Notes for POLS 607: Panel data*

Casey Crisman-Cox

Spring 2025

*These notes are for personal use only and not for distribution. They are likely rife with errors, typos, and are not referenced. This is not my own work, but simply notes for me to lecture from.

Contents

1 Review: Linear panel model basics

We will start with a review of the standard panel data models. Before getting into that we'll need some assumptions. The first of which describes what a basic panel is:

Assumption A1 *The data generating process is linear-in-the-parameters, such that*

$$y_{it} = \alpha_i + \beta' x_{it} + \gamma' z_i + \varepsilon_{it}.$$

Usually we think of i as "units" (individuals, states, countries, dyads, etc) and t as "within-unit" observations (typically time, but could be multiple individuals within a unit, etc). We will let $i = 1, \ldots, N$, $t = 1, \ldots, T$ and NT be the total number of observations. To make exposition easier, we will often assume a "balanced" panel where T is the same for each i. When necessary, we will talk about cases where this distinction matters. Neither x_{it} nor z_i contain a constant term, instead α_i reflects a general situation where each unit has its own constant term.

1.1 Pooled panel model

For now we will simplify that further and assume that $\alpha_i = \alpha$ for all i (**Assumption A1.A**). Additionally, x_{it} is a variable that changes both across and within units, while z_i is constant within units but variable across units.

Given this setup, we are unlikely to have independent observations. After all, if our panel is a collection of N separate time series, then assuming independence is a pretty long stretch from the start, but we will typically want a type of independence assumption

Assumption A2 *Each unit* $(x_i, z_i, \varepsilon_i)$ *is drawn iid.*

The notation $x_i = (x_{i1}, \ldots, x_{iT})$ used here refer to all observations of x_{it} within unit i. Here we are making the assumption that each block of observations is independent of the rest and that the units are drawn from some population process. This is not perhaps super convincing in some cases, but it is a convenience assumption and a start.

We are not currently making any assumptions about the dependency structure on the within-unit observations. However, we will need to make some kind of exogeneity assumption (as always)

Assumption A3 *There is strict exogeneity within units,* $\mathrm{E}[\varepsilon_{it}|x_i, z_i] = 0$.

This tells us that within-each unit, we assume that the error term is independent of the observables (i.e., no unobserved confounding within units).

Let $\mathbf{X} = [1\ X\ Z]$ and let $\theta = (\alpha, \beta, \gamma)'$, then the *pooled* OLS estimator for the panel model is

$$\begin{aligned}\hat{\theta}_p &= \left(\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x}_{it}\mathbf{x}_{it}'\right)^{-1}\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x}_{it}y_{it} \\ &= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'y,\end{aligned}$$

or (by A1.A)

$$\hat{\theta}_p = \theta + \left(\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x}_{it}\mathbf{x}_{it}'\right)^{-1}\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x}_{it}\varepsilon_{it}.$$

By strict exogeneity within unit (A3) and independence across units (A2) we have

$$\mathrm{E}[\varepsilon_{it}|x_i, z_i] = \mathrm{E}[\varepsilon_{it}|\mathbf{X}] = 0.$$

Thus we can apply iterated expectations to get

$$\begin{aligned}\mathrm{E}[\mathrm{E}[\hat{\theta}_p|\mathbf{X}]] &= \theta + \mathrm{E}\left[\left(\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x}_{it}\mathbf{x}_{it}'\right)^{-1}\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x}_{it}\,\mathrm{E}[\varepsilon_{it}|\mathbf{X}]\right] \\ &= \theta + 0.\end{aligned}$$

Which gives us our first property,

Property A1 *Under Assumptions A1.A, A2, and A3 the pooled estimator $\hat{\theta}_p$ is unbiased, if it exists.*

Note there will be times when we get to dynamics where strict within-unit exogeneity doesn't make sense. For now, we'll go with it.

Further, we will impose a rank condition

Assumption A4 *The matrix $\mathbf{Q} = \mathrm{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\mathbf{x}_{it}'\right] < \infty$ has full rank.*

With this assumption, we assert that the DGP for the T within-unit observations are well-behaved and $X_i'X_i$ has full rank for each unit. Likewise, since it is in a quadratic form, the matrix will be positive definite under this assumption.

Finally, we will include some hand-wavy moment conditions on the data and errors.

Assumption A5 *Additional moment assumptions that allow us to apply a central limit*

theorem (CLT).

For review, let's briefly restate these theorems along with some other useful results

Theorem 1 (Weak Law of Large Numbers) *Let $X_1, \ldots, X_N$ be an iid sequence of random variables, where each $X_i \in \mathbb{R}^k$ has a finite absolute first moment $\mathrm{E}[X_i] < \infty$. Then $\frac{1}{N}\sum_{i=1}^{N} X_i \xrightarrow{p} \mathrm{E}[X_i]$.*

Theorem 2 (Central Limit Theorem) *Let $X_1, \ldots, X_N$ be iid random variables with expected value μ and variance Ω (both finite), then for all $x \in \mathbb{R}$*

$$\sqrt{N}\left(\frac{1}{N}\sum_{i=1}^{N} X_i - \mu\right) \xrightarrow{d} N(0, \Omega).$$

Theorem 3 (Continuous mapping theorem) *Consider a sequence of random variables $X_n = X_1, \ldots X_N$ and a continuous function g*

- *If $X_n \xrightarrow{d} X$, then $g(X_n) \xrightarrow{d} g(X)$*
- *If $X_n \xrightarrow{p} X$, then $g(X_n) \xrightarrow{p} g(X)$*

Theorem 4 (Functions preserve iid) *Let g be a continuous function and let X and Y be random variables*

- *If X and Y are independent, then $g(X)$ and $g(Y)$ are independent random variables*
- *If X and Y are identically distributed, then $g(X)$ and $g(Y)$ are identically distributed*

Theorem 4 is a deceptively powerful result. We can now say that since we know x_i and ε_i are each iid, then $g(x_i)$ and $g(\varepsilon_i)$ will retain these properties for continuous g.

Theorem 5 (Slutsky's Theorem) *Let X_i and Y_i be sequences of random variables.*

1. *If $X_n \xrightarrow{d} X$ and $Y_n \xrightarrow{p} y$ (where y is a constant), then*
   - $X_n Y_n \xrightarrow{d} Xy$
   - $Y_n^{-1} X_n \xrightarrow{d} (y^{-1})X$, *if y^{-1} exists*
2. *If $X_n \xrightarrow{p} x$ and $Y_n \xrightarrow{p} y$ (where x and y are constants), then*
   - $X_n Y_n \xrightarrow{p} xy$
   - $Y_n^{-1} X_n \xrightarrow{p} (y^{-1})X$, *if y^{-1} exists*

We can now apply a standard LLN type argument

1. Let's start with the expression $\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\mathbf{x}_{it}'\right)$, what happens here as N grows? Assumption A4 will let us know that $\mathrm{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\mathbf{x}_{it}'\right]$ is finite. Likewise, Assumption A2 tells us that x_i and x_j are iid for $i \neq j$. Note that $\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\mathbf{x}_{it}'$ and $\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{jt}\mathbf{x}_{jt}'$ are functions of iid random variables and so are themselves iid by Theorem 4. As such we can apply the LLN to get

$$\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\mathbf{x}_{it}'\right) \xrightarrow{p} \mathrm{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\mathbf{x}_{it}'\right] = \mathbf{Q}$$

2. By Assumption A4, $\mathbf{Q}$ has full rank and will be positive definite. Because it's positive definite the inverse function will be continuous and we can apply the CMT, to get

$$\left[\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\mathbf{x}_{it}'\right)\right]^{-1} \xrightarrow{p} \mathbf{Q}^{-1}.$$

3. Now consider $\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\varepsilon_{it}\right)$. Using Assumption A2 again we know that $(x_i'\varepsilon_i)_{i=1}^{N}$ represents N iid random vectors. We can apply the LLN to see that

$$\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\varepsilon_{it}\right) \xrightarrow{p} \mathrm{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\varepsilon_{it}\right],$$

   where

$$\begin{aligned}\mathrm{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\varepsilon_{it}\right] &= \frac{1}{T}\sum_{t=1}^{T}\mathrm{E}\left[\mathbf{x}_{it}\varepsilon_{it}\right] \\ &= \mathrm{E}_x\left[\mathrm{E}\left[\mathbf{x}_{it}\varepsilon_{it}|\mathbf{X}\right]\right] \\ &= 0\end{aligned}$$

   by Assumption A2–A3.

4. We can now apply Slutsky's theorem to say:

$$\left(\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x}_{it}x_{it}'\right)^{-1}\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x}_{it}\varepsilon_{it} \xrightarrow{p} \mathbf{Q}^{-1}0 = 0.$$

5. Finally, since the sum operator is continuous we can again apply the continuous mapping

theorem to get

$$\begin{aligned}\hat{\theta}_p &= \theta + \left(\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x}_{it}x_{it}'\right)^{-1}\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x}_{it}\varepsilon_{it} \\ &\xrightarrow{p} \theta + 0 = \theta\end{aligned}$$

Property A2 *Under A1.A and A2–A4, as $N \to \infty$ the pooled estimator exists. The pooled estimator is consistent for θ.*

Asymptotic normality follows in a similar way, but let's recap it too,

1. Rewrite the estimator to look like Theorem 2.

$$\sqrt{N}(\hat{\theta}_p - \theta) = \left(\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x}_{it}\mathbf{x}_{it}'\right)^{-1}\frac{\sqrt{N}}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x}_{it}\varepsilon_{it}$$

2. From above we know

$$\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\mathbf{x}_{it}' \xrightarrow{p} \mathrm{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\mathbf{x}_{it}'\right] = \mathbf{Q}$$

3. The remaining term

$$\sqrt{N}\left(\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\varepsilon_{it}\right)\right)$$

   looks like the CLT, right? and we know that

$$\mathrm{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\varepsilon_{it}\right] = 0,$$

   from above. So, we can apply that to get to

$$\sqrt{N}\left(\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\varepsilon_{it}\right)\right) \xrightarrow{d} N\left(0, \Sigma_N\right).$$

   We will assume that $\Sigma_N = \mathrm{Var}\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\varepsilon_{it}\right)$ exists under Assumption A5.

4. Finally, we can combine terms using Slutsky's theorem to get

$$\sqrt{N}(\hat{\theta}_p - \theta) \xrightarrow{d} N(0, \mathbf{Q}^{-1}\Sigma_N\mathbf{Q}^{-1}).$$

All together this gives us our next property:

Property A3 *Under A1.A and A2–A5 the pooled estimator $\hat{\theta}_p$ is asymptotically normal such that*

$$\sqrt{N}(\hat{\theta}_p - \theta) \xrightarrow{d} N(0, \mathbf{Q}^{-1}\Sigma_N\mathbf{Q}^{-1})$$

This is a format you should be used to seeing by now. If we assumed within-unit independence and homoskedasticity we would get the classic OLS variance, how likely do you think that is?

We can estimate the asymptotic variance of θ using sample counterparts:

$$\begin{aligned}
\text{avar}\left(\hat{\theta}_p\right) &= \frac{1}{N}\mathbf{Q}^{-1}\Sigma_N\mathbf{Q}^{-1} \\
\widehat{\text{avar}}\left(\hat{\theta}_p\right) &= \frac{1}{N}\hat{\mathbf{Q}}^{-1}\widehat{\Sigma}_N\hat{\mathbf{Q}}^{-1} \\
\hat{\mathbf{Q}} &= \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x}_{it}\mathbf{x}_{it}' \\
\widehat{\Sigma}_N &= \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\hat{\varepsilon}_{it}\right]\left[\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_{it}\hat{\varepsilon}_{it}\right]' \\
\widehat{\text{avar}}\left(\hat{\theta}_p\right) &= \left[\sum_{i=1}^{N}\mathbf{x}_i'\mathbf{x}_i\right]^{-1}\left(\sum_{i=1}^{N}\mathbf{x}_i'\varepsilon_i\varepsilon_i'\mathbf{x}_i\right)\left[\sum_{i=1}^{N}\mathbf{x}_i'\mathbf{x}_i\right]^{-1}.
\end{aligned}$$

This variance matrix is called the *cluster-robust* or *clustered* variance matrix. The square root of the diagonal provides *clustered standard errors*. Note that the clustered variance matrix allows for *arbitrary* correlation among the errors within each unit. We haven't imposed any structure on them, not even stationarity. This is a powerful result that makes the clustered matrix very popular.

A warning, this sandwich is valid for large-N asymptotics. That's all we've done so far. An intutitive way to think about this is to note, that Σ_N is estimated by computing the variance within each unit and averaging over units As such, this estimator is only asymptotically valid **in** N. The standard rule of thumb is you have less than 50 units, the clustered matrix is probably not reliable and you may be better off with basic robust standard errors (if NT is large enough), or other alternatives based on large-T asymptotics that we'll get to later.

Some recent work has considered the issue with a small number of clusters. Some proposals here include:

1. A fixed number of clusters correction such as $\frac{N}{N-1}\widehat{\text{avar}}\left(\hat{\theta}_p\right)$.
2. Clustered bootstrap
3. Wild clustered bootstrap which one of your classmates will present on later

4. Jackknife

A clustered or block bootstrap. This *should* give you similiar results to the asymptotic standard errors, but they can diverge for any number of reasons.

To review, a bootstrap is a tool that relies on the *empirical distribution* to estimate the true distribution. Consider a sample $y_1, \ldots, y_N \overset{iid}{\sim} F(y)$ where F is an unknown CDF with some parameter of interest θ what is a function $\theta = T(F(y))$. We can consider the empirical distribution function (EDF) F_N as an approximation of F.

$$F_N(y) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(y_i \leq y),$$

which is a discrete distribution that puts probability $1/N$ on each observation. We want to be able to use this CDF to say something about true CDF and more importantly for us usually, some function of it T.

For example, if T is the expected value as in

$$T(F(y)) = \mathrm{E}[y] = \int y f(y) dy = \mu_y$$

and substituting the EDF gives us

$$T(F_N(y)) = \sum_{i=1}^{N} y_i f_N(y_i) = \frac{1}{N}\sum_{i=1}^{N} y_i = \hat{\mu}_y,$$

which better known as the sample mean. If we wanted to know the variance of this estimate without knowing anything else about the true distribution, we would ask, well do we think the EDF is a good approximation of the CDF? So long we say yes to that, then we can use the EDF as if it were the CDF and draw "new" samples from it.

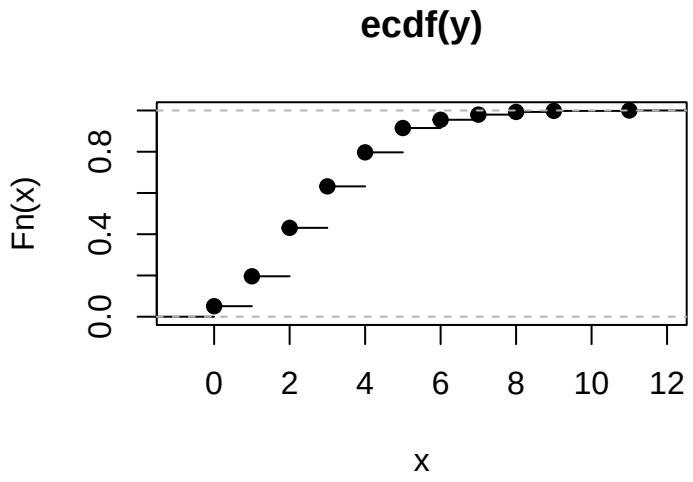
You may be familiar with how to sample from CDFs, typically we look for a solution based on inverse uniform sampling. This approach has us

1. Generate U which is a length-N vector of draws from the standard uniform
2. Generate $y^* = F^{-1}(U)$ which is our new sample.

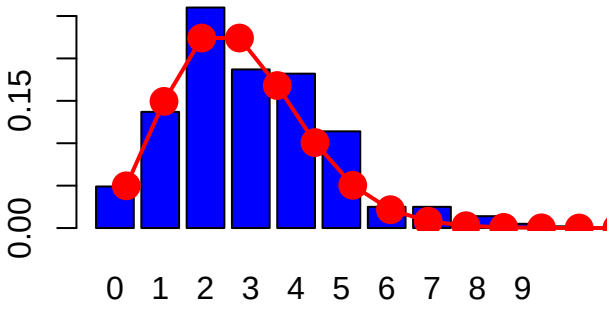
In the case of the EDF, we always have a step function and so inversion is just the quantile function. Which is nice.

```
set.seed(10)
y <- rpois(1000, lambda=3)
```

```
plot(ecdf(y))
```

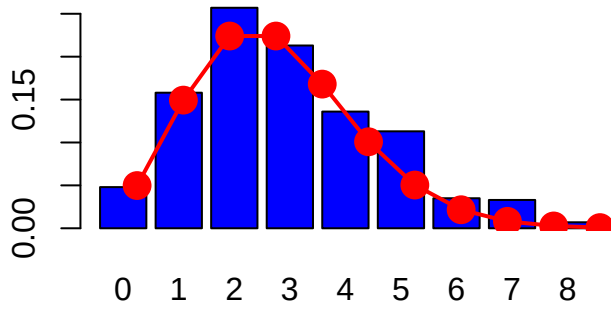



```
U <- runif(1000)
y2 <- quantile(y, probs=U, type=1)
barplot(table(y2)/1000, col="blue")
points(dpois(seq(0, 20,by=1), lambda=3), col="red", pch=16, cex=2)
lines(dpois(seq(0, 20,by=1), lambda=3), col="red", lwd=2)
```



This looks a little convoluted for what it actually is. Drawing samples from the EDF, is just a fancy way to say resample the data with replacement!

```
y3 <- sample(y, size=1000, replace=TRUE)
barplot(table(y3)/1000, col="blue")
points(dpois(seq(0, 20,by=1), lambda=3), col="red", pch=16, cex=2)
lines(dpois(seq(0, 20,by=1), lambda=3), col="red", lwd=2)
```



So that describes the ordinary bootstrap. When dealing with clustered data, however, we don't think that independence necessarily holds at the level of the individual observation. Instead, we sample at the level of the unit, reflecting Assumption A2. For bootstrap iteration $b = 1, 2, \ldots, B$:

1. Resample entire units in the data $(y_i, \mathbf{x}_i)$ with replacement to create a new data set
2. Fit the model using this new dataset
3. Save the estimates $\hat{\theta}^b$

We can then estimate the variance as

$$\frac{1}{B}\sum_{b=1}^{B}\left(\hat{\theta}^b - \hat{\bar{\theta}}\right)\left(\hat{\theta}^b - \hat{\bar{\theta}}\right)',$$

where $\hat{\bar{\theta}} = B^{-1}\sum_{b=1}^{B}\hat{\theta}^b$.

One issue with the standard clustered bootstrap comes into play when we're dealing with unbalanced panels. Now the sample size is changing with each bootstrap iteration, this can lead to numerical oddities. The more unbalanced, the more pronounced the issue. An alternative approach is called a Bayesian bootstrap, which comes from Rubin (1981). Here we think about a different sampling approach where we assign weights to each unit. In the traditional clustered bootstrap these weights are integers 0, 1, 2, ..., N, depending on how many times that unit appears in the data. These integer weights will sum to the number of units N but not the total sample size.

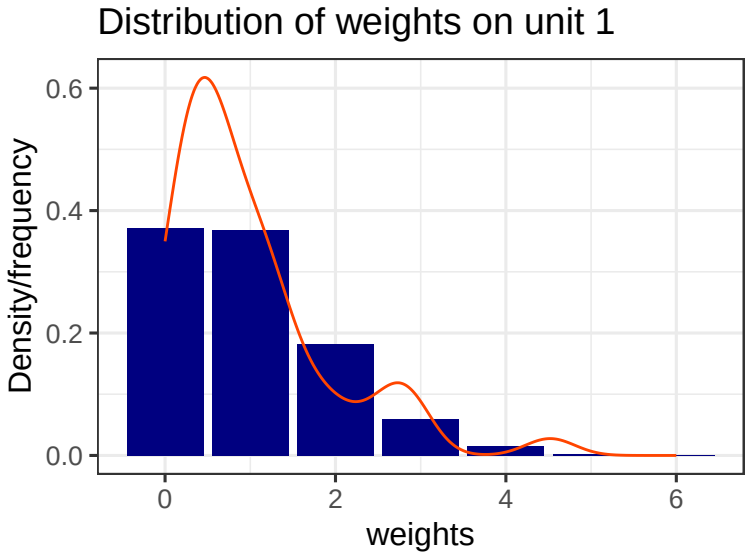
In the Bayesian version, we smooth the weights. The basic insight here is that the integer weights form a multinomial distribution, in Bayesian stats, the Dirichlet distribution is often paired with the multinomial as its continuous counterpart (glossing over details here). Visually we can think about this with the following

```
set.seed(1)
N <- 50
B <- 10000
```

```r
## draw a bunch of samples. Each row is a sample
multinomial <- matrix(sample(1:N, size=N*B, replace=TRUE), nrow=B)
m1 <- rowSums(multinomial==1)

gamma11 <-  matrix(rexp(N*B), nrow=B)
dircihlet <- gamma11/rowSums(gamma11)
d1 <- dircihlet[1,]*N

ggplot()+
  geom_bar(aes(x=m1, y=after_stat(prop)), fill="navyblue")+
  geom_density(aes(x=d1), color="orangered")+
  ggtitle("Distribution of weights on unit 1")+
  xlab("weights")+
  theme_bw(12)+
  ylab("Density/frequency")
```



Note the trick for generating Dirichlet weights:

1. N Gamma random variables with parameters α_i and β divided by their sum is distributed Dirichlet$(\alpha_1, \ldots, \alpha_N)$
2. Because we want a uniform probability of picking any unit, we'll use a Gamma(1,1), which is an Exponential(1)

So for iteration bootstrap b,

1. Draw N values from an Exponential(1) divide these by their sum to get the weights w.
2. Repeat each w_i T_i times where T_i is the number of observations within unit i to get weights for each observation.

