Using pgsc

Philip Barrett

2018-10-20

Introduction

The gsc package computes the generalized synthetic control estimator described in (Powell 2017). The estimator controls for a rich specification of omitted variables in a panel, beyond those covered by time and unit fixed effects (e.g. spatially-correlated time fixed effects). This vignette describes an extended example which illustrates how to generate point estimates, as well as how to test hypotheses using both a standard bootstrap and a constrained-estimator approach described in (Powell 2017).

Data generating process

We start by generating data using a process which cannot be recovered using a simple fixed effects model. The data generating process is:

$$Y_{it} = \alpha_i + \eta_t + \mu_i' \lambda_t + b' D_{it} + r' X_{it} + \epsilon_{it}$$

So the α_i and η_t are standard time and unit fixed effects, λ_t is a Q-dimensional vector of time-varying factors with unit-specific weights μ_i, the D_{it} is an M-dimensional vector of continuous treatments, and the X_{it} are vectors of observed confounding variables. The primary objective of inquiry is to recover b, the impact of the treatment. Because the λ_t and μ_i are unobserved if they are correlated with the treatments, then a standard time and unit fixed effects regression will fail to recover the parameter of interest, b.

We generate such a data set as follows:

```
rm(list=ls()) ; set.seed(42)
library(pgsc)

### Parameters
NN <- 15                    # Number of units
TT <- 50                    # Number of periods
MM <- 2                     # Number of treatments
RR <- 3                     # Number of covariates
SS <- 2                     # Number of unit FEs
QQ <- 3                     # Number of time FEs
b <- c(1,2)                 # Treatment coefficients
sq <- matrix( c( 1, .2, .3, -.9, -.1, .2 ), nrow=SS, ncol=QQ )
                            # Weighting matrix on time-varying factors
p <- t( matrix(c( -1, 0, .2, .5, .2, 0), nrow=MM, ncol=RR ) )
                            # The covariance of X and D
r <- .1 * c( .5, 1, 2)      # Coefficient on observed covariates
sig <- .2                   # Noise sd
sig.y <- 5                  # Unit FEs
sig.t <- 4                  # Time FE noise

### Data
set.seed(42)
```

```
fes <- matrix( rnorm(NN * SS, 0, sig.y), NN, SS )     # Unit fixed effects
tfes <- matrix( rnorm(TT * QQ, 0, sig.t ), TT, QQ )  # Time fixed effects
X <- array( rnorm(NN*RR*TT), c( NN, RR, TT ))        # Covariates
D <- array(NA, dim=c( NN, MM, TT ))
D[] <- apply(X, 3, function(x) x%*%p)
D <- D + array( rnorm(NN*MM*TT), c( NN, MM, TT ))              # Treatments, correl. w/ X
Y <- sapply( 1:TT, function(i) D[,,i] %*% b + X[,,i] %*% r ) +  # Treatment & covariates
  fes[,1] + rep(1,NN) %*% t(tfes[,1]) +                        # FEs and TFE
  fes %*% sq %*% t(tfes) +                                     # Time-unit interaction
  rnorm( NN*TT, 0, sig )                                       # Noise
pgsc.dta <- data.frame( n=state.abb[1:NN], t=rep(1:TT,each=NN), y=c(Y),
                   do.call(rbind,lapply(1:TT,function(i)D[,,i])),
                   do.call(rbind,lapply(1:TT,function(i)X[,,i])) )
names(pgsc.dta) <- c('n','t','y', paste0('D',1:MM), paste0('X',1:RR) )
    # Bind into a data frame
```

This dataset is also stored in the package and can be loaded using `data("pgsc.dta")`. Note that the parameter b that we wish to recover has value $(1, 2)'$.

