Investigating How Phenotype Differences Handles Information Asymmetry With Multiple Siblings

Klint Kanopka

10/19/2020

Load `MASS` before you load `dplyr` or `tidyverse`, or else things get weird with the `select()` function.

```
library(MASS)
library(tidyverse)
```

Simulation Function

First we create a data simulation function to create dichotomous phenotypes. This setup allows for specifying the within-family genetic correlation, ρ, the magnitude of the direct genetic effect, β, the magnitude of a pooled family-level genetic effect, γ, and the correlation between direct genetic and family genetic effects, ρ_G.

This function simulates a dichotomous phenotype, Y_i, for three siblings, $i \in \{0, 1, 2\}$. Note that sibling $i = 0$ is taken to be the genotyped sibling, though genotypes are generated for each sibling under the hood. Dichotomization is done by doing Bernoulli draws for each sibling where

$$P(Y_i = 1) = \sigma\big(\alpha + \beta g_{ij} + \gamma g_j + \varepsilon_{ij}\big)$$

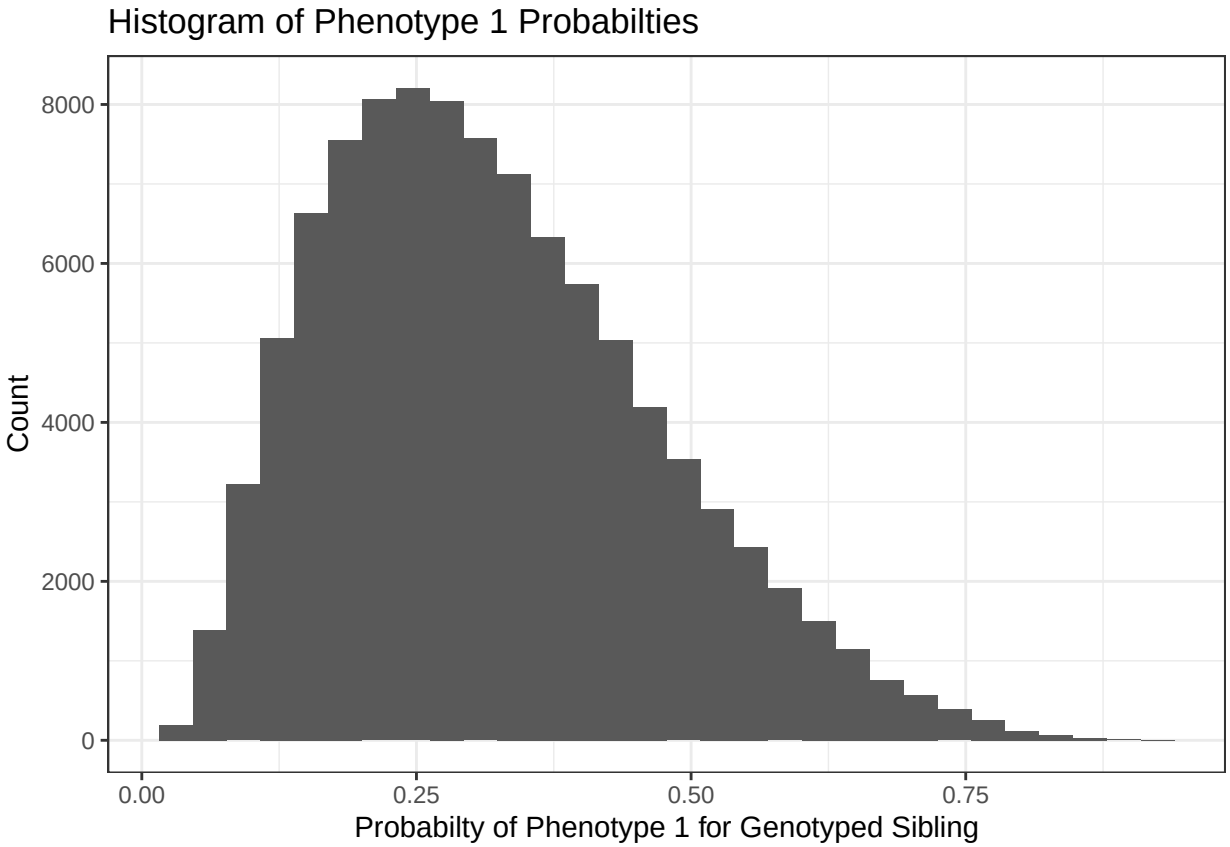
Where i indexes siblings, j indexes families and $\sigma(\cdot)$ is the logistic function. Note that in the simulations that follow, we will not estimate β directly - but instead fit a linear probability model to the simulated data.

```
sim.data <- function(N=1e5, alpha=-0.85, beta=1, rho=0.5, rho_g=0.4, gamma=1,
                     mu_dir=0, sd_dir=0.5, mu_fam=0, sd_ind=0.25, sd_fam=NULL){

  sim.pgs <- function(N=1e5, rho=0.5, rho_g=0.4,
                      mu_dir=0, sd_dir=1,
                      mu_fam=0, sd_fam=NULL){
    require(MASS)
    if (is.null(sd_fam)){sd_fam <- 0.7*sd_dir}
    var_mat <- matrix(c(sd_dir^2, rho*sd_dir^2, rho*sd_dir^2, rho_g*sd_dir*sd_fam,
                        rho*sd_dir^2, sd_dir^2, rho*sd_dir^2, rho_g*sd_dir*sd_fam,
                        rho*sd_dir^2, rho*sd_dir^2, sd_dir^2, rho_g*sd_dir*sd_fam,
                        rho_g*sd_dir*sd_fam, rho_g*sd_dir*sd_fam, rho_g*sd_dir*sd_fam, sd_fam^2),
                      ncol=4)
    pgs <- mvrnorm(n=N, mu=c(mu_dir, mu_dir, mu_dir, mu_fam),
                   Sigma=var_mat, empirical=TRUE)
    out <- data.frame(g0 = pgs[,1],
                      g1 = pgs[,2],
                      g2 = pgs[,3],
                      eps_fam = pgs[,4])
    return(out)
  }
```