The jackknife is similar to a bootstrap, but instead of resampling, we delete observations and see how much the variance changes. In a cross-sectional context, we typically delete one row at a time. In the panel context we will delete one unit at a time.

For $i = 1, \ldots, N$, refit the model without unit i. Call these estimates $\hat{\theta}_{-i}$. The jackknife variance estimator is then

$$\frac{N-1}{N} \sum_{i=1}^{N} \left(\hat{\theta}_{-i} - \hat{\bar{\theta}}\right)\left(\hat{\theta}_i - \hat{\bar{\theta}}\right)',$$

where $\hat{\bar{\theta}} = N^{-1} \sum_{i=1}^{N} \hat{\theta}_{-i}$.

So now we have some useful results for the panel model including is asymptotic variance matrix, along with several bootstrap procedures that may help us if we don't feel confident using asymptotic results. However, one of the main motivations for panel data is that it gives us repeated looks at individual units. So far we've only treated that as a nuisance that we correct for in the standard errors, but it actually provides us with some important ways to think about omitted variables and unobservables that are constant within units. This kind of unit-level heterogeneity has not yet appeared in our discussion as we have so far pooled all of our units together in estimation. The pooled model restricts us to only consider obervable differences across units (through z_i). To consider heterogeneity outside of the observables, we will need to expand our thinking.

1.2 Random effects

The random effects (RE) model also starts with Assumption A1

Assumption A1

$$y_{it} = \alpha_i + \beta' x_{it} + \gamma' z_i + \varepsilon_{it}$$

Where we diverge from the pooled model is that we will allow for heterogeneity across groups to enter in through α_i now such that we'll replace this model with

Assumption A1.R

$$y_{it} = \alpha + \beta' x_{it} + \gamma' z_i + \alpha_i + \varepsilon_{it}$$

Here the heterogeneity is modeled as an overall constant with random, unit-specific differences. This unit-level heterogeneity is time-invariant and contains any invariant factors that are

not included in z_i. In the RE world, each α_i is an iid draw from some distribution (thus the name), making it a stochastic component like the error term. Strict within-unit exogeneity (Assumption A3.R) in this context means

Assumptions A2.R, A3.R, A4.R, & A5.R

$$\begin{aligned}
\mathrm{E}[\varepsilon_{it}|\mathbf{x}_i] &= 0 \\
\mathrm{E}[\alpha_i|\mathbf{x}_i] &= 0 \\
\mathrm{E}[\alpha_i\varepsilon_i|\mathbf{x}_i] &= 0 \\
\mathrm{E}[\varepsilon_{it}\varepsilon_{jt'}|\mathbf{x}_i] &= 0 \\
\mathrm{E}[\alpha_i^2|\mathbf{x}_i] &= \sigma_\alpha^2 \\
\mathrm{E}[\varepsilon_{it}^2|\mathbf{x}_i] &= \sigma_\varepsilon^2
\end{aligned}$$

for all $i \neq j$ or $t \neq t'$.

Note that we also slid a few homoskedasticity assumptions (A4.R) and an iid assumption within units (A2.R). This means that we are in a world that is in some ways more restrictive than the pooled model. So we've generalized a bit through the unit-specific heterogeneity, but at the cost of some rather strong additional modeling assumptions. To see the restrictiveness of this model over the pooled model, consider that all the within-unit correlation here comes from the presence of α_i. As such it takes a very specific forms. In contrast, the basic model allowed for arbitrary within-unit correlation.

Additionally, note that in the pooled model we assumed that $\alpha_i = 0$, but what if instead we just assumed it was a set of time-invariant omitted variables? If the random effects assumptions are true then $\mathrm{E}[\alpha_i|\mathbf{x}_i] = 0$ and so the omitted variables are uncorrelated with the observables. This means that pooled estimator $\hat{\theta}_p$ is unbiased, consistent, and asymptotically normal for the the random effects model. However, it won't be the most efficient estimator for this model. So if we believe that RE assumptions, we will want to claw out some is efficiency improvements over the pooled estimator. This should trigger a memory in your brain. What tools do we have for cases like this where OLS is inefficient? (F)GLS and MLE.

Since we're thinking about efficiency, we'll want to be focus on the random component and what it looks like let $e_{it} = \alpha_i + \varepsilon_{it}$ be the combined stochastic component. Then under our above assumptions we can say that

$$\begin{aligned}
\mathrm{Var}(e_{it}|\mathbf{x}_i) &= \mathrm{Var}(\alpha_i|\mathbf{x}_i) + \mathrm{Var}(\varepsilon_{it}|\mathbf{x}_i) = \sigma_\alpha^2 + \sigma_\varepsilon^2 \\
\mathrm{Cov}(e_{it}, e_{it'}) &= \mathrm{Cov}(\alpha_i + \varepsilon_{it}, \alpha_i + \varepsilon_{it'}) = \sigma_\alpha^2
\end{aligned}$$

This means that the $T \times T$ covariance matrix Σ for unit i represented by $\mathrm{E}[\varepsilon_i \varepsilon_i' | \mathbf{x}_{it}]$ is dense with $\sigma_\alpha^2 + \sigma_\varepsilon^2$ on the diagonal and σ_α^2 everywhere else. The full $NT \times NT$ covariance matrix Ω is then block diagonal with Σ repeated N times diagonally.

There are several ways to fit random effects model. The first we'll consider is an FGLS approach as it is a little more general. Recall that

$$\hat{\theta}_{\text{FGLS}} = (\mathbf{X}' \widehat{\Omega}^{-1} \mathbf{X})^{-1} \mathbf{X}' \widehat{\Omega}^{-1} y,$$

we can make this a little bit easier on ourselves and our computers by exploiting symmetry.

Because Ω is block diagonal, we can work with just Σ. Already an improvement. We don't really need all of Σ^{-1} either, we really just need its square root in order to pre-treat the data for OLS. As you may recall, there are many different ways to think about the square root of a matrix (e.g., Choleskey). We'll focus on eigenvector decomposition for reasons that hopefully become clear.

Before we get into this, here's a few things you might want to remember about eigenvalues and eigenvectors.

1. A non-zero vector v_i is an **eigenvector** of a square matrix A if there exists a constant λ_i such that $Av_i = \lambda_i v_i$, where λ_i is called an **eigenvalue**.
2. For an $N \times N$ square matrix there will be N eigenvectors (each of length N) and N eigenvalues. These values may not be unique.
3. Because v_i is non-zero $\det(A - \lambda_i I) = 0$
4. $A = V \Lambda V^{-1}$ where V is a matrix where the ith column is v_i and Λ is a matrix with the corresponding eigenvalues on the diagonal.
5. A and A^{-1} have the same eigenvectors and the eigenvalues of A^{-1} are $1/\lambda$
6. $\det(A) = \prod_i \lambda_i$, and if A is triangular, then $\det(A)$ is also the product of its diagonal elements

So

$$\begin{aligned} \Sigma &= V \Lambda V^{-1} \\ \Sigma^{-1} &= V \Lambda^{-1} V^{-1} \\ \Sigma^{-1/2} &= V [\Lambda^{-1/2}] V^{-1}. \end{aligned}$$

where V is a matrix where each column is an eigenvector and Λ is a matrix with the eigenvalues of Σ on the diagonal. As such the diagonal of $\Lambda^{-1/2}$ is $1/\sqrt{\lambda}$ where λ are the eigenvalues of Σ.

We could compute these each time, but maybe there's a more general solution? Recall that

to find the eigenvalues of a matrix we need to solve

$$\det(\Sigma - \lambda I) = 0$$

for all possible values of λ. What do we know about this matrix?

$$\Sigma - \lambda I = I(\sigma_\alpha^2 + \sigma_\varepsilon^2 - \lambda) + \mathbf{1}\mathbf{1}'\sigma_\alpha^2.$$

With $T-2$ steps of Gaussian elimination you can get an upper diagonal matrix with diagonals:

$$(T\sigma_\alpha^2 + \sigma_\varepsilon^2 - \lambda, \sigma_\varepsilon^2 - \lambda, \ldots, \sigma_\varepsilon^2 - \lambda).$$

So the determinant of this matrix is the product of these diagonals and the eigenvalues are the values of λ that make this product 0. So what are the eigenvalues?

- 1 is $T\sigma_\alpha^2 + \sigma_\varepsilon^2$
- The other $T-1$ are σ_ε^2

Ok, remember the goal is to make an easy-to-use form of $\Sigma^{-1/2}$ that won't be dependent on sample size, so what's next? We have the eigenvalues for Σ, now we need them for Σ^{-1}. Thankfully that's as easy as

$$1/\lambda = \left(\frac{1}{T\sigma_\alpha^2 + \sigma_\varepsilon^2}, \frac{1}{\sigma_\varepsilon^2}, \ldots \frac{1}{\sigma_\varepsilon^2}\right).$$

This gives us the Λ matrix. Now we need eigenvectors, Working with Σ, let's start with the $T-1$ values that are σ_ε^2. Consider an arbitrary row t from Σ, we need it to solve (from property 1 above):

$$v_t\sigma_\alpha^2 + v_t\sigma_\varepsilon^2 + \sum_{s \neq t} v_s\sigma_\alpha^2 = \sigma_\varepsilon^2 v_t.$$

Note that σ_ε^2 appears only once on the RHS, so v_t for sure needs to be non-zero. This however means that $v_t\sigma_\alpha^2$ sticks around, so we need to cancel it out with something from the sum. The easiest way? For eigenvectors 2 through T let $v_t = 1$, $v_1 = -1$ and everything else be 0. This just leaves eigenvector 1 which solves

$$v_1\sigma_\alpha^2 + v_1\sigma_\varepsilon^2 + \sum_{s=2}^{T} v_s\sigma_\alpha^2 = v_1 T\sigma_\alpha^2 + v_1\sigma_\varepsilon^2.$$

The obvious? $v = 1$. Now we've got it so this gives us

$$V = \begin{bmatrix} 1 & -\mathbf{1}_{T-1} \\ \mathbf{1}_{T-1} & \mathbf{I}_{T-1} \end{bmatrix}.$$

So we've got all the pieces now for

$$\begin{aligned} \Sigma^{-1/2} &= V[\Lambda^{-1/2}]V^{-1} \\ &= \frac{1}{\sigma_\varepsilon}\left[I_T - \frac{\omega}{T}\mathbf{1}\mathbf{1}'\right] \\ \omega &= 1 - \frac{\sigma_\varepsilon}{\sqrt{T\sigma_\alpha^2 + \sigma_\varepsilon^2}}. \end{aligned}$$

And the FGLS estimates of the RE model become

$$\hat{\theta}_{\text{FGLS}} = (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}'\tilde{y},$$

where

$$\begin{aligned} \tilde{y}_i &= \Sigma^{-1/2}y_i = (y_i - \hat{\omega}\bar{y}_i)/\sigma_\varepsilon \\ \tilde{\mathbf{x}}_i &= \Sigma^{-1/2}\mathbf{x}_i = (\mathbf{x}_i - \hat{\omega}\bar{\mathbf{x}}_i)/\sigma_\varepsilon. \end{aligned}$$

Fortunately this contains only two parameters σ_α^2 and σ_ε^2, both of which can be estimated using the pooled residuals and the RE assumptions.

Here are the steps:

1. Fit the model using pooled OLS (consistent), call the residuals in this case $\hat{e}$ where

   $$\hat{e}_{it} = y_{it} - \hat{\theta}_p'\mathbf{x}_{it}.$$

   Note that the pooled residuals are estimates of e_{it}, and so $\hat{e}'\hat{e}/(NT)$ is a consistent estimator $\text{Var}(e_{it}|\mathbf{x}_i) = \text{Var}(u_i|\mathbf{x}_i) + \text{Var}(\varepsilon_{it}|\mathbf{x}_i) = \sigma_\alpha^2 + \sigma_\varepsilon^2$.

2. Likewise, the pooled residuals can also be used to estimate σ_α^2, how? It's the within-covariance of the residuals

   $$\hat{\sigma}_\alpha^2 = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T(T-1)/2}\sum_{t=2}^{T}\sum_{t'=1}^{t-1}\hat{e}_{it}\hat{e}_{it'},$$

   OR, you can use the within estimator residuals (below) to estimate σ_ε^2 (easier).

3. Use the relationship
$$\sigma_e^2 = \sigma_\alpha^2 + \sigma_\varepsilon^2$$
to back out the remaining quantity of interest.

4. Build $\hat{\omega}$ as described above and fit using OLS on the transformed data.

Property A4 *Under Assumptions A1.R–A5.R, as N increases, the random effects estimator will exist, be consistent for θ, and asymptotically normal. It is also unbiased and (weakly) more efficient than pooled OLS in finite samples.*

This property follows from ordinary (F)GLS results. The RE estimator will also consistent for θ in T. It will be inconsistent for σ_α^2 in T and this will affect anything we can say about efficiency in the big-T fixed N setting.

Note that for unbalanced panels, you'll need to

1. Be a little more careful estimating σ_α^2 and
2. Estimate ω_i separately for each unit

Both of these changes reflect the varying length T_i within each unit.

This version of the RE model is a semi-parametric way to consider heterogeneity across units. Basically, we assume that conditional on the observables all the remaining heterogeneity is mean-zero noise that is uncorrelated with the observables. We haven't put a distributional assumption on that noise yet except to say that it each u_i is an exogenous iid shock from an unknown distribution with several moments.

Identification in this case comes from correctly specifying the rest of the model. In this way, it is very similar to a standard cross-sectional linear models. The main identification comes from having no omitted variables in either z_i or x_{it} that are correlated with both the treatment of interest and the outcome of interest.

As such the assumptions needed for the RE model to identify an effect of interest are the same as the pooled model. What we've done is say, "look we recognize that there could be heterogeneity across units. We're going to model that heterogeneity in way that pooled OLS is consistent but inefficient. We can gain that efficiency back using the FGLS approach." Is that worth much? It's not nothing, but it very much relies on thinking the RE assumptions are reasonable.

It's worth asking the question at this point, what do to with the standard errors? Under the

RE assumptions we've made so far, the standard GLS variance matrix,

$$\text{Var}(\hat{\theta}_{RE}|\mathbf{X}) = \sigma^2_{FGLS}(\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1},$$

is correct, but recall that these assumptions are fairly strong (i.e., iid observations once we condition on u_i and two levels of homoskedasticity). Note that using the transformation above $\sigma^2_{FGLS} = 1$, some softwares do different transformations, so be careful. Often we are not so convinced of all our RE assumptions, but if we think they're mostly reasonable we may want to consider a clustered covariance matrix with the FGLS estimates. This is perhaps controversial as we make the efficiency claims largely on the basis of these assumptions, but then say we're not so sure about them if we cluster. The extent to which the clustered standard errors differ from the GLS standard errors may tell us something about how believable the RE assumptions are (e.g., King and Roberts).

Now two more points before we move on. First, this is not the only way to fit this model. Perhaps more common is a maximium likelihood approach that requires additional parametric assumptions

Assumption A6.R The stochastic components are normally distributed $u_i \sim N(0, \sigma^2_\alpha)$, $\varepsilon_{it} \sim N(0, \sigma^2_\varepsilon)$.

this parametric addition means we don't have to do a multi-step approach. Under this assumption

$$y_i|\mathbf{x}_i \sim N\left(\left[\alpha + \beta' x_{it} + \gamma' z_i\right]_{t=1}^{T_i}, \Sigma_i\right),$$

which suggests a straightforward log-likelihood function where each unit is a draw from this multivariate normal, giving us

$$L(\theta|y) = \sum_{i=1}^{N} -\frac{1}{2}\log(\det(\Sigma_i)) - \frac{1}{2}(y_i - \theta'\mathbf{x}_i)'\Sigma_i^{-1}(y_i - \theta'\mathbf{x}_i),$$

which we can simplify a bit further using what we know about the eigenvector decomposition of Σ_i, in that

$$\begin{aligned}
\det(\Sigma_i) &= (T_i\sigma^2_\alpha + \sigma^2_\varepsilon)(\sigma^2_\varepsilon)^{T_i - 1} \\
\frac{1}{2}\log(\det(\Sigma_i)) &= \frac{1}{2}\log(T_i\sigma^2_\alpha + \sigma^2_\varepsilon) + \frac{T_i - 1}{2}\log(\sigma^2_\varepsilon) \\
\frac{1}{2}(y_i - \theta'\mathbf{x}_i)'\Sigma_i^{-1}(y_i - \theta'\mathbf{x}_i) &= \frac{1}{2}[(e_i - \omega_i\bar{e}_i)/\sigma_\varepsilon]'[(e_i - \omega_i\bar{e}_i)/\sigma_\varepsilon]
\end{aligned}$$

Second, this approach to modeling heterogeneity is unlikely to satisfy many people because

the exogeneity assumption is quite strong. Likewise, the specific assumptions required for the RE estimator to be more efficient than the pooled estimator requires both within-unit independence and homoskedasticity at both the observation and unit levels. As such, we will set this framework aside for a moment and consider another approach to modelling unobserved heterogeneity.

1.3 The fixed effects model

All right, so how might we think about unobserved heterogeneity? Again, we start from **Assumption A1**

$$y_{it} = \alpha_i + \beta' x_{it} + \gamma' z_i + \varepsilon_{it}.$$

This time we change it to be **Assumption A1.F**

$$y_{it} = \alpha_i + \beta' x_{it} + \varepsilon_{it}$$

where

$$\alpha_i = \alpha + \gamma' z_i + u_i$$

Unlike the RE model, u_i are fixed parameters not draws from a random variable (thus the names) and contain everything unobserved about unit i that is time-invariant. By estimating this fixed, overall constant for each unit, we tuck $\gamma' z_i$ into α_i along with everything that is time-invariant. ote that this constant controls for **all** time-invariant heterogeneity, even things we didn't think of or can't measure. So we lose identification of γ, but we gain insulation from a range of omitted variables. In this way, the fixed effects model is a very important tool for fighting endogeneity as it eliminates any concerns about omitted variable bias from time-invariant sources. We will also return to iid units rather than observations (Assumption A2) and maintain strict exogeneity within units (Assumption A3).

This leaves us with a very similar setup to the pooled model. However, we haven't said anything about the correlation between u_i and the exogeneity of x_{it}. In the pooled and RE settings we assumed that any within-unit deviations from the overall constant α could be safely ignored by either a) knowing/including it z_i, b) leaving it outside the model as either either part of ε_{it} (pooled) or the random u_i (RE). Now however, we're going to ask, when is that assumption reasonable?

Suppose we fit the model in Assumption A1.F using either a pooled or RE estimator. In this case we include any observed z_i, but are leaving the unobserved u_i in the error term, as we've

done before. This leaves us with a joint error term $e_{it} = \varepsilon_{it} + u_i$, such that

$$\mathrm{E}[\hat{\theta}_p|\mathbf{x}_{it}] = \theta + \left(\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x}_{it}\mathbf{x}_{it}'\right)^{-1}\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x}_{it}\,\mathrm{E}[u_i|x_{it}],$$

which is our familiar omitted variable bias result. If there is any correlation between the variables in $\mathbf{x}_{it}$ and the unit-specific heterogeneity u_i, then the pooled estimator (and by extension the random effects estimator) are biased and inconsistent because of this endogeneity.

This poses a notable issue. What can we do to consider a non-parametric form of heterogeneity? Obviously, if we feel ok assuming that it is unrelated to either the treatment or outcome of interest then we're fine to return to the pooled estimator. However, in cases where that's unlikely to be true, we still have some options.

1.3.1 Within-estimator

The first approach we'll consider involves what's known as the "within transformation," using

$$M_i := \mathbf{I}_i - \mathbf{1}_i(\mathbf{1}_i'\mathbf{1}_i)^{-1}\mathbf{1}_i'.$$

Here M_i is the "demeaning" matrix. It's not insulting, but it does subtract the unit-specific mean of any matrices it meets such that

$$\begin{aligned} M_i y_i &= y_i - \bar{y}_i \\ M_i X_i &= \begin{bmatrix} X_{i1} - \bar{X}_{i1} & \dots & X_{iK} - \bar{X}_{iK} \end{bmatrix}. \end{aligned}$$

Let's consider this demeaning approach, such that

$$\begin{aligned} y_{it} - \bar{y}_i &= \beta'(x_{it} - \bar{x}_i) + (\alpha_i - \bar{\alpha}_i) + (\varepsilon_{it} - \bar{\varepsilon}_i) \\ y_{it} - \bar{y}_i &= \beta'(x_{it} - \bar{x}_i) + (\varepsilon_{it} - \bar{\varepsilon}_i) \\ M_i y_i &= M_i X_i \beta + M_i \varepsilon_i \end{aligned}$$

We can fit this model using OLS. Doing so is called the **within estimator**.

Note that because we are using OLS to fit the within model. We inherit all the good properties of OLS is so the within estimator is unbiased and consistent, if not efficient under Assumptions A1.F, A2, and A3. With appropriate rank conditions, we can also say that the correct variance matrix is the clustered variance matrix using the within transformed data.

Of additional note, supposed we have homoskedasticity and within-unit independence. From

what we know about OLS, this gives us

$$\text{Var}(\hat{\beta}_w^0|X_i) = \sigma_\varepsilon^2 \left(\sum_{i=1}^{N} X_i' M_i X_i \right)^{-1},$$

which as you'll show in a problem set is weakly greater than the variance of the pooled OLS estimator

$$\text{Var}(\hat{\beta}_p^0|X_i) = \sigma_\varepsilon^2 \left(\sum_{i=1}^{N} X_i' X_i \right)^{-1}.$$

The intuition behind this is that the demeaning process removes some information from each x variable (specifically the cross-section information).