Panel regression

The panel regression fails to recover the correct value of b due to the correlation of X_{it} with D_{it}.

```
### Panel regression
library(plm)
#> Loading required package: Formula
pan <- plm( y ~ D1 + D2 + X1 + X2 + X3, pgsc.dta, effect = 'twoways', index = c('n','t'))
summary(pan)
#> Twoways effects Within Model
#>
#> Call:
#> plm(formula = y ~ D1 + D2 + X1 + X2 + X3, data = pgsc.dta, effect = "twoways",
#>     index = c("n", "t"))
#>
#> Balanced Panel: n = 15, T = 50, N = 750
#>
#> Residuals:
#>       Min.    1st Qu.     Median    3rd Qu.       Max.
#> -117.80204  -15.48917   -0.54395   16.44607  132.34232
#>
#> Coefficients:
#>    Estimate Std. Error t-value Pr(>|t|)
#> D1  0.33096    1.27862  0.2588  0.79584
#> D2  0.27527    1.18746  0.2318  0.81675
#> X1 -1.02604    1.75666 -0.5841  0.55936
#> X2  3.55329    1.45885  2.4357  0.01512 *
#> X3 -0.58031    1.33523 -0.4346  0.66398
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Total Sum of Squares:    756860
#> Residual Sum of Squares: 745190
```

```
#> R-Squared:      0.015419
#> Adj. R-Squared: -0.082894
#> F-statistic: 2.13302 on 5 and 681 DF, p-value: 0.059783
```

The coefficients on the treatment variables are far from their true values, $(1, 2)'$

The source of the bias is that there is an omitted variable, which is time varying but spatially correlated. This interation of the spatial and time components means that time and period fixed effects cannot recover the true data generating process. If we have access to (something correlated with) this variable, then panel regression with fixed effects can, of course, recover the true data generating process:

```
### Panel regression
library(plm)
pgsc.dta.copy <- pgsc.dta
pgsc.dta.copy$ov <- c(fes %*% sq %*% t(tfes)) * 2 + rnorm(NN*TT) * 3
pan.2 <- plm( y ~ D1 + D2 + X1 + X2 + X3 + ov, pgsc.dta.copy, effect = 'twoways', index = c('n','t'))
summary(pan.2)
#> Twoways effects Within Model
#>
#> Call:
#> plm(formula = y ~ D1 + D2 + X1 + X2 + X3 + ov, data = pgsc.dta.copy,
#>     effect = "twoways", index = c("n", "t"))
#>
#> Balanced Panel: n = 15, T = 50, N = 750
#>
#> Residuals:
#>      Min.   1st Qu.    Median   3rd Qu.      Max.
#> -4.118867 -0.975830 -0.046624  0.983181  4.526300
#>
#> Coefficients:
#>      Estimate Std. Error  t-value  Pr(>|t|)
#> D1 1.05990920 0.06005611  17.6486 < 2.2e-16 ***
#> D2 1.99885524 0.05584710  35.7916 < 2.2e-16 ***
#> X1 0.16566058 0.08251751   2.0076  0.045083 *
#> X2 0.03777031 0.06879672   0.5490  0.583177
#> X3 0.18892907 0.06271522   3.0125  0.002687 **
#> ov 0.49870678 0.00089838 555.1170 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Total Sum of Squares:    756860
#> Residual Sum of Squares: 1640.8
#> R-Squared:      0.99783
#> Adj. R-Squared: 0.99761
#> F-statistic: 52165.3 on 6 and 680 DF, p-value: < 2.22e-16
```

The generalized synthetic control estimator: point estimates

The GSC estimator provides a vector of coefficients $\hat{b}$ and an estimated weighing matrix W solving:

$$(\hat{b}, \hat{W}) = \arg\min_{b,W} \frac{1}{2NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \left[Y_{it} - b' D_{it} - \sum_{j \neq i} w_{ij} \left(Y_{jt} - b' D_{jt} \right) \right]^2$$
$$\text{s.t.} \quad \forall\, i : \sum_{j \neq i} w_{ij} = 1$$

In other words, the GSC estimator minimizes the squared difference in outcomes unexplained by the treatment variable between each unit and a unit-specific counterfactual. The counterfactual itself is a weighted average of the other units, (i.e. the impact of the omitted variables).

