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.

```r
library(tidyverse)
library(parallel)
```

Simulation Function

Simulating genetic data

We first focus on a single trait. Note that in the simulation function, N specifies the number of *families*. For the jth family, we draw parental phenotypes $g_{jm}, g_{jp} \in \{0, 1, 2\}$ such that

$$g_{jm} \sim \text{Binomial}(2, p_{allele})$$

Where p_{allele} is the prevalence of the allele in the parent's generation. Note that g_{jp} is drawn the same way. This can be specified in the simulation function. Next, for the genotype of each of the three siblings birthed of these parents ($g_{ij} \in \{0, 1, 2\}$, where $i \in \{0, 1, 2\}$), one allele is drawn from each parent at random such that

$$g_{ij} \sim \text{Bernoulli}\left(\frac{g_{jm}}{2}\right) + \text{Bernoulli}\left(\frac{g_{jp}}{2}\right)$$

Simulating phenotypes

The `sim.data()` function allows for specifying the relative magnitudes of the direct genetic effect, β, a pooled family-level genetic and environmental effect, γ_1, and the individual error term, γ_2. We specify these three coefficients instead of specifying standard deviations for the ε_j and ε_{ij} terms for simplicity.

This function simulates a dichotomous phenotype, Y_{ij}, 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

$$Y_{ij} \sim \text{Bernoulli}(p_{ij})$$

where

$$p_{ij} = \sigma\big(\alpha + \beta g_{ij} + \gamma_1 \varepsilon_j + \gamma_2 \varepsilon_{ij}\big)$$

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

Adjust the beta/gamma_1/gamma_2 coefs to get R^2 ~ 0.001

```r
sim.data <- function(N=1e5, rho=0.5, rho_g=0.3, p_allele=0.3,
                     alpha=-1, beta=0.04, gamma_1=sqrt(0.35),   gamma_2=sqrt(0.649),
                     continuous=FALSE){

  sim.genes <- function(N=N, rho=rho, rho_g=rho_g, p_allele=p_allele){
    g_m <- rbinom(n=N, size=2, prob=p_allele)
    g_p <- rbinom(n=N, size=2, prob=p_allele)

    out <- data.frame(g_m = g_m,
                      g_p = g_p,
                      g0 = rbinom(N, 1, g_m/2) + rbinom(N, 1, g_m/2),
                      g1 = rbinom(N, 1, g_m/2) + rbinom(N, 1, g_m/2),
                      g2 = rbinom(N, 1, g_m/2) + rbinom(N, 1, g_m/2),
                      eps_fam = rnorm(N))
    return(out)
  }

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

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

  d <- sim.genes(N=N, rho=rho, rho_g=rho_g, p_allele=p_allele)

  if (continuous){
    d$Y0 <- a + b*d$g0 + gamma_1*d$eps_fam + gamma_2*rnorm(N)
    d$Y1 <- a + b*d$g1 + gamma_1*d$eps_fam + gamma_2*rnorm(N)
    d$Y2 <- a + b*d$g2 + gamma_1*d$eps_fam + gamma_2*rnorm(N)
  } else {
    d$p0 <- sigmoid(a + b*d$g0 + gamma_1*d$eps_fam + gamma_2*rnorm(N))
    d$p1 <- sigmoid(a + b*d$g1 + gamma_1*d$eps_fam + gamma_2*rnorm(N))
    d$p2 <- sigmoid(a + b*d$g2 + gamma_1*d$eps_fam + gamma_2*rnorm(N))

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

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.