What does this mean for us? Two things. First, it means that we have made a firm choice regarding what information matters. We are only interested in the within-unit variation. This is reflected in the fact that most software packages will report two (or three) different R^2 values for within estimation

$$\begin{aligned} R^2 &= 1 - \frac{SSR}{SST} = 1 - \frac{\hat{\varepsilon}'\hat{\varepsilon}}{(y - \bar{y})'(y - \bar{y})} \\ R^2_{\text{adj}} &= 1 - (1 - R^2)\frac{NT - 1}{NT - N - k} \\ R^2_{\text{within}} &= 1 - \frac{\hat{\varepsilon}'\hat{\varepsilon}}{\sum (y_i - \bar{y}_i)'(y_i - \bar{y}_i)} \end{aligned}$$

The differences between the overall R^2 and the within R^2 are that the former tells us how much of the total (cross-sectional and within) variance in y is explained by X plus the unit-specific heterogeneity. The latter tells us just how much of the within-unit changes in y variance is explained by X. The overall R^2 tends to be a lot higher as unit-specific heterogeneity tends to explain a lot.

Second, in this context it tells us that when we have panel data with unobserved heterogeneity and homoskedastic and independent errors, there is a bias-variance trade off. Ignoring the heterogeneity by estimating the pooling model will result in lower variance but more bias. Being robust to the heterogeneity by fitting the fixed-effects model using the within transformation will decrease bias but at the cost of variance. Generally, we're more concerned with bias in estimating treatment effects, but it's worth remembering that we don't get it for free.

1.3.2 Dummy variable estimator

Another way may be to just estimate the time invariant parameters directly for each unit. Let

$$\alpha_i = \alpha + \gamma' z_i + u_i,$$

then the model becomes

$$y_{it} = \beta' x_{it} + \alpha_i + \varepsilon_{it},$$

which can be fit using OLS with a dummy variable for each unit.

Let $\theta_{LSDV} = (\beta, \alpha_i)_{i=1}^N$ and redefine $\mathbf{X} = [X\ D]$ where D (no subscript) is a $NT \times N$ matrix of dummy variables where each column denotes if the observation is associated with unit i. Notice that we no longer have an overall constant, instead we have a constant for each unit.

This approach is identical to the within transformation such that $\hat{\beta}_w = \hat{\beta}_{LSDV}$. The within estimator saves us from estimating the N unit-specific parameters, which can be quite handy for larger N, but it does not direct estimate the constants. However, this drawback rarely matters to us in practice, at least for the linear model.

We will consider this equivalence in two steps. First, consider a model with no covariates:

$$y_{it} = \alpha_i + \varepsilon_{it}.$$

What would the least squares estimate be for α_i? The sample mean for group i (i.e, $\hat{\alpha}_i = \bar{y}_i$). This means that the residuals of $\hat{\varepsilon}_{it} = y_{it} - \bar{y}_i$, which of course is the residual vector from the within transformation.

Second, consider the model from Assumption A1.F

$$y_{it} = \alpha_i + \beta' x_{it} + \varepsilon_{it}.$$

In matrix form we can write this as

$$y = X\beta + D\alpha_i + \varepsilon,$$

Note that we can use the Firsch-Waugh-Lovell (FWL) theorem to consider the LSDV estimator of β separately from the LSDV estimator of α. First, let us remind ourselves what the FWL says,

Theorem 6 Firsch-Waugh-Lovell (FWL) *For a regression model of $y = X_1\beta_1 + X_2\beta_2 + \varepsilon$, with $\beta = (\beta_1, \beta_2)$, the OLS estimator of β_2 can be computed using the following algorithm:*

1. *Regress y on X_1 and save the residuals $\hat{e}_1$*
2. *Regress X_2 on X_1 and save the residuals $\hat{e}_2$*
3. *Regress $\hat{e}_1$ on $\hat{e}_2$ to get $\hat{\beta}_2$ and $\hat{\varepsilon}$*

In our case this means that we can compute $\hat{\beta}_{LSDV}$ in the following way

1. Regress y on D and save the residuals $\hat{e}_1$
2. Regress X on D and save the residuals $\hat{e}_2$
3. Regress $\hat{e}_1$ on $\hat{e}_2$ to get $\hat{\beta}_{LSDV}$ and $\hat{\varepsilon}$

Regressing anything on just D, as we showed above, **is** the within transformation. So steps 1 and 2 here are just conducting the within-transformation and step 3 is the within estimator.

The big deal here, is that the with the within/LSDV estimator, the estimated values of β are completely invariant to the values of the fixed effects α_i. This means that we do not need to put additional assumptions on α_i like we did with the pooled or random effects estimators. The composition of α_i can be correlated with the error terms and it doesn't matter.

As an additional note, when using degree of freedom corrections (or otherwise accounting for degrees of freedom), the correct number includes the N α_i terms even when using the within transformation. Why? Answer: The within transformation involves estimating $N(k+1)$ sample means, but these sample means are themselves directly related to the the N unit-specific intercepts such that

$$\hat{\alpha}_i = \bar{y}_i - \hat{\beta}'_w \bar{x}_i$$

, as such we are still using $N + k$ degrees of freedom even when we don't actually directly estimate the intercepts.

The LSDV/within approach also has a connection to the RE approach. Remember that we used the following weights for the RE estimator

$$\omega_i = 1 - \frac{\sigma_\varepsilon}{\sqrt{T_i \sigma_\alpha^2 + \sigma_\varepsilon^2}}.$$

and the RE-FGLS estimator was OLS applied to the transformed data $(y_i - \hat{\omega}\bar{y})$ and likewise for $\mathbf{x}$. A couple things to note:

1. If $\hat{\sigma}_\alpha^2 = 0$ then $\hat{\omega} = 0$ and $\hat{\theta}_{RE} = \hat{\theta}_p$

2. If $\hat{\sigma}_\varepsilon^2 >> T_i\hat{\sigma}_\alpha^2$, then $\hat{\omega} \approx 0$ and $\hat{\theta}_{RE} \approx \hat{\theta}_p$
3. If $T_i\hat{\sigma}_\alpha^2 >> \hat{\sigma}_\varepsilon^2$, then $\hat{\omega} \approx 1$ and $\hat{\beta}_{RE} \approx \hat{\beta}_w$.

Now note that β in this model reflects only the average within unit changes (thus the name). Basically, by applying the within transformation, we discard the cross-section information and focus on how the treatment affects each individual. At the other end is the pooled model, which uses both the within and the between unit information. In fact, we can consider what the other extreme might be: the **between** estimator

$$\hat{\theta}_{\text{btwn}} = \left(\sum_{i=1}^{N} \bar{\mathbf{x}}_i \bar{\mathbf{x}}_i'\right)^{-1} \sum_{i=1}^{N} \bar{\mathbf{x}} \bar{y}_i.$$

There is little practical use for the between estimator, but it does show us the extreme case where all we care about is the cross-sectional variation and where we care nothing about the within-unit variance. However, we can note the relationships between the pooled, between, and within estimators such that

$$\begin{aligned}
\hat{\theta}_p &= T_{XX}^{-1} T_{Xy} \\
\hat{\theta}_w &= W_{XX}^{-1} W_{Xy} \\
\hat{\theta}_b &= B_{XX}^{-1} B_{Xy} \\
T_{XX} &= \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i' \\
B_{XX} &= \sum_{i=1}^{N} \bar{\mathbf{x}}_i \bar{\mathbf{x}}_i' \\
W_{XX} &= T_{XX} - B_{XX} \\
\hat{\theta}_{RE} &= (W_{XX} + \lambda B_{XX})^{-1}(W_{Xy} + \lambda B_{Xy}) \\
\lambda &= (1-\omega)^2.
\end{aligned}$$

All of this to say that the random effects estimator can also be expressed as a weighted combination of the between and within estimators. When they're weighted equally ($\lambda = 1$) then we have the pooled estimator, as λ moves to 0 (favoring the within variance), we get the within estimator.

1.3.3 First differences

Yet another way to consider the FE model is to remove the heterogeneity by subtracting y_{t-1} from both sides

$$\begin{aligned} y_{it} - y_{t-1} &= (\alpha_i - \alpha_i) + \beta'(x_{it} - x_{it-1}) + \varepsilon_{it} - \varepsilon_{it-1} \\ \Delta y_{it} &= \beta' \Delta x_{it} + \Delta \varepsilon_{it}. \end{aligned}$$

All the time-invariant heterogeneity is removed and OLS becomes a good estimator for β.

In matrix form this looks like

$$\begin{aligned} \Delta_i y_i &= \Delta_i X_i + \Delta_i \varepsilon_i \\ \Delta_i &= \underbrace{\begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & -1 & 1 \end{bmatrix}}_{T_i - 1 \times T_i} \\ \hat{\beta}_{FD} &= \left(\sum_{i=1}^{N} X_i' \Delta_i' \Delta_i X_i \right)^{-1} \left(\sum_{i=1}^{N} X_i' \Delta_i' \Delta_i y_i \right) \end{aligned}$$

Because this is simply pooled OLS on the differenced data, we can obtain a few properties with ease:

Property A5 *Under Assumptions A1.F, A2, and A3 the first differences estimator $\hat{\beta}_{FD}$ is unbiased for β.*

With additional rank and moment assumptions, we can also obtain

Property A6 *$\hat{\beta}_{FD}$ is consistent in N and asymptotically normal with the clustered variance matrix on the differenced data.*

Note that when $T = 2$ the FD estimator will be identical to the within estimator, but this does not hold for $T > 2$. Additionally, if we wanted to make some independence and homoskedasticity assumptions on the undifferenced errors ε_{it} then we can note that the differenced errors are correlated within units

$$\text{Var}(\Delta_i \varepsilon_i | X_i) = \Delta_i \Delta_i' \sigma_\varepsilon^2,$$

where $\Delta_i \Delta_i'$ is a matrix with 2 on the diagonal, -1 on the first off-diagonals, and 0 everywhere else.

This means that we can eek out some improvements via GLS since we know the variance, such that

$$\hat{\beta}_{FD}^{GLS} = \left(\sum_{i=1}^{N} X_i'\Delta_i'(\Delta_i\Delta_i')^{-1}\Delta_i X_i\right)^{-1}\left(\sum_{i=1}^{N} X_i'\Delta_i'(\Delta_i\Delta_i')^{-1}\Delta_i y_i\right)$$

After some algebra, the inner parts work out to be

$$\Delta_i'(\Delta_i\Delta_i')^{-1}\Delta_i = \mathbf{I}_i - \mathbf{1}_i(\mathbf{1}_i'\mathbf{1}_i)^{-1}\mathbf{1}_i' := M_i.$$

This is the demeaning matrix again. So we can rewrite the GLS-FD estimator as

$$\hat{\beta}_{FD}^{GLS} = \left(\sum_{i=1}^{N} X_i' M_i X_i\right)^{-1}\left(\sum_{i=1}^{N} X_i' M_i y_i\right).$$

This is the within estimator! So if the errors are iid and homoskedastic then the within estimator is BLUE by the GLS properties. However, that's probably not going to be something we want to lean on very often, but it's 1 point in favor of within over FD. Of course, if $\Delta\varepsilon_{it}$ are iid and homoskedastic then FD is BLUE. More generally, the LSDV/within estimators will only be identical to the FD estimates when $T = 2$.

1.4 Model testing and comparisons

The next thing you should want to know is when do you want to use the pooled, or random effects FGLS, or the LSDV/within. As we've mentioned, the fixed effects estimators will be consistent in the widest set of cases, however, this can come at some efficiency losses. Likewise, in some cases, we now know that the decision may not matter too much (i.e., as T increases the differences between fixed and random effects will be less pronounced, all else equal). Table 1.1 outlines some of the important differences among the models and estimators we've discussed so far.

Okay, so now you're thinking I don't want inconsistent estimates, but efficiency is nice. How do I choose among these estimator?

As we mentioned, even if the random effects assumptions are good, the RE-FGLS estimator converges to the within-estimator (below) as T increases. So the efficiency gains are fleeting as T increases while the risk of bias and inconsistency remain. Recall that we can consider the closeness between the two estimators by just looking at the RE weights ω_i.

$$\omega_i = 1 - \frac{\sigma_\varepsilon}{\sqrt{\sigma_\varepsilon^2 + T_i\sigma_\alpha^2}}.$$

The closer ω_i is to 1 the more similar the estimates are, the fewer efficiency gains if the RE assumptions are correct. Likewise, if the RE assumptions are not satisfied the estimator is inconsistent.

However, in most cases there will be differences. In these cases, we have should have good reasons for choosing the estimators we do. Most of the time, we care about consistency more than efficiency. This is a good reason to make the within/LSDV your first choice (or the "mostly harmless" choice).

If you're not yet convinced that random effects are mostly meh, or you really think there's a good reason for that approach, you can consider two different types of hypothesis tests. The first is the common textbook text for this question: The Hausman test. The Hausman test should be considered for when you think random effects are right, and you want to provide evidence in support of that decision. It should *not* be used to make a selection when you're agnostic about fixed versus random effects. If you're agnostic, use the fixed effects because they require fewer assumptions.

The null hypothesis is that $\hat{\beta}_0$ and $\hat{\beta}_1$ are both consistent. The alternative hypothesis is that only $\hat{\beta}_1$ is consistent.. To put this another way, the null is that $q = \hat{\beta}_1 - \hat{\beta}_0 \xrightarrow{p} 0$. Note this is a slightly different kind of hypothesis than we're used to, because it relates to the limiting value of an estimate rather than whether the true parameters are a particularly value.

Hausman derives this hypothesis test for the case where $\hat{\beta}_0$ is the asymptotically efficient estimator of β (i.e., RE v FE). In this case we can consider the (joint) distribution of the estimators. We know that the sampling distributions are individually normal (asymptotically), and we will add the additional assumption that they are jointly normal.

So now we need to know the distribution of q, we start with the joint distribution:

$$\sqrt{N}\begin{bmatrix}\hat{\beta}_1 - \beta \\ \hat{\beta}_0 - \beta\end{bmatrix} \overset{asy}{\sim} N\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}\sigma_\varepsilon^2 \operatorname{E}[X'MX] & N\operatorname{Cov}(\hat{\beta}_1, \hat{\beta}_0) \\ N\operatorname{Cov}(\hat{\beta}_1, \hat{\beta}_0) & \operatorname{E}[X'\Omega^{-1}X]\end{bmatrix}\right)$$

and this means that $q = \hat{\beta}_1 - \hat{\beta}_0$ is also asymptotically normal with mean 0 and variance

$$\operatorname{Var}(q) = \operatorname{Var}(\hat{\beta}_1) + \operatorname{Var}(\hat{\beta}_0) - 2\operatorname{Cov}(\hat{\beta}_1, \hat{\beta}_0),$$

under the null. Now we don't typically know the covariance across estimators so $\operatorname{Cov}(\hat{\beta}_1, \hat{\beta}_0)$ is unclear (although we could—and maybe should—bootstrap it). We need to be clever. Let's

Table 1.1: Comparing panel estimators

	Pooled	RE-FGLS	FD	LSDV/within
Pooled model: $y_{it} = \alpha + \beta' x_{it} + \gamma' z_i + \varepsilon_{it}$ $(\mathbf{x}_i, \varepsilon_i)$ iid $\mathrm{E}[\varepsilon_{it}\|\mathbf{x}_i] = 0$	• Unbiased and consistent in N. In T, if data are stationary and ergodic. • BLUE if ε_{it} are iid homoskedastic. • Asymptotically efficient if ε_{it} are iid normal and homoskedastic.	• Unbiased and consistent in N for θ. In T, if data are stationary and ergodic. • No efficiency gains over the pooled estimator. • RE covariance matrix may be incorrect b/c of within-unit iid assumptions	• Unbiased and consistent in N for β. • In T if data are stationary and ergodic	• Unbiased and consistent in N for β. • In T if data are stationary and ergodic
RE model: $y_{it} = \alpha + \beta' x_{it} + \gamma' z_i$ $+\alpha_i + \varepsilon_{it}$ $\mathrm{E}[\alpha_i] = 0$ $\mathrm{Cov}(\mathbf{x}_{it}, \alpha_i) = 0$ $\mathrm{Cov}(\varepsilon_{it}, \alpha_i) = 0.$ $(\mathbf{x}_{it}, \varepsilon_{it})$ iid $\mathrm{E}[\varepsilon_{it}\|\mathbf{x}_i] = 0$ $\mathrm{E}[\varepsilon_{it}^2\|\mathbf{x}_i] = \sigma_\varepsilon^2$ $\mathrm{E}[\alpha_i^2\|\mathbf{x}_i] = \sigma_\alpha^2$	Unbiased and consistent N . In T if data are stationary and ergodic.	• Unbiased and consistent in N. In T if data are stationary and ergodic. • BLUE • Asymptotically efficient if α_i and ε_{it} are normal	See above	See above
FE model: $y_{it} = \beta' x_{it} + \alpha_i + u_{it}$ (x_i, ε_i) iid $\mathrm{E}[\varepsilon_{it}\|\mathbf{x}_i] = 0$	Biased and inconsistent	Biased and inconsistent in N. Consistent in T as $\omega \to 1$, if data are stationary and ergodic.	See above. BLUE if $\Delta\varepsilon_{it}$ are iid and homoskedastic	See above. BLUE if ε_{it} iid and homoskedastic

rewrite q such that we get

$$\begin{aligned}\hat{\beta}_1 &= \hat{\beta}_0 + q \\ \operatorname{Var}(\hat{\beta}_1) &= \operatorname{Var}(\hat{\beta}_0) + \operatorname{Var}(q) + 2\operatorname{Cov}(\hat{\beta}_0, q).\end{aligned}$$

Claim: $\operatorname{Cov}(\hat{\beta}_0, q) = 0$

Proof. Suppose not, that is, let $\operatorname{Cov}(\hat{\beta}_0, q) \neq 0$. We can define another estimator $\hat{\beta}_2 = \hat{\beta}_0 + rAq$, where A is an arbitrary matrix and r an arbitrary scalar. Because $q \xrightarrow{p} 0$, we know that $\hat{\beta}_2$ is consistent and asymptotically normal with variance

$$\operatorname{Var}(\hat{\beta}_2) = \operatorname{Var}(\hat{\beta}_0) + rA\operatorname{Cov}(\hat{\beta}_0, q) + r\operatorname{Cov}(\hat{\beta}_0, q)A' + r^2 A\operatorname{Var}(q)A'.$$

Let

$$f(r) = \operatorname{Var}(\hat{\beta}_2) - \operatorname{Var}(\hat{\beta}_0) = 2rA\operatorname{Cov}(\hat{\beta}_0, q)A' + r^2 A\operatorname{Var}(q)A' \geq 0$$

be the difference in variances between this new estimator and the efficient estimator $\hat{\beta}_0$. The derivative of f derivative wrt to r is

$$D_r f(r) = A\operatorname{Cov}(\hat{\beta}_0, q) + \operatorname{Cov}(\hat{\beta}_0, q)A' + 2rA\operatorname{Var}(q)A'.$$

Now consider the special case where $r = 0$ and $A = -\operatorname{Cov}(\hat{\beta}_0, q)$

$$D_r f(0) = -2\operatorname{Cov}(\hat{\beta}_0, q)'\operatorname{Cov}(\hat{\beta}_0, q).$$

If $\operatorname{Cov}(\hat{\beta}_0, q) \neq 0$, then this is a quadratic times a negative constant. As such, $D_r f(0) < 0$, which is to say that $f(r)$ is decreasing in r at $r = 0$.

But, note that $f(0) = 0$, so for some small $r > 0$, $f(r)$ will be negative. However, this contradicts the the fact that $\hat{\beta}_0$ is the efficient estimator. Therefore, we conclude that $\operatorname{Cov}(\hat{\beta}_0, q) = 0$. □

This tells us two identical things:

1. The variance of q

$$\begin{aligned}\operatorname{Var}(\hat{\beta}_1) &= \operatorname{Var}(\hat{\beta}_0) + \operatorname{Var}(q) + 0 \\ \operatorname{Var}(q) &= \operatorname{Var}(\hat{\beta}_1) - \operatorname{Var}(\hat{\beta}_0),\end{aligned}$$

2. The actual covariance of $\hat{\beta}_1$ and $\hat{\beta}_0$

$$\begin{aligned} \text{Cov}(\hat{\beta}_1 - \hat{\beta}_0, \hat{\beta}_0) &= 0 \\ \text{Cov}(\hat{\beta}_1, \hat{\beta}_0) - \text{Cov}(\hat{\beta}_0, \hat{\beta}_0) &= 0 \quad \text{Properties of covariance} \\ \text{Cov}(\hat{\beta}_1, \hat{\beta}_0) &= \text{Var}(\hat{\beta}_0) \end{aligned}$$

Returning to the test statistic, we can now construct a standard χ^2 test based on q such that

$$q' \left(\text{Var}(\hat{\beta}_1) - \text{Var}(\hat{\beta}_0) \right)^{-1} q \overset{asy}{\sim} \chi^2_k,$$

where k is the length of β.

Note, that the RE estimator is only more efficient under the RE assumptions, which include homoskedasticity and iid observations. If either of these fails, this test is suspect. As such, we can only use the "classical" variance matrices

$$\begin{aligned} V_1 &= \hat{\sigma}^2_\varepsilon \left[\sum_i X' M_i X \right]^{-1} \\ V_0 &= \mathbf{X}' \widehat{\Omega}^{-1} \mathbf{X} \qquad \text{restricted to } \beta. \end{aligned}$$

Because of this test's reliance on within-unit independence and homoskedasticity, we may want to consider alternatives. One that you'll think about in a problem set or something will consider the power and size of a version based on a clustered bootstrap.

Now because we want to know if the iid assumptions have any bite, we'll want to know if there is any autocorrelation in the residuals. Wooldridge recommends a panel version of the standard Bruesch-Godfrey test that simply regresses $\hat{\varepsilon}_{it}$ on $\hat{\varepsilon}_{it-1}$.