The function `pgsc` computes a number of variants of the GSC estimator using an iterative approach. It works by optimizing over the coefficients b and the weights W each in turn, iterating until the maximum difference in iterations is suitably close to zero.[1]

```
### Compute the point estimate
wt.init <- matrix( 1 / (NN-1), NN, NN-2 )
b.init <- pan$coefficients[c('D1','D2')]
sol.it <- pgsc(pgsc.dta, dep.var = 'y', indep.var = c('D1','D2'), b.init = b.init,
                      method='onestep')
#> Iteration report:
#>   1: diff = 1.549
#>   2: diff = 0.06803
#>   3: diff = 0.005163
#>   4: diff = 0.0004807
#>   5: diff = 4.295e-05
#>   6: diff = 7.269e-06
summary(sol.it)
#> Generalized synthetic control estimation, using 7 iterations
#>
#> Treatment coefficients:
#>        D1        D2
#> 0.8737682 1.8978044
#>
#> Max optimization error =  7.031578e-05
#> Max iteration difference =  7.268573e-06
```

While the resulting estimates are superior to the fixed effects panel estimates, they still differ considerably from the true value. The `pgsc` package therefore provides functionality to compute two-step estimators suggested in (Powell 2017), which re-weight the objective function to minimize the impact of the units where the model fit is bad. There are two two-step variants: an "average" one which uses the average unit-specific error from the one-step estimator, and an "individual" one which allows for unit-specific estimates of b in the first stage. These can be computed as follows:

```
### Compute point estimates from the two-step estimators
sol.2.step.aggte <- pgsc(pgsc.dta, dep.var = 'y', indep.var = c('D1','D2'),
                                 b.init = sol.it$b, method='twostep.aggte',
                                 print.level=-1)
sol.2.step.indiv <- pgsc(pgsc.dta, dep.var = 'y', indep.var = c('D1','D2'),
                                 b.init = sol.2.step.aggte$b, method='twostep.indiv',
                                 print.level=-2)
```

[1] This is slower than solving simultaneously for weights and parameters, but seems to be much more reliable. Presumably, this is because the simultaneous solution is a fourth order problem, whereas the individual weight/coefficient minimization problems are quadratic at each step. This improves convergence in a numerical optimizer.

```
#> i = 1 ; err = 3.018e-07
#> i = 2 ; err = 4.844e-07
#> i = 3 ; err = 4.561e-07
#> i = 4 ; err = 3.917e-05
#> i = 5 ; err = 6.568e-07
#> i = 6 ; err = 2.635e-07
#> i = 7 ; err = 5.925e-06
#> i = 8 ; err = 6.867e-07
#> i = 9 ; err = 0.0001681
#> i = 10 ; err = 7.866e-13
#> i = 11 ; err = 6.686e-07
#> i = 12 ; err = 5.142e-07
#> i = 13 ; err = 5.376e-06
#> i = 14 ; err = 0.0002276
#> i = 15 ; err = 3.486e-07
sol.compare <- rbind( pan$coefficients[c('D1','D2')], sol.it$b, sol.2.step.aggte$b,
                      sol.2.step.indiv$b, b )
rownames(sol.compare) <- c( 'Panel FEs', 'GSC onestep', 'Aggte two-step GSC',
                            'Indiv two-step GSC', 'Truth')
print(sol.compare)
#>                              D1        D2
#> Panel FEs           0.3309601 0.2752742
#> GSC onestep         0.8737682 1.8978044
#> Aggte two-step GSC  0.9874209 1.9934019
#> Indiv two-step GSC  1.0015500 1.9927136
#> Truth               1.0000000 2.0000000
```

Note that the GSC estimates are much closer than the panel regression estimates to the truth, particularly when weighted.

The generalized synthetic control estimator: hypothesis testing

The package also includes functions for hypothesis testing, using the bootstrap method proposed in (Powell 2017). This approach requires re-estimating the model under the hypothesized restriction $g(\theta) = 0$. If the null hypothesis is true, then the restriction does not bind. And so the slope of the objective function should be zero when the constraint is enforced. After computing the appropriate derivative under the restriction, one can approximate its variance under the null by bootstrap using a symmetry assumption. This provides a quick way to approximate the distribution of the gradient without having to re-solve the constrained problem, which may be time-consuming.