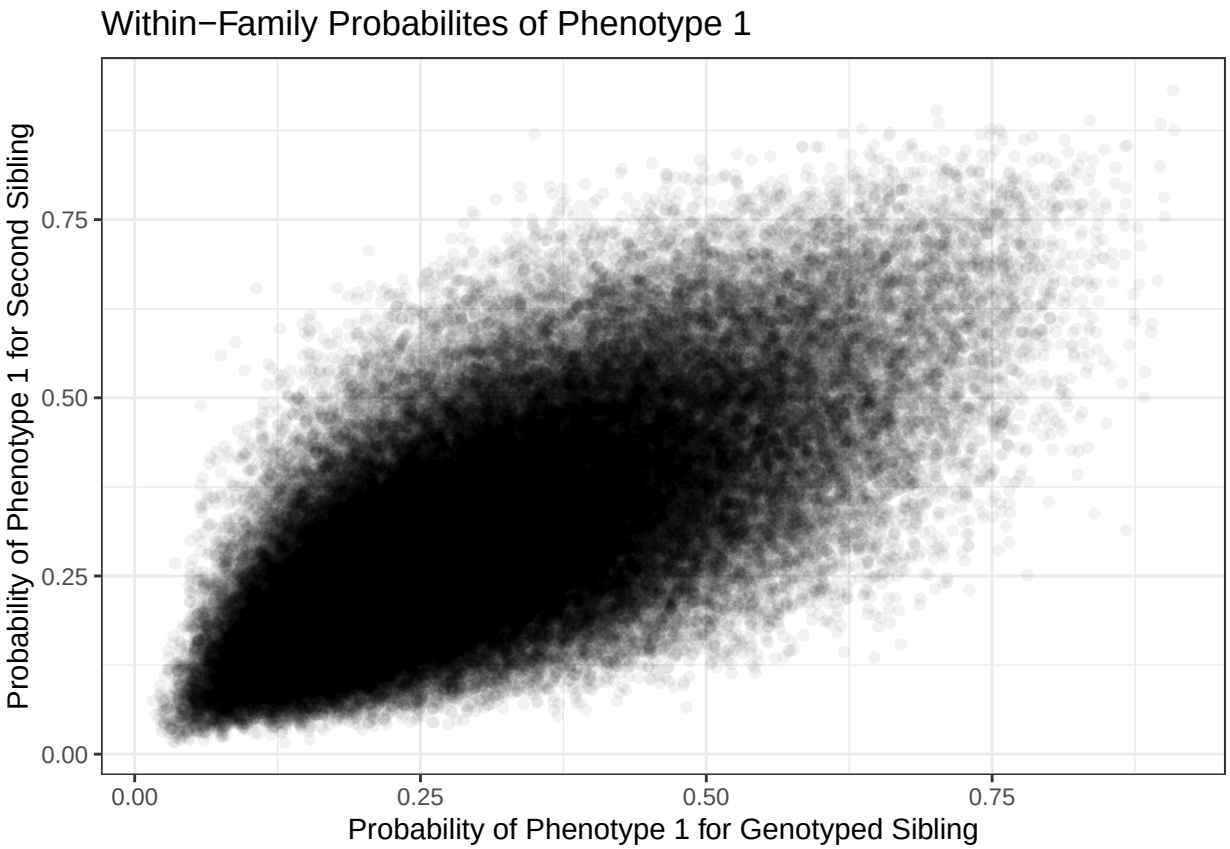
```
  sigmoid <- function(z){
    out <- 1 / (1 + exp(-z))
  }

  eps_0 <- rnorm(N, sd=sd_ind)
  eps_1 <- rnorm(N, sd=sd_ind)
  eps_2 <- rnorm(N, sd=sd_ind)

  a <- rep(alpha, N)
  b <- rep(beta, N)

  d <- sim.pgs(N=N, rho=rho, rho_g=rho_g,
               mu_dir=mu_dir, sd_dir=sd_dir,
               mu_fam=mu_fam, sd_fam=sd_fam)

  d$p0 <- sigmoid(a + b*d$g0 + gamma*d$eps_fam + eps_0)
  d$p1 <- sigmoid(a + b*d$g1 + gamma*d$eps_fam + eps_1)
  d$p2 <- sigmoid(a + b*d$g2 + gamma*d$eps_fam + eps_2)

  d$Y0 <- rbinom(N, 1, d$p0)
  d$Y1 <- rbinom(N, 1, d$p1)
  d$Y2 <- rbinom(N, 1, d$p2)

  # uncomment this line to destroy sibling genotype and probability data
  # before you ever get a chance to see it
  # d <- d[, c('g0', 'Y0', 'Y1', 'Y2')]

  return(d)
}
```

```
set.seed(8675309)
d <- sim.data()
```

The default parameters specified above provides us with a 0.317 prevalence of phenotype Y in the genotyped siblings. To get a sense of the underlying probabilities that generated the observed phenotypes, we look at the histogram of $P(Y_0 = 1)$. Note that adjusting the α parameter will move the mode of this distribution and all sibling data is generated using an identical procedure.



Additionally, we can look at the within-family relationship between $P(Y_0 = 1)$ and $P(Y_1 = 1)$. Here we see that the probabilities are related, but there is also a fair amount of within-family variation.



