments (GMM) method
##
## Exposure correlation matrix not specified. Exposures are assumed to be uncorrelated.
```

```
##
## Robust model with overdispersion heterogeneity.
##
## ------------------------------------------------------------------
##    Exposure Estimate Std Error  95% CI       p-value Cond F-stat
##  exposure_1    1.830     0.451  0.947, 2.713   0.000        20.3
##  exposure_2   -0.722     0.580 -1.858, 0.415   0.213        12.9
##  exposure_3    0.686     0.239  0.217, 1.155   0.004        13.3
## ------------------------------------------------------------------
##
## Overdispersion heterogeneity parameter estimate = 17.36878
##
## Heterogeneity test statistic = 24.9273
```

```
MVpcGMMObject <- mr_mvpcgmm(MVMRInputObject.cor, nx=rep(17723,3), ny=17723)

MVpcGMMObject
```

```
##
## Multivariable principal components generalized method of moments (PC-GMM) method
##
## Exposure correlation matrix not specified. Exposures are assumed to be uncorrelated.
##
## Number of principal components used : 25
##
## Robust model with overdispersion heterogeneity.
##
## ------------------------------------------------------------------
##    Exposure Estimate Std Error  95% CI       p-value Cond F-stat
##  exposure_1    1.695     0.517  0.681, 2.709   0.001        18.6
##  exposure_2   -0.754     0.759 -2.240, 0.733   0.321        10.2
##  exposure_3    0.682     0.292  0.110, 1.255   0.020        10.8
## ------------------------------------------------------------------
##
## Overdispersion heterogeneity parameter estimate = 25.5785
##
## Heterogeneity test statistic = 21.5116
```

Note that the *mr_mvpcgmm* example does not use variants from a single gene region, and so is provided to demonstrate that the code works, rather than to illustrate a recommended use case.

Summaries of multiple methods

The *mr_allmethods* function is provided to easily compare results (Estimate, Standard Error, 95% CI, and p-value) from multiple methods. One can look at results from a wide range of methods (*method = "all"*), or a limited set of results by setting method to *"egger"*, *"ivw"*, or *"median"*. The final option is *"main"*, which gives results from the simple median, weighted median, IVW, and MR-Egger methods only.

```
MRInputObject <- mr_input(bx = ldlc,
                          bxse = ldlcse,
                          by = chdlodds,
                          byse = chdloddsse)

MRAllObject_all <- mr_allmethods(MRInputObject, method = "all")
MRAllObject_all
```

```
##                     Method Estimate Std Error 95% CI        P-value
```

```
##              Simple median    1.755     0.740   0.305 3.205   0.018
##            Weighted median    2.683     0.419   1.862 3.504   0.000
##  Penalized weighted median    2.681     0.420   1.857 3.505   0.000
##
##                        IVW    2.834     0.530   1.796 3.873   0.000
##              Penalized IVW    2.561     0.413   1.752 3.370   0.000
##                 Robust IVW    2.797     0.307   2.195 3.399   0.000
##       Penalized robust IVW    2.576     0.251   2.083 3.069   0.000
##
##                   MR-Egger    3.253     0.770   1.743 4.762   0.000
##                (intercept)   -0.011     0.015  -0.041 0.018   0.451
##         Penalized MR-Egger    3.421     0.531   2.380 4.461   0.000
##                (intercept)   -0.022     0.011  -0.043 0.000   0.051
##             Robust MR-Egger    3.256     0.624   2.033 4.479   0.000
##                (intercept)   -0.015     0.021  -0.055 0.026   0.474
##  Penalized robust MR-Egger    3.502     0.478   2.566 4.438   0.000
##                (intercept)   -0.026     0.014  -0.054 0.003   0.075
```

```
MRAllObject_egger <- mr_allmethods(MRInputObject, method = "egger")
MRAllObject_egger
```

```
##                     Method Estimate Std Error 95% CI        P-value
##                   MR-Egger    3.253     0.770   1.743 4.762   0.000
##                (intercept)   -0.011     0.015  -0.041 0.018   0.451
##         Penalized MR-Egger    3.421     0.531   2.380 4.461   0.000
##                (intercept)   -0.022     0.011  -0.043 0.000   0.051
##             Robust MR-Egger    3.256     0.624   2.033 4.479   0.000
##                (intercept)   -0.015     0.021  -0.055 0.026   0.474
##  Penalized robust MR-Egger    3.502     0.478   2.566 4.438   0.000
##                (intercept)   -0.026     0.014  -0.054 0.003   0.075
```

```
MRAllObject_main <- mr_allmethods(MRInputObject, method = "main")
MRAllObject_main
```

