Detecting cross site variation

Miratrix

2019-08-21

Intro

In this document we review two ways to generate point estimates, confidence intervals, and significance tests for cross site treatment impact variation in multisite trials.

We first need some data, and so we simulate some:

```
df = gen.dat.no.cov( n.bar=40, J=20,
                     gamma.10 = 0,
                     tau.11.star = 0.2^2,
                     ICC = 0.55,
                     variable.n = TRUE )
head( df )
#>   sid        Y0         Y1 Z      Yobs
#> 1   1 0.2683617 0.08152846 0 0.2683617
#> 2   1 2.4363562 2.24952291 1 2.2495229
#> 3   1 1.3111425 1.12430924 0 1.3111425
#> 4   1 0.9737693 0.78693607 0 0.9737693
#> 5   1 1.7019158 1.51508251 1 1.5150825
#> 6   1 0.5793232 0.39248996 1 0.3924900
```

This data has a cross site impact variation of 0.2 (in effect size units)

The FIRC Model

The FIRC model is a fixed-intercept, random coefficient model, with separate variance parameters for treatment and control units. The standard results from the fitted model directly gives our point estimate:

```
re.mod <- nlme::lme(Yobs ~ 0 + Z + sid,
                    data = df,
                    random = ~ 0 + Z | sid,
                    weights = nlme::varIdent(form = ~ 1 | Z), na.action=na.exclude,
                    method = "ML",
                    control=nlme::lmeControl(opt="optim",returnObject=TRUE))

# Estimated average treatment impact (with SEs)
ATE = nlme::fixef( re.mod )[[1]]
SE.ATE = sqrt( vcov( re.mod )[1,1] )

# Estimated cross site variation.
vc <- nlme::VarCorr(re.mod)
tau.hat <- as.numeric( vc["Z","StdDev"] )
tau.hat
#> [1] 0.01370834
```

We test for presence of treatment using a null model with no cross site variation, and conducting a liklihood ratio test (which is the same as ANOVA):

```
re.mod.null <- nlme::gls(Yobs ~ 0 + Z + sid,
                         data=df,
                         method = "ML",
                         control=nlme::lmeControl(opt="optim",returnObject=TRUE))

myanova = anova(re.mod.null, re.mod)
myanova
#>              Model df      AIC      BIC    logLik   Test    L.Ratio p-value
#> re.mod.null     1 22 1741.267 1845.585 -848.6336
#> re.mod          2 24 1745.250 1859.050 -848.6248 1 vs 2 0.01759196  0.9912
p.value.anova = myanova[2,9]

p.variation = ( p.value.anova / 2 )  # divide by 2 by same logic as chi-squared test.
p.variation
#> [1] 0.4956213
```

When testing a variance at the boundary, we get to divide the p-value by 2 to adjust for the test statistic getting smashed up against 0 half the time.

We can also generate confidence intervals using normal approximations on the maximum likelihood estimators. These are not generally considered to be high quality.

```
intervals( re.mod, which="var-cov" )
#> Approximate 95% confidence intervals
#>
#>  Random Effects:
#>   Level: sid
#>               lower       est.    upper
#> sd(Z) 1.341334e-06 0.01370834 140.0982
#>
#>  Variance function:
#>       lower      est.    upper
#> 1 0.8962888 0.9900195 1.093552
#> attr(,"label")
#> [1] "Variance function:"
#>
#>  Within-group standard error:
#>     lower      est.     upper
#> 0.6183431 0.6622145 0.7091986
```

Note the discrepancy of the CI not including 0, but the test failing to reject the null of no cross site variation. Normal approximations are bad when up against boundaries. Also notice the completely bizzare upper bound. The approximation is really quite awful here!

Alternatively, we could use *profile confidence intervals*, but they are not implemented for the `nlme` package, apparently (which is needed to have separate variances for the treatment and control groups).

The Meta-Analysis approach

Q-statistics depend on the summary stats of the data. Once you have your estimated impacts and standard errors for each site, you are good to go.