A Fast Comparison of Vanilla OLS, Fixed-Effects, and Phenotype Differences

If we want to estimate the effect of a genotype, g on Y, we can use data from genotyped siblings and run a regression. This has the downside of being biased due to the presence of omitted variables (specifically at the family level) that are correlated with both our predictors and the outcome.

```
m_ols <- lm(Y0 ~ g0, data = d)
summary(m_ols)
```

```
##
## Call:
## lm(formula = Y0 ~ g0, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8577 -0.3364 -0.2213  0.5406  1.1302
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.317350   0.001419  223.66   <2e-16 ***
## g0          0.247546   0.002838   87.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.4487 on 99998 degrees of freedom
## Multiple R-squared:  0.07072,    Adjusted R-squared:  0.07071
## F-statistic:  7609 on 1 and 99998 DF,  p-value: < 2.2e-16
```

```
ols_est <- coef(m_ols)[2]
```

A fixed effects regression gets around this. Here I use genotype data from all three simulated siblings to give our best estimate of the effect we are looking to recover:

```
# apply within transformation
fe_d <- data.frame(
  Y = c(d$Y0, d$Y1, d$Y2),
  g = c(d$g0, d$g1, d$g2),
  family = rep(1:nrow(d), 3)) %>%
  group_by(family) %>%
  mutate(Y_bar = mean(Y),
         g_bar = mean(g)) %>%
  ungroup() %>%
  transmute(Y = Y - Y_bar,
            g = g - g_bar)

m_fe <- lm(Y ~ g, data = fe_d)
summary(m_fe)
```

```
##
## Call:
## lm(formula = Y ~ g, data = fe_d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.84943 -0.29992 -0.01173  0.26951  0.92058
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.262e-18  6.631e-04    0.00        1
## g            1.967e-01  2.297e-03   85.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3632 on 299998 degrees of freedom
## Multiple R-squared:  0.02386,    Adjusted R-squared:  0.02386
## F-statistic:  7333 on 1 and 299998 DF,  p-value: < 2.2e-16
```

```
fe_est <- coef(m_fe)[2]
```

Unfortunately, it is not always the case that we have sets of siblings that are all genotyped. Here is where phenotype differences methods are helpful, as they are unbiased when we observe two phenotypes within the same family and only one genotyped sibling. Note that phenotype differences with a randomly selected sibling is unbiased for the direct genetic effect, and it does not matter which sibling we use:

```
d <- d %>%
  select(g0, Y0, Y1, Y2) %>%
  mutate(g = 0.5 * g0,
         keep = 1 - max(Y1, Y2),
         delta_1 = Y0 - Y1,
         delta_2 = Y0 - Y2,
         delta_max = Y0 - max(Y1, Y2))
```

```r
m1 <- lm(delta_1 ~ g, data = d)
m2 <- lm(delta_2 ~ g, data = d)
summary(m1)
```

```
##
## Call:
## lm(formula = delta_1 ~ g, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.17791 -0.06024  0.00738  0.07258  1.15450
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.002320   0.002006  -1.157    0.247
## g            0.186296   0.008023  23.220   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6343 on 99998 degrees of freedom
## Multiple R-squared:  0.005363,   Adjusted R-squared:  0.005353
## F-statistic: 539.2 on 1 and 99998 DF,  p-value: < 2.2e-16
```

```r
summary(m2)
```

```
##
## Call:
## lm(formula = delta_2 ~ g, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.21079 -0.06280  0.00878  0.07817  1.18203
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.004070   0.002009  -2.026   0.0427 *
## g            0.196860   0.008034  24.502   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6352 on 99998 degrees of freedom
## Multiple R-squared:  0.005968,   Adjusted R-squared:  0.005958
## F-statistic: 600.4 on 1 and 99998 DF,  p-value: < 2.2e-16
```

A Motivating Example

Next imagine the case where there are three phenotyped siblings and we ask the genotyped sibling, "Do either of your siblings have phenotype Y?" Some data in the UKB is collected this way. Additionally, it is easy to imagine a selection design that does this implicitly - providing us with a nonrandom phenotyped sibling. As one would expect, this case provides an estimate of the direct genetic effect that is biased:

```r
m3 <- lm(delta_max ~ g, data = d)
summary(m3)
```

```
##
## Call:
## lm(formula = delta_max ~ g, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8577 -0.3364 -0.2213  0.5406  1.1302
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.682650   0.001419 -481.12   <2e-16 ***
## g            0.495093   0.005676   87.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4487 on 99998 degrees of freedom
## Multiple R-squared:  0.07072,    Adjusted R-squared:  0.07071
## F-statistic:  7609 on 1 and 99998 DF,  p-value: < 2.2e-16
```

```
max_est <- coef(m3)[2]
```

We will use a ratio to quantify the amount of bias in the PD estimate relative to the OLS estimate. Specifically:

$$R = \frac{\hat{\beta}_{PD} - \hat{\beta}_{FE}}{\hat{\beta}_{OLS} - \hat{\beta}_{FE}}$$

Note that here, the value of our ratio is $R = 5.869$, signaling that PD does much worse than OLS in this case.

