

MendelianRandomization v0.3.0: an R package for performing Mendelian randomization analyses using summarized data

Created and maintained by Stephen Burgess (sb452@medschl.cam.ac.uk) and Olena Yavorska (olena.yavorska@ludwig.ox.ac.uk)

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

```
library(MendelianRandomization)
```

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 object in parts. In order to make an *MRInput* object, one can do one of the following :

- assign values to each slot separately
- extract it from a PhenoScanner .csv output file

We focus on the first method.

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 outlining the 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.
- *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, several of the methods do not require *bxse* to be specified; 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.

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*, *fastglucose*: 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.

Methods

The MendelianRandomization package supports three main methods for causal estimation: the inverse-variance weighted method, the median-based method, and the MR-Egger method.

Inverse-variance weighted method

The inverse-variance method is the equivalent to the standard IV 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) causal 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.

ADDED IN VERSION 0.2.0 The default option for the weights is "simple", which corresponds to an inverse-variance weighted meta-analysis of the ratio estimates by/bx using standard errors $byse/bx$. The expression for the standard error of the ratio estimate is the first-order term of a delta expansion for the variance of a ratio. The corresponding weights in the regression analysis are $byse^{-2}$. These weights have been demonstrated to give reasonable inferences in realistic settings. However, they do not take into account uncertainty in the genetic associations with the exposure. Alternatively, the second-order term of the delta expansion is $\sqrt{byse^2/bx^2 + by^2*bxse^2/bx^4 - 2*psi*by*bxse*byse/bx^3}$, where *psi* is the correlation for each variant between its association with the exposure and with the outcome. This correlation will be zero (the default value) in a strict two-sample setting in which there is no overlap between the individuals used to estimate associations with the exposure and with the outcome. Paradoxically, although the standard error expression implied by the second-order weights (obtained by setting *weights* equal to "delta") is larger, resulting in wider confidence intervals for a single genetic variant, the use of second-order weights often leads to less heterogeneity between the estimates for the individual variants, and so in narrower confidence intervals.

```
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 = 99.5304 on 27 degrees of freedom, (p-value = 0.0000)

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 = fastglucose, 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 = 2.0530 on 5 degrees of freedom, (p-value = 0.8418)

```

Median-based method

The median-based method calculates a median of the SNP-specific causal 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.

The median-based method requires data on at least 3 genetic variants. Variants must be uncorrelated provided that the correlation matrix is specified.

```
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 median-based 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)
## I2GX statistic: 91.9%

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

EggerObject.corr <- mr_egger(mr_input(bx = calcium, bxse = calciumse,
                                       by = fastgluc, byse = fastglucose, 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      2.413 -3.427, 6.031  0.589
##           (intercept)   0.009      0.021 -0.032, 0.050  0.666
##           -----
## 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

ADDED IN VERSION 0.2.0 An additional method for causal estimation 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. This method has not been used much in the literature, due to its similarity to the inverse-variance weighted method (both methods use the same underlying model), and its relative complexity: the method involves maximizing a likelihood that has one parameter for each genetic variant, plus a causal effect parameter. For large numbers of genetic variants, this entails optimization over a large parameter space, which can lead to numerical issues in the optimization algorithm.

The method naturally assumes a fixed-effect model -- the same causal effect is estimated by each of the genetic variants. However, if there is heterogeneity between the causal estimates from the different genetic variants, then confidence intervals from a fixed-effect model will be overly precise. We implement a random-effects model by multiplying the standard error from the fixed-effect model by an estimate of the residual standard error obtained: the likelihood ratio heterogeneity test statistic divided by the number of genetic variants less 1, and then square rooted (provided this quantity is greater than 1, otherwise no modification to the standard error is made). This is similar to the residual standard error in a regression model, which can be calculated from the Cochran Q heterogeneity test statistic by the same steps. As with the IVW method, the default option is to implement a fixed-effect model with three variants or fewer, and a random-effects analysis otherwise.

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 ψ is set to zero (the default option). If the associations are estimated in the same individuals (complete sample overlap), then the value of ψ should be the observed correlation between the exposure and the outcome.