Here we use simple OLS to obtain these, but whatever is traditionally done should be done here:

```
s = length( unique( df$sid ) )
ols <- nlme::gls(Yobs ~ 0 + Z*sid - Z,
                 data=df,
                 method = "ML",
                 control=nlme::lmeControl(opt="optim",returnObject=TRUE))

bj <- ols$coefficients[(s+1):(2*s)]
vj <- (coef(summary(ols))[(s+1):(2*s),2])^2
wj <- 1/vj

# Look at our point estimates, just for kicks
summary( bj )
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> -0.49263 -0.13812  0.01570  0.02199  0.23083  0.43542
```

The `bj` are our individual site impact estimates, and the `vj` are the site-specific squared standard errors. The `wj` are the inverse of these, giving our weights. Let's dissect the testing approach to see how we get point estimates:

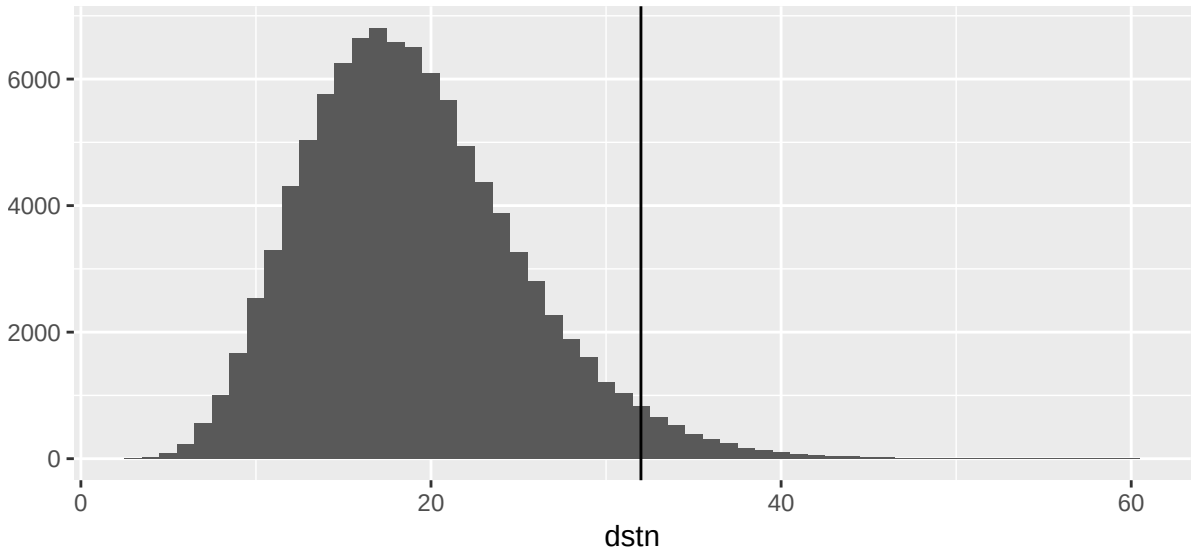
```
# ATE
bbar <- sum(wj*bj)/(sum(wj))
bbar
#> [1] -0.01352921

# Dispersion measure
q <- sum((bj - bbar)^2/vj)
q
#> [1] 31.99257

# P-value
pval <- pchisq(q,df=(s-1),lower.tail=FALSE)
pval
#> [1] 0.03131497
```

To illustrate, here is the reference distribution and our observed test statistics:

```
dstn = rchisq( 100000, df= (length(bj)-1))
qplot( dstn, binwidth= 1 ) + geom_vline(xintercept=q )
```



Now we make a confidence interval

```
tau_test <- seq(0, 2, 0.005)

alpha = 0.05

lowbound <- qchisq(alpha,s-1)
highbound <- qchisq(1 - alpha,df=(s-1))
lowbound
#> [1] 10.11701
highbound
#> [1] 30.14353
```

We are going to reject any tau with associated Q stat outside these bounds:

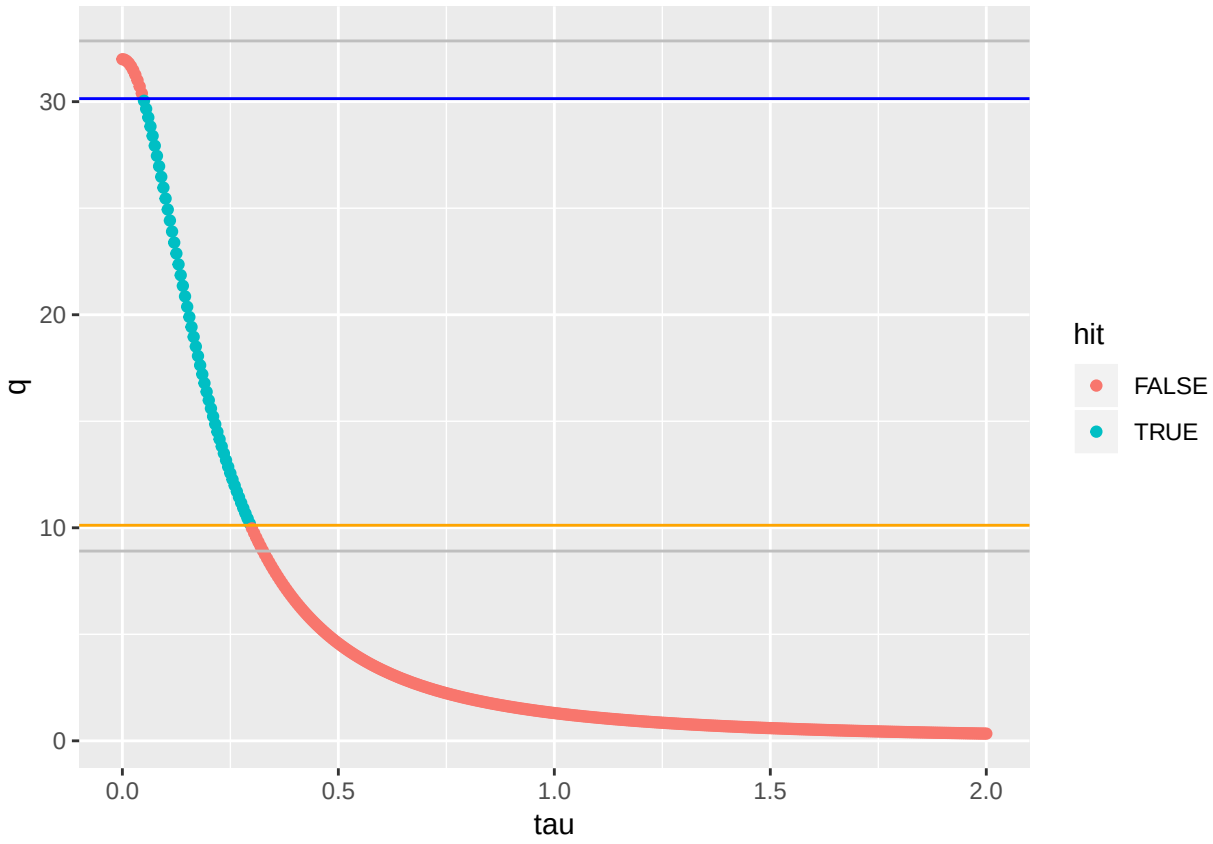
```
q_invert <- c()
CI_95 <- c()
for (i in 1:length(tau_test)){
    #calculate new denominator that takes into account tau^2
    denom <- vj + tau_test[i]^2
    #new q-stat
    q_invert[i] <- sum((bj - bbar)^2/denom)
    #Compare q.stat to lower and upperbounds (Weiss et al, JREE p. 55)
    CI_95[i] <- (q_invert[i]>=lowbound & q_invert[i]<=highbound)
}
```

Let's plot our results

```
df = data.frame( tau = tau_test, q = q_invert, hit = CI_95 )

# For illustration
lowlowbound <- qchisq(alpha/2,s-1)
highhighbound <- qchisq(1 - alpha/2,df=(s-1))

ggplot( df, aes( tau, q, col=hit ) ) +
    geom_point() +
    geom_hline(yintercept=c(lowbound,highbound, lowlowbound, highhighbound),
               col=c("orange","blue", "grey", "grey")  )
```