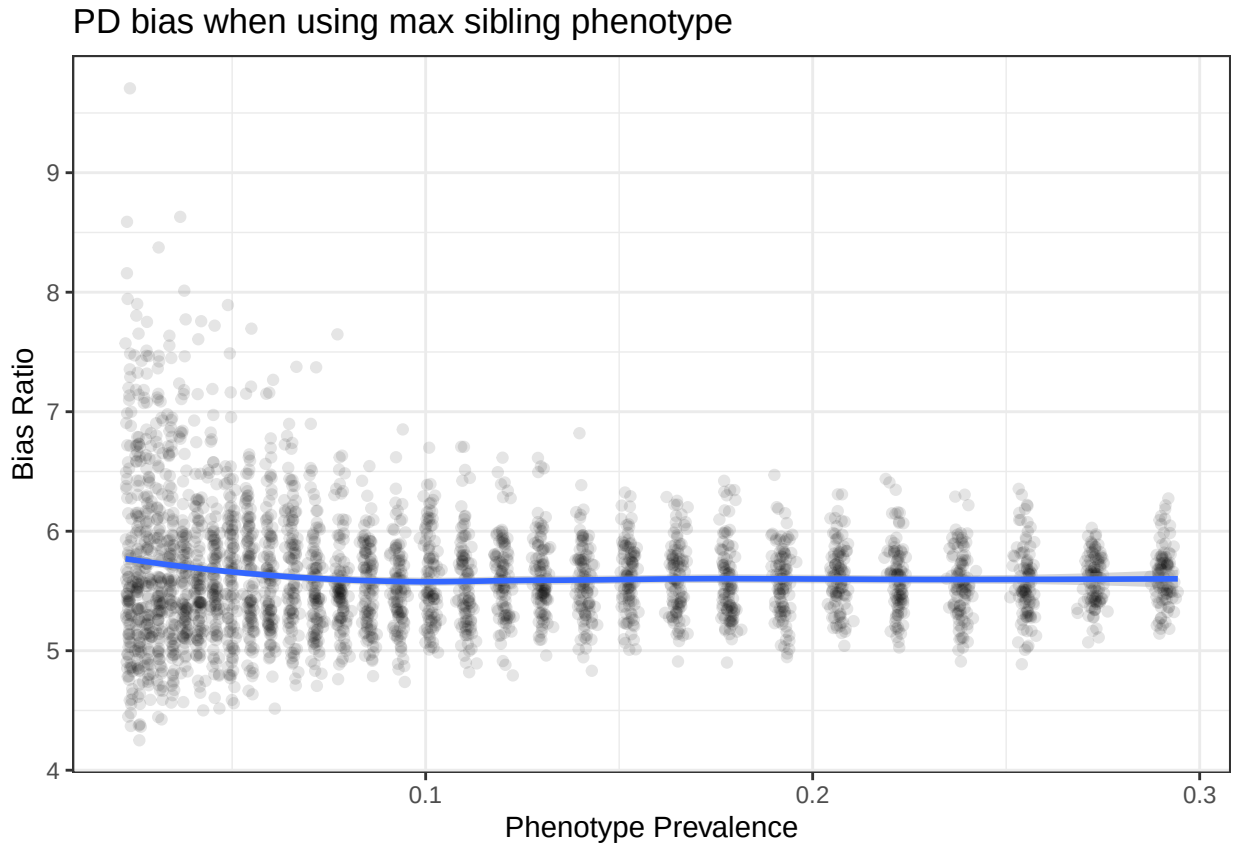
Bias as a Function of Phenotype Prevalence

First we run a simulation to see how this bias ratio responds as a function of phenotype prevalence. Note that, here, the phenotype difference that forms the dependent variable is:

$$PD = Y_0 - \max(Y_1, Y_2)$$

```
alpha <- seq(-4, -1, by=0.1)

prevalence <- c()
ratio <- c()

reps <- 100

for (i in 1:length(alpha)){
  for (j in 1:reps){
    d_sim <- sim.data(alpha=alpha[i])
    prevalence <- c(prevalence, mean(d_sim$Y0))

    fe_d <- data.frame(Y = c(d_sim$Y0, d_sim$Y1, d_sim$Y2),
                       g = c(d_sim$g0, d_sim$g1, d_sim$g2),
                       family = rep(1:nrow(d_sim), 3)) %>%
      group_by(family) %>%
      mutate(Y_bar = mean(Y),
             g_bar = mean(g)) %>%
```

```
      ungroup() %>%
      transmute(Y = Y - Y_bar,
                g = g - g_bar)

    m_ols <- lm(Y0 ~ g0, data=d_sim)
    ols_est <- coef(m_ols)[2]

    m_sim_fe <- lm(Y ~ g, data = fe_d)
    fe_est <- coef(m_sim_fe)[2]

    d_sim <- d_sim %>%
      transmute(g = 0.5 * g0,
                delta_max = Y0 - max(Y1,Y2))
    m_sim_pd <- lm(delta_max ~ g, data = d_sim)
    max_est <- coef(m_sim_pd)[2]

    ratio <- c(ratio, (max_est - fe_est)/(ols_est - fe_est))
  }
}

sim_results <- data.frame(alpha, prevalence, ratio)

ggplot(sim_results, aes(x = prevalence, y = ratio)) +
  geom_point(alpha = 0.1) +
  geom_smooth(method = 'loess', formula=y~x) +
  labs(y='Bias Ratio',
       x='Phenotype Prevalence',
       title='PD bias when using max sibling phenotype') +
  theme_bw()
```



