armor_mr

Jeff Walker

April 5, 2018

History

```
# reboot Aug 9 2013
# reboot II April 5 2018
```

The original motivation was a Mendelian Randomization re-analysis of Marchinko, K.B., 2009. Predation's role in repeated phenotypic and genetic divergence of armor in threespine stickleback. Evolution, 63(1), pp.127-138. The new goal is to add the Schluter xxx data.

We need a DAG model of the potential paths

$$\begin{tikzcd}
 & X_{SL} \arrow[dr, "\beta_7"] \arrow[d, leftrightarrow, bend right, "r2.z"] & \\
Z_{eda} \arrow[ur, "\beta_8"] \arrow[r, "\beta_1"] \arrow[dr, "\beta_3"] \arrow[d, leftrightarrow, bend right, "r1"'] & X_{armor} \arrow[r, "\beta_2"] & Y_w \\
U_g \arrow[ur, "\beta_5"] \arrow[r, "\beta_6"'] & U_x \arrow[ur, "\beta_4"'] &
\end{tikzcd}$$

MR estimates β_2. An unbiased Mendelian randomization (MR) estimate of β_2 assumes

1. $\beta_3 \beta_4 = 0$
2. $r \beta_6 \beta_4 = 0$

The estimate of the "direct selection on" X_{armor} (β_2), conditioning on Z_{eda}, is an unbiased estimate of β_2 if

1. $\beta_5 \beta_6 \beta_4 = 0$

Load libraries

```
library(readxl)
library(lme4)
library(ggplot2)
library(data.table)

source("../R/replicate.R") # for knit, drop the "../" for console
source("../R/mendelian_randomization.R") # for knit, drop the "../" for console
```

Import Rennison data

```
dir_path <- "data/" # for console
dir_path <- "../data/" # for knit
```

$$\begin{tikzcd}
Z_{eda} \arrow[r, "\beta_1"] \arrow[dr, "\beta_3"] \arrow[d, leftrightarrow, bend right, "r"'] & X_{armor} \arrow[r, "\beta_2"] & w \\
U_g \arrow[ur, "\beta_5"'] \arrow[r, "\beta_6"'] & U_x \arrow[ur, "\beta_4"'] &
\end{tikzcd}$$

Figure 1: Full model.

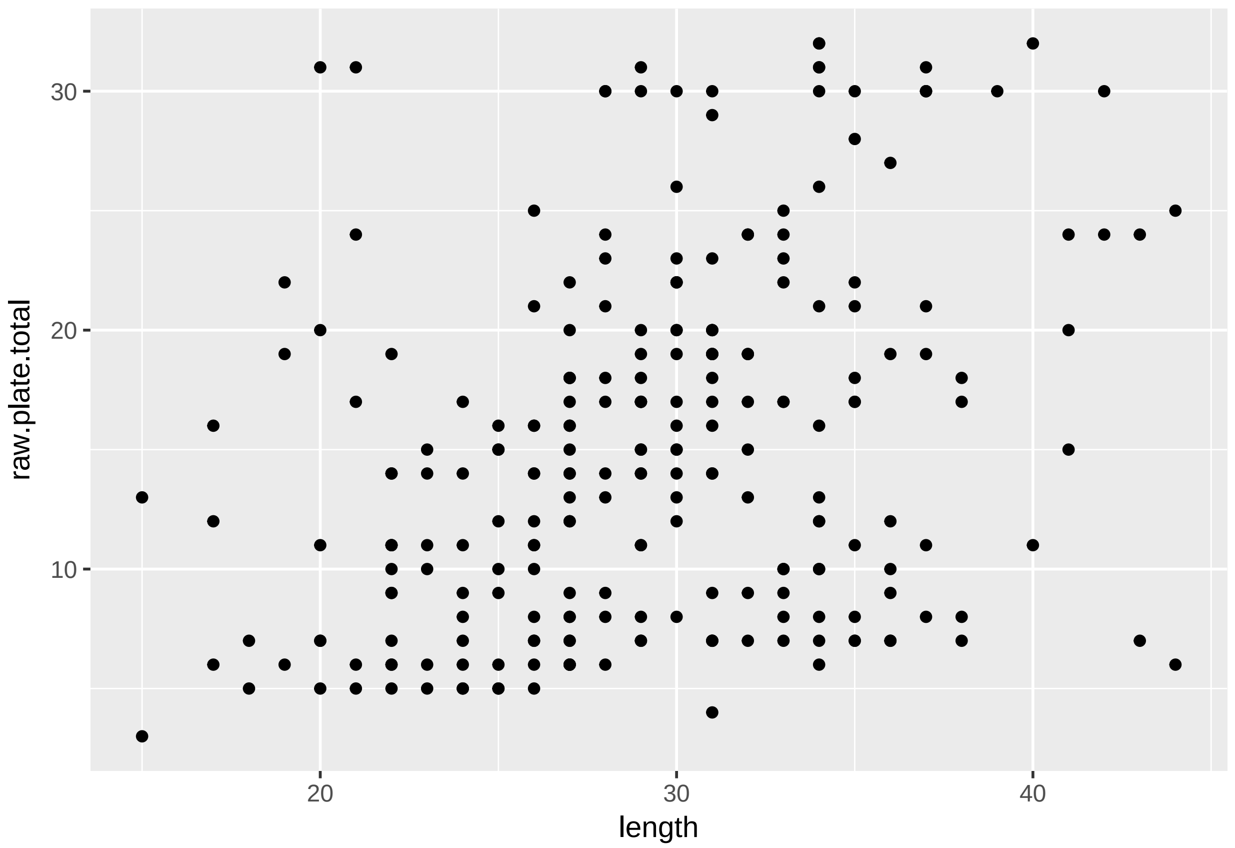
```
file_name <- "Rennison-am_nat-2016.txt"
file_path <- paste(dir_path, file_name, sep='')
rennison <- data.table(fread(file_path))
rennison[, month:=factor(month, c('06-Sep', '06-Oct', '06-Nov'))]
```

