MendelianRandomization v0.1.0: an R package for performing Mendelian randomization analyses using summarized data

Created and maintained by Olena Yavorska (yavorsko@tcd.ie) and Stephen Burgess (sb452@medschl.cam.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, hdlse, 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.

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.

```
IVWObject <- mr_ivw(MRInputObject,
                    model = "default",
                    robust = FALSE,
                    penalized = FALSE,
                    correl = FALSE,
                    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 = 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 = 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)
## 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)
```

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.

```
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               2.734     0.339   2.069  3.399   0.000
## Robust IVW                  2.797     0.307   2.195  3.399   0.000
## Penalized robust IVW        2.730     0.257   2.226  3.235   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          3.589     0.456   2.695  4.483   0.000
## (intercept)                 -0.027    0.010   -0.046 -0.007  0.007
## 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   3.626     0.441   2.762  4.489   0.000
## (intercept)                 -0.030    0.011   -0.051 -0.009  0.005
## ------------------------------------------------------------------
```

```
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          3.589     0.456   2.695  4.483   0.000
## (intercept)                 -0.027    0.010   -0.046 -0.007  0.007
## 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   3.626     0.441   2.762  4.489   0.000
## (intercept)                 -0.030    0.011   -0.051 -0.009  0.005
## ------------------------------------------------------------------
```

```
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    -0.011   0.770  0.015  1.743
## ----------------------------------------------------------
```

Graphical summaries of results

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:

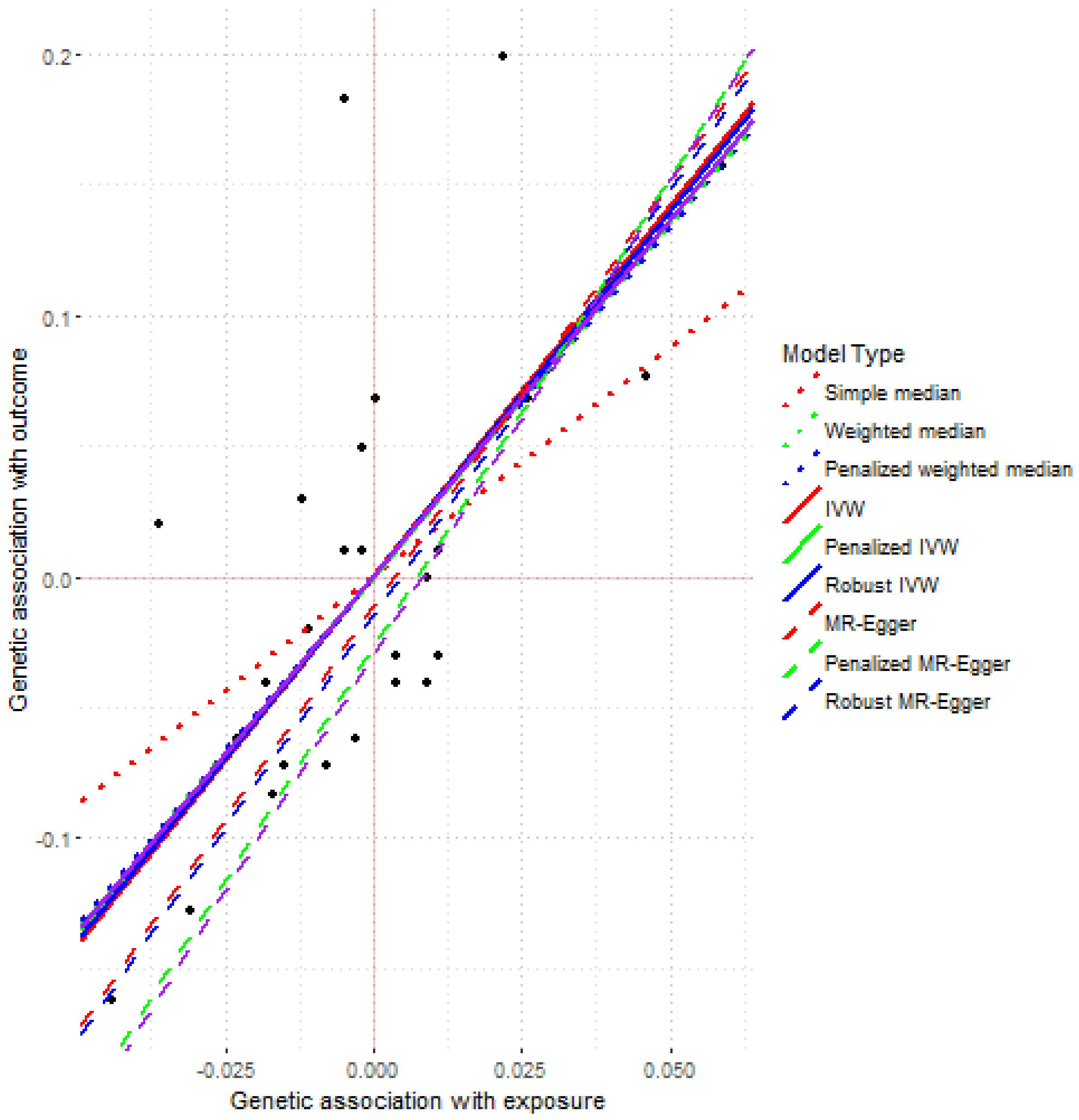
```
mr_plot(mr_input(bx = ldlc, bxse = ldlcse, by = chdlodds, byse = chdloddsse), error = TRUE,
   orientate = FALSE, line = "ivw")
```