Exploring Improvements

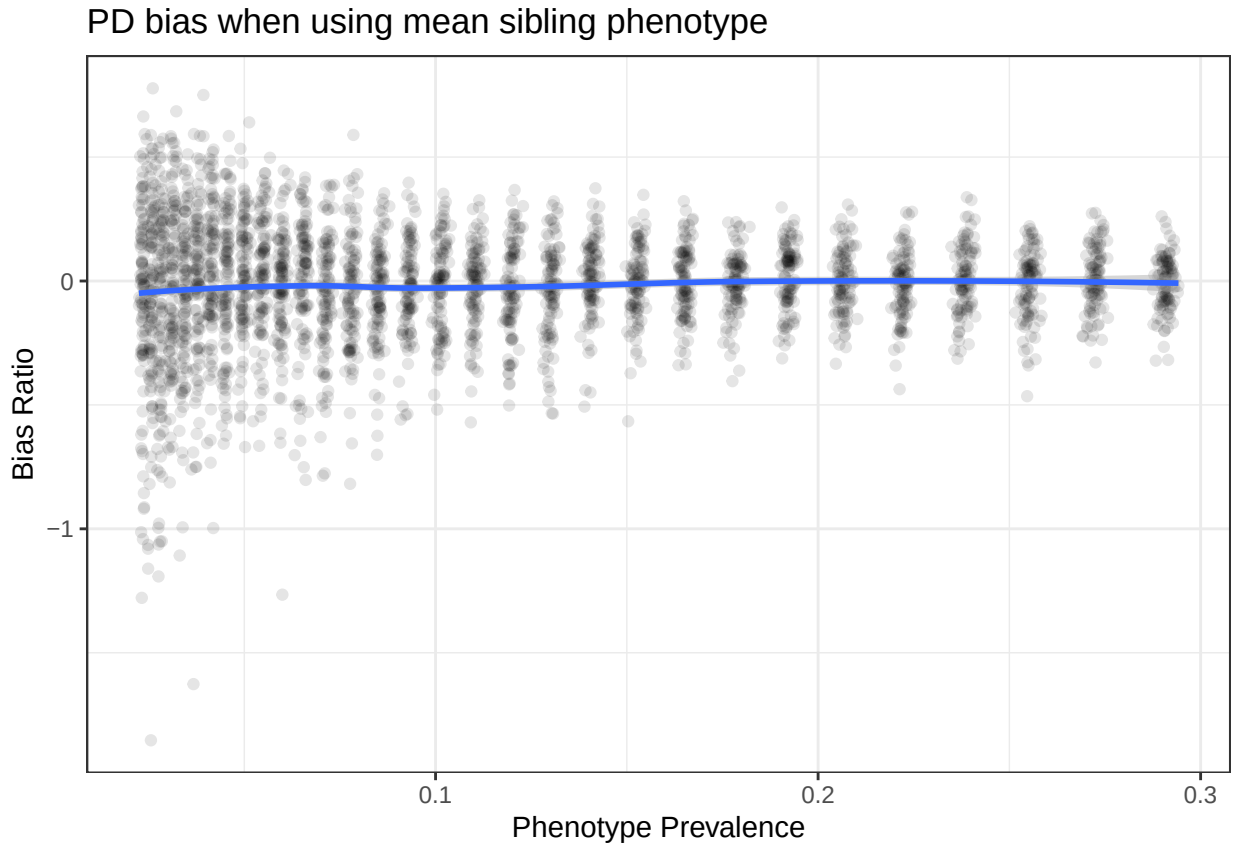
Using mean sibling phenotype

Were we to have complete information about both sibling phenotypes, we could use the mean sibling phenotype and modify our phenotype difference to be:

$$PD = Y_0 - \frac{Y_1 + Y_2}{2}$$

```
alpha <- seq(-4, -1, by=0.1)

prevalence <- c()
ratio <- c()

reps <- 100

for (i in 1:length(alpha)){
  for (j in 1:reps){
    d_sim <- sim.data(alpha=alpha[i])
    prevalence <- c(prevalence, mean(d_sim$Y0))

    fe_d <- data.frame(Y = c(d_sim$Y0, d_sim$Y1, d_sim$Y2),
                       g = c(d_sim$g0, d_sim$g1, d_sim$g2),
                       family = rep(1:nrow(d_sim), 3)) %>%
      group_by(family) %>%
```

```
      mutate(Y_bar = mean(Y),
             g_bar = mean(g)) %>%
      ungroup() %>%
      transmute(Y = Y - Y_bar,
                g = g - g_bar)

    m_ols <- lm(Y0 ~ g0, data=d_sim)
    ols_est <- coef(m_ols)[2]

    m_sim_fe <- lm(Y ~ g, data = fe_d)
    fe_est <- coef(m_sim_fe)[2]

    d_sim <- d_sim %>%
      transmute(g = 0.5 * g0,
                delta_mean = Y0 - (Y1+Y2)/2)
    m_sim_pd <- lm(delta_mean ~ g, data = d_sim)
    mean_est <- coef(m_sim_pd)[2]

    ratio <- c(ratio, (mean_est - fe_est)/(ols_est - fe_est))
  }
}

sim_results <- data.frame(alpha, prevalence, ratio)

ggplot(sim_results, aes(x = prevalence, y = ratio)) +
  geom_point(alpha = 0.1) +
  geom_smooth(method = 'loess', formula=y~x) +
  labs(y='Bias Ratio',
       x='Phenotype Prevalence',
       title='PD bias when using mean sibling phenotype') +
  theme_bw()
```



Here we see this fixes the bias problem, but requires more information than our motivating example provides. Specifically, when the max phenotype is zero, we know that $Y_1 = Y_2 = 0$. We can make no such sure statement when $\max(Y_1, Y_2) = 1$. Next we try three different strategies for resolving this problem.