```r
set.seed(8675309)
d <- sim.data(continuous=TRUE)
m_ols <- lm(Y0 ~ g0, data = d)
summary(m_ols)
```

```
##
## Call:
## lm(formula = Y0 ~ g0, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1359 -0.6747  0.0000  0.6754  4.2476
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.999638   0.003963 -252.25   <2e-16 ***
## g0           0.042877   0.003995   10.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.002 on 99998 degrees of freedom
## Multiple R-squared:  0.00115,    Adjusted R-squared:  0.00114
## F-statistic: 115.2 on 1 and 99998 DF,  p-value: < 2.2e-16
```

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
## -3.1842 -0.4439 -0.0003  0.4453  3.3192
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.822e-17  1.204e-03    0.00        1
## g           3.875e-02  3.226e-03   12.01   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6595 on 299998 degrees of freedom
## Multiple R-squared:  0.0004805,  Adjusted R-squared:  0.0004772
## F-statistic: 144.2 on 1 and 299998 DF,  p-value: < 2.2e-16
```

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 (theoretically) unbiased when we observe two phenotypes within the same family and only one genotyped sibling. Note that here, PD does not appear to recover the correct estimates.

```r
d <- d %>%
  mutate(g_pd = 0.5 * g0,
         delta_g1 = g0 - g1,
         delta_g2 = g0 - g2,
         delta_Y1 = Y0 - Y1,
         delta_Y2 = Y0 - Y2,
         delta_max = Y0 - max(Y1, Y2),
         keep = 1 - max(Y1, Y2))

#FD Models
m1 <- lm(delta_Y1 ~ delta_g1, data = d)
m2 <- lm(delta_Y2 ~ delta_g2, data = d)
summary(m1)
```

```
##
## Call:
## lm(formula = delta_Y1 ~ delta_g1, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8525 -0.7746 -0.0016  0.7709  5.5038
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.001182   0.003618  -0.327    0.744
## delta_g1     0.045520   0.005616   8.105 5.33e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.144 on 99998 degrees of freedom
## Multiple R-squared:  0.0006565,  Adjusted R-squared:  0.0006465
## F-statistic: 65.69 on 1 and 99998 DF,  p-value: 5.333e-16
```

```r
summary(m2)
```

```
##
## Call:
## lm(formula = delta_Y2 ~ delta_g2, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.2439 -0.7740  0.0018  0.7690  4.7704
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.754e-05  3.604e-03  -0.016    0.987
## delta_g2     3.599e-02  5.557e-03   6.477 9.42e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.14 on 99998 degrees of freedom
## Multiple R-squared:  0.0004193,  Adjusted R-squared:  0.0004093
## F-statistic: 41.95 on 1 and 99998 DF,  p-value: 9.42e-11
```

```r
#PD Models
m1 <- lm(delta_Y1 ~ g_pd, data = d)
m2 <- lm(delta_Y2 ~ g_pd, data = d)
summary(m1)
```

```
##
## Call:
## lm(formula = delta_Y1 ~ g_pd, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8707 -0.7747 -0.0014  0.7693  5.4987
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.008852   0.004528  -1.955  0.05058 .
## g_pd         0.025598   0.009131   2.804  0.00506 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.145 on 99998 degrees of freedom
## Multiple R-squared:  7.859e-05,  Adjusted R-squared:  6.859e-05
## F-statistic:  7.86 on 1 and 99998 DF,  p-value: 0.005056
```

```r
summary(m2)
```

```
##
## Call:
## lm(formula = delta_Y2 ~ g_pd, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.2740 -0.7737  0.0014  0.7692  4.7565
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.005943   0.004510  -1.318   0.1876
## g_pd         0.019802   0.009093   2.178   0.0294 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.14 on 99998 degrees of freedom
## Multiple R-squared:  4.742e-05,  Adjusted R-squared:  3.742e-05
## F-statistic: 4.742 on 1 and 99998 DF,  p-value: 0.02944
```

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:

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

```
##
## Call:
## lm(formula = delta_max ~ g_pd, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1359 -0.6747  0.0000  0.6754  4.2476
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -4.468203   0.003963 -1127.51   <2e-16 ***
## g_pd         0.085754   0.007991    10.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.002 on 99998 degrees of freedom
## Multiple R-squared:  0.00115,    Adjusted R-squared:  0.00114
## F-statistic: 115.2 on 1 and 99998 DF,  p-value: < 2.2e-16
```

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