We start by defining the restriction(s) to be tested. We test the restriction:

$$\frac{b_1}{1 - b_2} = A$$

This is true when $A = -1$. Both the function and gradient are required, which we define as follows:

```
A <- - 1
g <- function(b) b[1] / ( 1 - b[2] ) - A
g.grad <- function(b) c( 1 / ( 1 - b[2] ), b[1] / ( 1 - b[2] )^2 )
```

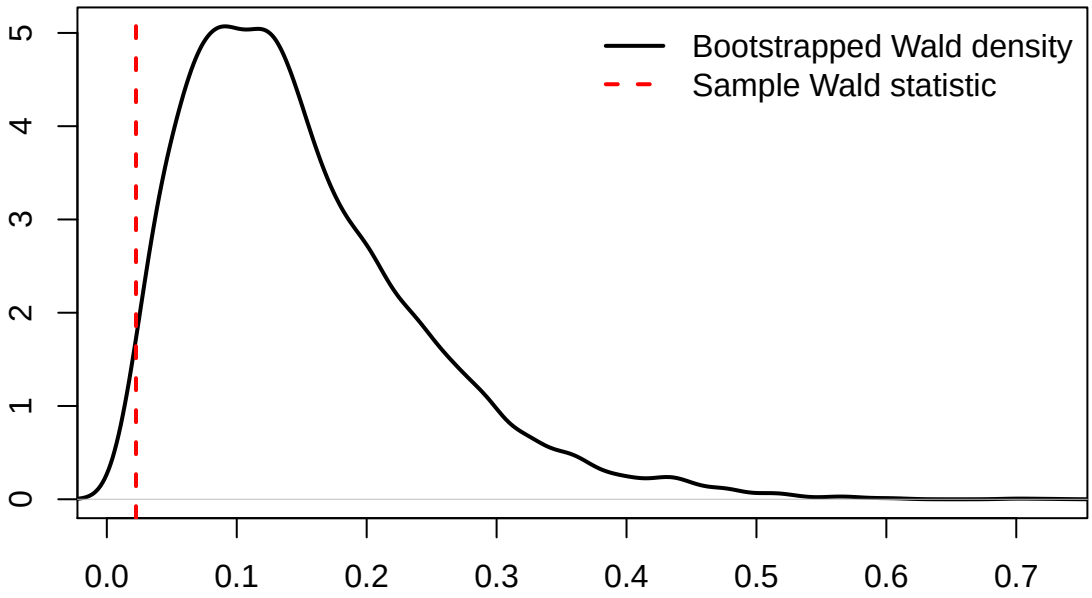
Note that the current version of the package only allows for single equation restrictions to be tested. Extending this is a point for further development. In the meantime, multiple restrictions can be implemented as a sum of squares of individual restrictions.

To test the restriction, we first solve the restricted model, using the auxilary argument `g.i`, and then test that $g(b) = 0$.

```
sol.2.step.rest <- pgsc(pgsc.dta, dep.var = 'y', indep.var = c('D1','D2'), b.init = b.init,
                        method='twostep.indiv', g.i=g, g.i.grad=g.grad )
#> Iteration report:
#>   1: diff = 1.573
#>   2: diff = 0.03383
#>   3: diff = 0.00175
#>   4: diff = 0.0001191
#>   5: diff = 1.906e-05
#>   6: diff = 1.178e-06
#> i = 1 ; err = 3.752e-07
#> i = 2 ; err = 4.192e-07
#> i = 3 ; err = 2.906e-07
#> i = 4 ; err = 1.503e-05
#> i = 5 ; err = 3.989e-07
#> i = 6 ; err = 2.625e-07
#> i = 7 ; err = 3.515e-06
#> i = 8 ; err = 6.874e-07
#> i = 9 ; err = 0.0001866
#> i = 10 ; err = 2.67e-11
#> i = 11 ; err = 5.239e-07
#> i = 12 ; err = 6.073e-07
#> i = 13 ; err = 9.724e-06
#> i = 14 ; err = 9.089e-07
#> i = 15 ; err = 3.527e-07
#> Iteration report:
#>   1: diff = 0.09708
#>   2: diff = 0.0128
#>   3: diff = 0.002606
#>   4: diff = 0.0004848
#>   5: diff = 9.04e-05
#>   6: diff = 1.681e-05
#>   7: diff = 2.984e-06
wald.test.g <- pgsc.wald.test( pgsc.dta, dep.var = 'y', indep.var = c('D1','D2'),
                               sol.rest = sol.2.step.rest, n.boot = 10000 )
```