1.4.1 CRE

There is another approach though which can accommodate more interesting covariance structures without concern. This is based on work by Mundlak, who gifted us *another* estimator for the linear fixed effects model that is also equivalent to the LSDV/within called

the **correlated random effects** estimator. Here we adjust the fixed effects model such that

$$\begin{aligned}
y_{it} &= \alpha_i + \beta' x_{it} + \varepsilon_{it} \\
\alpha_i &= \alpha + \gamma' \bar{x}_i + u_i \\
u_i &\sim f(0, \sigma_\alpha^2) \\
\mathbf{x}_{it} &= \begin{bmatrix} 1 & x_{it} & \bar{x}_i \end{bmatrix} \\
\mathrm{E}[u_i | \mathbf{x}_i] &= \mathrm{E}[u_i \varepsilon_i | \mathbf{x}_i] = 0 \\
\mathrm{Cov}(u_i + \varepsilon_{it}, u_i + \varepsilon_{it'}) &= \sigma_\varepsilon^2 I_T + \sigma_\alpha^2 \mathbf{1}\mathbf{1}'
\end{aligned}$$

What's happening here? Well we're blending the RE and FE models a bit. If the RE assumptions are correct, then we should find that $\gamma = 0$ and then these α_i simplify to the standard random intercept from that approach. However, if $\gamma \neq 0$ then we have incorporated a way for the observe covariates x to be correlated with the unobserved heterogeneity α_i. Note that we are retaining iid within-unit observations here (from the RE setup), so the only within-unit autocorrelation is in the form of the constant u_i.

Essentially, we are accommodating the unobserved heterogeneity by modeling it's relationship to the observables and controlling for that. We are directly controlling for deviations from the within means (as in the within model), while deviations of y_{it} from $\bar{y}_i$ and u_{it} from u_i are captured in α_i and α. And indeed $\hat{\beta}_{CRE} = \hat{\beta}_{LSDV} = \hat{\beta}_w$. To see this consider the following alternative derivation starting with the within model:

$$\begin{aligned}
y_{it} - \bar{y}_i &= \beta_w'(x_{it} - \bar{x}_i) + (\varepsilon_{it} - \bar{\varepsilon}_i) \\
y_{it} &= \beta_w'(x_{it} - \bar{x}_i) + (\varepsilon_{it} - \bar{\varepsilon}_i) + \bar{y}_i \\
\bar{y}_i &= \alpha + u_i + \beta_b' \bar{x}_i + \bar{\varepsilon}_i \\
y_{it} &= \beta_w'(x_{it} - \bar{x}_i) + (\varepsilon_{it} - \bar{\varepsilon}_i) + \alpha + u_i + \beta_b' \bar{x}_i + \bar{\varepsilon}_i \\
&= \beta_w' x_{it} + (\beta_w - \beta_b)' \bar{x}_i + \alpha + u_i + \varepsilon_{it} \\
&= \beta_w' x_{it} + \gamma' \bar{x}_i + \alpha + u_i + \varepsilon_{it}
\end{aligned}$$

Which is to say that we get the within estimates back on x_{it} and the difference between the within and between estimates back on $\bar{x}_i$. This gives us another nice alternative to dummy variables that uses fewer parameters.

Now it also suggests a specification test. Namely if $\alpha = 0$ (i.e., $\beta_w = \beta_b$), then the there are no real differences between the within and between estimators and we can use the more efficient RE estimator (or the pooled estimator if we still want to avoid questionable RE assumptions). This becomes an ordinary Wald test of the hypothesis that $\gamma = 0$. Unlike the

Hausman test, the Wald test is well defined and applicable with (cluster) robust covariance matrices.

Note that u_i can be safely ignored here regardless of whether it is correlated with the observables or not because we have obtained the within estimates on the observables. As such we can treat it as either fixed or random. We can leave it in the error term (i.e., fit the above equation with the pooled estimator) or treat it as random and fit the GLS. Regardless, we'll get the within estimates for β.

1.5 Application

In this example we're going to be working with data from Choulis, Escribá-Folch, and Mehri (2024, *JOP*).[1] In this paper, they consider how the presence of secret police within a country affect anti-regime protests.

The outcome of interest is a latent measure based on combining information from several different protest datasets. The treatment of interest is whether there is a secret police organization within that country-year observation (binary). They also consider several control variables include population, GDP per captia, economic growth, politically exclude ethnic groups, protests in neighboring countries, civil conflict, and coup attempts. We will take their specification at face value and observe the following specification

$$\text{Protests}_{it} = \alpha_i + \beta_1 \text{Secret Police}_{it} + x_{it}'\gamma + \varepsilon_{it}.$$

We will consider pooling, random effects, and FE estimation.

```
## data manipulation packages
library(readstata13)
library(data.table)

## econometrics packages
library(lmtest)
library(car)
library(sandwich)
library(fixest)
library(lme4)
library(clubSandwich)
```

[1] https://doi.org/10.1086/729953

```r
## tables and figures
library(modelsummary)

## checking out the data
protests <- read.dta13("Rcode/datasets/Replication_secpol_protestComplete.dta")
protests <- data.table(protests)
protests <- protests[order(ccode, year),]

colnames(protests)
```

```
##  [1] "ccode"             "year"              "country"
##  [4] "Region"            "secretpol_revised" "pop"
##  [7] "gdp_pc"            "intrastate"        "polity2"
## [10] "attempt"           "theta_mean"        "physint"
## [13] "disap"             "kill"              "polpris"
## [16] "tort"              "Capacity"          "v2clrspct"
## [19] "v2stfisccap"       "v2terr"            "v2cseeorgs"
## [22] "v2csreprss"        "v2csprtcpt"        "v2csantimv"
## [25] "v2csstruc_1"       "solschdum"         "urbanpop"
## [28] "l12gr"             "xpers"             "lexclpop"
## [31] "effectivenumber"   "mean3"             "mean5"
## [34] "nbr_mean3"         "nbr_mean5"
```

```r
## panel dimenions
length(unique(protests$ccode))
```

```
## [1] 208
```

```r
summary(protests[, length(year), by = ccode])
```

```
##      ccode             V1
##  Min.   :  2.0   Min.   : 1.00
##  1st Qu.:313.8   1st Qu.:69.00
##  Median :466.0   Median :69.00
##  Mean   :479.9   Mean   :63.76
##  3rd Qu.:694.5   3rd Qu.:69.00
##  Max.   :990.0   Max.   :69.00
```

```r
## adjust the variables based on their replication file

## Normalize the latent varaible to be  mean 0, var 1
protests[, Protest := scale(mean5)]
protests[, nbr_protest := scale(nbr_mean5)]

## create the controls: lag(log(pop)), lag(log(gdp_pc), lag(excluded population))
protests[, `:=` (l.ln_pop = shift(log(pop+1)),
                 l.ln_gdppc = shift(log(gdp_pc)),
                 l.lexclpop = shift(lexclpop)),
         by=ccode]

## model formula
f1 <- Protest~ secretpol_revised + l.ln_pop + l.ln_gdppc  + l12gr+  l.lexclpop+
  nbr_protest+intrastate+attempt


## Fitting with the pooled esetimator
pooled  <- lm(f1, data=protests, x=TRUE)
summary(pooled)
```

```
##
## Call:
## lm(formula = f1, data = protests, x = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.18241 -0.48782  0.00395  0.48523  1.98517
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -6.265451   0.185713 -33.737  < 2e-16 ***
## secretpol_revised -0.072502   0.031246  -2.320   0.0204 *
## l.ln_pop           0.337756   0.009246  36.529  < 2e-16 ***
## l.ln_gdppc         0.115105   0.010665  10.793  < 2e-16 ***
## l12gr             -0.011531   0.001964  -5.872 4.73e-09 ***
```

```
## l.lexclpop          0.101536   0.043745   2.321   0.0203 *
## nbr_protest         0.158305   0.013073  12.109  < 2e-16 ***
## intrastate          0.192116   0.031781   6.045 1.66e-09 ***
## attempt             0.217309   0.048047   4.523 6.32e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6917 on 3245 degrees of freedom
##   (10009 observations deleted due to missingness)
## Multiple R-squared:  0.4165, Adjusted R-squared:  0.415
## F-statistic: 289.5 on 8 and 3245 DF,  p-value: < 2.2e-16
```

```
## To make life easy
## We're going to restrict ourselves to just the used sample
protests <- protests[as.numeric(row.names(pooled$model)), ]

## Let's consider the residual autocorrelation
## in choosing standard errors
protests[, e.hat := pooled$residuals]
protests[, L.e.hat := shift(e.hat), by =ccode]
summary(lm(e.hat~L.e.hat, data=protests)) #that's pretty high!
```

```
##
## Call:
## lm(formula = e.hat ~ L.e.hat, data = protests)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1104 -0.1078 -0.0032  0.1080  0.7920
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.003252   0.003175   1.024    0.306
## L.e.hat     0.970443   0.004629 209.648   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.1779 on 3139 degrees of freedom
##   (113 observations deleted due to missingness)
## Multiple R-squared:  0.9333, Adjusted R-squared:  0.9333
## F-statistic: 4.395e+04 on 1 and 3139 DF,  p-value: < 2.2e-16
```

```
## Clustering the standard errors
Vcl.pooled <- vcovCL(pooled, cluster=protests$ccode)
round(coeftest(pooled, Vcl.pooled), 5)
```

```
##
## t test of coefficients:
##
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -6.26545    0.72706 -8.6175  < 2e-16 ***
## secretpol_revised -0.07250    0.10565 -0.6863  0.49259
## l.ln_pop           0.33776    0.03442  9.8120  < 2e-16 ***
## l.ln_gdppc         0.11510    0.05135  2.2416  0.02505 *
## l12gr             -0.01153    0.00384 -3.0029  0.00269 **
## l.lexclpop         0.10154    0.15857  0.6403  0.52200
## nbr_protest        0.15830    0.05752  2.7523  0.00595 **
## intrastate         0.19212    0.08844  2.1722  0.02992 *
## attempt            0.21731    0.07627  2.8493  0.00441 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## Suppose we wanted to bootstrap we have the clustered bootstrap
pooled.boot <- t(replicate(50, {
  idx <- sample(unique(protests$ccode),
                size=length(unique(protests$ccode)),
                replace=TRUE)
  d <- copy(protests)
  d <- d[unlist(sapply(idx, \(x){which(d$ccode==x)}))]
  pooled.bs <- lm(f1, dat=d)
  pooled.bs$coef
}))
round(coeftest(pooled, var(pooled.boot)), 5)
```

```
##
```

```
## t test of coefficients:
##
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -6.26545    0.90032 -6.9591  < 2e-16 ***
## secretpol_revised -0.07250    0.14023 -0.5170  0.60519
## l.ln_pop           0.33776    0.04292  7.8701  < 2e-16 ***
## l.ln_gdppc         0.11510    0.06132  1.8772  0.06059 .
## l12gr             -0.01153    0.00398 -2.9005  0.00375 **
## l.lexclpop         0.10154    0.15421  0.6584  0.51031
## nbr_protest        0.15830    0.05551  2.8516  0.00438 **
## intrastate         0.19212    0.08414  2.2832  0.02248 *
## attempt            0.21731    0.07474  2.9075  0.00367 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## And the clustered bayesian bootstrap
pooled.bayes.boot <- t(replicate(50, {
  Ti <- table(protests$ccode)
  d <- copy(protests)
  weight <- rexp(length(unique(protests$ccode)))
  weight <- weight/sum(weight)
  d$weight <- rep(weight*length(unique(protests$ccode)), Ti)
  lm(f1, dat=d, weights=weight)$coef
}))
round(coeftest(pooled, var(pooled.bayes.boot)), 5)
```

```
##
## t test of coefficients:
##
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -6.26545    0.75865 -8.2586  < 2e-16 ***
## secretpol_revised -0.07250    0.10021 -0.7235  0.46942
## l.ln_pop           0.33776    0.03859  8.7515  < 2e-16 ***
## l.ln_gdppc         0.11510    0.05014  2.2956  0.02176 *
## l12gr             -0.01153    0.00319 -3.6107  0.00031 ***
## l.lexclpop         0.10154    0.15421  0.6584  0.51030
## nbr_protest        0.15830    0.05755  2.7508  0.00598 **
```

```
## intrastate          0.19212    0.09509  2.0203  0.04343 *
## attempt             0.21731    0.07011  3.0995  0.00196 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## We can consider fixed effects estimators too. Starting with the LSDV
lsdv  <- lm(update(f1, .~. -1 + factor(ccode)), data=protests)
Vcl.lsdv <- vcovCL(lsdv, cluster=protests$ccode)
round(coeftest(lsdv, Vcl.lsdv)[1:8,], 4)
```

```
##                   Estimate Std. Error t value Pr(>|t|)
## secretpol_revised  -0.2716     0.0926 -2.9334   0.0034
## l.ln_pop            0.6411     0.1077  5.9507   0.0000
## l.ln_gdppc         -0.0180     0.0804 -0.2236   0.8231
## l12gr              -0.0041     0.0026 -1.5970   0.1104
## l.lexclpop         -0.0128     0.1075 -0.1190   0.9053
## nbr_protest         0.1088     0.0664  1.6380   0.1015
## intrastate          0.1851     0.0542  3.4151   0.0006
## attempt             0.1141     0.0432  2.6393   0.0083
```

```
## Within transformation
var.names <- colnames(pooled$model)
protests[,paste0(var.names, ".within"):=lapply(.SD, \(x){x- mean(x)}),
         by=ccode, .SDcols=var.names ]
fwithin <- paste0(var.names[1], ".within ~ -1 + ",
                  paste0(var.names[-1], ".within", collapse=" + "))
within1 <- lm(fwithin, data=protests)
Vcl.within1 <- vcovCL(within1, cluster=protests$ccode)
round(coeftest(within1, Vcl.within1), 4)
```

```
##
## t test of coefficients:
##
##                           Estimate Std. Error t value Pr(>|t|)
## secretpol_revised.within  -0.2716     0.0910 -2.9858   0.0028 **
## l.ln_pop.within            0.6411     0.1058  6.0571   <2e-16 ***
## l.ln_gdppc.within         -0.0180     0.0790 -0.2276   0.8199
## l12gr.within              -0.0041     0.0025 -1.6255   0.1042
```

```
## l.lexclpop.within         -0.0128     0.1056 -0.1211   0.9036
## nbr_protest.within         0.1088     0.0653  1.6673   0.0956 .
## intrastate.within          0.1851     0.0533  3.4762   0.0005 ***
## attempt.within             0.1141     0.0425  2.6865   0.0073 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## The fixest is the better way to go here.  It takes
## a formula of the form y~x|heterogeneity. And automatically
## clusters the variance
within2 <- feols(Protest~ secretpol_revised + l.ln_pop + l.ln_gdppc  + l12gr+  l.lexclpo
                   nbr_protest+intrastate+attempt|ccode, data=protests)
summary(within2)
```

```
## OLS estimation, Dep. Var.: Protest
## Observations: 3,254
## Fixed-effects: ccode: 113
## Standard-errors: Clustered (ccode)
##                    Estimate Std. Error   t value   Pr(>|t|)
## secretpol_revised -0.271642   0.090992 -2.985330 3.4792e-03 **
## l.ln_pop           0.641114   0.105861  6.056172 1.9048e-08 ***
## l.ln_gdppc        -0.017976   0.078980 -0.227601 8.2037e-01
## l12gr             -0.004095   0.002520 -1.625256 1.0692e-01
## l.lexclpop        -0.012796   0.105648 -0.121119 9.0381e-01
## nbr_protest        0.108823   0.065281  1.666995 9.8309e-02 .
## intrastate         0.185147   0.053270  3.475647 7.2605e-04 ***
## attempt            0.114058   0.042462  2.686092 8.3295e-03 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 0.422676     Adj. R2: 0.77316
##                  Within R2: 0.222534
```

```
## truely the same
max(abs(lsdv$residuals-within1$residuals))
```

```
## [1] 3.28626e-14
```

```r
max(abs(lsdv$residuals-within2$residuals))
```

```
## [1] 3.153033e-14
```

```r
## here we can see the difference between the
## total and within r-squared
c(summary(lsdv)$r.sq, summary(within1)$r.sq)
```

```
## [1] 0.7825909 0.2225345
```

```r
## why are these different?
## which of these are unbiased estimates? Which are consistent?
c(summary(lsdv)$sigma, summary(within1)$sigma, sqrt(summary(within2)$sigma2))
```

```
## [1] 0.4307607 0.4231964 0.4307607
```

```r
## build the weights for RE=GLS
Ti <- table(protests$ccode) #unbalanced panel so each unit has different weight
Ti <- rep(Ti, Ti)
sigma2.eps <- within2$sigma2 #unbiased and consistent
sigma2.a <- mean(pooled$residuals^2) -sigma2.eps
protests$omega.hat <- 1- sqrt(sigma2.eps/(Ti*sigma2.a+sigma2.eps) )
mean(protests$omega.hat) ## fairly similar on this measure
```

```
## [1] 0.8587313
```

```r
protests[,paste0(var.names, ".gls"):=lapply(.SD, \(x){x-omega.hat*mean(x)}),
         by=ccode, .SDcols=var.names ]
protests[,const.gls:=1-omega.hat]
fgls <- paste0(var.names[1], ".gls ~ -1 + const.gls + ",
               paste0(var.names[-1], ".gls", collapse=" + "))

gls <- lm(fgls, data=protests)
summary(gls)
```

```
##
## Call:
## lm(formula = fgls, data = protests)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -1.25424 -0.32143 -0.02315  0.29313  1.63801
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## const.gls               -8.420488   0.366711 -22.962  < 2e-16 ***
## secretpol_revised.gls -0.252149   0.035191  -7.165 9.57e-13 ***
## l.ln_pop.gls             0.516274   0.022047  23.417  < 2e-16 ***
## l.ln_gdppc.gls           0.044860   0.020036   2.239 0.025230 *
## l12gr.gls               -0.004943   0.001356  -3.644 0.000272 ***
## l.lexclpop.gls          -0.042889   0.051030  -0.840 0.400712
## nbr_protest.gls          0.138695   0.017534   7.910 3.49e-15 ***
## intrastate.gls           0.191343   0.025758   7.429 1.40e-13 ***
## attempt.gls              0.110429   0.031822   3.470 0.000527 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4359 on 3245 degrees of freedom
## Multiple R-squared:  0.2234, Adjusted R-squared:  0.2212
## F-statistic: 103.7 on 9 and 3245 DF,  p-value: < 2.2e-16
```

```
Vcl.gls <- vcovCL(gls, cluster=protests$ccode)
round(coeftest(gls, Vcl.gls), 4)
```

```
##
## t test of coefficients:
##
##                         Estimate Std. Error t value Pr(>|t|)
## const.gls               -8.4205     1.1471 -7.3409   <2e-16 ***
## secretpol_revised.gls  -0.2521     0.0844 -2.9864   0.0028 **
## l.ln_pop.gls             0.5163     0.0688  7.5047   <2e-16 ***
## l.ln_gdppc.gls           0.0449     0.0623  0.7197   0.4718
## l12gr.gls               -0.0049     0.0025 -1.9586   0.0502 .
## l.lexclpop.gls          -0.0429     0.1002 -0.4280   0.6687
## nbr_protest.gls          0.1387     0.0615  2.2561   0.0241 *
## intrastate.gls           0.1913     0.0526  3.6370   0.0003 ***
## attempt.gls              0.1104     0.0422  2.6144   0.0090 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## The lme4 package is the more common way to go here.  It takes
## a formula of the form y~x+(1|heterogeneity).
## However, it does not work with the sandwich package, so
## we move the clubSandwich package for clustering.
## It also doesn't like the lmtest package that much
re  <- lmer(update(f1, . ~ . + (1|ccode)), data=protests)
Vcl.re <- vcovCR(re, cluster=protests$ccode, type="CR1")
coef_test(re, Vcl.re)
```

```
##              Coef. Estimate      SE t-stat d.f. (Satt) p-val (Satt) Sig.
##         (Intercept)  -8.8185 1.27158 -6.935        67.0       <0.001  ***
##  secretpol_revised  -0.2585 0.08628 -2.996        22.3       0.0066   **
##            l.ln_pop   0.5493 0.07827  7.018        60.0       <0.001  ***
##          l.ln_gdppc   0.0285 0.06664  0.427        23.1       0.6731
##              l12gr  -0.0047 0.00252 -1.870        22.8       0.0744    .
##          l.lexclpop  -0.0365 0.10105 -0.361        18.5       0.7219
##         nbr_protest   0.1307 0.06265  2.087        65.0       0.0408    *
##          intrastate   0.1898 0.05268  3.604        48.7       <0.001  ***
##             attempt   0.1109 0.04224  2.625        47.8       0.0116    *
```

```
## hausman  (with iid)
Htest <- c(within2$coefficients - re@beta[-1]) %*%
  solve(within2$cov.iid - vcov(re)[-1,-1]) %*%
  c(within2$coefficients - re@beta[-1])
pchisq(drop(Htest), df=length(within2$coefficients), lower=FALSE)
```

```
## [1] 0.0004674246
```

```
##hausman  (with clustering) but this version is sus
Htest.cl <- c(within2$coefficients - re@beta[-1]) %*%
  solve(vcov(within2) - Vcl.re[-1,-1]) %*%
  c(within2$coefficients - re@beta[-1])
pchisq(drop(Htest.cl), df=length(within2$coefficients), lower=FALSE)
```

```
## [1] 1
```

```
### Mundlak--pooled
Xnames <- colnames(pooled$model)[-1]
protests[,paste0(var.names, ".bar"):=lapply(.SD, \(x){mean(x)}),
         by=ccode, .SDcols=var.names ]
mundlak.add <- paste(".~.",paste0("+", Xnames, ".bar",collapse = "" ))
mundlak.formula <- update(f1, mundlak.add)
mundlak <- lm(mundlak.formula, data=protests)
Vcl.m <- vcovCL(mundlak, cluster=protests$ccode)
round(coeftest(mundlak, Vcl.m), 4)
```

```
##
## t test of coefficients:
##
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -6.1546     0.8482 -7.2564   <2e-16 ***
## secretpol_revised       -0.2716     0.0911 -2.9816   0.0029 **
## l.ln_pop                 0.6411     0.1060  6.0487   <2e-16 ***
## l.ln_gdppc              -0.0180     0.0791 -0.2273   0.8202
## l12gr                   -0.0041     0.0025 -1.6233   0.1046
## l.lexclpop              -0.0128     0.1058 -0.1210   0.9037
## nbr_protest              0.1088     0.0654  1.6649   0.0960 .
## intrastate               0.1851     0.0533  3.4714   0.0005 ***
## attempt                  0.1141     0.0425  2.6828   0.0073 **
## secretpol_revised.bar    0.3140     0.1828  1.7180   0.0859 .
## l.ln_pop.bar            -0.3116     0.1130 -2.7576   0.0059 **
## l.ln_gdppc.bar           0.1318     0.0958  1.3748   0.1693
## l12gr.bar               -0.0383     0.0198 -1.9360   0.0530 .
## l.lexclpop.bar           0.0655     0.2354  0.2782   0.7809
## nbr_protest.bar          0.0648     0.0866  0.7480   0.4545
## intrastate.bar          -0.0091     0.2013 -0.0450   0.9641
## attempt.bar              1.1473     0.6343  1.8088   0.0706 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
linearHypothesis(mundlak, paste0(Xnames, ".bar=0"), vcov=Vcl.m)
```

```
## Linear hypothesis test
```

```
##
## Hypothesis:
## secretpol_revised.bar = 0
## l.ln_pop.bar = 0
## l.ln_gdppc.bar = 0
## l12gr.bar = 0
## l.lexclpop.bar = 0
## nbr_protest.bar = 0
## intrastate.bar = 0
## attempt.bar = 0
##
## Model 1: restricted model
## Model 2: Protest ~ secretpol_revised + l.ln_pop + l.ln_gdppc + l12gr +
##     l.lexclpop + nbr_protest + intrastate + attempt + secretpol_revised.bar +
##     l.ln_pop.bar + l.ln_gdppc.bar + l12gr.bar + l.lexclpop.bar +
##     nbr_protest.bar + intrastate.bar + attempt.bar
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df     F  Pr(>F)
## 1   3245
## 2   3237  8 2.562 0.00876 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
### Mundlak--CRE
cre  <- lmer(update(mundlak.formula, . ~ . + (1|ccode)), data=protests)
Vcl.cre <- vcovCR(cre, cluster=protests$ccode, type="CR1")
coef_test(cre, Vcl.cre)
```

```
##                  Coef. Estimate      SE t-stat d.f. (Satt) p-val (Satt) Sig.
##             (Intercept) -6.10516 0.84238 -7.248        45.2      < 0.001  ***
##      secretpol_revised -0.27164 0.09088 -2.989        20.7      0.00707   **
##                l.ln_pop  0.64111 0.10573  6.064        41.7      < 0.001  ***
##              l.ln_gdppc -0.01798 0.07888 -0.228        18.1      0.82229
##                   l12gr -0.00409 0.00252 -1.627        22.6      0.11753
##              l.lexclpop -0.01280 0.10552 -0.121        17.5      0.90486
```

```
##            nbr_protest  0.10882 0.06520  1.669        60.3      0.10029
##             intrastate  0.18515 0.05320  3.480        48.2      0.00107   **
##                attempt  0.11406 0.04241  2.689        47.7      0.00983   **
##  secretpol_revised.bar  0.31416 0.19237  1.633        45.4      0.10938
##          l.ln_pop.bar -0.31137 0.11380 -2.736        54.1      0.00839   **
##         l.ln_gdppc.bar  0.12956 0.09593  1.351        37.9      0.18482
##             l12gr.bar -0.02341 0.01857 -1.260        13.7      0.22858
##         l.lexclpop.bar  0.03421 0.24765  0.138        37.9      0.89086
##        nbr_protest.bar  0.07145 0.08324  0.858        46.3      0.39515
##         intrastate.bar -0.04938 0.21551 -0.229        34.1      0.82012
##            attempt.bar  1.32038 0.57089  2.313        20.6      0.03116    *
```

```
## CRE R squared
1-sum(residuals(cre)^2)/sum((protests$Protest-mean(protests$Protest))^2)
```

```
## [1] 0.7812683
```

```
##Matches the LSDV closely
max(abs(lsdv$residuals-residuals(cre)))
```

```
## [1] 0.2097183
```

```
### between
protests[, Protest.bar := mean(Protest), by=ccode]
f.btwn <- as.formula(paste("Protest.bar ~",
                           paste0(Xnames, ".bar",collapse = " + " ) ))
btwn <- lm(f.btwn, data=protests)

cbind(within2$coefficients, mundlak$coef[2:9])
```

```
##                           [,1]         [,2]
## secretpol_revised -0.271641863 -0.271641863
## l.ln_pop           0.641113791  0.641113791
## l.ln_gdppc        -0.017975958 -0.017975958
## l12gr             -0.004094911 -0.004094911
## l.lexclpop        -0.012795881 -0.012795881
## nbr_protest        0.108823234  0.108823234
## intrastate         0.185147476  0.185147476
## attempt            0.114057919  0.114057919
```