```
##           Method Estimate Std Error 95% CI        P-value
##    Simple median    1.755     0.740   0.305 3.205   0.018
##  Weighted median    2.683     0.419   1.862 3.504   0.000
##              IVW    2.834     0.530   1.796 3.873   0.000
##         MR-Egger    3.253     0.770   1.743 4.762   0.000
##      (intercept)   -0.011     0.015  -0.041 0.018   0.451
```

Graphical summaries of results

Applied to MRInput object - display of data

The *mr_plot* function has two different functionalities. First, if the function is applied to an *MRInput* object, then the output is an interactive graph that can be used to explore the associations of the different genetic variants and look for outliers, which may represent pleiotropic variants.

The syntax is:

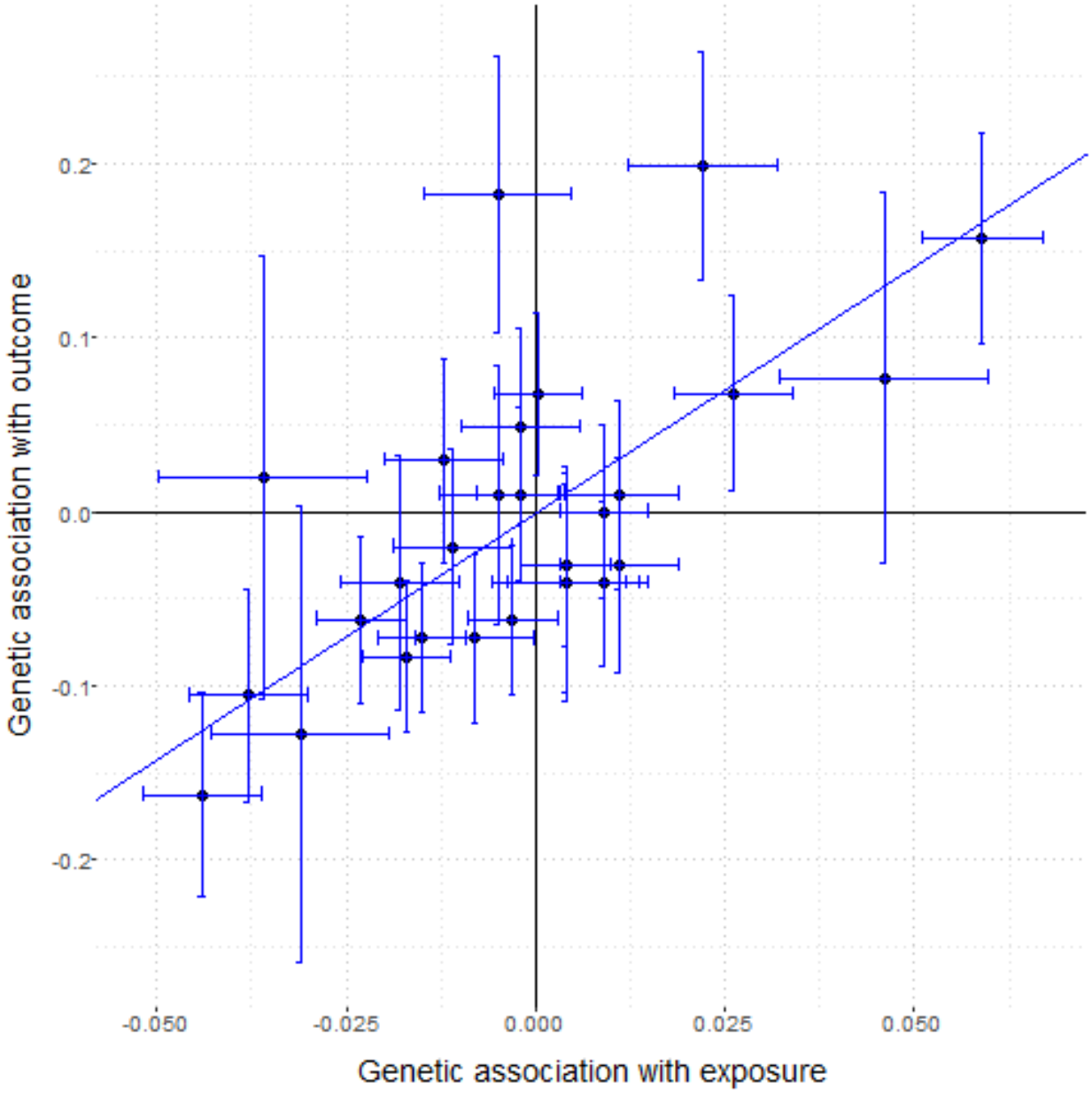
```
mr_plot(mr_input(bx = ldlc, bxse = ldlcse, by = chdlodds, byse = chdloddsse),
 error = TRUE, orientate = FALSE, line = "ivw")
```

An interactive graph does not reproduce well in a vignette, so we encourage readers to input this code for themselves. The interactive graph allows the user to pinpoint outliers easily; when the user mouses over one of the points, the name of the variant is shown.

The option *error = TRUE* plots error bars (95% confidence intervals) for the associations with the exposure and with the outcome. The option *orientate = TRUE* sets all the associations with the exposure to be positive, and re-orientates the associations with the outcome if needed. This option is encouraged for the MR-Egger method (as otherwise points having negative associations with the exposure can appear to be far from the regression line), although by default it is set to *FALSE*. The *line* option can be set to *"ivw"* (to show the inverse-variance weighted estimate) or to *"egger"* (to show the MR-Egger estimate).

A static version of this graph is also available, by setting *interactive = FALSE*:

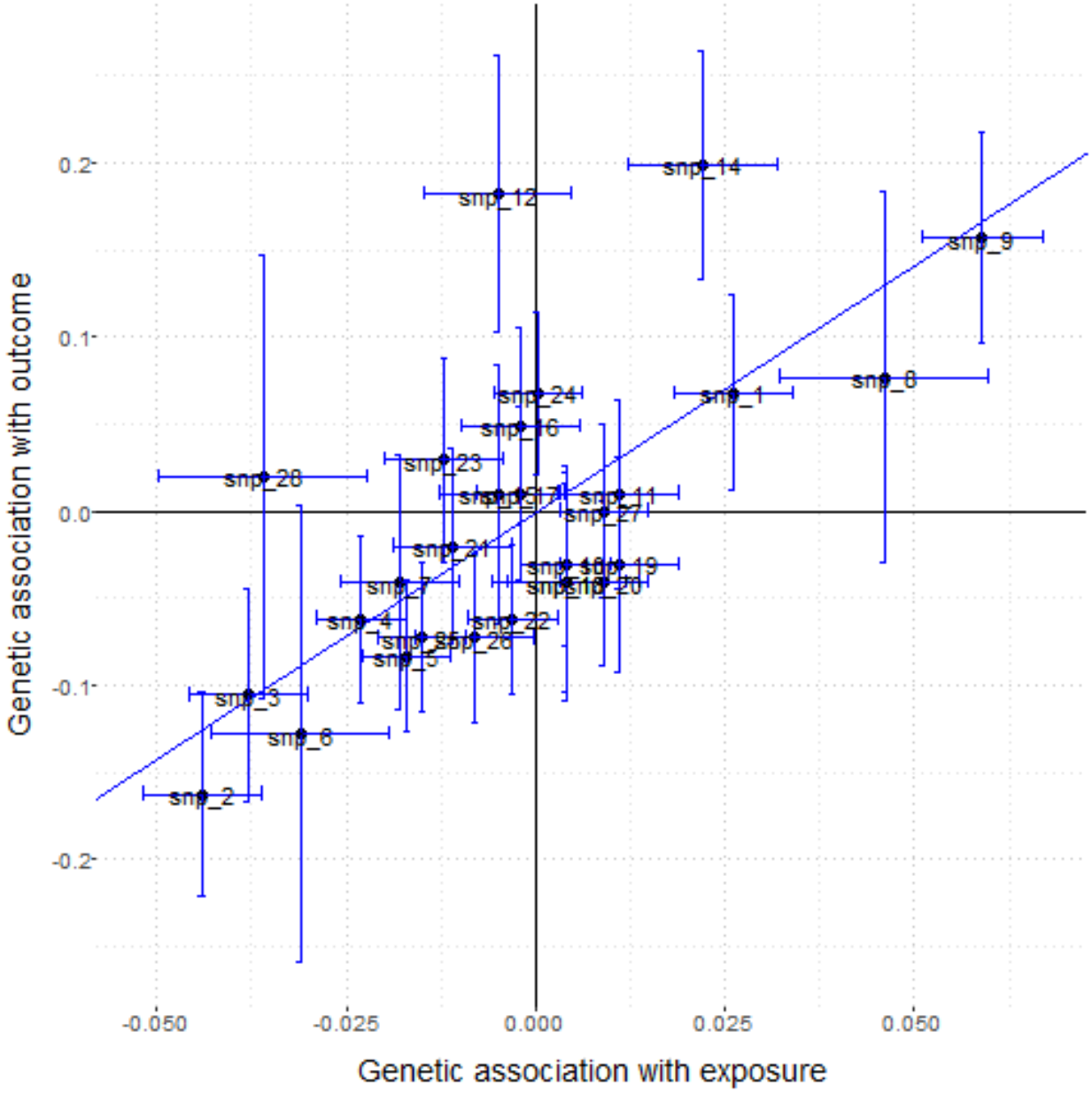
```
mr_plot(mr_input(bx = ldlc, bxse = ldlcse, by = chdlodds, byse = chdloddsse),
 error = TRUE, orientate = FALSE, line = "ivw", interactive = FALSE)
```



This version of the graph is less useful for detecting outliers, but easier to save as a file and share with colleagues.

Outliers can be identified using the *labels = TRUE* option:

```
mr_plot(mr_input(bx = ldlc, bxse = ldlcse, by = chdlodds, byse = chdloddsse),
 error = TRUE, orientate = FALSE, line = "ivw", interactive = FALSE, labels = TRUE)
```

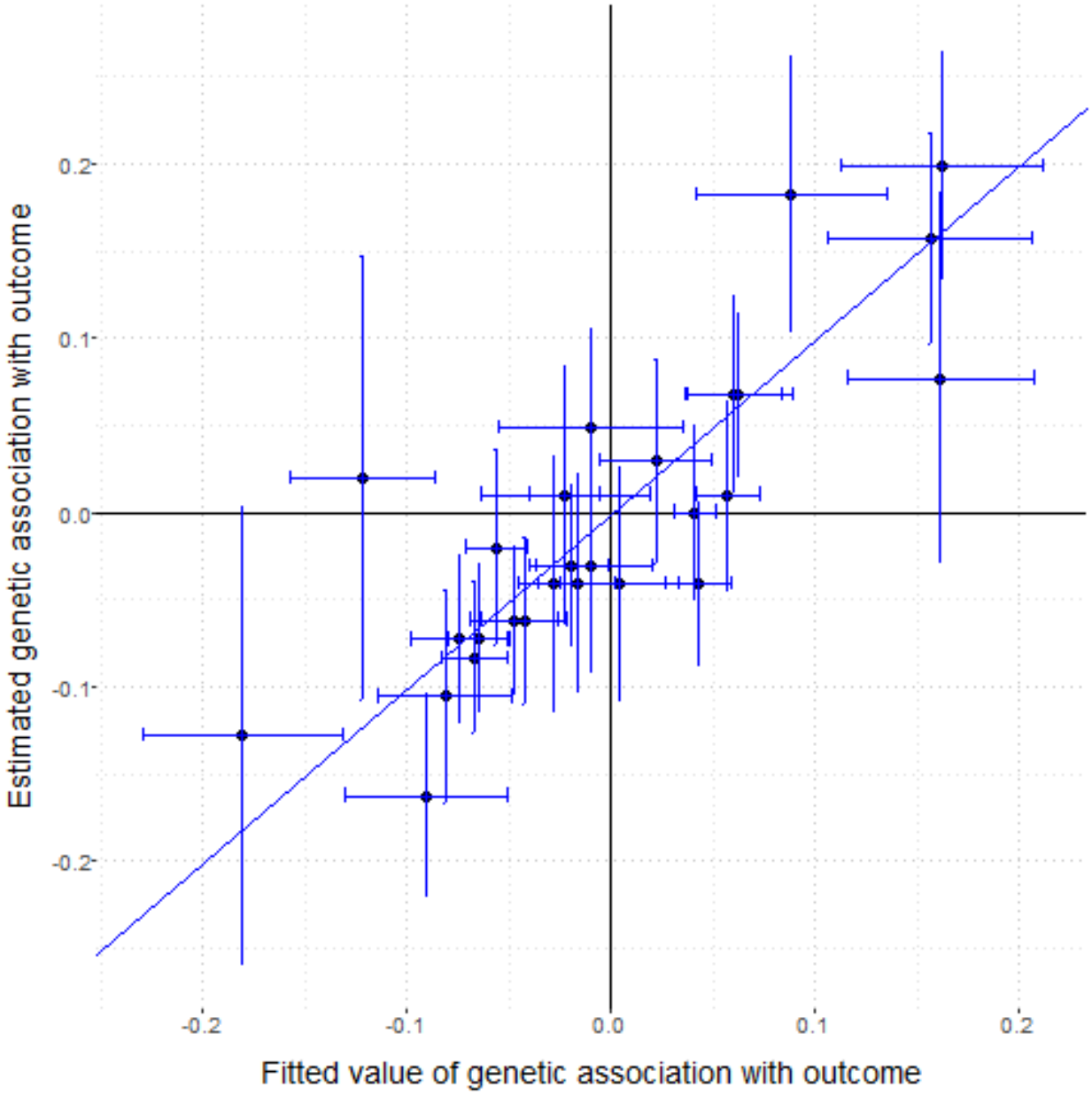


The resulting graph is quite ugly, but it is easy to identify the individual points.

Applied to MRMVInput object - display of data