Adjust the sibling phenotypes by a fixed amount

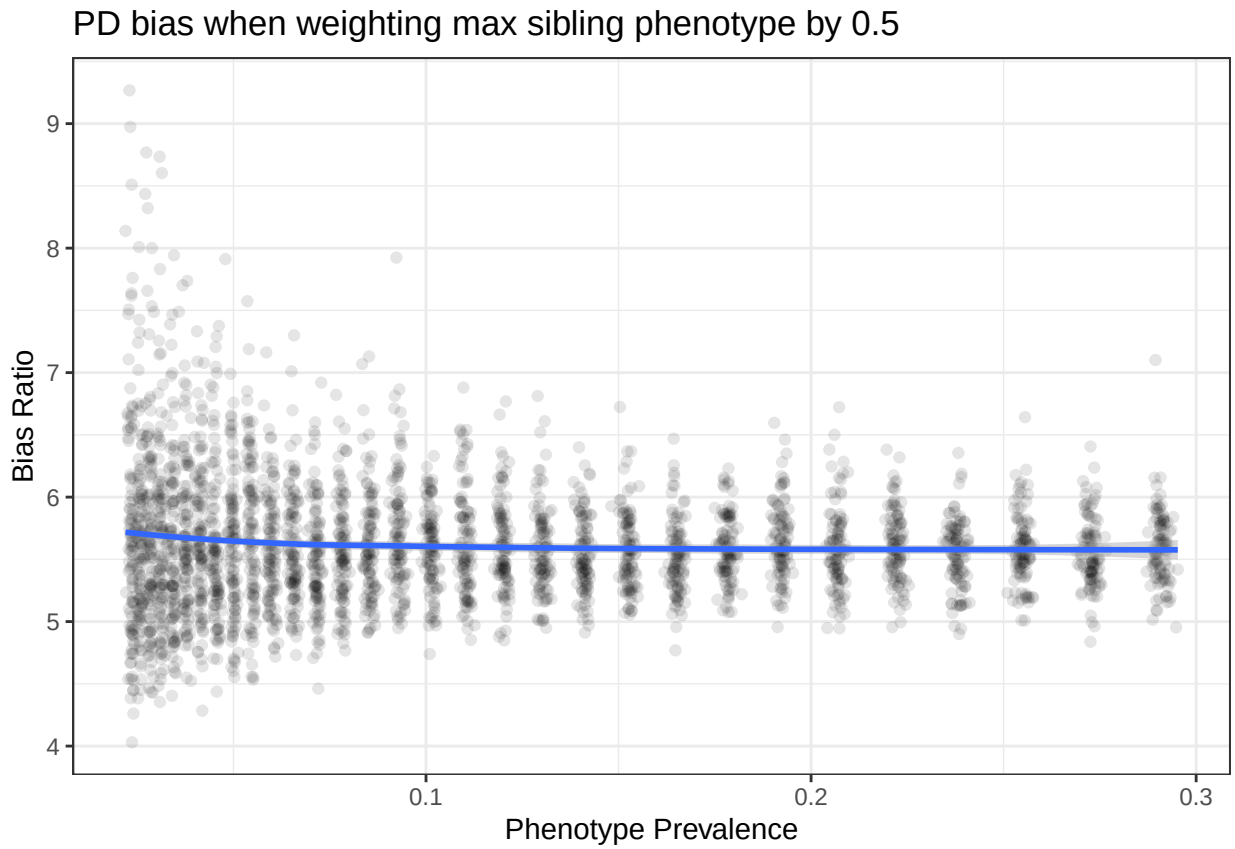
Here, we adjust sibling phenotypes by a factor of 0.5. This makes our phenotype difference:

$$PD = Y_0 - \frac{\max(Y_1, Y_2)}{2}$$

Given the approaches above, this assumes that there are only two cases for the siblings: Both siblings have phenotype zero, or one sibling has phenotype zero.

```
alpha <- seq(-4, -1, by=0.1)

prevalence <- c()
ratio <- c()

reps <- 100

for (i in 1:length(alpha)){
  for (j in 1:reps){
    d_sim <- sim.data(alpha=alpha[i])
    prevalence <- c(prevalence, mean(d_sim$Y0))

    fe_d <- data.frame(Y = c(d_sim$Y0, d_sim$Y1, d_sim$Y2),
```

```
                        g = c(d_sim$g0, d_sim$g1, d_sim$g2),
                        family = rep(1:nrow(d_sim), 3)) %>%
      group_by(family) %>%
      mutate(Y_bar = mean(Y),
             g_bar = mean(g)) %>%
      ungroup() %>%
      transmute(Y = Y - Y_bar,
                g = g - g_bar)

    m_ols <- lm(Y0 ~ g0, data=d_sim)
    ols_est <- coef(m_ols)[2]

    m_sim_fe <- lm(Y ~ g, data = fe_d)
    fe_est <- coef(m_sim_fe)[2]

    d_sim <- d_sim %>%
      transmute(g = 0.5 * g0,
                delta_max = Y0 - 0.5*max(Y1,Y2))
    m_sim_pd <- lm(delta_max ~ g, data = d_sim)
    max_est <- coef(m_sim_pd)[2]

    ratio <- c(ratio, (max_est - fe_est)/(ols_est - fe_est))
  }
}

sim_results <- data.frame(alpha, prevalence, ratio)

ggplot(sim_results, aes(x = prevalence, y = ratio)) +
  geom_point(alpha = 0.1) +
  geom_smooth(method = 'loess', formula=y~x) +
  labs(y='Bias Ratio',
       x='Phenotype Prevalence',
       title='PD bias when weighting max sibling phenotype by 0.5') +
  theme_bw()
```



This appears to shrink PD bias slightly compared to just using the maximum phenotype, but not by much.