	Pooled	RE-GLS	RE-MLE	LSDV	Within	Mundlak	CRE
Secret police	−0.07	−0.25	−0.26	−0.27	−0.27	−0.27	−0.27
	(0.11)	(0.08)	(0.09)	(0.09)	(0.09)	(0.09)	(0.09)
Num.Obs.	3254	3254	3254	3254	3254	3254	3254
R2	0.416	0.223		0.783	0.782	0.447	
R2 Within					0.223		

```
cbind(BtwnDiff=btwn$coef[-1]-within2$coefficients,
      mundlak$coef[10:17])
```

```
##                            BtwnDiff
## secretpol_revised.bar  0.314034175  0.314034175
## l.ln_pop.bar          -0.311639916 -0.311639916
## l.ln_gdppc.bar         0.131772527  0.131772527
## l12gr.bar             -0.038255641 -0.038255641
## l.lexclpop.bar         0.065499295  0.065499295
## nbr_protest.bar        0.064750150  0.064750150
## intrastate.bar        -0.009054607 -0.009054607
## attempt.bar            1.147308716  1.147308716
```

```
modelsummary(list("Pooled"=pooled,
                  "RE-GLS"=gls,
                  "RE-MLE"=re,
                  "LSDV"=lsdv,
                  "Within"=within2,
                  "Mundlak"=mundlak,
                  "CRE"=cre),
             vcov=list(Vcl.pooled, Vcl.gls, Vcl.re,
                       Vcl.lsdv, vcov(within2),
                       Vcl.m, Vcl.cre),
             fmt=2,
             coef_map=c("secretpol_revised"="Secret police",
                        "secretpol_revised.gls"="Secret police",
                        "secretpol_revised.within"="Secret police"),
             gof_map=c("nobs", "r.squared", "r2.within"))
```

1.6 Two-way heterogeneity

Having considered the one-way heterogeneity model to some extent, we may want start considering extensions. Given that panels are often seen as N separate time-series we may want to start by thinking about a model of the form

$$y_{it} = \beta' x_{it} + \alpha_i + f(t) + \varepsilon_{it}.$$

As in the above format we can consider the heterogenity as functions of observable and unobservable factors such that

$$\alpha_i = \alpha + \gamma' z_i + u_i$$

There are many ways to think about time here.
The easiest is a simple time trend: $f(t) = \tau t$. This could be made more flexible by using polynomials $f(t) = \tau_1 t + \tau_2 t^2 + \ldots$ or splines. Alternatively, we could do a time trend by individual $f_i(t) = \tau_i t$, in which case we would interact t with the unit-dummies.

These kind of functional forms can be appealing, but they tend to require a relatively strong assumption on how time works in the empirical model. Should it be linear? Some kind of cycle? Cycles may get weird. As such a non-parametric form based on the full specification above may be preferred, as in

$$f(t) = \tau_t + v_t.$$

This approach is called two-way heterogeneity or the two-way fixed effects model. In matrix form we can write this for a balanced panel as

$$y = \begin{bmatrix} X & (I_N \otimes 1_T) & (1_T \otimes I_N) \end{bmatrix} \theta + \varepsilon,$$

where $\theta = (\beta, \alpha_i, \tau_t)$. For examples on Kronecker products

```
N <- 3
T <- 2
diag(N) %x% rep(1, T)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    1    0    0
## [3,]    0    1    0
## [4,]    0    1    0
```

```
## [5,]    0    0    1
## [6,]    0    0    1
```

```
rep(1, T) %x% diag(N)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
## [4,]    1    0    0
## [5,]    0    1    0
## [6,]    0    0    1
```

```
diag(N) %x% matrix(1, ncol=T, nrow=1)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    1    0    0    0    0
## [2,]    0    0    1    1    0    0
## [3,]    0    0    0    0    1    1
```

```
matrix(1, ncol=T, nrow=1) %x% diag(N)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    0    0    1    0    0
## [2,]    0    1    0    0    1    0
## [3,]    0    0    1    0    0    1
```

```
A <- matrix(1:4, ncol=2)
A %x% matrix(1, ncol = 2, nrow=2)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    1    3    3
## [2,]    1    1    3    3
## [3,]    2    2    4    4
## [4,]    2    2    4    4
```

```
matrix(1, ncol = 2, nrow=2) %x% A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    1    3
## [2,]    2    4    2    4
```

```
## [3,]    1    3    1    3
## [4,]    2    4    2    4
```

Now when we believe that two-way heterogeneity is present we cannot ignore either form. Omitted variables are of course one problem, but even with no correlation we have a problem if either dimension is small. For example, consider a large N survey over a small number of waves T and consider the one-way within estimator:

$$\begin{aligned}
\hat{\beta}_w &= \left[\sum_i \sum_t (x_{it} - \bar{x}_i)(x_{it} - \bar{x}_i)'\right]^{-1} \left[\sum_i \sum_t (x_{it} - \bar{x}_i)(y_{it} - \bar{y}_i)\right] \\
&= \beta + \left[\frac{1}{NT}\sum_i \sum_t (x_{it} - \bar{x}_i)(x_{it} - \bar{x}_i)'\right]^{-1} \left[\frac{1}{NT}\left(\sum_i \sum_t (x_{it} - \bar{x}_i)(\tau_t - \bar{\tau})\right.\right. \\
&\qquad \left.\left. + \sum_i \sum_t (x_{it} - \bar{x}_i)(\varepsilon_{it} - \bar{\varepsilon}_i)\right)\right] \\
&= \beta + \left[\frac{1}{NT}\sum_i \sum_t (x_{it} - \bar{x}_i)(x_{it} - \bar{x}_i)'\right]^{-1} \left[\frac{1}{NT}\left(\sum_i \sum_t (x_{it} - \bar{x}_i)(\tau_t - \bar{\tau}) + 0\right)\right]
\end{aligned}$$

So far so good, but with a little algebra we get to

$$\frac{1}{NT}\sum_i \sum_t (x_{it} - \bar{x}_i)(\tau_t - \bar{\tau}) = \frac{1}{T}\sum_t (\bar{x}_t - \bar{x})(\tau_t - \bar{\tau}).$$

This will converge to its expected value (zero if x is uncorrelated with the time effects), but this convergence is in T! If T is relatively small, then we can't rely on that. To put this another way, even if they are uncorrelated with the observables, time effects can still bias the estimates of β if T is not large!

The consequence of this is that unless you believe that the time effects are constant $\tau_t = \bar{\tau}$ for all t, if this dimension is small, then we should consider time heterogeneity as an important bias to control for.

We can do this in the same three ways we described above:

1. A two-way within transformation:

$$M_2 = \underbrace{I_{NT} - (I_N \otimes 1_T 1_T'/T)}_{\text{Unit demeaning}} - \underbrace{(1_N 1_N'/N \otimes I_T)}_{\text{Time means}} + \underbrace{\frac{1}{NT} 1_{NT} 1_{NT}'}_{\text{overall mean}}.$$

   Note that here (the balanced case) we have the original group-wise demeaning, then

time-wise demeaning and then we add back in the overall mean, as in

$$M_2 X = \left[x_{it} - \bar{x}_i - \bar{x}_t + \bar{x} \right].$$

This transformation is more involved with unbalanced panels.

2. Dummies: As before, just include dummies for each i and each t. 1 of these will need to be removed to avoid colinearity
3. CRE: When the panel is balanced, then CRE with time means and group means will still be equivalent. This equivalence does not hold for unbalanced panels.

The within transformation in unbalanced panels is slightly convoluted, but we can see how it maps into the above.

$$M_2 = M - M\Delta_T[\Delta_T' M \Delta_T]^{-1}\Delta_T' M$$
$$M = I_{NT} - \Delta_N[\Delta_N'\Delta_N]^{-1}\Delta_N'.$$

Here, Δ_N and Δ_T are matrices of unit and time dummies, respectively. We remove the first (or any) column from Δ_T to avoid colinearity. Note that M here is the unit-demeaning matrix for the whole sample (diagonal binding the M_is).

When the panel is balanced,

$$\Delta_N[\Delta_N'\Delta_N]^{-1}\Delta_N' = (I_N \otimes 1_T 1_T')/T,$$

which gives us a block diagonal matrix of $1/T$, and

$$M\Delta_T[\Delta_T' M \Delta_T]\Delta_T' M = (1_N 1_N' \otimes I_T)/N - \frac{1}{NT} 1_{NT} 1_{NT}'.$$

In the unbalanced case we get weighted averages for the time and overall means based on how often they appear in the sample.

1.6.1 Asymptotics in T

While we're considering the different dimensions of the panel, we should also be clear about fixed-N asymptotics. In survey data and many other contexts, fixed-T-large-N makes sense. However, in other parts of political science we often have a fixed (or fairly fixed) N. For example, the number of U.S. states or countries of the world don't increase all that often and are fairly static in many cases for which we collect data.

In these cases, it may make more sense to think about large T asymptotics. After all, in country-year data we typically have a fairly fixed N, but T is increasing as we move forward in

time and data collection continues. So what does it mean to think about the panel estimators in that context?

The pooled estimator should now be rewritten as

$$\hat{\theta}_p = \theta + \left[\frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_{it}\mathbf{x}_{it}'\right)\right]^{-1}\left(\frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_{it}\varepsilon_{it}\right)\right).$$

Before we can do anything with this, we'll need to add a few assumptions. First, instead of allowing for arbitrary correlation within-units we'll impose some constraints on the time series. Note that if N is also reasonably large these won't be as important, but here we're assuming that appealing to asymptotics in N is a tough sell.

Assumption A6 *The sequence* $(\mathbf{x}_t, \varepsilon_t)$ *is strictly stationary and ergodic*

Note that here $\mathbf{x}_t = (\mathbf{x}_{1t}, \mathbf{x}_{2t}, \ldots, \mathbf{x}_{Nt})$ and likewise for ε_t.

Assumption A7 *The matrix* $\mathrm{E}[\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_{it}\mathbf{x}_{it}']$ *has full rank.*

To review, a sequence y_t is (strictly) *stationary* if the joint distribution of $(y_t, \ldots y_{t+k})$ is independent of both t and k. This basically means that distribution of y_t does not change with time so its mean and variance are constant, but also that the relationships between parts of the time series are constant over time. For example, the covariance between y_1 and y_3 is the same as y_5 and y_7.

Some example of stationary series include $y_t = x_t + \theta x_{t-1}$, $y_t = x_t$, and $y_t = x$, where x_t is iid with $|\theta| < 1$, $\mathrm{E}[x_t] = 0$ and x is a single realization of a random variable.

A stationary series y_t is **erogodic** if, well that is complicated and takes awhile to really explain. However, at its core, an ergodic sequence can never get "stuck." An ergodic y_t will eventually visit every value in its support if the sequence lasts long enough and it can move from any part of its support to any other with positive probability. Another way to think about this is that when we have an ergodic sequence any long-enough sub-sample will have the same statistical properties like the mean.

To make life easier, we will also assume that y_t is *mixing*. This means that as ℓ increases the $\mathrm{Cov}(y_t, y_{t-\ell})$ goes to 0. As the time between points increases, they provide less and less information about each other. Note that mixing implies ergodicity, but not vice-versa.

Of import to us is the following theorem

Theorem 7 *Let* y_t *be a strictly stationary and ergodic random variable and let* f *be a*

continuous function. Then $X_t = f(y_t, y_{t-1}, \ldots)$ *is also strictly stationary and ergodic.*

This theorem tells us that stationarity is preserved by continuous transformations that consider some or part of the history of y_t. Don't lose any sleep over it other than to remember the intuition part.

What this gives us now is a time-series version of a law of large numbers

Theorem 8 *(Ergodic LLN) Let* y_t *be stationary and ergodic with* $\mathrm{E}[y_t] < \infty$*, then* $\frac{1}{T}\sum_{t=1}^{T} y_t \xrightarrow{p} \mathrm{E}[y_t]$.

We will also replace assumption A3 with A3′ **Assumption A3′** $\mathrm{E}[\varepsilon_{it}|\mathbf{x}_{it}] = 0$.

This gives us something to work with now

$$\frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_{it}\mathbf{x}_{it}'\right) \xrightarrow{p} \mathrm{E}\left[\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_{it}\mathbf{x}_{it}'\right]$$
$$\frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_{it}\varepsilon_{it}\right) \xrightarrow{p} \mathrm{E}\left[\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_{it}\varepsilon_{it}\right]$$
$$\mathrm{E}\left[\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_{it}\varepsilon_{it}\right] = \mathrm{E}_{\mathbf{x}_{it}}\left[\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_{it}\,\mathrm{E}\left[\varepsilon_{it}|\mathbf{x}_{it}\right]\right] = 0$$

From here, the usual applications of Slutsky's theorem follows and we get that pooled OLS is consistent in T under Assumptions A1.A, A2, A3′, A6, & A7 the pooled estimator is consistent and surely exists for large enough T. If we want to include strict-within unit exogeneity (increasingly unlikely as T increases), then we also get unbiased estimates.

Assumption A8 *Additional technical assumptions that allow us to use a central limit theorem for dependent data*

We now present a central limit theorem for stationary mixing sequences

Theorem 9 *Let* z_t *be strictly stationary and mixing with* $\mathrm{E}[z_t] = 0$ *and some other conditions. Then*

$$\sqrt{T}\left(\frac{1}{T}\sum_{t=1}^{T} z_t\right) \xrightarrow{d} N(0, \Sigma_T)$$

For ease let $z_t = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_{it}\varepsilon_{it}$ be stationary and mixing. This gives us the following to work

with

$$\sqrt{T}\frac{1}{T}\sum_{t=1}^{T}\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_{it}\varepsilon_{it} = \sqrt{T}\frac{1}{T}\sum_{t=1}^{T}z_t \xrightarrow{d} N(0,\Sigma_T)$$

$$\Sigma_T = \text{Var}\left(T^{-1/2}\sum_{t=1}^{T}z_t\right)$$

Using some time series results for stationary and ergodic series we can write this as

$$\Sigma_T = \lambda(0) + \sum_{\ell=1}^{T}\left(1-\frac{\ell}{T+1}\right)(\lambda(\ell)+\lambda(\ell)'),$$

where $\lambda(\ell)$ are the covariances matrix of the tth observation with the ℓth lag

$$\lambda(\ell) = \sum_{t=\ell+1}^{T}\mathbf{x}_t'\varepsilon_t\varepsilon_{t-\ell}'\mathbf{x}_{t-\ell},$$

This makes $\lambda(0)$ the contemporary variance, which in this case is the meat of a cluster-robust covariance matrix where we cluster on time.

As $T\to\infty$, full consideration of Σ_T becomes unbearable and will contain many irrelevant lags that add little-to-no information and probably some excess noise because there aren't as many lags of that length to average over.

To avoid this we can exploit the diminishing nature of the dependency (i.e., the mixing component) to get

$$\widehat{\Sigma}_T(L) = \hat{\lambda}(0) + \sum_{\ell=1}^{L}\left(1-\frac{\ell}{L+1}\right)\left(\hat{\lambda}(\ell)+\hat{\lambda}(\ell)'\right).$$

However, we now have to choose a maximum lag value have to choose L and we should choose L such that it increases with T, for example $T^{1/4}$ is a frequent default and not a bad starting point, other more thoughtful options exist.

The whole covariance estimator is then

$$\widehat{\text{avar}}(\hat{\theta};L) = [\mathbf{X}'\mathbf{X}]^{-1}\widehat{\Sigma}_T(L)[\mathbf{X}'\mathbf{X}]^{-1},$$

note that when $L=0$, this simplifies into a covariance matrix that is clustered by time. Similar analysis will demonstrate this for the within and RE estimators. This particular variance matrix is sometimes called the Driscoll-Kraay covariance matrix after their 1998 article. Note that because the baseline matrix clusters on time, that it allows for arbitrary correlation cross-sectionally (partially relaxing the assumption of iid units), while making the

most of the long-T time series within each unit.

Note that if you have large N and believe that the N units are iid, then you're probably better off with the clustered variance matrix above as it allows for arbitrary within-unit correlations, but this gives you something to do in the case where T is large and N is not.

If you are blessed enough to have both N and T going to infinity, you can stick with within-unit clustering, but you also have an option to use what are known as two-way clustered standard errors. Here we relax the assumption of iid units and suppose that errors are arbitrary correlated across time periods and within units. This still rules out correlation between observations it and js where $i \neq j$ & $t \neq s$.

Consider the estimated OLS variance

$$\widehat{\text{Var}}(\hat{\beta}) = (X'X)^{-1}\Omega(X'X)^{-1}$$

When we have within unit clustering

$$\begin{aligned}
\Omega &= \sum_{i=1}^{N} X_i'\hat{\varepsilon}_i\hat{\varepsilon}_i'X_i \\
&= X'\left(\hat{\varepsilon}\hat{\varepsilon}' \cdot \begin{bmatrix} 1_{T_1}1_{T_1}' & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 1_{T_N}1_{T_N}' \end{bmatrix}\right)X \\
&= X'(\hat{\varepsilon}\hat{\varepsilon}' \cdot S_N)X.
\end{aligned}$$

Here we use $\cdot$ to be element-by-element multiplication. The matrix S_N is block-diagonal where each block is a $T_i \times T_i$ matrix of 1s. So each ij element in S_N is 1 if observations i and j are in the same unit, where i and j are individual observations not units.