The *mr_plot* command can also be applied to an *MRMVInput* object created using the *mr_mvinput()* function. In this case, the horizontal axis does not present the associations with any single exposure, but the fitted value from the multivariable IVW method, representing the expected genetic association with the outcome based on the genetic associations with the exposures.

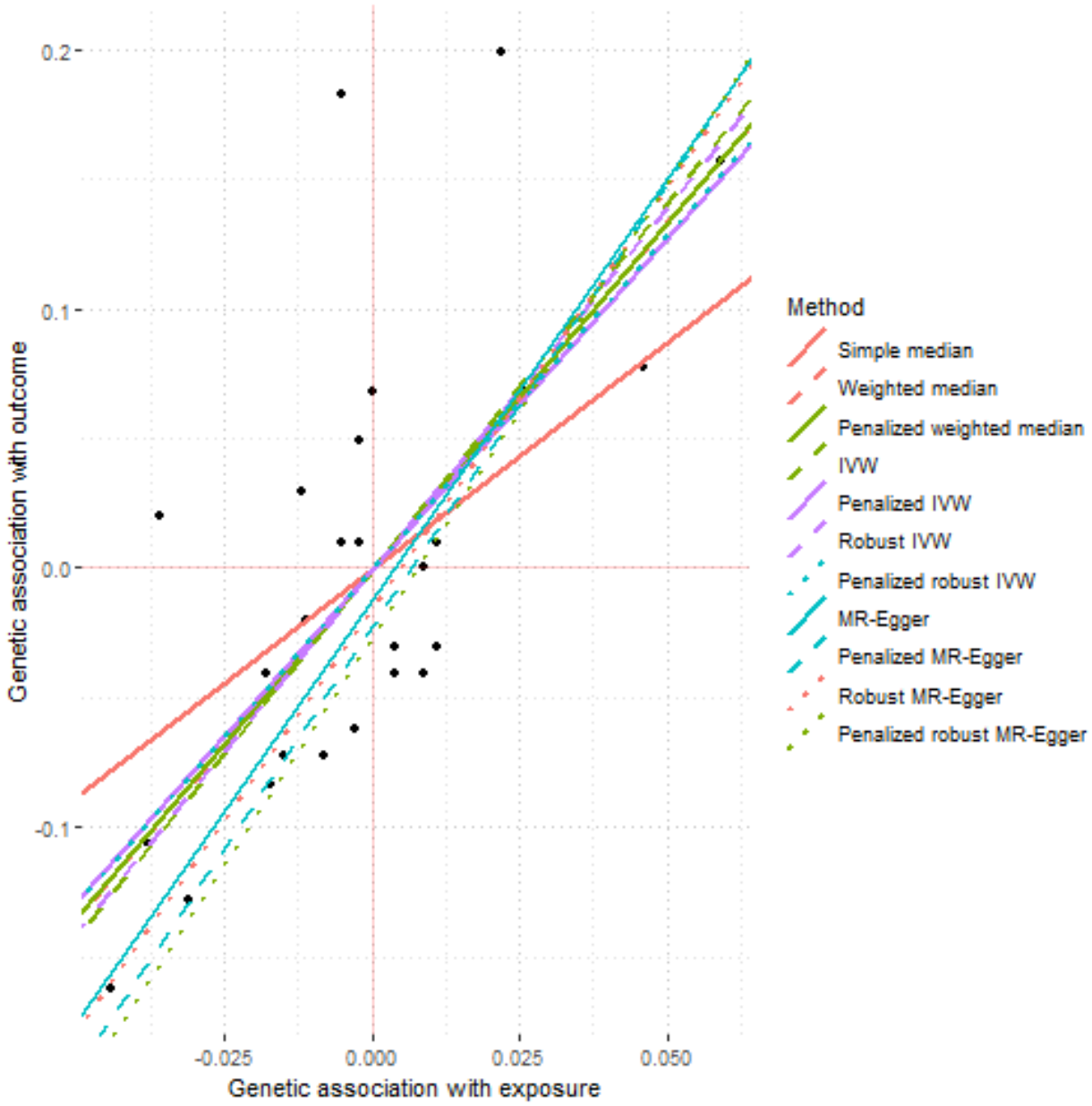
```