```
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 = fastglucose, 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)

```

Multivariable Mendelian randomization

ADDED IN VERSION 0.3.0 An additional method for causal estimation 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, causal 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*:

```

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

```

```

MRMVInputObject
##      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	

The `mr_mvivw` command performs an extension of the inverse-variance weighted method for multivariable Mendelian randomization, performing multivariable weighted linear regression (the standard IVW 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.

```
MRMVObject <- mr_mvivw(MRMVInputObject,
                        model = "default",
                        correl = FALSE,
                        distribution = "normal",
```

```

alpha = 0.05)

MRMVObject <- mr_mvivw(MRMVInputObject)

MRMVObject

##
## 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)

```

Mode-based estimation method of Hartwig et al

ADDED IN VERSION 0.3.0 An additional method for robust causal 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 mean of these estimates, the mode-based method obtains the "mode" of the 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 above (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

ADDED IN VERSION 0.3.0 An additional method for robust causal 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 (2017) "Improving on a modal-based estimation method: model averaging for consistent and efficient estimation in Mendelian randomization when a plurality of candidate instruments are valid", doi: 10.1101/175372.

There are several aspects of the mode-based estimation method that could be improved upon. The choice of bandwidth is arbitrary and has a large effect on estimates and their confidence intervals. The bootstrap procedure is not ideal, as it makes the assumption that the estimate is normally distributed. The method is not particularly efficient.

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 causal estimates. As such, the method will identify the subset with the largest number (by weight) of variants having similar causal 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 causal 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 choosing the minimum (*CIMin*) and maximum (*CIMax*) values in the search, as well as the step size (*CISStep*). The default values will not be suitable for all applications.

```
HetPenObject <- mr_hetpen(MRInputObject,
                          prior = 0.5,
                          CIMin = -1,
                          CIMax = 1,
                          CISStep = 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, CISStep = 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 range of values.
```

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 all methods (`method = "all"`), or a partial result setting method to `"egger"`, `"ivw"`, or `"median"` to get a more selective set of results. 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
## Penalyzed weighted median  2.681    0.420  1.857 3.505   0.000
##
##              IVW             2.834    0.530  1.796 3.873   0.000
## Penalyzed IVW             2.561    0.413  1.752 3.370   0.000
##              Robust IVW      2.797    0.307  2.195 3.399   0.000
## Penalyzed 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.

The syntax is:

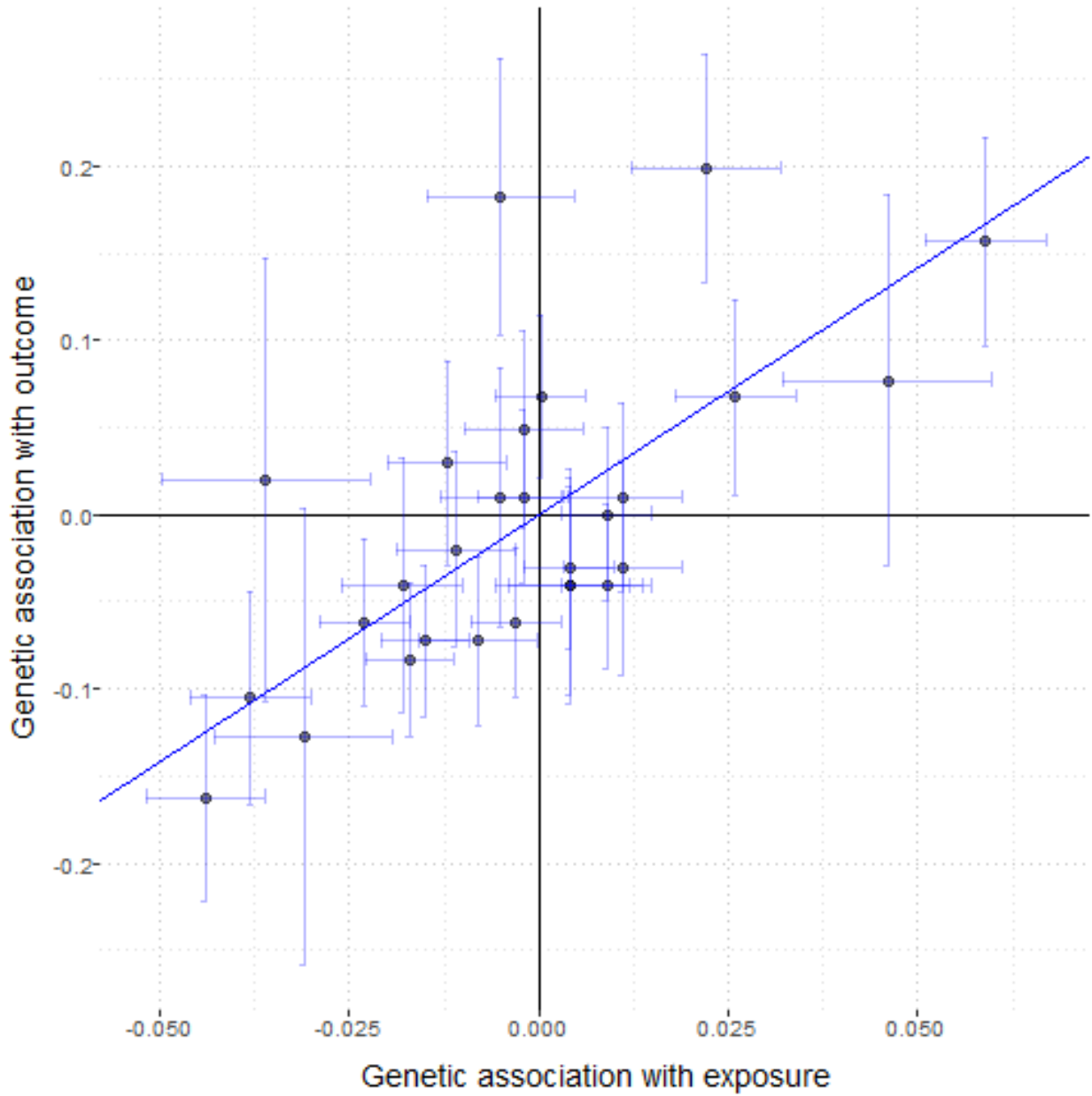
```
mr_plot(mr_input(bx = ldlc, bxse = ldlcse, by = chdlodds, byse = chdloddsse),
  error = TRUE, orientate = FALSE, line = "ivw")
```

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).

ADDED IN VERSION 0.2.0 A static version of this graph is also available, by setting `interactive = FALSE`:

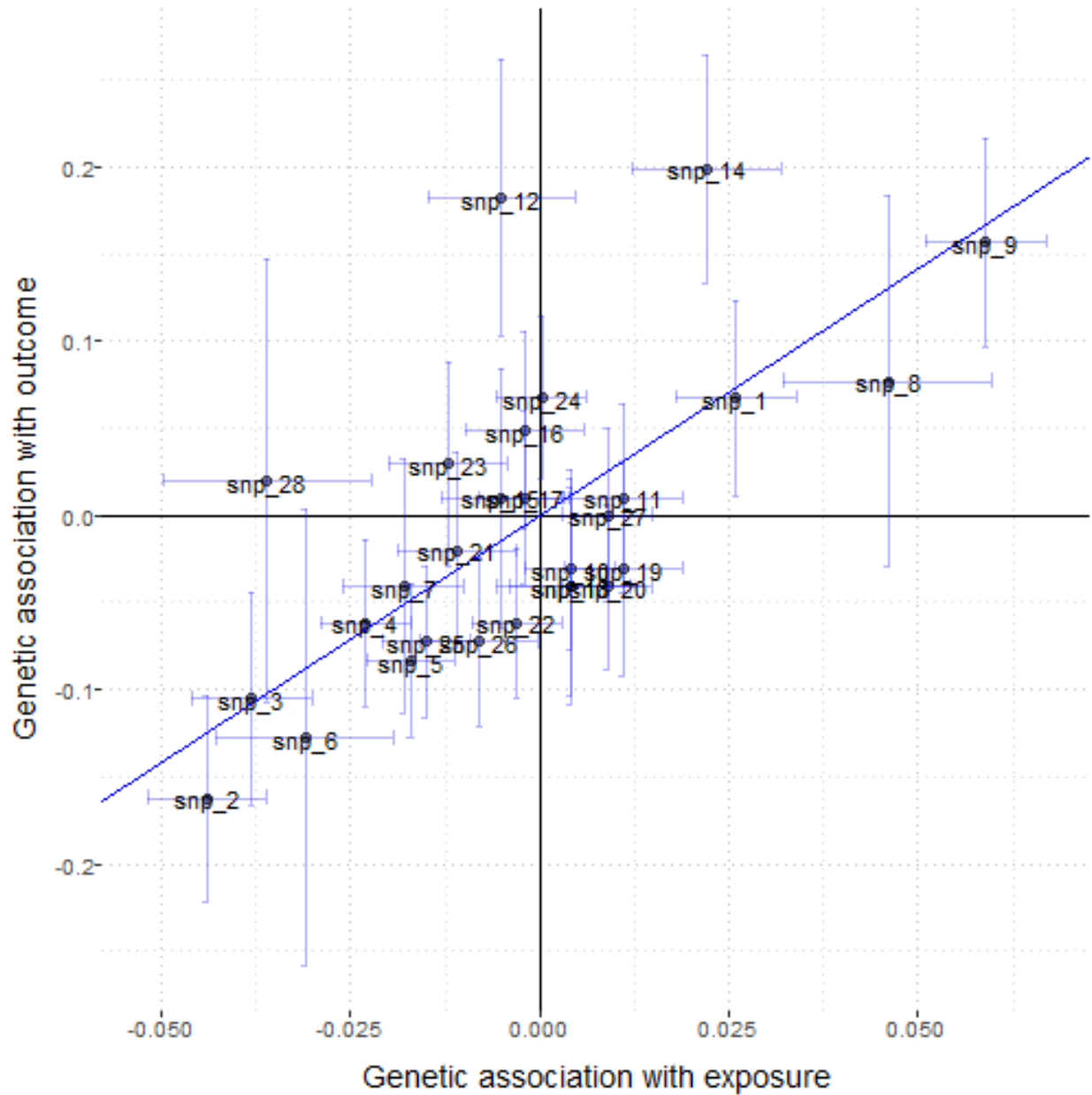
```
mr_plot(mr_input(bx = ldlc, bxse = ldlcse, by = chdlodds, byse = chdloddsse),
  error = TRUE, orientate = FALSE, line = "ivw", interactive = FALSE)
```