In a two-way case, each observation belongs to two groups and so we would want an equivalent matrix S_{NT} where the ij element is 1 if observations i and j if i and j are *either* in the same unit or the same time period. We can build such a matrix

$$S_{NT} = S_N + S_T - S_{N\cap T}.$$

This last term subtracts one from cases where i and j are in the same unit and the same time period (i.e., $i = j$). So let's plug this in

$$\Omega_2 = X'(\hat{\varepsilon}\hat{\varepsilon}' \cdot S_N)X + X'(\hat{\varepsilon}\hat{\varepsilon}' \cdot S_T)X - X'(\hat{\varepsilon}\hat{\varepsilon}' \cdot S_{N\cap T})X.$$

In our case, we said that $S_{N\cap T} = I_{NT}$ so this term becomes

$$X'(\hat{\varepsilon}\hat{\varepsilon}' \cdot S_{N\cap T})X = X'(I_{NT}(\hat{\varepsilon} \cdot \hat{\varepsilon}))X$$

This is a clustered matrix where each observation is a "cluster" this is otherwise known as the meat for the standard heteroskedasticity robust variance matrix (i.e., White or Huber-White) with the full matrix given as

$$\widehat{\text{avar}_0}(\hat{\theta}) = [\mathbf{X}'\mathbf{X}]^{-1}\left(\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{x_{it}}\mathbf{x_{it}}'\hat{\varepsilon}_{it}^2\right)[\mathbf{X}'\mathbf{X}]^{-1}.$$

This means that two-way clustering is fairly straight forward in the sense that we end up with

$$\widehat{\text{avar}_2}(\hat{\theta}) = \underbrace{\widehat{\text{avar}}(\hat{\theta};0)}_{\text{Clustered on time}} + \underbrace{\widehat{\text{avar}}(\hat{\theta})}_{\text{Clustered on unit}} - \underbrace{\widehat{\text{avar}_0}(\hat{\theta})}_{\text{Robust standard errors}}.$$

Here we have

1. Arbitrary correlation across space within each year $\widehat{\text{avar}}(\hat{\theta};0)$. requires large T because we're averaging the cross-sectional correlations over time.
2. Arbitrary correlation within units $\widehat{\text{avar}}(\hat{\theta})$. Requires large N because we're averaging the within-unit correlations over units
3. Remove the double counted observation-level heterogeneity $\text{avar}_0(\hat{\theta})$

This method can be extended to 3 or more dimensions (Cameron, Gelbach, and Miller 2011) and to persistent shocks within groups which allow for some correlation between observations it and js (Thompson 2011). One of your classmates will present more on choosing a clustering dimension.

Here we're looking at data from Meierrieks & Auer (2022) who study the effect of corruption on terrorism.

```
library(data.table)
library(fixest)
library(readstata13)

terror <- read.dta13("Rcode/corruption_terrorism/subsample.dta")

within1 <- feols( nattack ~ v2x_corr+sp_pop_totl+ ny_gdp_pcap_kd
                  +kg_democracy +statefailure|id,data=terror)
```

```
summary(within1)
```

```
## OLS estimation, Dep. Var.: nattack
## Observations: 6,837
## Fixed-effects: id: 170
## Standard-errors: Clustered (id)
##                 Estimate Std. Error   t value   Pr(>|t|)
## v2x_corr        0.386718   0.486018  0.795687 4.2733e-01
## sp_pop_totl     1.226421   0.252814  4.851086 2.7724e-06 ***
## ny_gdp_pcap_kd -0.072627   0.152929 -0.474910 6.3546e-01
## kg_democracy    0.130249   0.206882  0.629582 5.2982e-01
## statefailure    0.340789   0.051979  6.556298 6.4270e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 1.18963     Adj. R2: 0.586532
##                 Within R2: 0.148268
```

```
within2 <- feols( nattack ~ v2x_corr+sp_pop_totl+ ny_gdp_pcap_kd
                  +kg_democracy +statefailure|id+year,data=terror)
summary(within2)
```

```
## OLS estimation, Dep. Var.: nattack
## Observations: 6,837
## Fixed-effects: id: 170,  year: 48
## Standard-errors: Clustered (id)
##                Estimate Std. Error t value   Pr(>|t|)
## v2x_corr       0.856508   0.475351 1.80184 7.3352e-02 .
## sp_pop_totl    1.933857   0.392966 4.92118 2.0300e-06 ***
## ny_gdp_pcap_kd 0.418864   0.219191 1.91095 5.7703e-02 .
## kg_democracy   0.366543   0.196599 1.86442 6.3997e-02 .
## statefailure   0.322644   0.049112 6.56949 5.9896e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 1.11304     Adj. R2: 0.635483
##                 Within R2: 0.134426
```

```r
within2a <- feols( nattack ~ v2x_corr+sp_pop_totl+ ny_gdp_pcap_kd
                   +kg_democracy +statefailure|id+year,
                   cluster=~year,
                   data=terror)
summary(within2a)
```

```
## OLS estimation, Dep. Var.: nattack
## Observations: 6,837
## Fixed-effects: id: 170,  year: 48
## Standard-errors: Clustered (year)
##                Estimate Std. Error  t value   Pr(>|t|)
## v2x_corr       0.856508   0.153166  5.59202 1.1093e-06 ***
## sp_pop_totl    1.933857   0.099712 19.39444  < 2.2e-16 ***
## ny_gdp_pcap_kd 0.418864   0.074586  5.61588 1.0215e-06 ***
## kg_democracy   0.366543   0.066627  5.50139 1.5167e-06 ***
## statefailure   0.322644   0.019319 16.70112  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 1.11304     Adj. R2: 0.635483
##                 Within R2: 0.134426
```

```r
within2b <- feols( nattack ~ v2x_corr+sp_pop_totl+ ny_gdp_pcap_kd
                   +kg_democracy +statefailure|id+year,
                   vcov="DK",
                   panel.id=c("id", "year"),
                   data=terror)
summary(within2b)
```

```
## OLS estimation, Dep. Var.: nattack
## Observations: 6,837
## Fixed-effects: id: 170,  year: 48
## Standard-errors: Driscoll-Kraay (L=2)
##                Estimate Std. Error  t value   Pr(>|t|)
## v2x_corr       0.856508   0.205488  4.16816 1.3069e-04 ***
## sp_pop_totl    1.933857   0.147251 13.13310  < 2.2e-16 ***
## ny_gdp_pcap_kd 0.418864   0.115101  3.63909 6.7883e-04 ***
## kg_democracy   0.366543   0.089222  4.10823 1.5821e-04 ***
```

```
## statefailure   0.322644   0.027830 11.59334 2.1961e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 1.11304     Adj. R2: 0.635483
##                 Within R2: 0.134426
```

```
within2c <- feols( nattack ~ v2x_corr+sp_pop_totl+ ny_gdp_pcap_kd
                   +kg_democracy +statefailure|id+year,
                   vcov="twoway",
                   data=terror)
summary(within2c)
```

```
## OLS estimation, Dep. Var.: nattack
## Observations: 6,837
## Fixed-effects: id: 170,  year: 48
## Standard-errors: Clustered (id & year)
##                Estimate Std. Error t value   Pr(>|t|)
## v2x_corr       0.856508   0.471167 1.81785 7.5465e-02 .
## sp_pop_totl    1.933857   0.389793 4.96125 9.5888e-06 ***
## ny_gdp_pcap_kd 0.418864   0.222266 1.88452 6.5688e-02 .
## kg_democracy   0.366543   0.192196 1.90713 6.2626e-02 .
## statefailure   0.322644   0.049099 6.57126 3.6589e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 1.11304     Adj. R2: 0.635483
##                 Within R2: 0.134426
```

```
sqrt(diag(vcov(within2) + vcov(within2a)
          - vcov(update(within2, vcov="hetero"))))
```

```
##       v2x_corr    sp_pop_totl ny_gdp_pcap_kd   kg_democracy   statefailure
##     0.46944915     0.38826932     0.22156593     0.19146068     0.04894522
```

2 Classically advanced topics and moment estimators

In this chapter we're going to cover the topics that econometrics books list as the advanced panel models. These include attrition/sample selection, dynamic models (e.g., lagged variables), and how to consider time-invariant factors without sacrificing the credibility benefits

of the fixed effects models. This section will be more applied than the last as we will focus on specific implementation issues for the issues that arise.

2.1 Instrumental variables (refresher and update)

Before proceeding we will spend a little time refreshing ourselves on instrumental variables as they are key to many of the following techniques. Recall that we use the method of instrumental variable when we are concerned that the treatment of interest is endogenous (i.e., $\mathrm{E}[\varepsilon_{it}x_{it}] \neq 0$). This can occur for any number of reasons, including

1. Omitted variables that are correlated with x_{it} and y_{it}
2. Measurement error in x_{it}
3. Feedback/dynamics
4. Attrition (i.e., exiting the panel early)

One way to work around this endogeneity is to use 1 or more instruments z_{it} such that z is exogenous, relevant, and not redundant. In lay terms this means that:

1. The instrument is correlated with the treatment (relevance)
2. The instrument *only* influences values of the outcome through its effect on the treatment (exogeneity/validity). In other words, it is not in the structural equation and is uncorrelated with any omitted variables itself (conditional on the observables).

Note that the fixed effects models buy us some insulation from endogeneity concerns. Specifically, they allow for unobserved heterogeneity that is correlated with the treatment to exist so long as it is time invariant. This means that any ommitted variables that are time invariant are not a concern because they are swept away by the within-transformation.

For working in the IV framework we'll list our assumptions, so that we can be more clear about the above.

Assumption B1 *The outcome y_{it} is linear-in-the-parameters such that*

$$y_{it} = \beta' x_{it} + \alpha_i + \varepsilon_{it}.$$

Move to two-way heterogeneity doesn't change much of this or the following, so we'll keep it easy.

Assumption B2 *The units $(x_i, z_i, \varepsilon_i)$ are iid*

Assumption B3 *The instruments are strictly exogeneous within units* $\mathrm{E}[\varepsilon_{it}|z_i] = 0$

Assumption B4 *The instruments are not redundant:* $\mathrm{E}[z_i' M_i z_i]$ *exists and has full rank*

Assumption B5 *The instruments are relevant:* $rank(\mathrm{E}[z_i' M_i x_i]) \geq dim(x_{it})$

Assumption B6 *Additional moment assumptions*

When the above assumptions are met, we can use the two-stage-least squares (2SLS) with the within-transformed data. Let $\dot{Z} = MZ$ and $\dot{Z}_i = M_i Z_i$, then the 2SLS estimator is such that

$$\hat{\beta}_{2SLS} = [\dot{X}'\dot{Z}(\dot{Z}'\dot{Z})^{-1}\dot{Z}'\dot{X}]^{-1}(\dot{X}'\dot{Z}(\dot{Z}'\dot{Z})^{-1}\dot{Z}'\dot{y})$$

In the case where $\dim(z_i) = \dim(x_i)$ this simplifies to

$$\hat{\beta}_{2SLS} = [\dot{Z}'\dot{X}]^{-1}\dot{Z}'\dot{y}$$

Of note in the above is that we will also want to considering the within estimator in the first stage. That is to say we consider the first-stage reduced form equations

$$\dot{X} = \dot{Z}\Gamma + \dot{\nu}.$$

We can then consider the first-stage tests (i.e., the first stage F-statistic) using this regression. Note that the appropriate covariance matrix for Γ, under typical assumptions, would be the clustered variance matrix.

Because nothing actually changes as we move from cross-sectional IV to panel IV, we can import some results that we know.

1. $\hat{\beta}_{2SLS}$ is biased, but consistent (in N or T)
2. $\hat{\beta}_{2SLS}$ is asymptotically normal
3. As $N \to \infty$
$$\sqrt{N}(\hat{\beta}_{2SLS} - \beta) \xrightarrow{d} N\left(0, V_{2sls}\right).$$
Let $Q_Z = \mathrm{E}[\dot{Z}_i'\dot{Z}_i]$ and $Q_X = \mathrm{E}[\dot{Z}_i'\dot{X}_i]$, then the variance becomes
$$V_{2sls} = (Q_X' Q_Z^{-1} Q_X)^{-1}(Q_X' Q_Z^{-1}\,\mathrm{E}[\dot{Z}_i{}'\varepsilon_i\varepsilon_i'\dot{Z}_i]Q_Z^{-1}Q_X)(Q_X' Q_Z^{-1} Q_X)^{-1}.$$
All can be estimated using its standard sample counterparts. Generally, you would not write these out yourself. The `feols` package is built to handle instruments, or you can use `ivreg` with your own within transformations.

Note that the 2SLS estimator also gives us another way to motivate the Mundlak estimator. The problem with pooled OLS is, as you recall, the omitted variable bias from the unobserved unit-level heterogeneity. The within-transformed variables $\dot{x}_{it}$ are in fact excellent instruments for x_{it} in that they are correlated with x and exogeneous with respect to the omitted time-invariant variables. This gives us the equations

$$\begin{aligned} y_{it} &= \beta' x_{it} + \alpha_i + \varepsilon_{it} \\ &= \beta' x_{it} + e_{it} \\ x_{it} &= \Gamma \dot{x}_{it} + u_{it} \end{aligned}$$

where α_i is unobserved. This creates the joint error term $e_{it} = \alpha_i + \varepsilon_{it}$. The variables are endogeneous to the extent that they are correlated with the unobserved α_i. We can express this as a correlation between u_{it} and e_{it} which we can write as

$$e_{it} = \rho u_{it} + \nu_{it}$$

We can then substitute this into the first equation to get

$$y_{it} = \beta' x_{it} + \rho u_{it} + \nu_{it},$$

where x_{it} is exogeneous conditional on u_{it}. Unfortunately we don't observe u directly, but we do observe x and $\dot{x}$, let's plug those in

$$y_{it} = \beta' x_{it} + \rho'(x_{it} - \Gamma \dot{x}_{it}) + \nu_{it}.$$

Of note is that Γ in this case will be an identity matrix (I may have you show this in a problem set), so we now get

$$\begin{aligned} y_{it} &= \beta' x_{it} + \rho(x_{it} - x_{it} + \bar{x}_i) + \nu_{it} \\ &= \beta' x_{it} + \rho' \bar{x}_i + \nu_{it} \end{aligned}$$

which is of course the Mundlak estimator. Of importance here is that the variables are now fully exogeneous of the new error term.

Before moving on, there will be times in panels where one dimension is large and you need to do something yourself. In these cases sparse matrices tools can be your friends. A sparse matrix is one that is dominated by zeros. These zeros take up memory and can slow computation despite the face that they are canceling things left and right. Sparse matrices

avoid these problems by only actually saving the non-zero elements and their coordinates in memory and working around the zeros.

However, for many small or medium sized problems, deploying sparse matrices can make the problem slower. But for larger problems it can make a huge difference. When to make the switch depends on the problem size and the amount of memory at your disposal.

The main R package for sparse matrices is `Matrix`. It works reasonably well for 99% of things, but if you're still having issues, Matlab is the king of matrix computation and their sparse tools are hard to beat. The combination of `numpy` and `scipy` for python is very good and tends to be my go to for very complex problems.

2.1.1 Application

```
library(data.table)
library(readstata13)
library(fixest)
library(ivreg)
library(sandwich)
library(lmtest)
library(car)
library(Matrix) #sparse matrices

terror <- read.dta13("Rcode/corruption_terrorism/subsample.dta")

terror <- data.table(terror)

## lead the outcome by one (lag everything else)
terror[, f.nattack := shift(nattack, -1), by=id]

baseline.ols <- feols(f.nattack~v2x_corr+sp_pop_totl+ny_gdp_pcap_kd+
                        kg_democracy+statefailure|id+year, data=terror)

within.2sls <- feols(f.nattack~sp_pop_totl+ny_gdp_pcap_kd+
                       kg_democracy+statefailure|id+year|
                       v2x_corr~iv_region, data=terror)
summary(within.2sls, stage=1:2)
```

```
## IV: First stage: v2x_corr
## TSLS estimation - Dep. Var.: v2x_corr
##                   Endo.    : v2x_corr
##                   Instr.   : iv_region
## First stage: Dep. Var.: v2x_corr
## Observations: 6,561
## Fixed-effects: id: 167,  year: 47
## Standard-errors: Clustered (id)
##                 Estimate Std. Error   t value   Pr(>|t|)
## iv_region       0.518865   0.147550  3.516536 0.00056392 ***
## sp_pop_totl     0.055358   0.043617  1.269182 0.20615276
## ny_gdp_pcap_kd -0.050708   0.021659 -2.341162 0.02041189 *
## kg_democracy   -0.082769   0.026242 -3.154015 0.00191193 **
## statefailure   -0.000652   0.002544 -0.256356 0.79799330
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 0.078755     Adj. R2: 0.928392
##                  Within R2: 0.141782
## F-test (1st stage): stat = 515.8, p < 2.2e-16, on 1 and 6,509 DoF.
##
## IV: Second stage
## TSLS estimation - Dep. Var.: f.nattack
##                   Endo.    : v2x_corr
##                   Instr.   : iv_region
## Second stage: Dep. Var.: f.nattack
## Observations: 6,561
## Fixed-effects: id: 167,  year: 47
## Standard-errors: Clustered (id)
##                Estimate Std. Error t value   Pr(>|t|)
## fit_v2x_corr   7.496943   2.452749 3.05655 2.6100e-03 **
## sp_pop_totl    1.533674   0.467091 3.28346 1.2505e-03 **
## ny_gdp_pcap_kd 0.810152   0.303067 2.67318 8.2639e-03 **
## kg_democracy   0.770970   0.358895 2.14818 3.3150e-02 *
## statefailure   0.291289   0.050492 5.76906 3.8099e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## RMSE: 1.23553     Adj. R2:  0.548268
##                 Within R2: -0.0715
## F-test (1st stage), v2x_corr: stat = 515.8, p < 2.2e-16, on 1 and 6,509 DoF.
##                   Wu-Hausman: stat = 115.4, p < 2.2e-16, on 1 and 6,342 DoF.
```

```
terror2 <- terror[as.numeric(within.2sls$obs_selection$obsRemoved)]
length(unique(terror2$id))
```

```
## [1] 167
```

```
summary(terror2[,length(year), by=id]$V1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00   28.00   47.00   39.29   47.00   47.00
```

```
## Sparse functions
DeltaN <- sparse.model.matrix(~factor(id)-1, data=terror2)
DeltaT <- sparse.model.matrix(~factor(year)-1, data=terror2)[,-1]
M <- Diagonal(nrow(terror2)) - DeltaN %*% solve(crossprod(DeltaN)) %*% t(DeltaN)


## build two-way transformation
M2 <- M - M %*% DeltaT  %*% solve(t(DeltaT) %*% M %*% DeltaT) %*% t(DeltaT) %*% M

var.names<- c("f.nattack", "v2x_corr", "sp_pop_totl",
              "ny_gdp_pcap_kd", "kg_democracy", "statefailure", "iv_region")
terror2[ , paste0(var.names, ".within") := lapply(.SD, \(x){as.numeric(M2 %*%x)}),
         .SDcols=var.names ]


within.2sls2 <- ivreg(f.nattack.within~v2x_corr.within+
                        sp_pop_totl.within+ny_gdp_pcap_kd.within+
                        kg_democracy.within+statefailure.within-1|
                        iv_region.within+
                        sp_pop_totl.within+ny_gdp_pcap_kd.within+
                        kg_democracy.within+statefailure.within-1,
                      data=terror2)

within.2sls2 <- ivreg(f.nattack.within~v2x_corr.within+
```

```
                        sp_pop_totl.within+ny_gdp_pcap_kd.within+
                        kg_democracy.within+statefailure.within-1|
                        iv_region.within+
                        sp_pop_totl.within+ny_gdp_pcap_kd.within+
                        kg_democracy.within+statefailure.within-1,
                      data=terror2)
summary(within.2sls2, vcov=\(x){vcovCL(x,cluster=terror2$id)})
```

```
##
## Call:
## ivreg(formula = f.nattack.within ~ v2x_corr.within + sp_pop_totl.within +
##     ny_gdp_pcap_kd.within + kg_democracy.within + statefailure.within -
##     1 | iv_region.within + sp_pop_totl.within + ny_gdp_pcap_kd.within +
##     kg_democracy.within + statefailure.within - 1, data = terror2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5.79047 -0.74103 -0.05355  0.65796  5.20596
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## v2x_corr.within        7.49694    2.44320   3.068 0.002160 **
## sp_pop_totl.within     1.53367    0.46527   3.296 0.000985 ***
## ny_gdp_pcap_kd.within  0.81015    0.30189   2.684 0.007301 **
## kg_democracy.within    0.77097    0.35750   2.157 0.031075 *
## statefailure.within    0.29129    0.05029   5.792 7.29e-09 ***
##
## Diagnostic tests:
##                   df1  df2 statistic p-value
## Weak instruments    1 6556    12.455 0.00042 ***
## Wu-Hausman          1 6555     9.508 0.00205 **
## Sargan              0   NA        NA      NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.236 on 6556 degrees of freedom
```

```
## Multiple R-Squared: -0.0715, Adjusted R-squared: -0.07232
## Wald test: 17.36 on 5 and 6556 DF,  p-value: < 2.2e-16
```

```
within.first <- lm(v2x_corr.within~
                     iv_region.within+
                     sp_pop_totl.within+ny_gdp_pcap_kd.within+
                     kg_democracy.within+statefailure.within-1,
                   data=terror2)
coeftest(within.first, vcov=vcovCL(within.first, terror2$id))
```

```
##
## t test of coefficients:
##
##                           Estimate  Std. Error t value  Pr(>|t|)
## iv_region.within       0.51886530  0.14702026  3.5292 0.0004197 ***
## sp_pop_totl.within     0.05535764  0.04346016  1.2738 0.2027951
## ny_gdp_pcap_kd.within -0.05070781  0.02158148 -2.3496 0.0188232 *
## kg_democracy.within   -0.08276893  0.02614817 -3.1654 0.0015559 **
## statefailure.within   -0.00065216  0.00253482 -0.2573 0.7969707
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
linearHypothesis(within.first, "iv_region.within",
                 vcov=vcovCL(within.first, terror2$id))
```

```
## Linear hypothesis test
##
## Hypothesis:
## iv_region.within = 0
##
## Model 1: restricted model
## Model 2: v2x_corr.within ~ iv_region.within + sp_pop_totl.within + ny_gdp_pcap_kd.wit
##     kg_democracy.within + statefailure.within - 1
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1   6557
```

```
## 2   6556  1 12.455 0.0004197 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So where does that wacky first-stage F come from in `feols`? Well, to track that down, I did the following:

1. I searched the output of `fixest:::print.fixest` because we know it appears when we print the object. Note that there are 3 colons now, because I know it exists in there but they don't actively export it. Here I find that they call a function called `fitstat` with an argument `ivf1` to get the first stage
2. Next I look at `fixest::fitstat` this only needed two colons because it is visible. Here I find the following code associated with `ivf1`