mr_plot(MVMRInputObject, interactive = FALSE)
```



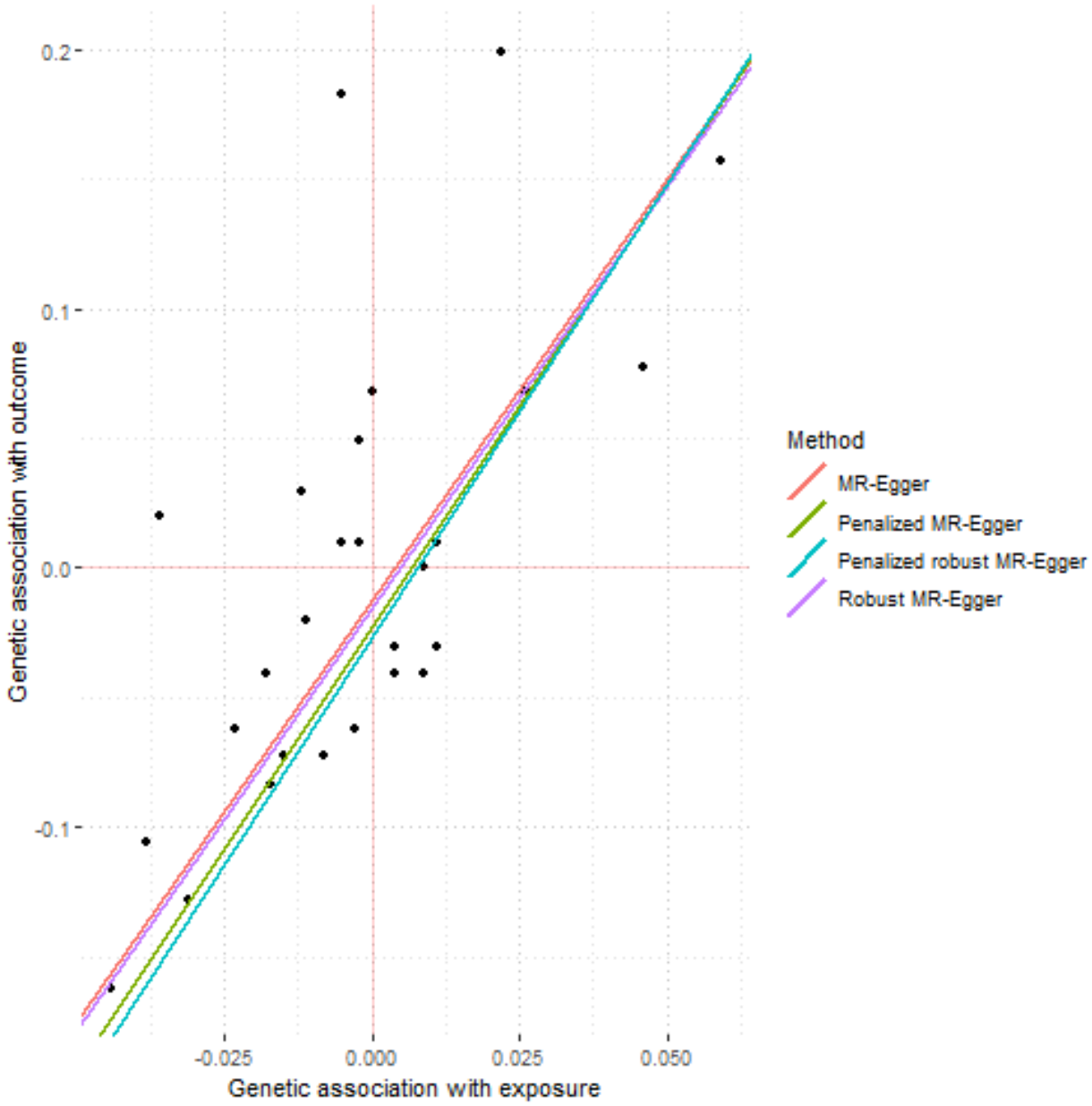
Applied to MRAll object - comparison of estimates

Finally, if the *mr_plot* function is applied to the output of the *mr_allmethods* function, estimates from the different methods can be compared graphically.

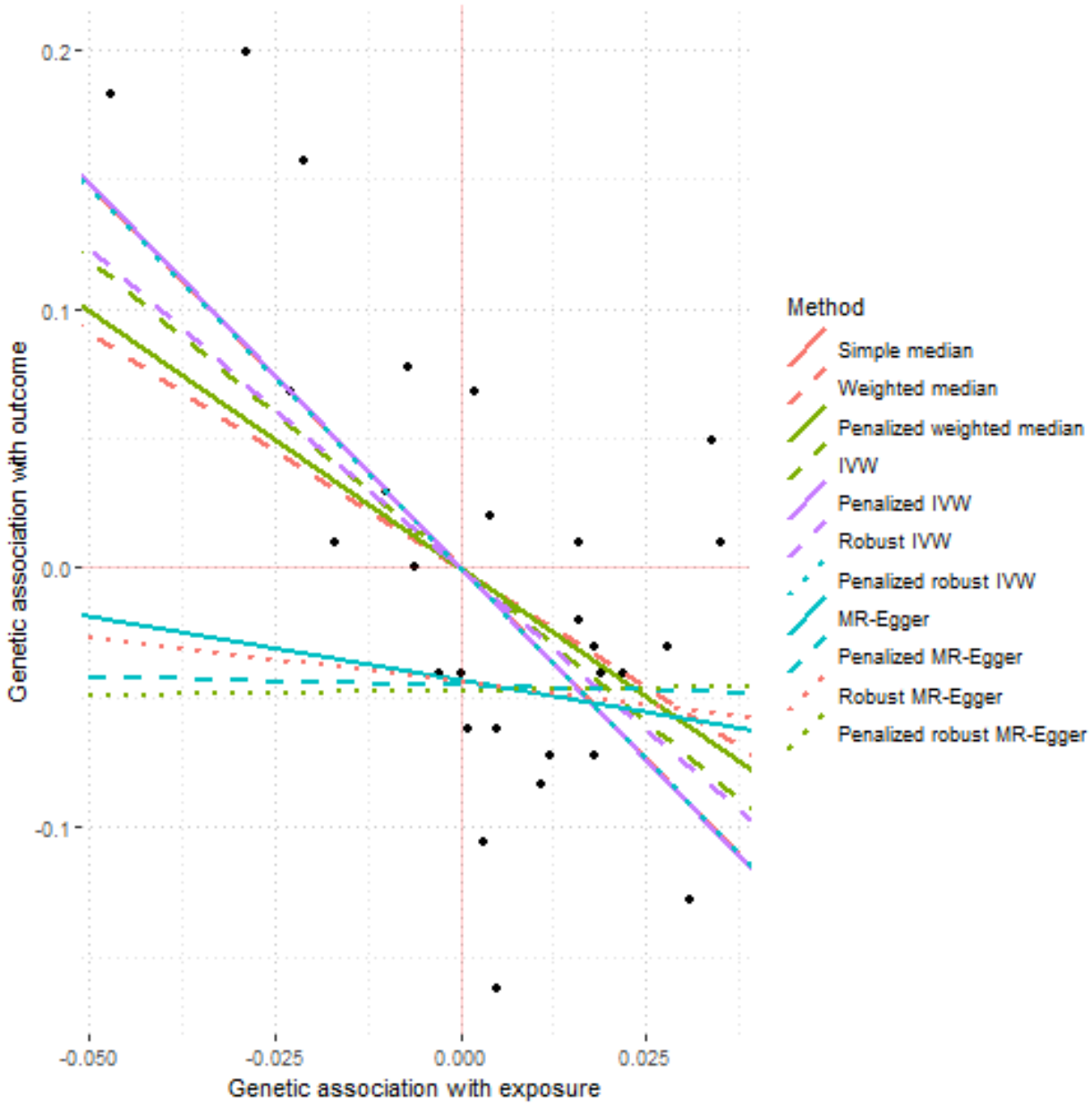
```
mr_plot(MRAllObject_all)
```