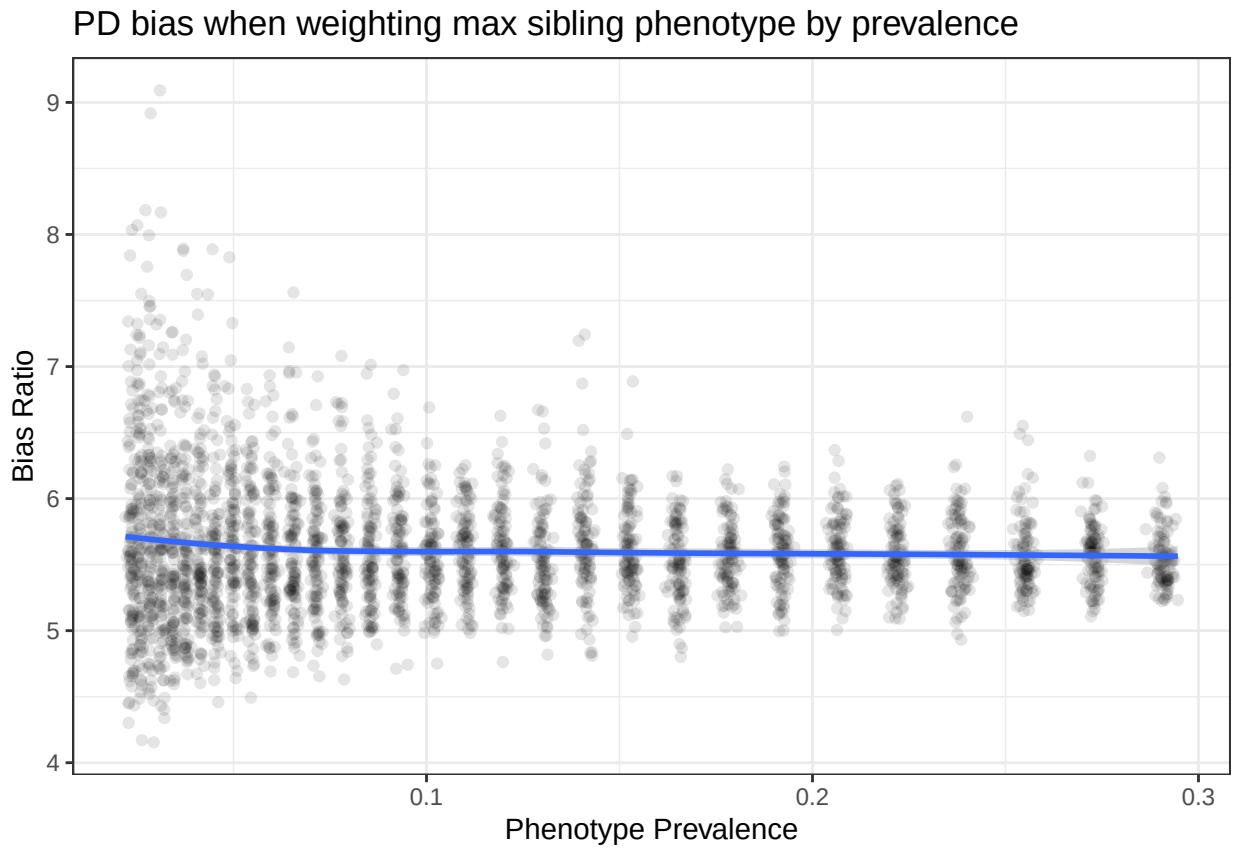
Scale by an estimate of the phenotype prevalence

Here we apply a little bit of trickery. We start by estimating $P\big(Y_0 = 1 | \max(Y_1, Y_2) = 1\big)$. We take this as our best guess at phenotype prevalence within these families. We then adjust our phenotype difference to be:

$$PD = Y_0 - \frac{1 + P\big(Y_0 = 1 | \max(Y_1, Y_2) = 1\big)}{2} \max(Y_1, Y_2)$$

```
alpha <- seq(-4, -1, by=0.1)

prevalence <- c()
ratio <- c()

reps <- 100

for (i in 1:length(alpha)){
  for (j in 1:reps){
    d_sim <- sim.data(alpha=alpha[i])
    prevalence <- c(prevalence, mean(d_sim$Y0))

    fe_d <- data.frame(Y = c(d_sim$Y0, d_sim$Y1, d_sim$Y2),
                       g = c(d_sim$g0, d_sim$g1, d_sim$g2),
                       family = rep(1:nrow(d_sim), 3)) %>%
      group_by(family) %>%
```

```
      mutate(Y_bar = mean(Y),
             g_bar = mean(g)) %>%
      ungroup() %>%
      transmute(Y = Y - Y_bar,
                g = g - g_bar)

    m_ols <- lm(Y0 ~ g0, data=d_sim)
    ols_est <- coef(m_ols)[2]

    m_sim_fe <- lm(Y ~ g, data = fe_d)
    fe_est <- coef(m_sim_fe)[2]

    d_sim <- d_sim %>%
      transmute(g = 0.5 * g0,
                Y0 = Y0,
                max_Y = max(Y1,Y2))

    p_est <- mean(d_sim$Y0[d_sim$max_Y == 1])
    d_sim$max_Y = 0.5*(1+p_est)*d_sim$max_Y

    d_sim <- d_sim %>%
      mutate(delta_max = Y0 - max_Y)

    m_sim_pd <- lm(delta_max ~ g, data = d_sim)
    max_est <- coef(m_sim_pd)[2]

    ratio <- c(ratio, (max_est - fe_est)/(ols_est - fe_est))
  }
}

sim_results <- data.frame(alpha, prevalence, ratio)

ggplot(sim_results, aes(x = prevalence, y = ratio)) +
  geom_point(alpha = 0.1) +
  geom_smooth(method = 'loess', formula=y~x) +
  labs(y='Bias Ratio',
       x='Phenotype Prevalence',
       title='PD bias when weighting max sibling phenotype by prevalence') +
  theme_bw()
```



This appears to perform similarly to scaling by a flat 0.5, with the added complication of performing much worse at low phenotypic prevalence.