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 detected 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)
```

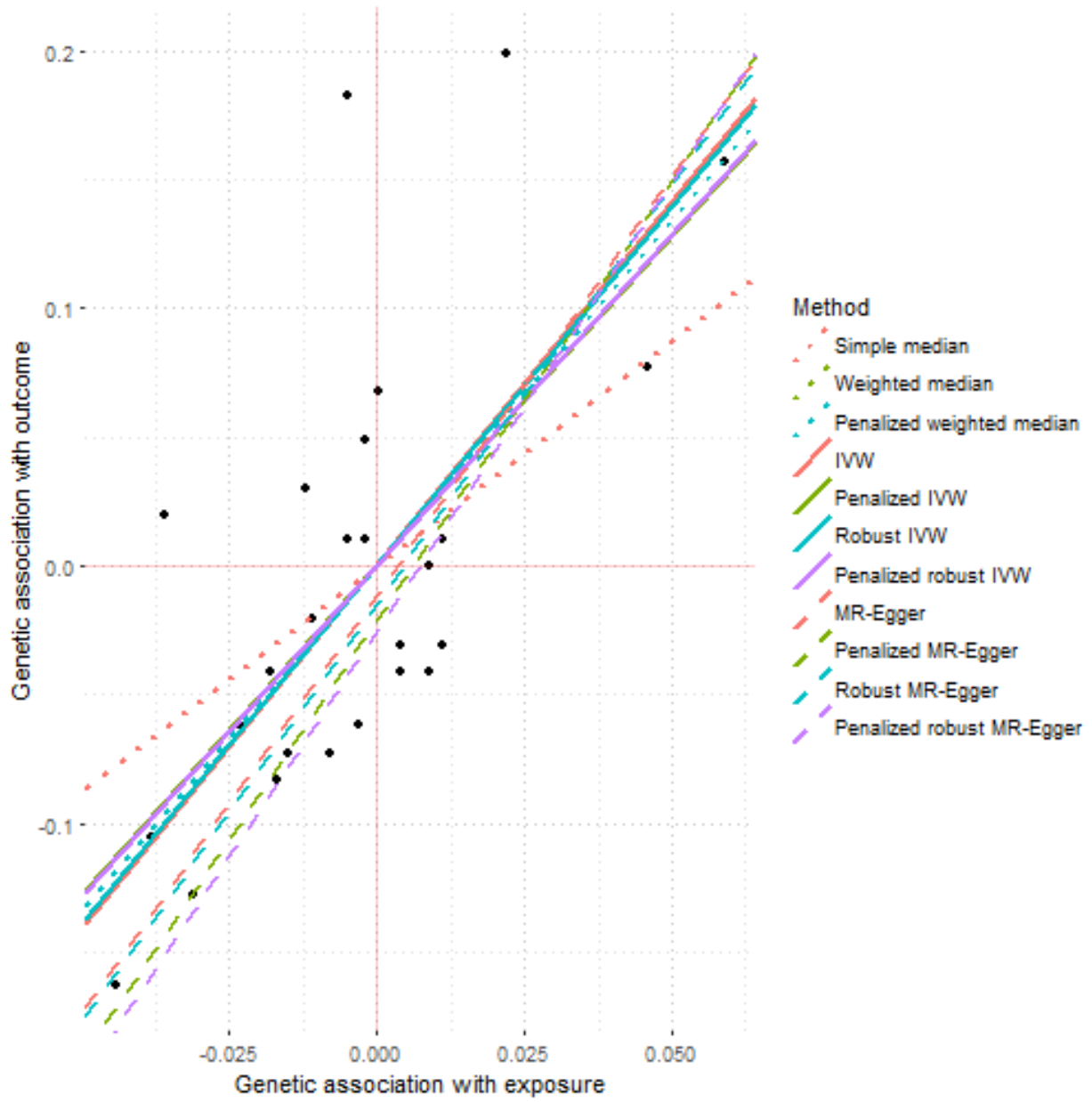


The resulting graph is quite ugly, but it is easy to identify the individual points.

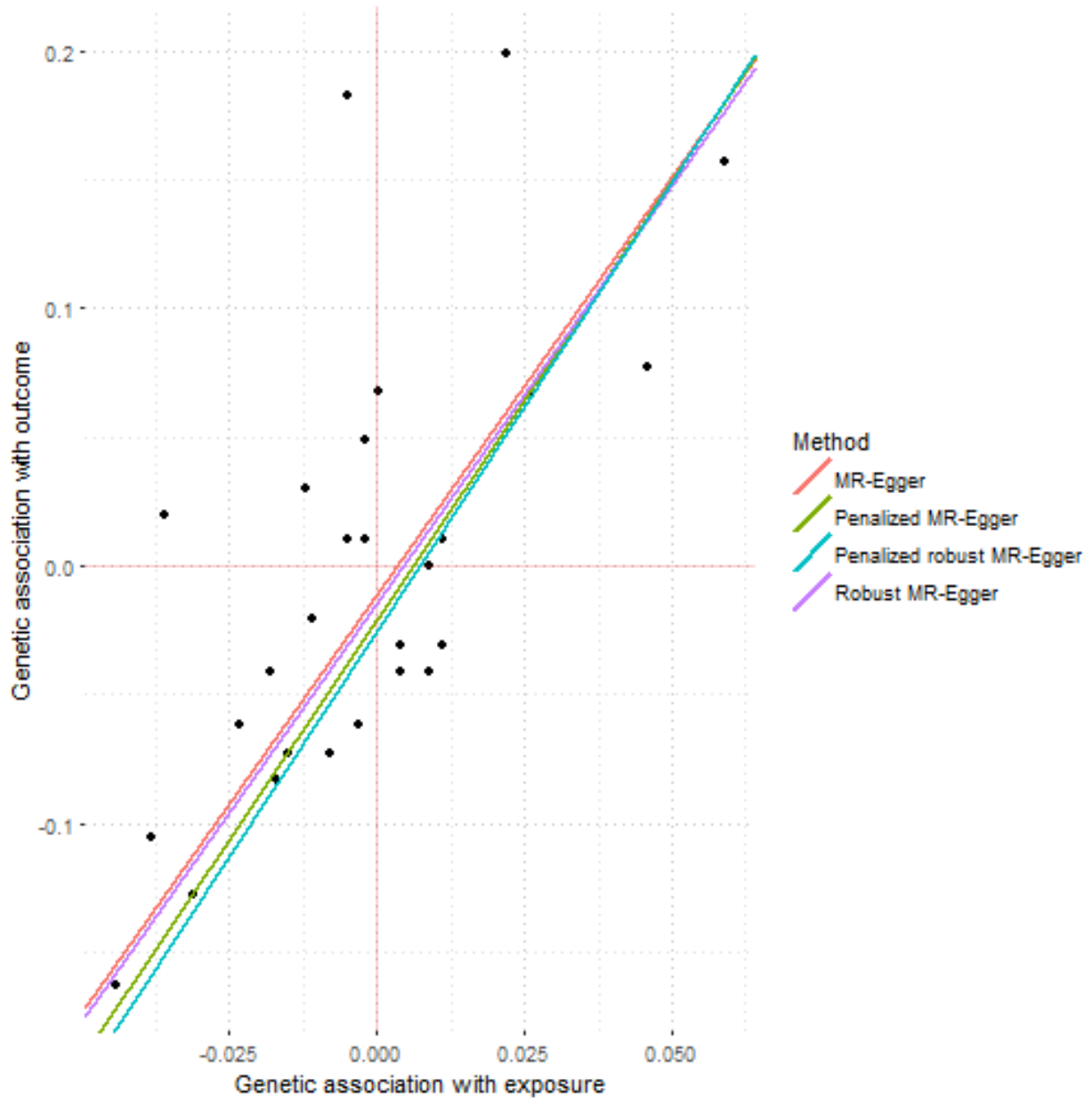
Applied to MRAll object - comparison of estimates

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

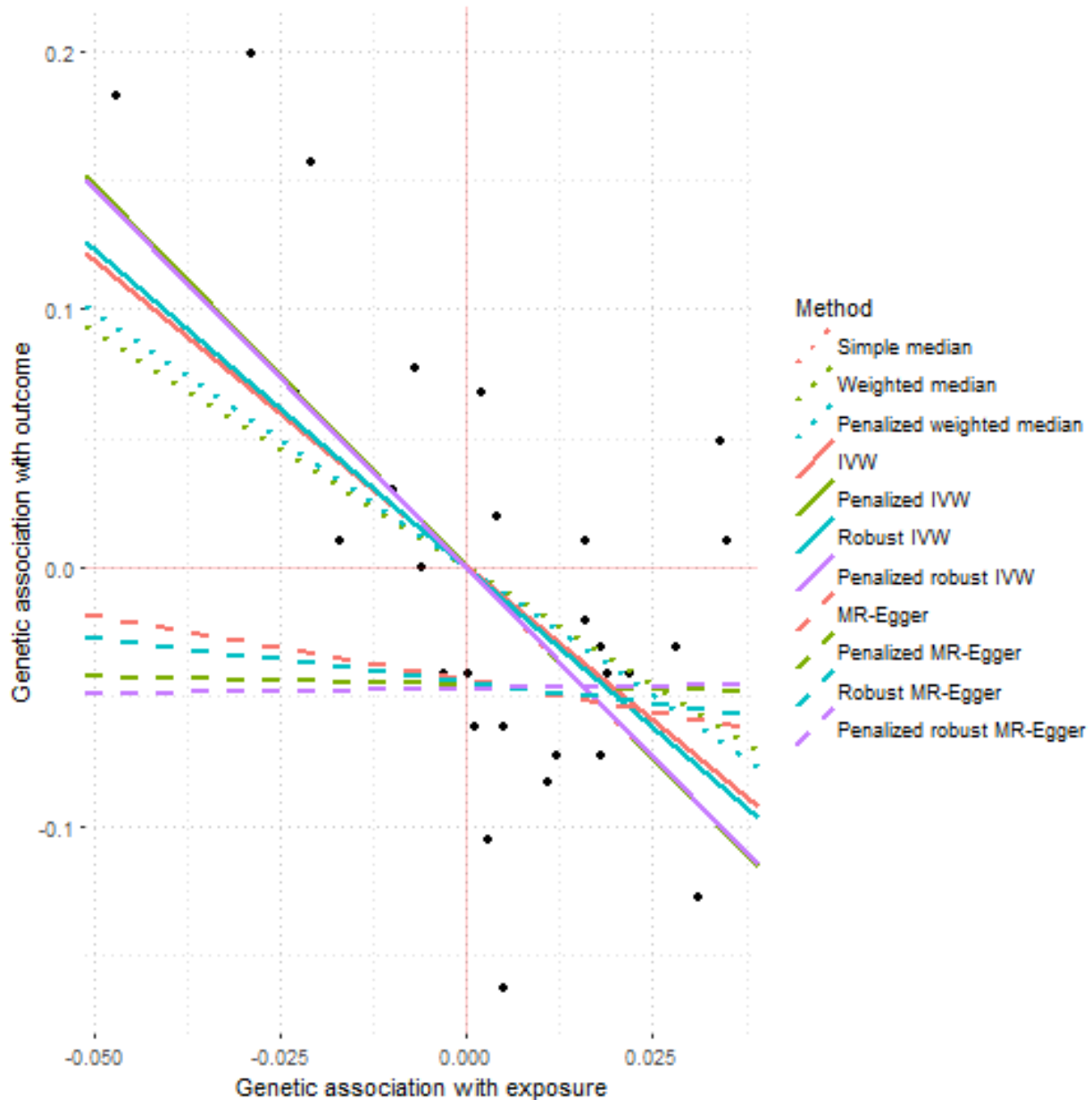
```
mr_plot(MRAllObject_all)
```



`mr_plot(MRA11Object_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.

Extracting association estimates from publicly available datasets

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 350 million association results and over 10 million unique genetic variants, mostly single nucleotide polymorphisms.

Our desire is to enable PhenoScanner to be queried directly from the MendelianRandomization package. Currently, PhenoScanner is only available via a web browser. The `extract.pheno.csv()` function