```
mr_plot(MRAllObject_egger)
```




```
mr_plot(mr_allmethods(mr_input(bx = hdlc, bxse = hdlcse,
  by = chdlodds, byse = chdloddsse)))
```

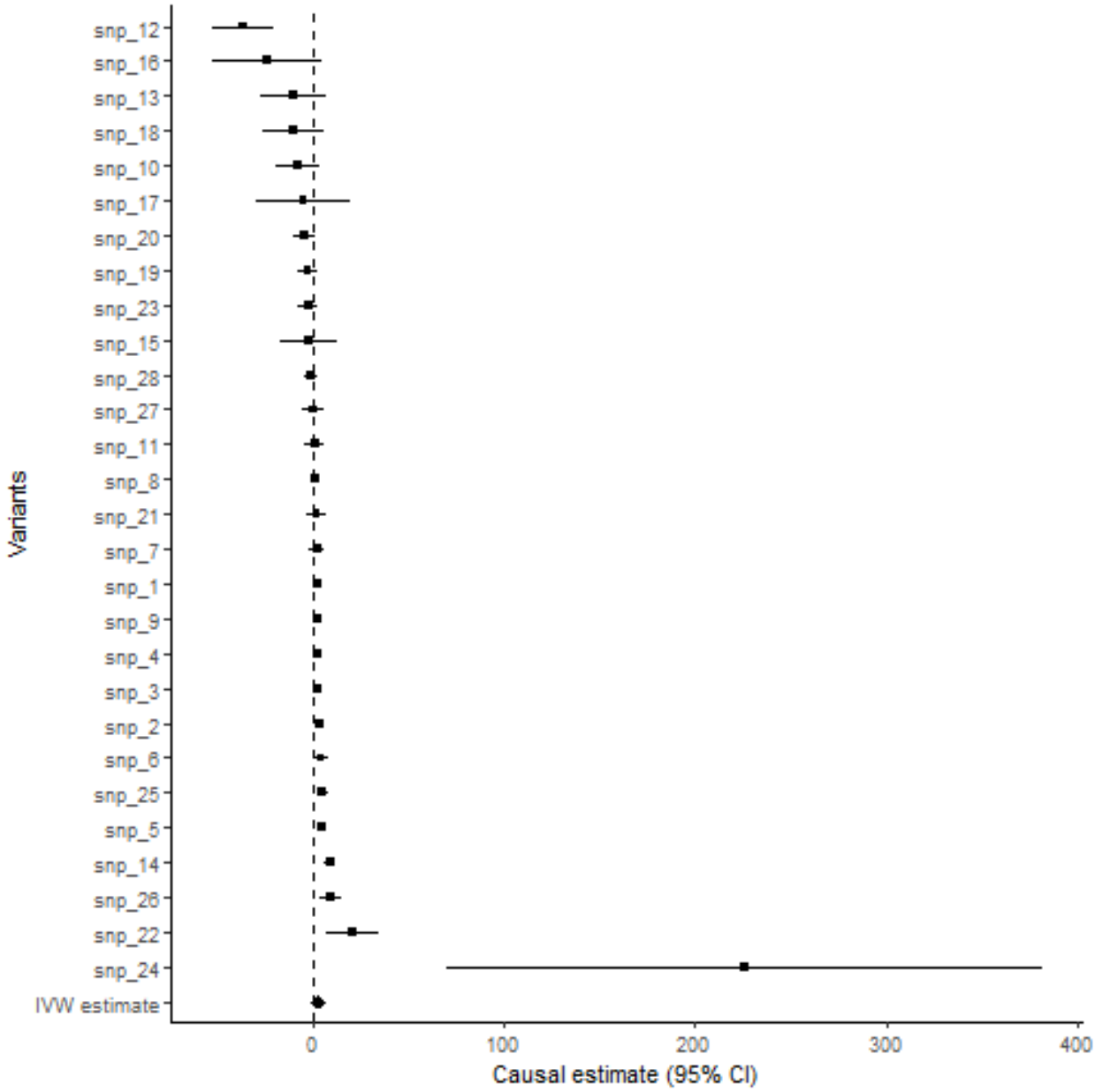


We see that estimates from all methods are similar when LDL-cholesterol is the risk factor, but the MR-Egger estimates differ substantially when HDL-cholesterol is the risk factor.

Other graphical functions

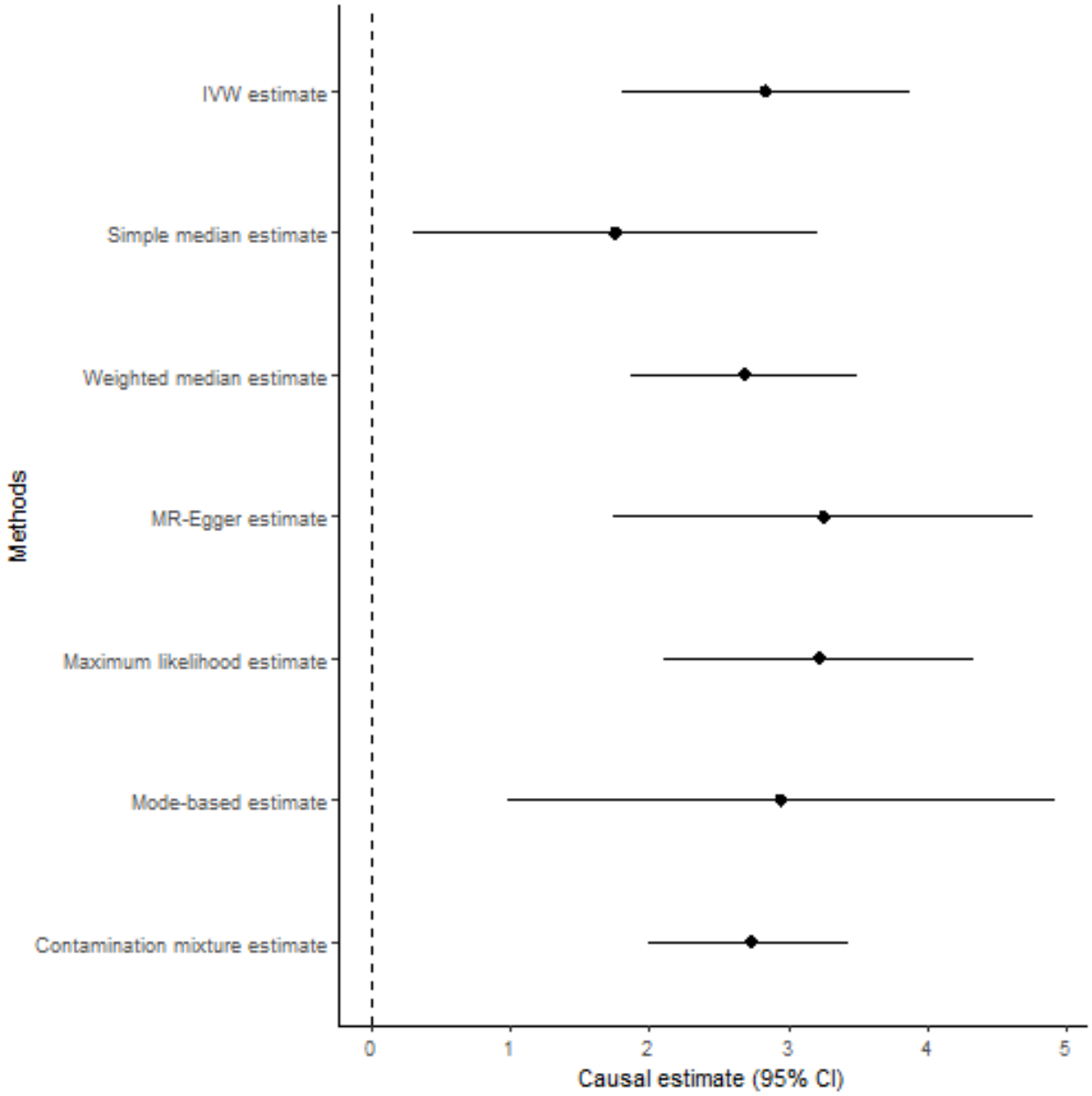
Other graphical functions are the *mr_forest* command, which can either provide a forest plot of the variant-specific estimates:

```
mr_forest(MRInputObject, ordered=TRUE)
```



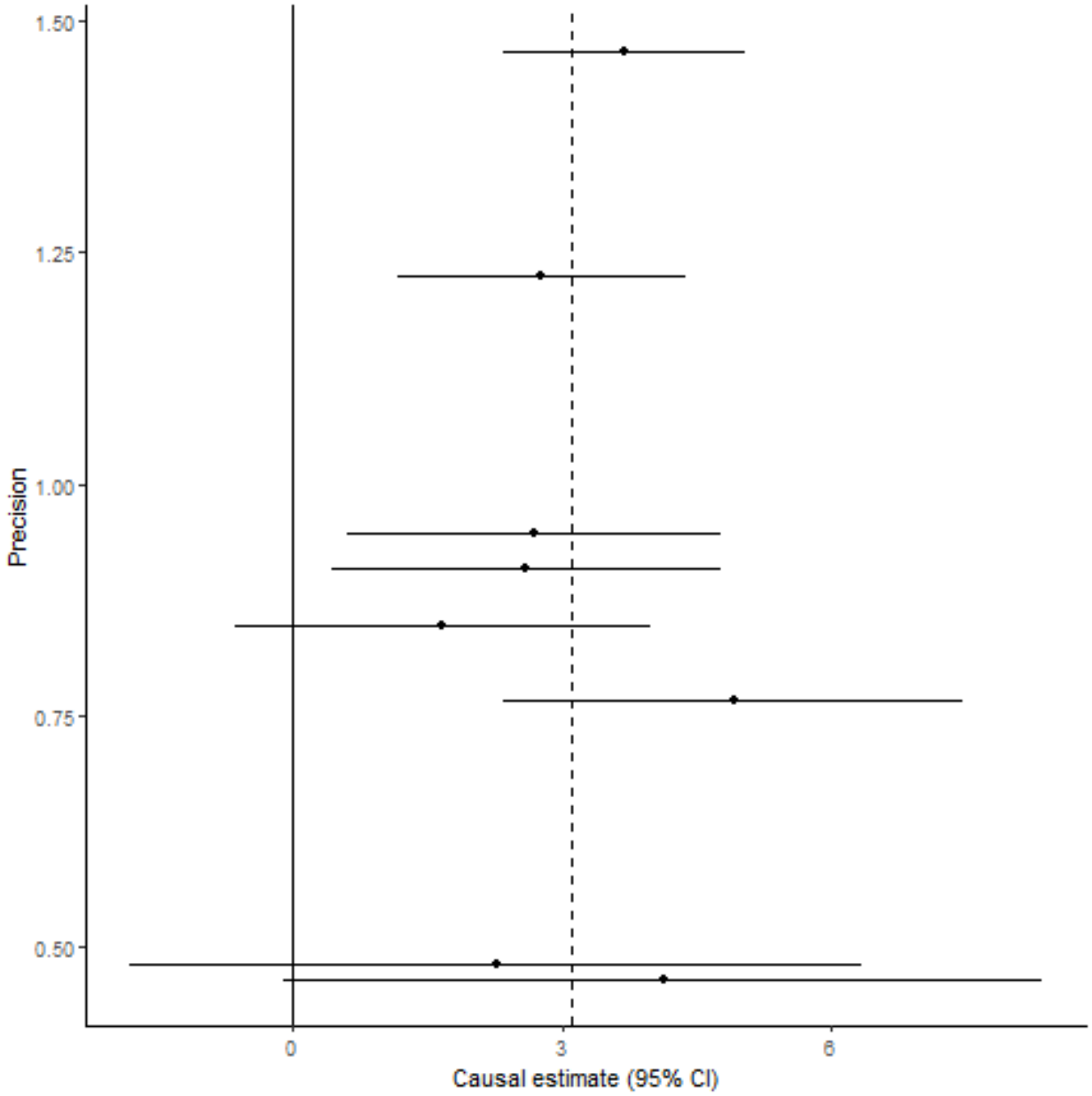
Or the comparison of various estimates from different methods:

```
mr_forest(MRInputObject,
          methods = c("ivw", "median", "wmedian", "egger", "maxlik", "mbe", "conmix"),
          snp_estimates = FALSE)
```



The *mr_funnel* command, which provides a funnel plot of the variant-specific estimates against their precision (that is, the reciprocal of the standard error):

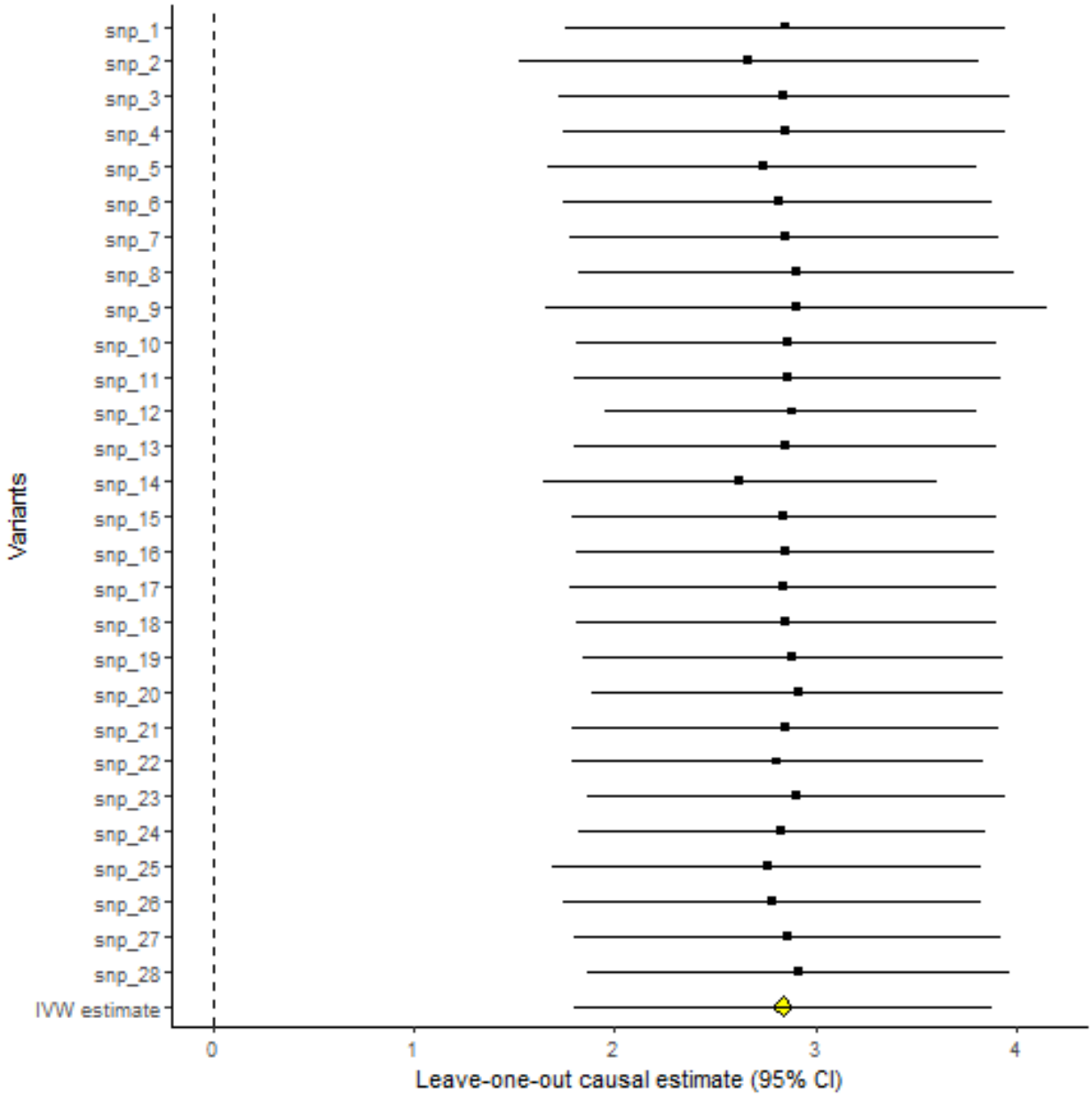
```
mr_funnel(mr_input(bx = ldlc[1:8], bxse = ldlcse[1:8],
          by = chdlodds[1:8], byse = chdloddsse[1:8]))
```



(Note we only plot the first 8 variants here)

And the *mr_loo* command, which provides the leave-one-out estimates - that is, the estimate based on all variants except variant 1, then the based on all variants except variant 2, and so on.

```
mr_loo(MRInputObject)
```



These plots are all *ggplot* objects, and so can be edited using commands from the *ggplot2* package. For example, to set the x-axis to run from -5 to +5:

```
library(ggplot2)
forest  = mr_forest(mr_input(ldlc, ldlcse, chdlodds, chdloddsse))
forest2 = forest + coord_cartesian(xlim=c(-5,5))
forest2
```

Extracting association estimates from PhenoScanner

The PhenoScanner bioinformatic tool (http://phenoscanner.medschl.cam.ac.uk) is a curated database of publicly available results from large-scale genetic association studies. The database currently contains over 65 billion associations and association results and over 150 million unique genetic variants, mostly single nucleotide polymorphisms.

PhenoScanner can be called directly from the MendelianRandomization package using the *pheno_input()* function. This creates an *MRInput* function, which can be directly used as an input to any of the estimation functions. For example:

```
mr_ivw(pheno_input(snps=c("rs12916", "rs2479409",
                          "rs217434", "rs1367117",
                          "rs4299376", "rs629301",
                          "rs4420638", "rs6511720"),
                   exposure = "Low density lipoprotein",
                   pmidE = "24097068",
                   ancestryE = "European",
                   outcome = "Coronary artery disease",
                   pmidO = "26343387",
                   ancestryO = "Mixed"))
```

(We do not implement this code here, as it requires a connection to the internet, and hence produces an error if an internet connection cannot be found. But please copy it and try it for yourself!)

In order to obtain the relevant summary estimates, run the *pheno_input()* function with:

- *snps* is a character vector giving the rsid identifiers of the genetic variants.
- *exposure* is a character vector giving the name of the risk factor.
- *pmidE* is the PubMed ID of the paper where the association estimates with the exposure were first published.
- *ancestryE* is the ancestry of the participants on whom the association estimates with the exposure were estimated. (For some traits and PubMed IDs, results are given for multiple ancestries.) Usually, ancestry is *"European"* or *"Mixed"*.
- *outcome* is a character vector giving the name of the outcome.
- *pmidO* is the PubMed ID of the paper where the association estimates with the outcome were first published.
- *ancestryO* is the ancestry of the participants on whom the association estimates with the exposure were estimated.

We note that the spelling of the exposure and outcome, the PubMed ID, and the ancestry information need to correspond exactly to the values in the PhenoScanner dataset. If these are not spelled exactly as in the PhenoScanner dataset (including upper/lower case), the association estimates will not be found.

Final note of caution

Particularly with the development of Mendelian randomization with summarized data, two-sample Mendelian randomization (where associations with the risk factor and with the outcome are taken from separate datasets), and bioinformatic tools for obtaining and analysing summarized data (including this package), Mendelian randomization is becoming increasingly accessible as a tool for answering epidemiological questions. This is undoubtably a Good Thing.

However, it is important to remember that the difficult part of a Mendelian randomization analysis is not the computational method, but deciding what should go into the analysis: which risk factor, which outcome, and (most importantly) which genetic variants. Hopefully, the availability of these tools will enable less attention to be paid to the mechanics of the analysis, and more attention to these choices.

The note of caution is that tools that make Mendelian randomization simple to perform run the risk of encouraging large numbers of speculative analyses to be performed in an unprincipled way. It is important that Mendelian randomization is not performed in a way that avoids critical thought. In releasing this package, the hope is that it will lead to more comprehensive and more reproducible causal inferences from Mendelian randomization, and not simply add more noise to the literature.