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.

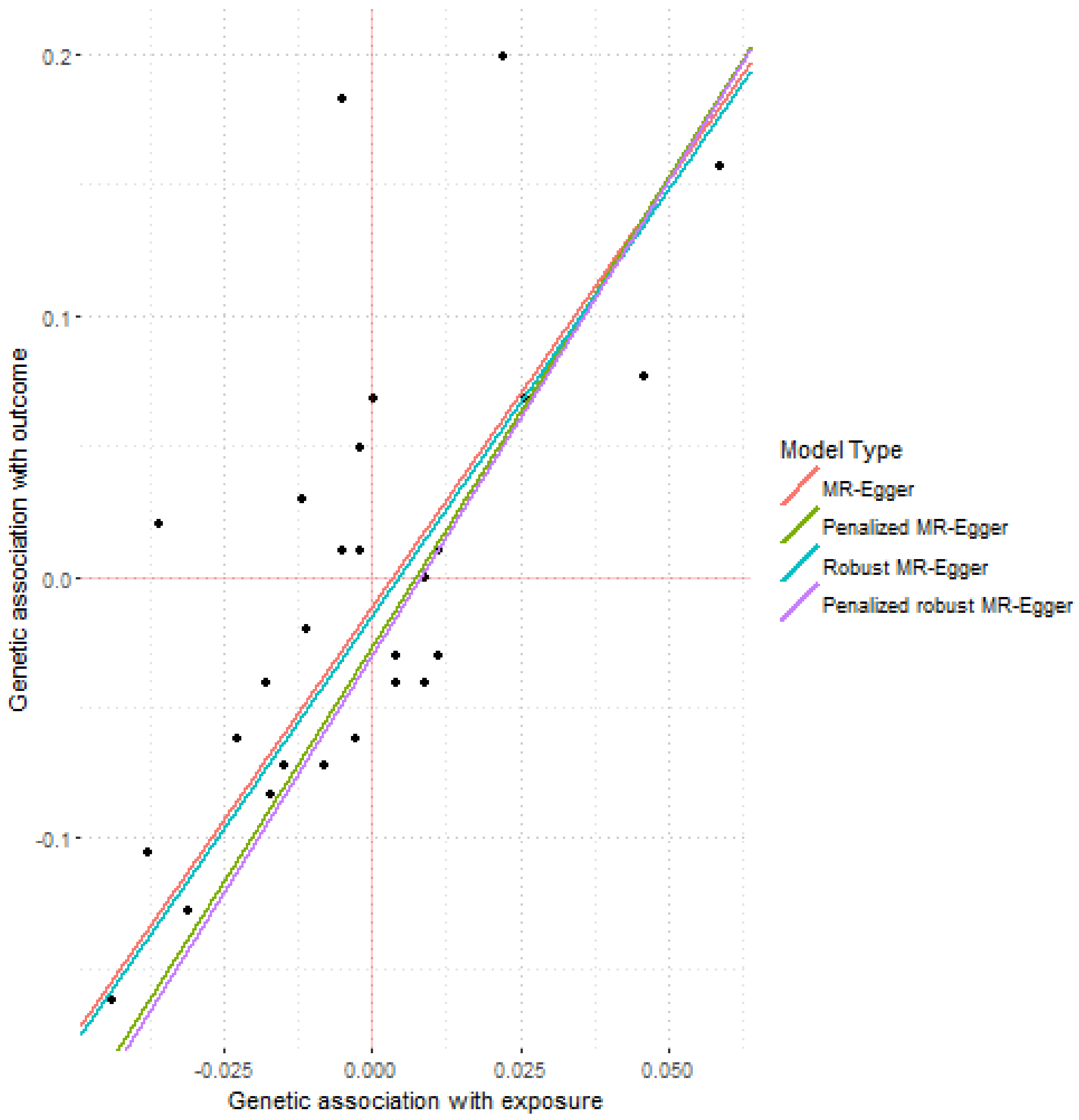
The option `error = TRUE` plots error bars (95% confidence intervals) for the associations with the exposure and with the outcome. The option `orientate = TRUE` sets all the associations with the exposure to be positive, and re-orientates the associations with the outcome if needed. This option is encouraged 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).