takes the output from the web version of PhenoScanner, and converts this into an `MRInput` object. PhenoScanner is still under development, and so `extract.pheno.csv()` should be considered as an experimental function. This function is designed for output from PhenoScanner version 1.1 (Little Miss Sunshine).

The initial steps required to run the `extract.pheno.csv()` function are:

1. Open <http://www.phenoscanner.medschl.cam.ac.uk/> in your browser.
2. Input the SNPs in question by uploading a file with the rsIDs or hg19 chrN:pos names (separated by newline).
3. Choose whether to include proxies or not for SNPs that do not have association data for the given risk factor and/or outcome (and if so, the R-squared threshold value for defining a proxy variant). PhenoScanner currently allows correlation estimates to be taken from the 1000 Genomes or HapMap datasets.
4. Run the analysis and download the resulting association .csv file.

In order to obtain the relevant SNP summary estimates, run the `extract.pheno.csv()` function with:

- `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.
- `file` is the file path to the PhenoScanner output .csv file.
- `rsq.proxy` is the threshold R-squared value for proxies to be included in the analysis. If a proxy variant is used as part of the analysis, this is reported. The default value of `rsq.proxy` is 1, meaning that only perfect proxies will be used.
- `snps` is the SNPs that will be included in the analysis. The default value is "all", indicating that all SNPs in the .csv file will be used in the analysis. If only a limited number of SNPs are to be included, `snps` can be set as a character vector with the named to the SNPs to be included in the analysis.

Two example .csv files from PhenoScanner are included as part of the package: `vitD_snps_PhenoScanner.csv` (which does not include proxies), and `vitD_snps_PhenoScanner_proxies.csv` (which includes proxies at an R-squared threshold of 0.6).

