An integrative multi-context Mendelian randomization method for identifying risk genes across human tissues

Yihao Lu, Ke Xu, Bowei Kang, Brandon L. Pierce, Fan Yang and Lin S. Chen

Introduction

This vignette provides an introduction to the `mintMR` package. R package `mintMR` implements the mintMR method for integrative multi-context Mendelian randomization.

Install the development version of `mintMR` by use of the `devtools` package. Note that `mintMR` depends on the `CCA`, `Rcpp`, and `RcppArmadillo` package.

To install this package, run the following command in R

```
library(devtools)
install_github("ylustat/mintMR")
```

Load the package using the following command:

```
library("mintMR")
```

This vignette depends on R packages `MendelianRandomization`, `ggplot2` and `tidyverse`, you can load these two packages using the following command:

```
suppressMessages(library("tidyverse"))
library("ggplot2")
library("MendelianRandomization")
```

Fit mintMR for correlated SNPs using simulated data

In this section, we fit mintMR using simulated data (provided in the package) as an example to illustrate basic usage of mintMR. `gamma_hat` is the IV-to-Exposure effect and standard error, `Gammah_hat` is the IV-to-Outcome effect and standard error. `LD` is the estimated genetic correlations among IVs. `latent status` is the true underlying effect status, where `1` indicates true non-zero effect and `0` indicates true zero effect.

```
data(example_data)
names(example_data)
```

```
## [1] "gamma_hat"     "Gamma_hat"     "se_g"          "se_G"
## [5] "LD"            "latent_status"
```

The example data include 50 simulated genes, each of them with 5 contexts.

```
L <- length(example_data$gamma_hat)
K1 <- K2 <- 5
L
```

```
## [1] 50
```

The mintMR requires the information for which context is included for each exposure (the `group` variable). The `group` variable is a list, where each element is a vector of column indices for the `gamma_hat` and `se_g`.

```
group <- list(exposure1 = 1:K1, exposure2 = 1:K2+K1)
# column 1 to 5 in gamma_hat and se_g are from exposure 1
# column 6 to 10 in gamma_hat and se_g are from exposure 2
group
```

```
## $exposure1
## [1] 1 2 3 4 5
##
## $exposure2
## [1]  6  7  8  9 10
```

In addition, mintMR takes IV-to-exposure effect (a list of $I_g \times \sum K_l$ matrices with length L), IV-to-outcome effect and corresponding standard errors. The LD matrices (`corr_mat`) and sample overlap (`Lambda`) are optional.

Default prior parameters used in the algorithm is

```
opts <- get_opts(L)
names(opts)
```

```
##  [1] "a_gamma" "b_gamma" "a_alpha" "b_alpha" "a_beta"  "b_beta"  "a"
##  [8] "b"       "maxIter" "thin"    "burnin"
```

It can be changed by specifying the parameters you want to use. For example:

```
opts <- get_opts(L, maxIter = 1000)
```

MintMR can be run without specifying LD or sample overlap using following command.

```
set.seed(1)
res_no_LD <- mintMR(gammah = example_data$gamma_hat,
                    Gammah = example_data$Gamma_hat,
                    se1 = example_data$se_g,
                    se2 = example_data$se_G,
                    group = group)
```

When the LD information is available and formatted as a list of matrices, it can be accounted for:

```
set.seed(1)
res <- mintMR(gammah = example_data$gamma_hat,
              Gammah = example_data$Gamma_hat,
              se1 = example_data$se_g,
              se2 = example_data$se_G,
              corr_mat = example_data$LD,
              group = group)
```

When sample overlap information $\widehat{C}$ is available as a $(1 + \sum_l K_l) \times (1 + \sum_l K_l)$ matrix, it can be used in the function as

```
set.seed(1)
# Example correlation matrix for sample overlap. In real data, it can be pre-estimated.
C <- 0.9 * diag(1,(1+K1+K2)) +
  0.1 * matrix(1,nrow=1+K1+K2,ncol=1+K1+K2)

Lambda <- solve(C)

res_LD_sample_overlap <- mintMR(gammah = example_data$gamma_hat,
                                Gammah = example_data$Gamma_hat,
```

```
                                se1 = example_data$se_g,
                                se2 = example_data$se_G,
                                corr_mat = example_data$LD,
                                Lambda = Lambda,
                                group = group)
```

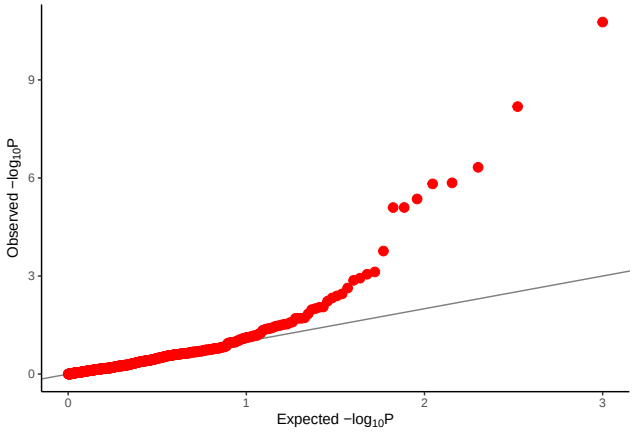
The output includes the effect estimates and p-values.

```
names(res)
```

```
## [1] "Estimate" "Pvalue"
```

