Supplemental Appendix for Improving the Interpretation of Random Effects Regression Results

Soren Jordan*
Andrew Q. Philips†

December 10, 2021

Forthcoming at *Political Studies Review*

Contents

*sorenjordanpols@gmail.com. Associate Professor, Department of Political Science, Auburn University, Haley 8024, Auburn, AL 36849.

†andrew.philips@colorado.edu. Assistant Professor, Department of Political Science, University of Colorado Boulder, UCB 333, Boulder, CO 80309-0333.

1 Code to Evalaute Demeaning

1.1 Stata

```
* Example showing quasi-demeaning and obtaining post-estimation variances
* ------------------------------------------------------------------------------

* ----------- PRELIMINARIES ----------------------------
* open up Stata's longitudinal dataset of women's wages:
webuse nlswork, clear
xtset idcode year
* create a time-invariant variable, just to show that quasi-demeaning applies here as well:
gen white = .
replace white = 0 if race != .
replace white = 1 if race == 1
drop if missing(ln_wage) | missing(wks_work) | missing(union) | missing(age) | missing(white)

* ------------------------------------------------------------------------------
* 1. Estimate model and store all variable names:
xtreg ln_wage wks_work union age white, re theta
global varnames = "ln_wage wks_work union age white _cons" // DV, IVs and constant
* ------------------------------------------------------------------------------

* ------------------------------------------------------------------------------
* 2. create max(T_i):
bysort idcode: gen maxT = _n
bysort idcode: replace maxT = maxT[_N]
* ------------------------------------------------------------------------------

* ------------------------------------------------------------------------------
* 3. create \bar{x}_i:
foreach var of global varnames {
bysort idcode: egen `var'_bar = mean(`var')
}
* ------------------------------------------------------------------------------

* ------------------------------------------------------------------------------
* 4. compute theta_i:
gen theta_i = 1 - sqrt(e(sigma_e)^2 / (maxT*e(sigma_u)^2 + e(sigma_e)^2))
preserve // run all the way to restore
collapse theta_i, by(idcode)
su theta_i, det // this will match exactly w/ theta option in xtreg
restore
* ------------------------------------------------------------------------------

* ------------------------------------------------------------------------------
* 5. quasi-demean:
foreach var of global varnames {
gen `var'_qdmean = `var' - theta_i*`var'_bar
}
* ------------------------------------------------------------------------------

* ------------------------------------------------------------------------------
* 6. Check: does OLS with quasi-demeaned vars get the same result as RE-FGLS?
```

```
reg ln_wage_qdmean wks_work_qdmean union_qdmean age_qdmean white_qdmean _cons_qdmean, nocon
xtreg ln_wage wks_work union age white, re
* ------------------------------------------------------------------------------

* ------------------------------------------------------------------------------
* 7. Summarize and compare quasi-demeaned variance vs variance in original variable as
*  needed for any counterfactual claims
* e.g., looking at weeks worked:
su wks_work wks_work_qdmean
twoway kdensity wks_work || kdensity wks_work_qdmean, legend(order(1 "Original" 2 "Quasi-demeaned"))

* age
su age age_qdmean
twoway kdensity age || kdensity age_qdmean, legend(order(1 "Original" 2 "Quasi-demeaned"))
* ------------------------------------------------------------------------------
```

1.2 R

```
# This should work in most instances. The code is simple enough: pass the plm
#  model object, the predictor as a string, and the grouping variable as a string.
#  It returns the quasi demeaned variable, which should be investigated in any
#  standard number of ways (summarized, quantiles, etc.) to determine the
#  appropriate change to discuss.

re.demean.plm <- function(model, predictor, group) {
data <- data.frame(model$model, index(model), summary(model)$ercomp$theta)
names(data) <- c(names(model$model), names(index(model)), "theta")
data$xmean <- as.numeric(ave(data[[predictor]], data[[group]],
                             FUN = function(x) mean(x, na.rm = T)))
x.full <- as.numeric(data[[predictor]])
x.qdmean <- x.full - data$theta*data$xmean
x.qdmean
}




# This should work in most instances. It requires a bit more: the dataset (since
#  lmer doesn't output the data used in the model object that I can find), the lmer
#  model object, the predictor as a string, the grouping variable as a string, and
#  the dependent variable as a string. It returns the quasi demeaned
#  variable, which should be investigated in any standard number of ways
#  (summarized, quantiles, etc.) to determine the appropriate change to discuss.

re.demean.lmer <- function(data, model, predictor, group, dv) {
data.lmer <- data.frame(get(group, data), get(dv, data))
for(i in 2:length(names(fixef(model.lmer)))) {
data.lmer <- data.frame(data.lmer, get(names(fixef(model.lmer))[i], data))
}
names(data.lmer) <- c(group, dv, names(fixef(model.lmer))[2:length(names(fixef(model.lmer)))])
data.lmer <- na.omit(data.lmer)
data.lmer$xmean <- as.numeric(ave(data.lmer[[predictor]], data.lmer[[group]],
                                  FUN = function(x) mean(x, na.rm = T)))
x.full <- as.numeric(data.lmer[[predictor]])
```

```
maxT <- as.numeric(ave(data.lmer[[group]], data.lmer[[group]], FUN = function(x) length(x)))
sig_u <- as.data.frame(VarCorr(model))[as.data.frame(VarCorr(model))$grp == group, "sdcor"]
sig_e <- as.data.frame(VarCorr(model))[as.data.frame(VarCorr(model))$grp == "Residual", "sdcor"]
theta <- 1 - sqrt(sig_e^2/(maxT*(sig_u)^2 + sig_e ^2))
x.qdmean <- x.full - theta*data.lmer$xmean
x.qdmean
}
```