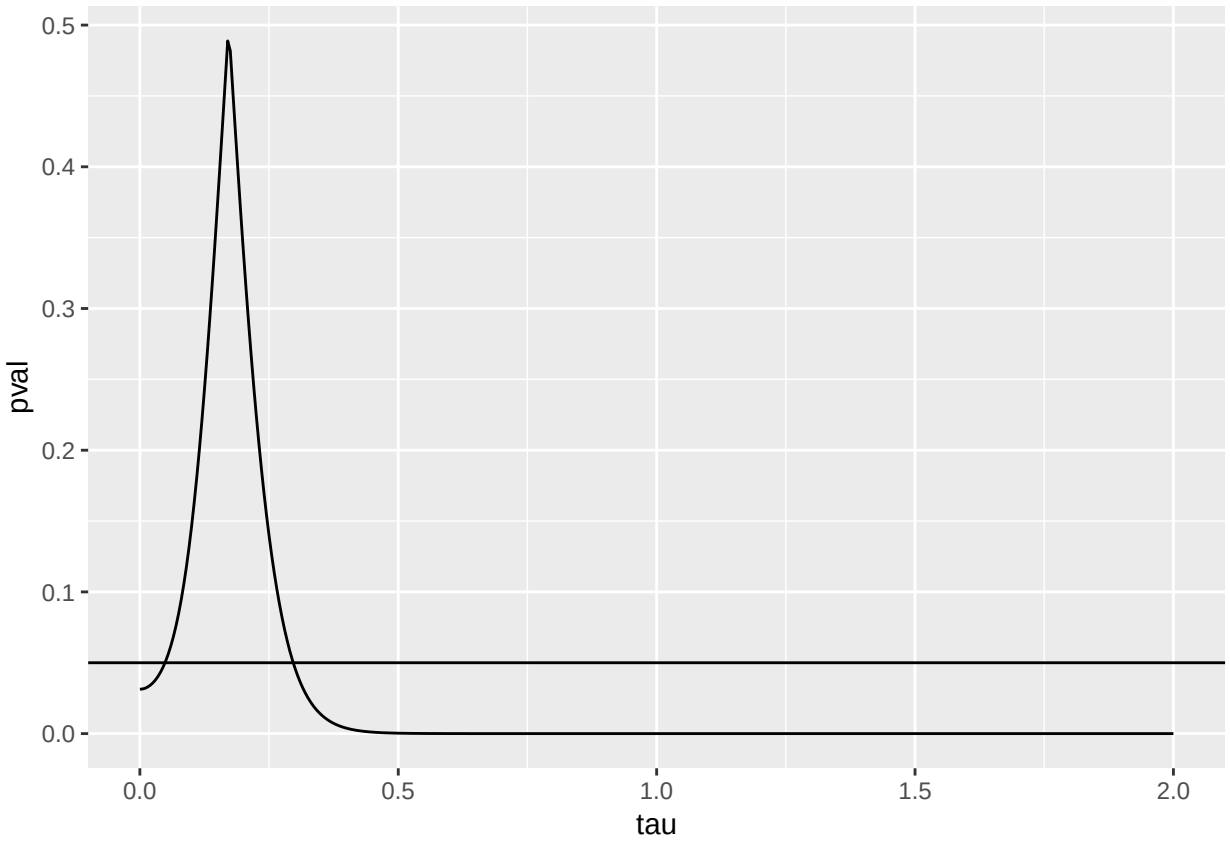
Everything between the bounds correspond to values for our confidence interval

```
if ( length(tau_test[CI_95 == 1]) == 0 ) {
    CI_low <- NA
    CI_high <- NA
} else {
    CI_high <- max(tau_test[CI_95 == 1])
    CI_low <- min(tau_test[CI_95 == 1])
}

CI_low
#> [1] 0.05
CI_high
#> [1] 0.295
```

Now let's get a point estimate. For each Q stat we can get a p-value:

```
pval.low <- pchisq(q_invert,df=(s-1),lower.tail=FALSE)
pval.high <- pchisq(q_invert,df=(s-1),lower.tail=TRUE)
df$pval = pmin( pval.low, pval.high )
ggplot( df, aes( tau, pval ) ) +
    geom_line() +
    geom_hline( yintercept = alpha )
```




```
pos = which.max( df$pval )

tau.hat.2 = df$tau[[ pos ]]
tau.hat.2
#> [1] 0.17
```

Our point estimate is 0.17, compared with 0.014 from the FIRC model.

In the above figure, note how all the τ corresponding to p-values greater than 0.05 correspond to our confidence interval.

Also note this is a 90% confidence interval, not 95%. These align a bit better, I think, with the testing for treatment variation at the 0.05 level since that test is a one-sided test, and our interval's boundary should include or exclude 0 depending on our test. If we did a 95% confidence interval, our tails would only hold 2.5% and thus our interval would contain 0 in this case. Note the grey lines that demark the thresholds for the Q statistic for this more stringent test. Our highest Q is less than our upper gray line, indicating we would not reject 0 at the 0.025 level, which we would need to do for our two-tailed 95% confidence interval.