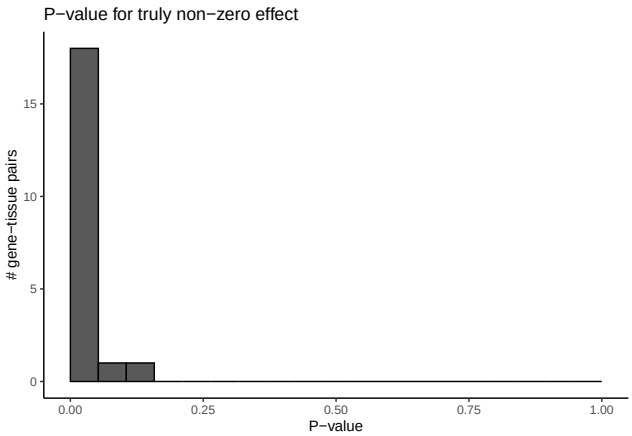
The p-value distribution for all gene-tissue pairs analyzed can be visualized as below.

```
p <- data.frame(mintMR = as.vector(res$Pvalue))
p %>%
  arrange(mintMR) %>%
  mutate(unif = ppoints(n())) %>%
  mutate(value = -log10(mintMR), unif = -log10(unif)) %>%
  ggplot(aes(y = value, x = unif))+
  geom_abline(intercept = 0, slope = 1, alpha = 0.5)+
  geom_point(size = 3, col = "red", alpha=1)+
  theme_classic()+
  labs(y = expression(paste("Observed -log"[10], plain(P))),
       x = expression(paste("Expected -log"[10], plain(P))))
```



The p-value distribution for truly non-zero effects:

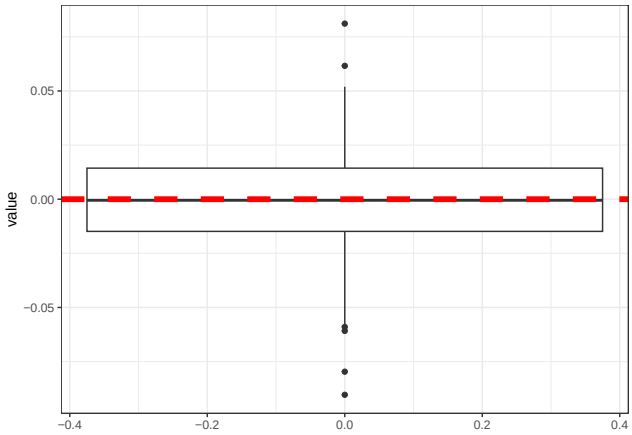
```
p <- data.frame(mintMR = res$Pvalue[example_data$latent_status == 1])
p %>%
  ggplot(aes(x = mintMR))+
  geom_histogram(col="black", boundary = 0, bins=20)+
  theme_classic()+
  scale_x_continuous(limits = c(NA,1))+
  labs(x = "P-value", y = "# gene-tissue pairs",
       title = "P-value for truly non-zero effect")
```



The effect estimates for truly zero effects:

```
estimate <- res$Estimate
estimate[example_data$latent_status == 1] <- NA

estimate %>%
  reshape2::melt() %>%
  tidyr::drop_na() %>%
  ggplot(aes(y = value))+
  theme_bw()+
  geom_boxplot()+
  geom_hline(yintercept = 0,col="red",linetype="dashed",linewidth=2)
```



In real data, the parameter `b_beta` is recommended to be estimated first using other MR methods. For example,

```
ivw <- mapply(function(bx,by,bxse,byse,ld) {
  res <- mr_mvivw(mr_mvinput(bx = bx, by = as.numeric(by),
                             bxse = bxse, byse = as.numeric(byse),
                             correlation = ld))
  return(list(p = res$Pvalue, b = res$Estimate))
},example_data$gamma_hat,example_data$Gamma_hat,
example_data$se_g,example_data$se_G,
example_data$LD,SIMPLIFY = F)

b_ivw <- lapply(ivw, function(x) x$b) %>% do.call(rbind,.)
p_ivw <- lapply(ivw, function(x) x$p) %>% do.call(rbind,.)
```

```
b_beta <- var(b_ivw[p_ivw > 0.05]) * (K1+K2) / 2
opts <- get_opts(L, b_beta = rep(b_beta,L))
```

The new parameter can be used in the mintMR by setting the `opts` option.

```
set.seed(1)
res <- mintMR(gammah = example_data$gamma_hat,
              Gammah = example_data$Gamma_hat,
              se1 = example_data$se_g,
              se2 = example_data$se_G,
              corr_mat = example_data$LD,
              group = group,
              opts = opts)

p <- data.frame(mintMR = as.vector(res$Pvalue))
p %>%
  arrange(mintMR) %>%
  mutate(unif = ppoints(n())) %>%
  mutate(value = -log10(mintMR), unif = -log10(unif)) %>%
  ggplot(aes(y = value, x = unif))+
  geom_abline(intercept = 0, slope = 1, alpha = 0.5)+
  geom_point(size = 3, col = "red", alpha=1)+
  theme_classic()+
  labs(y = expression(paste("Observed -log"[10], plain(P))),
       x = expression(paste("Expected -log"[10], plain(P))))
```