```
if (root == "ivf1") {
  if (isTRUE(x$iv)) {
    df1 = degrees_freedom(x, vars = x$iv_inst_names_xpd,
                          stage = 1)
    df2 = degrees_freedom(x, "resid", stage = 1)
    if (x$iv_stage == 1) {
      stat = ((x$ssr_no_inst - x$ssr)/df1)/(x$ssr/df2)
      p = pf(stat, df1, df2, lower.tail = FALSE)
      vec = list(stat = stat, p = p, df1 = df1,
                 df2 = df2)
      res_all[[type]] = set_value(vec, value)
    }
    else {
      x_first = x$iv_first_stage
      for (endo in names(x_first)) {
        stat = ((x_first[[endo]]$ssr_no_inst -
                   x_first[[endo]]$ssr)/df1)/(x_first[[endo]]$ssr/df2)
        p = pf(stat, df1, df2, lower.tail = FALSE)
        vec = list(stat = stat, p = p, df1 = df1,
                   df2 = df2)
        res_all[[paste0(type, "::", endo)]] = set_value(vec,
                                                        value)
      }
```

```
    }
  }
  else {
    res_all[[type]] = NA
  }
}
```

This is something we can work with

```
x <- within.2sls
isTRUE(x$iv)
```

```
## [1] TRUE
```

```
df1 = degrees_freedom(x, vars = x$iv_inst_names_xpd,
                      stage = 1)
df2 = degrees_freedom(x, "resid", stage = 1)
print(c(df1, df2)) ## number of IVs and N-K for the first stage
```

```
## [1]    1 6509
```

```
x$iv_stage == 1
```

```
## [1] FALSE
```

```
x_first = x$iv_first_stage
names(x_first)
```

```
## [1] "v2x_corr"
```

```
endo <- names(x_first)
stat = ((x_first[[endo]]$ssr_no_inst -
           x_first[[endo]]$ssr)/df1)/(x_first[[endo]]$ssr/df2)
stat
```

```
## [1] 515.8203
```

```
## This is it so what are we looking at?



ssrR <- sum(feols(v2x_corr~sp_pop_totl + ny_gdp_pcap_kd+kg_democracy+statefailure|id+yea
```

```
                  data=terror2)$resid^2)
print(c(ssrR, x_first[[endo]]$ssr_no_ins))
```

```
## [1] 43.91802 43.91802
```

```
ssrU <- sum(feols(v2x_corr~iv_region+sp_pop_totl + ny_gdp_pcap_kd+kg_democracy+statefail
                  data=terror2)$resid^2)
print(c(ssrU, x_first[[endo]]$ssr))
```

```
## [1] 40.6932 40.6932
```

Ok now we're getting somewhere.

Recall that with classical variance matricies we can form the F statistic for comparing nested models as

$$F = \frac{(R_U^2 - R_R^2)/(k-\ell)}{(1-R_U^2)/(N-k)} \sim F(k-\ell, N-k)$$

It looks like they're going for something like that

```
SST <- sum( (terror2$v2x_corr.within)^2)## R squared w/o an overall constant drops the
(((1-ssrU/SST) - (1-ssrR/SST))/df1)  / (((ssrU/SST))/df2)
```

```
## [1] 515.8203
```

```
((ssrR-ssrU)/df1) / (ssrU/df2)
```

```
## [1] 515.8203
```

```
stat
```

```
## [1] 515.8203
```

```
## Spot on. It is just them using classical variance matrix on the first stage
```

It will also behoove us to refresh our memory on alternatives to 2SLS for some of the topics we're going to cover. Specifically, generalized method of moments (GMM) estimators are common in this literature. Recall that moments are specific characteristics of random variables. The method of moments (MoM) works by equating sample (empirical) moments with theoretical moments and then solving for the the parameter of interest. GMM generalizes this to situations where we have/want to use more empirical moments than theoretical ones.

To recap, the original method moments consider data $x_1, \ldots, x_N$ that we believe to have come from a uniform distribution. Further suppose that we know that the lower bound of

this distribution is 0 but we don't know the upper limit. So we have that $x_1, \ldots, x_N$ are iid $U(0, \theta)$ where we want to estimate θ.

We have 1 parameter so we need one moment. In this case we take the first empirical moment $\bar{x} = \sum_{i=1}^{N} x_i$ and relate it to the theoretical moment given by $E[X]$, which for the uniform is $\mathrm{E}[X] = \frac{1}{2}(\theta + 0)$. As such we set these equal:

$$\begin{aligned} \bar{x} &= \frac{1}{2}(\theta) \\ 2\bar{x} &= \hat{\theta}_{\mathrm{MoM}}. \end{aligned}$$

So the MoM estimator is twice the sample mean.

If we needed to estimate both ends (i.e, $X \sim U(\theta_1, \theta_2)$) then we would need the first 2 moments

$$\begin{aligned} N^{-1} \sum_{i=1}^{N} x_i &= \bar{x} = \frac{\theta_1 + \theta_2}{2} = E[X] \\ N^{-1} \sum_{i=1}^{N} x_i^2 &= \overline{x^2} = \frac{\theta_1^2 + \theta_1\theta_2 + \theta_2^2}{3} = E[X^2]. \end{aligned}$$

Solving these for the parameters we get

$$\begin{aligned} \hat{\theta}_1 &= \bar{x} - \sqrt{3\left(\overline{x^2} - \bar{x}^2\right)} \\ \hat{\theta}_2 &= \bar{x} + \sqrt{3\left(\overline{x^2} - \bar{x}^2\right)} \end{aligned}$$

for example

```
set.seed(1)
X <- runif(500, -2, 5)
barx <- mean(X)
barx2 <- mean(X^2)
theta1.hat <- barx - sqrt(3 *(barx2 - barx^2))
theta2.hat <- barx + sqrt(3 *(barx2 - barx^2))
c(theta1.hat, theta2.hat)
```

```
## [1] -1.962062  4.901231
```

For cross-sectional linear model, the theoretical and empirical moments of interest are

$$\frac{1}{N}\sum_i x_i(y_i - \beta' x_i) = \mathrm{E}[x_i \varepsilon_i]$$
$$\frac{1}{N}\sum_i x_i(y_i - \beta' x_i) = 0$$
$$\hat{\beta} = \left(\frac{1}{N}\sum_i x_i x_i'\right)^{-1} \frac{1}{N}\sum_i x_i y_i,$$

which of course means that the OLS estimator is also MoM estimator for the basic linear model.

The GMM case formalizes this procedure a little bit and extends it. Let $g(x_i, y_i; \theta)$ be a function that takes in data (x_i, y_i) and a guess at the parameters θ, and then and measures how close the population moments to the sample moments are for this guess of θ. In the population model, we will have

$$\mathrm{E}[g(x_i, y_i; \theta)] = 0$$

only if we guess the true θ. The model is identified if there is a unique mapping from g to θ, which in this case means that there is a unique solution to the above equation. The model is **just identified** if we have $\dim(\beta) = \dim(g(x_i, y_i; \theta))$ equations in g and **over identified** if we have more equations than unknowns $\dim(\beta) < \dim(g(x_i, y_i; \theta))$. In regular method of moments we only deal with just-identifed cases, in the GMM case we can have either just or over identifed models $\dim(\beta) \leq \dim(g(x_i, y_i; \theta))$.

Going back to our examples. For the uniform we have

$$g(x_i; \theta) = \begin{bmatrix} x_i - \frac{\theta_1 + \theta_2}{2} \\ x_i^2 - \frac{\theta_1^2 + \theta_1\theta_2 + \theta_2^2}{3}, \end{bmatrix}$$

which has 2 parameters and 2 equations. For the pooled panel model we have

$$g(y_{it}, \mathbf{x}_{it}; \theta) = \mathbf{x}_{it}(y_{it} - \theta' \mathbf{x}_{it}),$$

which is also a just identified case.

Moving back to the IV framework, we find a situation where GMM can offer us something interesting over least-squares approaches. The reason for this is that in the IV framework we can be over identified. That is to say we have more estimating equations (i.e., more instruments) than parameters.

Returning to the within case recall that the identifying conditions are

$$\mathrm{E}[\dot{z}_i \varepsilon_i] = 0$$

such that

$$g(x_i, z_i; \beta) = \dot{z}_i'(\dot{y}_i - \beta' \dot{x}_i)$$

In the just identified case we get the 2SLS estimator, but in the over identified case (more instruments than endogenous variables) we cannot simply solve sample moment conditions

$$\frac{1}{N} \sum_{i=1}^{N} \dot{z}_i'(\dot{y}_i - \beta' \dot{x}_i) = 0,$$

as $\dot{z}_i''$ is $\ell \times T_i$ and $\dot{y}_i - \beta' \dot{x}_i$ is $T_i \times k$ and so we end up with ℓ equations and k unknowns with $\ell > k$.

With this conundrum in mind, we introduce can introduce a concept of distance. How close can we get these systems of equations to 0? This is a minimum distance problem, and so we need to define our distance criteria. Perhaps by using least squares. Such that we want to minimize the "error" in the equation

$$\begin{aligned} \eta &= \dot{Z}'\dot{y} - \dot{Z}'\dot{X}\beta \\ \dot{Z}'\dot{y} &= \dot{Z}'\dot{X}\beta + \eta \\ \ddot{y} &= \ddot{X}\beta + \eta. \end{aligned}$$

This looks like a regression equation! So the minimum squared error estimator will be

$$\begin{aligned} \hat{\beta} &= (\ddot{X}'\ddot{X})^{-1}\ddot{X}'\ddot{y} \\ &= (\dot{X}'\dot{Z}\dot{Z}'\dot{X})^{-1}\dot{X}'\dot{Z}\dot{Z}'\dot{y} \end{aligned}$$

This is a start and will minimize $\eta'\eta$, but we also know that we can make this more efficient by weighting if η is not spherical. The GMM estimator is then

$$\begin{aligned} \hat{\beta}_{GMM} &= (\ddot{X}'W\ddot{X})^{-1}\ddot{X}'W\ddot{y} \\ &= (\dot{X}'\dot{Z}W\dot{Z}'\dot{X})^{-1}\dot{X}'\dot{Z}W\dot{Z}'\dot{y}. \end{aligned}$$

More generally, the GMM estimator of some parameters θ is given by

$$\hat{\theta}_{GMM} = \underset{\theta}{\operatorname{argmin}}\, N \left[\frac{1}{N}\sum_i g(y_i;\theta)\right]' W \left[\frac{1}{N}\sum_i g(y_i;\theta)\right].$$

We are now left with the choice of W. The good news here is that it doesn't matter too much. So long as W is positive definite, the GMM estimator will be consistent and asymptotically normal for its model. As such $W = I$ is typically a fine choice for starting out, however it will rarely be the best choice.

There are few options to consider with GMM, the first is to use a "one-step" estimator where W is fixed to a specific value like I. In the above example, if $W = (\dot{Z}'\dot{Z})^{-1}$ when the GMM is identical to the over-identified 2SLS.

It turns out that the most efficient GMM results from $W^* = \text{Var}(\dot{Z}'\varepsilon)^{-1}$. With this in mind, what does that tell us about the 2SLS here and its relative efficency to the GMM? Basically, 2SLS will only be the most efficient if

$$\text{Var}(\dot{Z}'\varepsilon)^{-1} \propto (\dot{Z}'\dot{Z})^{-1},$$

which will be the case when we have iid and homoskedastic errors.

In cases where we do not believe that (i.e., most panel settings) we can try to estimate the efficient weighting matrix in a two-step setting

$$\begin{aligned}\hat{W}^{*-1} &= \frac{1}{N}\sum_{i=1}^{N} g_i(\hat{\beta}_1)g_i(\hat{\beta}_1)' \\ &= \frac{1}{N}\sum_{i=1}^{N} \dot{z}_i'\hat{\varepsilon}_i\hat{\varepsilon}_i'\dot{z}_i\end{aligned}$$

Where $\hat{\beta}_1$ is a consistent first-stage estimate of β that produce consistent estimates of $\hat{\varepsilon}_i$. Natural candidates include either GMM with $W = I$ or 2SLS. Note that $\hat{W}$ here is "meat" of the cluster-robust covariance matrix. This will generally be the case for the efficient GMM in the linear model.

To estimate the clustered standard errors for the panel GMM models with weights W we have

$$\text{avar}(\hat{\beta}_{GMM}) = (\ddot{X}'W\ddot{X})^{-1}\left(\ddot{X}'W\left[\frac{1}{N}\sum_{i=1}^{N}\dot{z}_i\hat{\varepsilon}_i\hat{\varepsilon}_i'\dot{z}_i\right]W\ddot{X}\right)(\ddot{X}'W\ddot{X})^{-1}$$

Although, in this case you would probably use the efficient

$$W^* = \left[\frac{1}{N}\sum_{i=1}^{N}\dot{z}_i'\hat{\varepsilon}_i\hat{\varepsilon}_i'\dot{z}_i\right]^{-1}$$

so this becomes

$$\begin{aligned}\text{avar}(\hat{\beta}_{GMM}) &= (\ddot{X}'W\ddot{X})^{-1}(\ddot{X}'W^*W^{*-1}WW^*\ddot{X})(\ddot{X}'W\ddot{X})^{-1} \\ &= (\ddot{X}'W^*\ddot{X})^{-1}(\ddot{X}'W^*\ddot{X})(\ddot{X}'W\ddot{X})^{-1} \\ &= (\ddot{X}'W^*\ddot{X})^{-1}\end{aligned}$$

The last topic I want want to cover in GMM is how Sargan's (1958) applies to the over identified GMM. Recall that Sargan's test relies on the face that in the over-identified case it is unlikely that we will find estimates $\hat{\beta}$ that perfectly satisfy the moment conditions that $\mathrm{E}[\dot{Z}_i\varepsilon_i] = 0$. As such we can treat that as a hypothesis. We can use the moment conditions to form the test statistic as in

$$\begin{aligned}H_0 :& E[\dot{Z}_i\varepsilon_i] = 0 \\ \bar{g}(\hat{\beta}) &= \frac{1}{N}\sum_{i=1}^{N}\dot{z}_i'\hat{\varepsilon} \\ J &= \bar{g}(\hat{\beta})'W^*\bar{g}(\hat{\beta}) \\ J &\xrightarrow{d} \chi^2_{\ell-k}.\end{aligned}$$

This is test of the exogeneity of the instruments. Rejecting this null means that we have evidence against the exogeneity of the instruments. Note that we need W^* here which is the efficient weighting matrix.

2.2 Invariant-regressors

At some point you may find yourself in a pickle where you want to consider the effect of a time-invariant variable but you do not want to lose the benefits of a fixed effects estimator. As we know, the within and LSDV estimators remove any time-invariant characteristics. While we tend to think of this as a net positive, there may be times where you actually want to know something about a time-invariant trait.

One option may be to use a CRE estimator and just include the covariate. However, it's not clear that this is the best approach. Here we'll consider the model

$$y_{it} = x_{it}'\beta + z_i'\gamma + \alpha_i + \varepsilon_{it}$$

. We maintain the basic panel assumptions, most importantly strict exogeneity within units, which we can now write as

$$\mathrm{E}[\mathbf{x}_{it}\varepsilon_{is}] = 0, \;\; \forall i \in \{1\ldots,N\} \;\&\; (s,t) \in \{1,\ldots,T\}^2$$

We all suppose that z_i is uncorrelated with the individual specific intercepts, i.e., $\mathrm{E}[z_i\alpha_i] = 0$, while we leave x_{it} unrestricted in this sense. This means that we have a model where z_i is exogenous wrt to both ε and α, while x_{it} is only exogeneous wrt to ε. We leave α_i as unobserved.

To consistently estimate β we would typically need to use a fixed-effects estimator. However, applying the within transformation would remove $z_i'\gamma$ and in this context we also want to know something about γ. To work around this, we will consider an IV approach. For the instruments to be valid we need them to be uncorrelated with $\alpha_i + \varepsilon_{it}$. We already assumed that for z_i, so they can instrument for themselves. For x_{it}, what if we used the within transformed variables $\dot{x}_{it}$?
The transformation removes the relationship wih u_i and so would be valid!

The moment conditions then become

$$\mathrm{E}[z_i(y_i - \beta' x_i - \gamma' z_i)] = 0$$
$$\mathrm{E}[\dot{x}_i(y_i - \beta' x_i - \gamma' z_i)] = 0$$

In this case we have just as many instruments as endogeneous variables and so it doesn't matter if we use 2SLS or GMM.

The above approach is algebraically equivalent to another two-step method where:

1. Estimate *beta* using the within estimator and compute the estimated unit constants

$$\hat{\alpha}_i = \bar{y}_i - \bar{x}_i\hat{\beta}_w.$$

   Note that these residuals "contain" the omitted variables z_i.
2. Estimate γ by regressing $\hat{\alpha}_i$ on z_i.

This equivalence follows from the above moment conditions. In sample these are

$$Z'(y - X\beta - Z\gamma) = 0$$
$$\dot{X}'(y - X\beta - Z\gamma) = 0$$

Recall that the within transformation removes all cross-sectional variance so the covariance of $\dot{X}$ and Z is 0, making $\dot{X}'Z = 0$. The bottom line is then

$$\dot{X}'(y - X\beta) = 0$$

The value of β that solves this? The within estimates! We saw this before when we used

these instruments to motivate the Mundlak estimator. Plug those into the first set of line and we get:

$$\begin{aligned}\left[z_i'(y_i - x_i'\hat{\beta}_w - z_i'\gamma)\right]_{i=1}^N &= \left[z_i'(\bar{y}_i - \bar{x}_i'\hat{\beta}_w - z_i'\gamma)\right]_{i=1}^N \\ &= Z'(\hat{\alpha} - Z\gamma),\end{aligned}$$

which is of course the second step in that two-step routine just described.

In practice, you would not, of course, do the two-step when the one-step version with 2SLS exists. I include it here for you to see the intuition of this estimator, which is that we consider $\hat{\alpha}_i$ to contain the information on $z_i'\gamma$ even when z_i is uncorrelated with the true α_i. To extract that information use this $\hat{\alpha}_i$ as dependent variable.

The main restriction in this model is the assumption that the invariant variables z are uncorrelated with the true α_i. While largely a theoretical question, it seems unlikely to me in probably most interesting cases. Hausman and Taylor generalize the above model to the following case

$$y_{it} = \beta_1'x_{1it} + \beta_2'x_{2it} + \gamma_1'z_{1i} + \gamma_2'z_{2i} + \alpha_i + \varepsilon_{it},$$

where

- x_{1it} contains k_1 time-varying exogenous variables $\mathrm{E}[x_{1it}\alpha_i] = 0$
- z_{1i} contains ℓ_1 time-invariant exogenous variables $\mathrm{E}[z_{1i}\alpha_i] = 0$
- x_{2it} contains k_2 time-varying endogenous variables $\mathrm{E}[x_{2it}\alpha_i] \neq 0$
- z_{2i} contains ℓ_2 time-invariant endogenous variables $\mathrm{E}[z_{2i}\alpha_i] \neq 0$

Our goals is to estimate $\theta = (\beta, \gamma) = (\beta_1, \beta_2, \gamma_1, \gamma_2)$. As before we will consider the use of instruments to help us identify these parameters. The within transformed variables $\dot{x}_{2it}$ will once against be great choices, and z_{1i} can instrument for itself. In theory x_{1it} could instrument for itself, but let's go ahead and include $\dot{x}_{1it}$ to be safe. This leave the choice for z_2. Hausman and Taylor propose using $\bar{x}_{1i}$ as these should at least be exogenous.

Collect the regressors in $\mathbf{x}_i t$, then we have the following moment conditions

$$\begin{aligned}\mathrm{E}[\dot{x}_{1it}'(y_{it} - \theta'\mathbf{x}_i t)] &= 0 \\ \mathrm{E}[\dot{x}_{2it}'(y_{it} - \theta'\mathbf{x}_i t)] &= 0 \\ \mathrm{E}[\bar{x}_{1i}'(y_{it} - \theta'\mathbf{x}_i t)] &= 0 \\ \mathrm{E}[z_1'(y_{it} - \theta'\mathbf{x}_i t)] &= 0.\end{aligned}$$

This gives us $2k_1 + k_2\ell_1$ moment conditions and $k_1 + k_2 + \ell_1 + \ell_2$ parameters. So the main

identification condition is that $k_1 \geq \ell_2$. This model can be fit with either 2SLS or GMM, with the latter being potentially advantageous in the

2.2.1 Example

As an example, we will consider