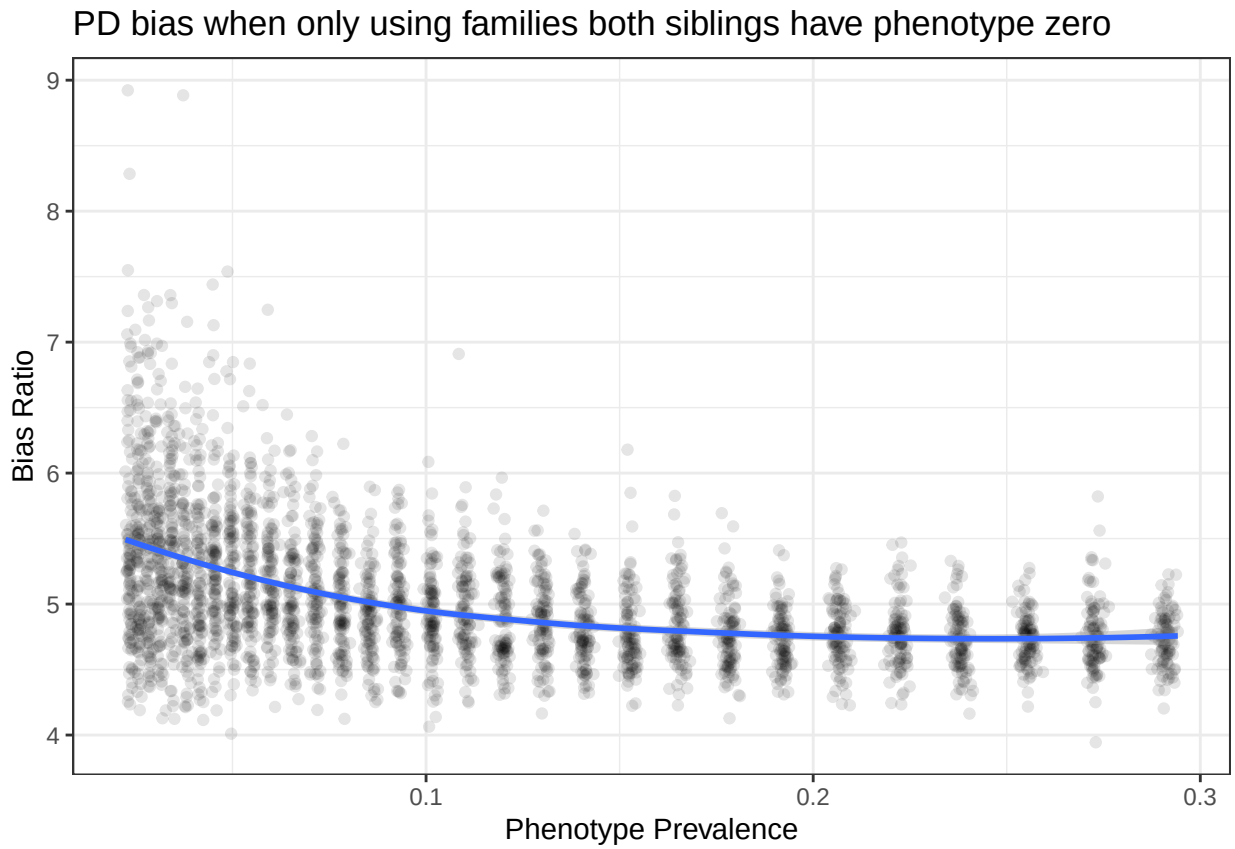
Only use information you are certain of

Here we leverage the asymmetry of the information provided by the motivating case. We have imperfect information when $\max(Y_1, Y_2) = 1$, but when $\max(Y_1, Y_2) = 0$, we know that $Y_1 = Y_2 = 0$. In this simulation, we only consider families where $\max(Y_1, Y_2) = 0$. This makes the phenotype difference:

$$PD = Y_0 - 0 = Y_0$$

```
alpha <- seq(-4, -1, by=0.1)

prevalence <- c()
ratio <- c()

reps <- 100

for (i in 1:length(alpha)){
  for (j in 1:reps){
    d_sim <- sim.data(alpha=alpha[i])
    prevalence <- c(prevalence, mean(d_sim$Y0))

    if (sum((d_sim$Y1 + d_sim$Y2) == 0) > 1){

      fe_d <- data.frame(Y = c(d_sim$Y0, d_sim$Y1, d_sim$Y2),
```

```
                        g = c(d_sim$g0, d_sim$g1, d_sim$g2),
                        family = rep(1:nrow(d_sim), 3)) %>%
        group_by(family) %>%
        mutate(Y_bar = mean(Y),
               g_bar = mean(g)) %>%
        ungroup() %>%
        transmute(Y = Y - Y_bar,
                  g = g - g_bar)

      m_ols <- lm(Y0 ~ g0, data=d_sim)
      ols_est <- coef(m_ols)[2]

      m_sim_fe <- lm(Y ~ g, data = fe_d)
      fe_est <- coef(m_sim_fe)[2]

      d_sim <- d_sim %>%
        mutate(keep = Y1 + Y2) %>%
        filter(keep == 0) %>%
        transmute(g = 0.5 * g0,
                  delta_max = Y0 - max(Y1, Y2))
      m_sim_pd <- lm(delta_max ~ g, data = d_sim)
      max_est <- coef(m_sim_pd)[2]

    ratio <- c(ratio, (max_est - fe_est)/(ols_est - fe_est))
    }
  }
}

sim_results <- data.frame(alpha, prevalence, ratio)

ggplot(sim_results, aes(x = prevalence, y = ratio)) +
  geom_point(alpha = 0.1) +
  geom_smooth(method = 'loess', formula=y~x) +
  labs(y='Bias Ratio',
       x='Phenotype Prevalence',
       title='PD bias when only using families both siblings have phenotype zero') +
  theme_bw()
```



Here we see this adjustment outperforms the adjustments to the max phenotype, but still retains significant bias. Additionally, this picks up the issue of having the sample size shrink as phenotype prevalence increases.

Conclusions

The information asymmetry created by asking questions of the "do any of your siblings" form creates problems for estimation of genetic effects when only a single phenotype is observed. Multiple potential corrections are explored, but none perform better than vanilla OLS that ignores sibling data. Additionally, the bias ratio we estimate is highly variable in regions of low phenotype prevalence, which is the sweet spot for many of the dichotomous phenotypes in studies like UKB.