Replicate Rennison

Replicate size adjustment

It would be good to simulate process to know how sensitive result is to adjustment vs. simply using length as covariate in linear model.

```
# plot plates vs. size
qplot(x=length, y=raw.plate.total, data=rennison)
```



```
# check size adjustment - can't replicate
rennison[, logplates.s:=ifelse(length <=34.0, size_corrected(raw.logplates, length, origin=34.0), raw.lo
# limit adjusted plates to max of log(32)
rennison[, logplates.s:=ifelse(logplates.s > log(32), log(32), logplates.s)]
x <- rennison[genotype=='CL', raw.logplates]
sl <- rennison[genotype=='CL', length]
fit <- lm(x~sl)
x_34 <- predict(fit, newdata=data.frame(sl=34.0))
logplates.s <- x_34 + residuals(fit)
```

Replicate multivariate selection differentials

The bootstrap implemented below does not bootstrap the whole process including the length standardization step

```
# this comes closer to replicating adj.plates than using eda_add column
Xa <- rennison[month==levels(month)[1] & eda_add!=-1, .SD, .SDcols=c('adj.plates', 'eda_dom')]
Xb <- rennison[month==levels(month)[2] & eda_add!=-1, .SD, .SDcols=c('adj.plates', 'eda_dom')]
res1 <- mv_selection_differential(Xa, Xb, std=TRUE, bootstrap=TRUE)
apply(res1,2, quantile, c(0.025, 0.5, 0.975))
```

```
##        adj.plates    eda_dom
## 2.5%  -0.07807069 -0.6580128
## 50%    0.29447588 -0.2776027
## 97.5%  0.65601220  0.1005373
```

```r
Xa <- rennison[month==levels(month)[2] & eda_add!=-1, .SD, .SDcols=c('adj.plates', 'eda_dom')]
Xb <- rennison[month==levels(month)[3] & eda_add!=-1, .SD, .SDcols=c('adj.plates', 'eda_dom')]
res2 <- mv_selection_differential(Xa, Xb, std=TRUE, bootstrap=TRUE)
apply(res2,2, quantile, c(0.025, 0.5, 0.975))
```

```
##        adj.plates     eda_dom
## 2.5%  -0.7280472 -0.47655398
## 50%   -0.2066366  0.03831118
## 97.5%  0.2220305  0.49700138
```

```r
#
```

Rennison Mendelian Randomization

```r
inc <- c(which(rennison[eda_add!=-1, month]=='06-Sep'), which(rennison[eda_add!=-1, month]=='06-Oct'))
x <- rennison[inc, adj.plates]
y <- rennison[inc, month]
y <- factor(y, c('06-Sep', '06-Oct'))
z <- rennison[inc, eda_dom]
res <- mendelian_randomization(x,y,z,std=TRUE, bootstrap=TRUE)
apply(res,2, quantile, c(0.025, 0.5, 0.975))
```

```
##           beta.xy    beta.zy    beta.zx          mr
## 2.5%  -0.2156724 -0.9401866 -0.1150887 -0.11262524
## 50%    0.2114904 -0.5717488  0.4126132 -0.01059150
## 97.5%  0.6515719 -0.2382387  0.9188905  0.07710673
```