The `wald.test.g` object has associated summary and plot methods

```
summary(wald.test.g)
#> Non-linear Wald test of restricted hypothesis, using 10000 bootstrapped samples
#>
#> Restricted coefficients:
#>        D1        D2
#> 0.9973405 1.9973405
#>
#> Wald statistic =  0.02246179
#> p-value =  0.986
plot(wald.test.g)
```

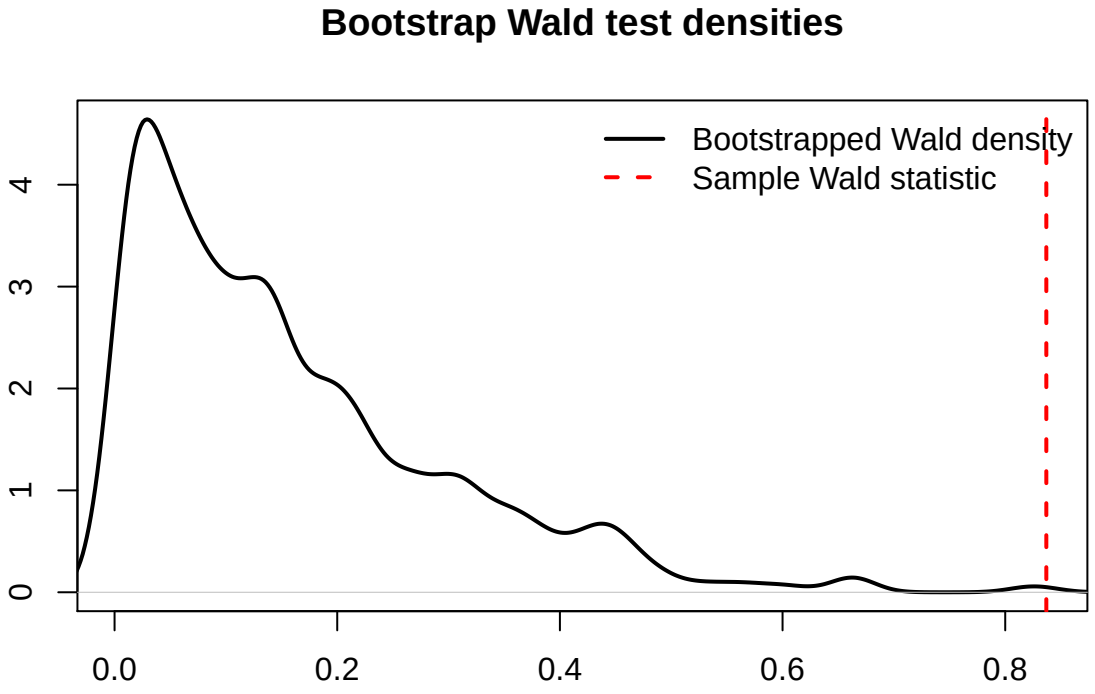


With a p-value of over 0.98, the hypothesis cannot be rejected at any reasonable significance level. This is relief, as it is true; when evaluated at the true value of $b = (1, 2)$, $g(b) = 0$.

We can also check that a false hypothesis is rejected by re-solving the under a false restriction:

The resulting p-value is close to zero, so we can comfortably reject the hypothesis.

```
summary(wald.test.g.2)
#> Non-linear Wald test of restricted hypothesis, using 10000 bootstrapped samples
#>
#> Restricted coefficients:
#>        D1        D2
#> 0.4797704 0.7601148
#>
#> Wald statistic =  0.836904
#> p-value =  5e-04
plot(wald.test.g.2)
```



References

Powell, David. 2017. "Synthetic Control Estimation Beyond Case Studies: Does the Minimum Wage Reduce Employment?"