```
path.noproxy <- system.file("extdata", "vitD_snps_PhenoScanner.csv",
  package = "MendelianRandomization")
path.proxies <- system.file("extdata", "vitD_snps_PhenoScanner_proxies.csv",
  package = "MendelianRandomization")
```

```
extract.pheno.csv(
  exposure = "log(eGFR creatinine)", pmidE = 26831199, ancestryE = "European",
  outcome = "Tanner stage", pmidO = 24770850, ancestryO = "European",
  file = path.noproxy)
```

```
##          SNP log(eGFR creatinine).beta log(eGFR creatinine).se
## 1 rs10741657          -0.0018          0.00092
## 2 rs17217119          -0.0066          0.00110
## 3 rs2282679          -0.0016          0.00100
##  Tanner stage.beta Tanner stage.se
## 1          -0.0136          0.0135
## 2           0.0227          0.0165
## 3          -0.0042          0.0148
```

```
extract.pheno.csv(
```

```

exposure = "log(eGFR creatinine)", pmidE = 26831199, ancestryE = "European",
outcome = "Tanner stage", pmidO = 24770850, ancestryO = "European",
rsq.proxy = 0.6,
file = path.proxies)

## Variant rs4944958 used as a proxy for rs12785878 in association
## with the outcome. R^2 value: 1.000.

##          SNP log(eGFR creatinine).beta log(eGFR creatinine).se
## 1 rs10741657          -0.0018          0.00092
## 2 rs12785878           0.0024          0.00100
## 3 rs17217119          -0.0066          0.00110
## 4 rs2282679           -0.0016          0.00100
## Tanner stage.beta Tanner stage.se
## 1          -0.0136          0.0135
## 2           0.0102          0.0155
## 3           0.0227          0.0165
## 4          -0.0042          0.0148

extract.pheno.csv(
  exposure = "log(eGFR creatinine)", pmidE = 26831199, ancestryE = "European",
  outcome = "Asthma", pmidO = 20860503, ancestryO = "European",
  rsq.proxy = 0.6,
  file = path.proxies)

## Variant rs2060793 used as a proxy for rs10741657 in association
## with the outcome. R^2 value: 1.000.
## Variant rs7944926 used as a proxy for rs12785878 in association
## with the outcome. R^2 value: 1.000.

##          SNP log(eGFR creatinine).beta log(eGFR creatinine).se Asthma.beta
## 1 rs10741657          -0.0018          0.00092          0.00897
## 2 rs12785878           0.0024          0.00100         -0.00353
## 3 rs17217119          -0.0066          0.00110         -0.03906
## 4 rs2282679           -0.0016          0.00100         -0.01241
## Asthma.se
## 1           0.0202
## 2           0.0229
## 3           0.0255
## 4           0.0220

```

The output from the *extract.pheno.csv* function is an MRInput object, which can be plotted using *mr_plot*, causal estimates can be obtained using *mr_ivw*, and so on.

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 undoubtedly 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.