Import Marchinko data

```r
dir_path <- "data/" # for console
dir_path <- "../data/" # for knit
file_name <- "Marchinko_PaxtonData.xls"
file_path <- paste(dir_path, file_name, sep='')

marchinko <- data.table(read_excel(file_path))
    # columns are
        # population
        # family
        # treatment {no = -insect, pred = +insect}
        # individual
        # standlength
        # ant.dorspine
        # sec.dorspine
        # pelspine.len
        # pelgirdle.len
        # plate.number
        # eda.genotype {AA, Aa, aa}
        # pelgirdle.presence {1=yes, 0 = no}
        # pelspine.presence {1 = yes, 0 = no}
```

```
  marchinko[, family:=factor(family)]
  marchinko[, treatment:=factor(treatment, c('no','pred'))]
```

Replicate Marchinko

Replication requires scaling armor variables within families and replacing values=0.0 with NA. Still, I cannot replicate *ant.dorspine.s*

```
  table_1_replicate <- round(replicate_marchinko(marchinko),3)
```

Re-analysis using multi-level model

```
y_list <- c('ant.dorspine', 'sec.dorspine', 'pelspine.len', 'pelgirdle.len')
table_1_mlm <- numeric(length(y_list))
for(j in 1:length(y_list)){
  y <- y_list[j]
  # fit to get treatment effect
  model_formula <- formula(paste(y, " ~ standlength + treatment + (standlength|family)", sep=''))
  fit <- lmer(model_formula, data=marchinko)
  b_xy <- coefficients(summary(fit))['treatmentpred', 'Estimate']
  # get sd from control group only
  model_formula <- formula(paste(y, " ~ standlength + (standlength|family)", sep=''))
  fit <- lmer(model_formula, data=marchinko[treatment=='no',])
  sd_x <- unlist(as.data.table(VarCorr(fit))[grp=='Residual','sdcor'])
  sd_x <- 1
  table_1_mlm[j] <- b_xy/sd_x
}
table_1_mlm
```

```
## [1] -0.04118469 -0.02889154 -0.02478541 -0.00759773
```

Mendelian randomization of Marchinko armor data

It doesn't make sense to compute this for each armor trait separately because then by definition there assumptions are violated (Z has a path to W independent of X). Possible solution here: "Multivariable Mendelian randomization: the use of pleiotropic genetic variants to estimate causal effects", doi: 10.1093/aje/kwu283.

β_{zy} is the treatment effect on *eda* genotype.

```
# additive model of eda
marchinko[, eda_add:=ifelse(eda.genotype=="aa",-1,ifelse(eda.genotype=="Aa",0,1))]
# dominance model of eda
marchinko[, eda_dom:=ifelse(eda.genotype=="aa",0,1)]

fit <- lmer(eda_add ~ treatment + (1|family), data=marchinko)
b_zy <- coefficients(summary(fit))['treatmentpred', 'Estimate']

  # get sd_z from control group only
  fit <- lmer(eda_add ~ (1|family), data=marchinko[treatment=='no',])
  sd_z<- unlist(as.data.table(VarCorr(fit))[grp=='Residual','sdcor'])

b_zy_prime <- b_zy/sd_z
```

β_{zx} is the effect of *eda* genotype on armor phenotype in the control group.

```
p <- length(y_list)
b_zx <- numeric(p)
sd_x <- numeric(p)
b_zx_prime <- numeric(p)
table_1_mr <- numeric(p)
table_1_mr_prime <- numeric(p)
for(j in 1:length(y_list)){
  y <- y_list[j]
  # fit to get treatment effect
  model_formula <- formula(paste(y, " ~ standlength + eda_add + (standlength|family)", sep=''))
  fit <- lmer(model_formula, data=marchinko[treatment=='no'])
  b_zx[j] <- coefficients(summary(fit))['eda_add', 'Estimate']

  # get sd_x from control group only and from non-RM model
  model_formula <- formula(paste(y, " ~ standlength + (standlength|family)", sep=''))
  fit <- lmer(model_formula, data=marchinko[treatment=='no',])
  sd_x[j] <- unlist(as.data.table(VarCorr(fit))[grp=='Residual','sdcor'])

  table_1_mr[j] <- b_zy/b_zx[j]
  b_zx_prime[j] <- b_zx[j]*(sd_z/sd_x[j])
  table_1_mr_prime[j] <- b_zy_prime/b_zx_prime[j]
}
table_1_mr
```

```
## [1] -7.398726 -9.473280 -7.419083 -5.903709
```

```
table_1_mr_prime
```

```
## [1] -1.707147 -2.228926 -2.624435 -5.364176
```

```
table_1_mlm
```

```
## [1] -0.04118469 -0.02889154 -0.02478541 -0.00759773
```

```
table_1_replicate
```

```
##     standlength  ant.dorspine.s  sec.dorspine.s  pelspine.len.s
##           0.403          -0.225          -0.065           0.011
## pelgirdle.len.s
##           0.011
```