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.

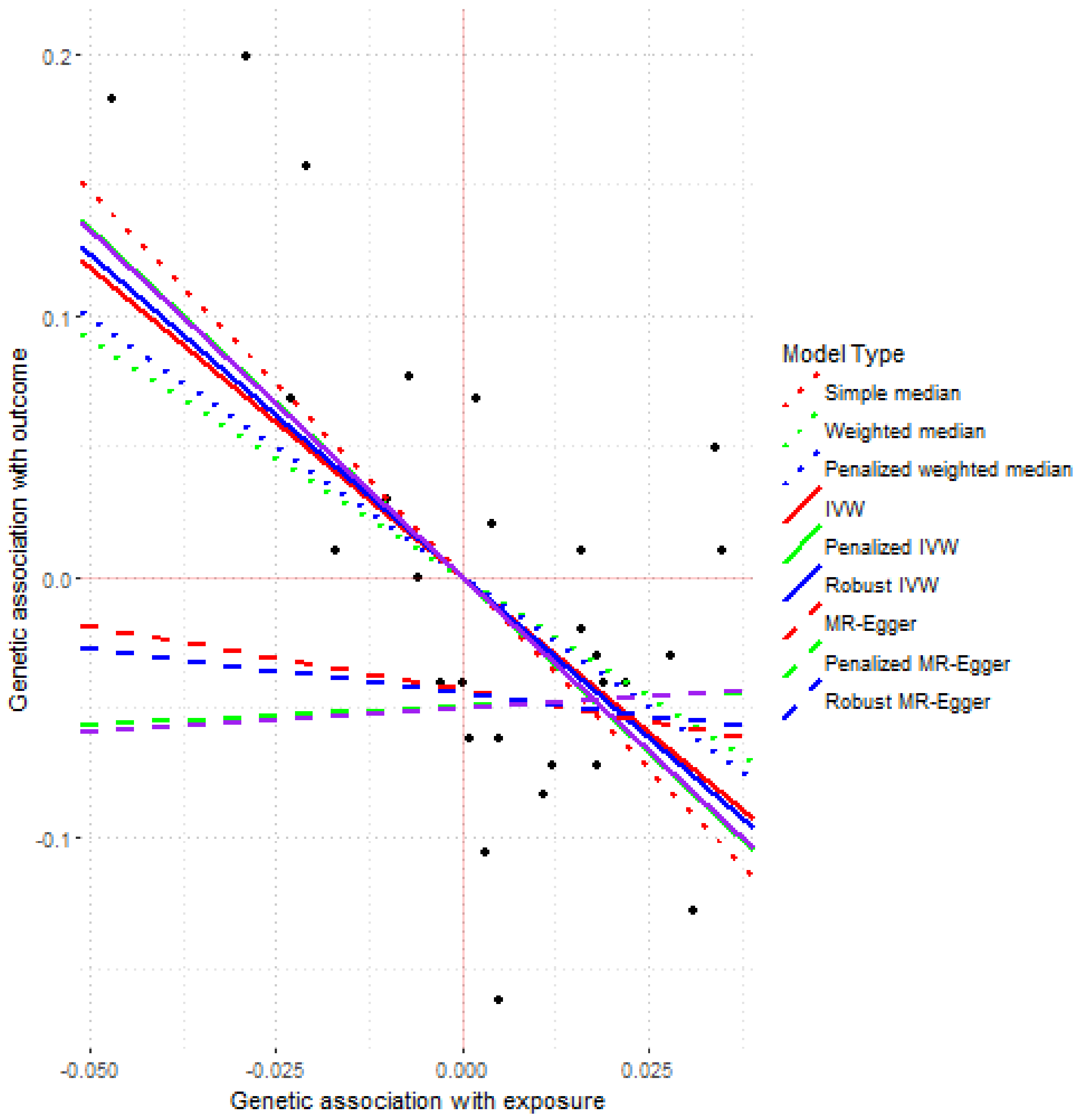
```
mr_plot(MRAllObject_all)
```



```
mr_plot(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.

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.
- `ancestryE` 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  rs2282679                   -0.0066                 0.00110
## 3 rs17217119                   -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  rs2282679                   -0.0066                 0.00110
## 4 rs17217119                   -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  rs2282679                   -0.0066                 0.00110    -0.03906
## 4 rs17217119                   -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 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.