```
library(data.table)
library(readstata13)

library(sandwich)
library(fixest)
library(ivreg)
library(lmtest)

library(modelsummary)
aid <- data.table(read.dta13("Rcode/aid_migration/finaldata.dta"))

### setup in the paper###
aid[, `:=`(lcommit3a=log(1000000*commit3a+1),
           lpopulation =log(1000*population),
           listock = log(istock+1),
           lgdpcap = log(gdpcap+1),
           lexports = log(exports+1),
           ldist = log(distance+1),
           lusmil = log(usmil+1),
           ldisaster = log(disaster +1))]

aid[, `:=`(lpopulation_lag= shift(lpopulation),
           listock_lag=shift(listock),
           lgdpcap_lag=shift(lgdpcap),
           lexports_lag = shift(lexports),
           lusmil_lag = shift(lusmil),
           fh_lag = shift(fh),
           civilwar_lag = shift(civilwar),
           ldisaster_lag = shift(ldisaster)),
    by=dyad]
```

```r
aid <- aid[year > 1992 & year < 2009]

## Main outcome
# lcommit3a: foreign aid commitments from donor to receipient (USD log)

## Main regressors
# listock_lag: Size of the migrant population from the recipient country
#                in the donor (log, lag)
# lgdpcap_lag: Recipient GDP per capita (USD/person log, lag)
# lpopulation_lag: Recipient population (log, lag)
# lexports_lag: Exports from donor to the recipient  (USD log, lag)
# ldist: Distance from donor to recipient
# colony: Recipient is a former colony of the donor
# lusmil_lag: US military aid (log lag)
# fh_lag: 1-7 measure of democracy (lag)
# civilwar_lag: Binary, is there civil war (lag)
# ldisaster_lag: Number of people affected by a natural disaster (log, lag)



m1 <- feols(lcommit3a~listock_lag+lgdpcap_lag+lpopulation_lag+ lexports_lag +
              ldist+ colony+ lusmil_lag+ fh_lag+ civilwar_lag+ldisaster_lag|
              donor+year, data=aid)
summary(m1)
```

```
## OLS estimation, Dep. Var.: lcommit3a
## Observations: 33,181
## Fixed-effects: donor: 22,  year: 16
## Standard-errors: Clustered (donor)
##                  Estimate Std. Error    t value   Pr(>|t|)
## listock_lag      0.572146   0.071801   7.968504 8.7724e-08 ***
## lgdpcap_lag     -2.109841   0.144451 -14.605945 1.7958e-12 ***
## lpopulation_lag  0.577321   0.151597   3.808254 1.0266e-03 **
## lexports_lag     0.204072   0.057430   3.553430 1.8796e-03 **
## ldist           -0.838207   0.277692  -3.018472 6.5396e-03 **
## colony           3.147186   0.810626   3.882414 8.6027e-04 ***
```

```
## lusmil_lag       0.061323   0.017605   3.483345 2.2177e-03 **
## fh_lag           0.092279   0.057136   1.615071 1.2122e-01
## civilwar_lag    -0.139480   0.197864  -0.704926 4.8860e-01
## ldisaster_lag    0.084973   0.022365   3.799323 1.0487e-03 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 5.18145     Adj. R2: 0.486778
##                 Within R2: 0.370701
```

```
m2 <- feols(lcommit3a~listock_lag+lgdpcap_lag+lpopulation_lag+ lexports_lag +
              ldist+ colony+ lusmil_lag+ fh_lag+ civilwar_lag+ldisaster_lag|
              dyad+year, data=aid)
summary(m2)
```

```
## OLS estimation, Dep. Var.: lcommit3a
## Observations: 33,181
## Fixed-effects: dyad: 3,129,  year: 16
## Standard-errors: Clustered (dyad)
##                  Estimate Std. Error   t value   Pr(>|t|)
## listock_lag      0.303379   0.052167  5.815484 6.6555e-09 ***
## lgdpcap_lag     -0.583753   0.344623 -1.693886 9.0387e-02 .
## lpopulation_lag -1.205957   0.970822 -1.242202 2.1426e-01
## lexports_lag     0.094556   0.018695  5.057827 4.4850e-07 ***
## lusmil_lag       0.041247   0.007308  5.644173 1.8080e-08 ***
## fh_lag           0.200216   0.062172  3.220382 1.2933e-03 **
## civilwar_lag     0.016697   0.132610  0.125908 8.9981e-01
## ldisaster_lag    0.013605   0.006253  2.175740 2.9649e-02 *
## ... 2 variables were removed because of collinearity (ldist and colony)
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 3.67935     Adj. R2: 0.714453
##                 Within R2: 0.00923
```

```
aid.sam <- aid[m2$obs_selection$obsRemoved]



Years <- model.matrix(~factor(year)-1, data=aid.sam)[,-1]
colnames(Years) <- paste0("year", 1994:2008)
```

```
aid.sam <- cbind(aid.sam, Years)


var.names<- c("listock_lag", "lgdpcap_lag", "lpopulation_lag",
              "lexports_lag", "lusmil_lag", "fh_lag", "civilwar_lag",
              "ldisaster_lag",paste0("year", 1994:2008) )
aid.sam[ , paste0(var.names, ".within") := lapply(.SD, \(x){x-mean(x)}),
         by=dyad,
         .SDcols=var.names ]



fx <- ~listock_lag+lgdpcap_lag+lpopulation_lag+ lexports_lag +
  ldist+ colony+ lusmil_lag+ fh_lag+ civilwar_lag+ldisaster_lag +
  year1994+year1995+year1996+year1997+year1998+year1999+
  year2000+year2001+year2002+year2003 +year2004+
  year2005+year2006+year2007+year2008-1


fz <- ~lgdpcap_lag.within+lgdpcap_lag.within+lpopulation_lag.within+
  lexports_lag.within +
  lusmil_lag.within+
  fh_lag.within+ civilwar_lag.within+ldisaster_lag.within +
  year1994.within+year1995.within+year1996.within+year1997.within+
  year1998.within+year1999.within+
  year2000.within+year2001.within+year2002.within+year2003.within +
  year2004.within+ year2005.within+year2006.within+year2007.within+
  year2008.within +
  listock_lag.within +
  ldist+ colony-1#z1
## (no z2 here, so no need to include xbar as additional instruments)

ht <- ivreg(update(fx, lcommit3a ~.), fz, data=aid.sam)
ht.vcl <-vcovCL(ht, aid.sam$dyad)
coeftest(ht, vcov=ht.vcl)[1:10,]
```

```
##                    Estimate  Std. Error    t value     Pr(>|t|)
## listock_lag      0.30337884 0.052149342  5.8175008 6.028210e-09
## lgdpcap_lag     -0.58375251 0.344503853 -1.6944731 9.018481e-02
## lpopulation_lag -1.20595676 0.970485097 -1.2426330 2.140119e-01
## lexports_lag     0.09455581 0.018688465  5.0595813 4.224190e-07
## ldist            3.35030046 1.955755681  1.7130465 8.671333e-02
## colony           5.45199487 0.742772789  7.3400574 2.183973e-13
## lusmil_lag       0.04124711 0.007305377  5.6461304 1.654394e-08
## fh_lag           0.20021649 0.062150113  3.2214984 1.276452e-03
## civilwar_lag     0.01669668 0.132564014  0.1259518 8.997708e-01
## ldisaster_lag    0.01360516 0.006250953  2.1764944 2.952534e-02
```

```
modelsummary(list("Donor-FE"=m1,
                  "Dyad-FE"=m2,
                  "Dyad-FE (HT)"=ht),
             vcov=list(vcov(m1),vcov(m2),
                       ht.vcl),
             fmt=2,
             coef_map=c("listock_lag"="Migrant population (log)",
                        "colony"="Former colony",
                        "lgdpcap_lag"="GDP per cap. (log)",
                        "lpopulation_lag"="Population (log)",
                        "lexports_lag"="Exports (log)",
                        "ldist"="Distance (log)",
                        "lusmil_lag"= "U.S. Military aid (log)",
                        "fh_lag"="Democracy",
                        "civilwar_lag"="Civil war",
                        "ldisaster_lag"="Diaster"),
             gof_map=c("nobs"))
```

Note that HT (1981) impose some random effects style structure on the model by introducing

	Donor-FE	Dyad-FE	Dyad-FE (HT)
Migrant population (log)	0.57	0.30	0.30
	(0.07)	(0.05)	(0.05)
Former colony	3.15		5.45
	(0.81)		(0.74)
GDP per cap. (log)	−2.11	−0.58	−0.58
	(0.14)	(0.34)	(0.34)
Population (log)	0.58	−1.21	−1.21
	(0.15)	(0.97)	(0.97)
Exports (log)	0.20	0.09	0.09
	(0.06)	(0.02)	(0.02)
Distance (log)	−0.84		3.35
	(0.28)		(1.96)
U.S. Military aid (log)	0.06	0.04	0.04
	(0.02)	(0.01)	(0.01)
Democracy	0.09	0.20	0.20
	(0.06)	(0.06)	(0.06)
Civil war	−0.14	0.02	0.02
	(0.20)	(0.13)	(0.13)
Diaster	0.08	0.01	0.01
	(0.02)	(0.01)	(0.01)
Num.Obs.	33 181	33 181	33 181

the following RE (and unnecessary) assumptions

$$\begin{aligned}
\mathrm{E}[\alpha_i] = \mathrm{E}[\alpha_i|x_{1it}, z_{1it}] &= 0 \\
\mathrm{Var}(\alpha_i|x_{1it}, x_{2it}, z_{1i}, z_{2i}) &= \sigma_\alpha^2 \\
\mathrm{E}[\alpha_i\varepsilon_{it}|x_{1it}, x_{2it}, z_{1i}, z_{2i}] &= 0 \\
\mathrm{Var}(\varepsilon_{it} + \alpha_i|x_{1it}, x_{2it}, z_{1i}, z_{2i}) &= \sigma_e^2 = \sigma_\alpha^2 + \sigma_\varepsilon^2 \\
\mathrm{Cov}(\varepsilon_{it} + \alpha_i, u_{is} + \alpha_i|x_{1it}, x_{2it}, z_{1i}, z_{2i}) &= \sigma_\alpha^2
\end{aligned}$$

Note that like the RE models above this also introduces iid within unit observations. As such, canned versions of HT will be GLS estimators but based on the same instruments. The HT estimator is an FGLS, that takes the following form:

1. Use the FE estimator to obtain estimates of β. Use the residuals from this to produce $\hat{\sigma}_\varepsilon^2$.

2. Construct the within-group constants using the FE estimates of β

$$\hat{\alpha}_i = \bar{y}_i - \bar{x}_i\hat{\beta}_{FE} = \gamma' z_i + \alpha_i$$

   Generate $\hat{\alpha}_{it} = \hat{\delta}_i \otimes 1_T$.

3. Using 2SLS regress $\hat{alpha}$ on z_1 and z_2 with instruments z_1 and x_1. This will produce consistent estimates of γ. We could stop here since everything is consistent, but to clean but some inefficiency we move on.

4. The variance of the residuals from the 2SLS in the last step is a consistent estimate of

$$\sigma_e^2 = \sigma_\alpha^2 + \sigma_\varepsilon^2/T$$

   . We have a consistent estimate of σ_ε^2 from above, so we can use these two things to back out σ_α^2. The FGLS weights for the RE model are, as we know from above,

$$\lambda = 1 - \frac{\sigma_\varepsilon}{\sqrt{\sigma_\varepsilon^2 + T\sigma_\alpha^2}}.$$

5. We now have data, instruments, and weights, we're set. Let

$$\begin{aligned}\tilde{\mathbf{x}}_{it} &= \mathbf{x}_{it} - \lambda\bar{\mathbf{x}}_i \\ \tilde{y}_{it} &= y_{it} - \lambda\bar{y}_i \\ \mathbf{z}_{it} &= (\dot{x}_{1it}, \dot{x}_{2it}, z_{1i}, \bar{x}_{1i}),\end{aligned}$$

then are estimates are

$$\hat{\theta}_{HT} = [\tilde{\mathbf{X}}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\tilde{\mathbf{X}}]^{-1}[\tilde{\mathbf{X}}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\tilde{y}],$$

which is the over-identified 2SLS with regressors $\tilde{\mathbf{X}}$ and instruments $\mathbf{Z}$. However, it's not at all clear if the additional RE assumptions buy you anything since they are almost surely wrong.

2.3 Dynamic panel models

Dynamic panel models take the form of

$$y_{it} = \rho y_{it-1} + \beta' x_{it} + \alpha_i + \varepsilon_{it}.$$

Where the main thing to note here is that the presence of y_{it-1} in the regressors rules out assumption A3 (strict within-unit exogeneity). To see this, first recall that strict exogeneity means that ε_{it} is independent of of all values of the regressors (past and future). Then note that

$$\begin{aligned}y_{it+1} &= \rho y_{it} + \beta' x_{it+1} + \alpha_i + \varepsilon_{it+1} \\ &= \rho(\rho y_{it-1} + \beta' x_{it} + \alpha_i + \varepsilon_{it}) + \beta' x_{it+1} + \alpha_i + \varepsilon_{it+1}\end{aligned}$$

which is to say that ε_{it} is related to y_{it+1}, which is also part of the regressors and breaks strict exogeneity. As such, the pooled, random effects, and fixed effects estimators will all be biased.

However, **if** we can credibly claim weak exogeneity (Assumption A3′), then we may have something we can work with asymptotically. Let's start with the easier case where the unobserved heterogeneity is uncorrelated with x_{it}. Note that it will still be correlated with y_{it-1}. However, can we still get an okay estimate of β with the pooled estimator? A decent number of papers try this identification approach, so how does it do?

If x_{it} is static is might do ok. To see this consider the following three DAGs. In the first, we have a static x_{it} and unobserved heterogeneity that is uncorrelated with x_{it}. In this case,

there is actually no reason to include the lagged dependent variable. The pooled model that regresses y_{it} on x_{it} should be just fine here as omitting both y_{it-1} and a_i induce no bias.

In the second case, we have a dynamic x_{it} and unobserved heterogeneity that is still uncorrelated with x_{it}. In this case, x_{it-1} is a confounder it has a path to both x_{it} and y_{it}. We need to close the path between them by controlling for either it or for y_{it-1}. However, note that y_{it-1} is also confounded by the unobserved a_i. This means that controlling for y_{it-1} would introduce a new omitted variable bias that isn't present otherwise. If we regress y_{it} on just x_{it} and y_{it-1}, then we have opened up a new path between x_{it} and y_{it}. This particular structure is known as M-bias. To close it we need to control for either x_{it-1} or a_i. Given that a_i is unobserved, the best approach would be to control for x_{it-1}. However, this is a case where the RE estimator actually offers a real improvement over the pooled model. Fitting the RE model will act as a kind of control for a_i that is appropriate as a_i is uncorrelated with x_{it} and x_{it-1}. In this case, RE and FE estimators will be preferred to pooled model (but this comes with its own caveats to follow). As such, controlling for just x_{it-1} seems like a soundest strategy in this case.

In the third case, we have some more interesting dynamics where some feedback is present. Here past values of y affect current values of both x and y. For example, consider trade and war. Being at war in $t-1$ likely affects both trade and the probability of being in war at t, while trade levels at $t-1$ likely affect both of these things as well. This case is very similar to the last one, but with the note that both y_{it-1} and x_{it-1} are now both classic confounders. Controlling for just y_{it-1} blocks the path from x_{it-1} to y_t (good), but again introduce M bias that needs to be blocked by either controlling for x_{it-1} or a_i. Since a_i is still uncorrelated with x_{it}, either the RE or FE estimators will work to block that path.

The final case returns us to unobserved heterogeneity that is correlated with x_{it}. Here, controlling for a_i is no longer just a possibility, it is a requirement for consistently estimating β. Likewise, the lagged DV is still a confounder that needs to be addressed. However, controlling for x_{it-1} is no longer required and it no longer helps us with β as its paths are blocked by controlling for α_i and y_{it-1}. Because a_i is correlated with x_{it} however, we need a fixed effects estimator. We turn to this task next and uncover from concerning truths.

What this comes back to is that any exogeneity assumption you make will be violated under omitted variables and we need to be careful with that. Even time-invariant omitted variables that are uncorrelated with x will be correlated with the lagged y. However, even fixed effects estimators (within/LSDV or FD) will present additional issues that were not present in the static case. Both will be biased with a lagged dependent variable (because we lose strict

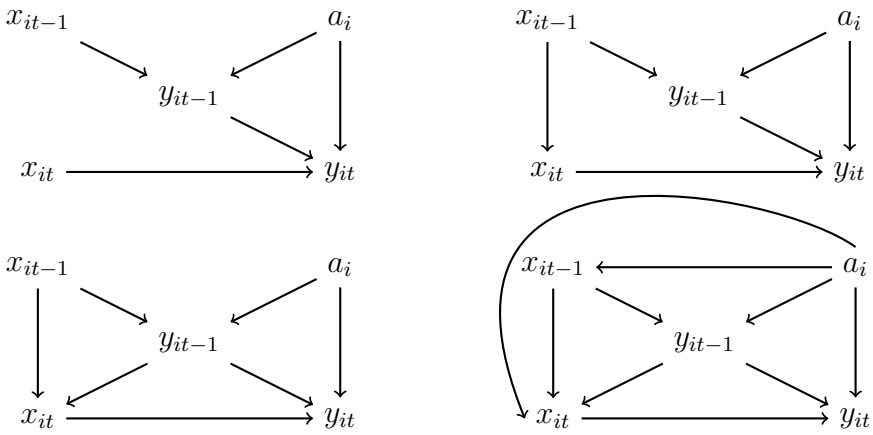


Figure 2.1: Why including a lagged dependent variable can introduce M-bias in the pooled model even when the unobserved heterogeneity is uncorrelated with x.

exogeneity within units). However, note that the within estimator is still consistent with weak exogeneity (in either N or T if there is not a lagged dependent variable).

The FD estimator, is not consistent with weak exogeneity under our current assumptions even when there is no lagged dependent variable. We could work around this by making an assumption on weak exogeneity in the differences $\mathrm{E}[\Delta\varepsilon_{it}\Delta x_{it}] = 0$ instead. However, once we add the lagged DV

$$y_{it} - y_{t-1} = \beta'(x_{it} - x_{it-1}) + \rho(y_{it-1} - y_{it-2}) + \varepsilon_{it} - \varepsilon_{it-1}.$$

To acheive consistency we would need the assumption on the differences to hold, but

$$\mathrm{E}[\Delta\varepsilon_{it}\Delta y_{it-1}] = \mathrm{E}[\varepsilon_{it}y_{it-1}] - \mathrm{E}[\varepsilon_{it-1}y_{it-1}] - \mathrm{E}[\varepsilon_{it}y_{it-2}] + \mathrm{E}[\varepsilon_{it-1}y_{it-2}]$$

even assuming fully iid errors, the term $\mathrm{E}[\varepsilon_{it-1}y_{it-1}]$ will definitely not be 0. Ok so the FD estimator is biased and inconsistent.
What can we say about the performance of the within estimator with a lag?

2.3.1 Within estimator with a lagged dependent variable

Let's consider a simple version of the model with

$$y_{it} = \rho y_{it-1} + \alpha_i + \varepsilon_{it}$$

Where we will let $|\rho| < 1$, α_i be a fixed constant, and ε_{it} is iid noise. These conditions are sufficient for y_{it} to be stationary and we can calculate it's mean and variance conditional on

u_{it} using the complete history of lags:

$$\begin{aligned} y_{it} = \sum_{s=0}^{\infty} \rho^s (\alpha_i + \varepsilon_{it}) \, \mathrm{E}[y_{it}|\alpha_i] \quad &= \frac{1}{1-\rho}\alpha_i \\ \mathrm{Var}(y_{it}|\alpha_i) = \frac{1}{1-\rho^2}\sigma_\varepsilon^2 & \end{aligned}$$

Note that the fixed unit-level intercepts move the mean, but not the variance, as we might intuit.

Ok, so how do we go about estimating ρ in this case? Well off the bat, the within or LSDV estimator seems promising, yes? Ok suppose we want to use the within estimator and we have a panel with $T = 3$. Note that this means that we actually only get 2 observations per unit because we lose 1 to the lag. This means that the within estimator will be identical to the FD estimator so we can write

$$\begin{aligned} \hat{\rho} &= \left(\sum_{i=1}^{N} \Delta y_{i2}^2 \right)^{-1} \left(\sum_{i}^{N} \Delta y_{i2} \Delta y_{i3}) \right) \\ &= \rho + \left(\sum_{i=1}^{N} \Delta y_{i2}^2 \right)^{-1} \left(\sum_{i}^{N} \Delta y_{i2} \Delta \varepsilon_{i3} \right), \end{aligned}$$

which takes us back to what we had above

$$\begin{aligned} \mathrm{E}[\Delta \varepsilon_{i3} \Delta y_{i2}] &= \mathrm{E}[\varepsilon_{i3} y_{i2}] - \mathrm{E}[\varepsilon_{i2} y_{i2}] - \mathrm{E}[\varepsilon_{i3} y_{i1}] + \mathrm{E}[\varepsilon_{i2} y_{i1}] \\ &= -\sigma_\varepsilon^2. \end{aligned}$$

Does this bias get better with increasing N or T? Fixing $T = 3$ and taking $N \to \infty$ we get

$$\hat{\rho} - \rho \xrightarrow{p} \frac{\mathrm{E}[\Delta y_{i2} \Delta \varepsilon_{i3}]}{\mathrm{E}[\Delta y_{i2}^2]} = \frac{\rho + 1}{-2}.$$

Note that this bias is always negative and the estimator is only consistent if $\rho = -1$. This is both unlikely and would probably lead to other issues as it takes us beyond our assumption of $|\rho| < 1$. Also if $\rho \geq 0$, then the bias is between $-1/2$ and 1 which is fairly large for a correlation.

What about as T increases? Nickell (1981) provides us with an expression of the asymptotic bias of ρ in this context (as $N \to \infty$) for any given T, which is

$$\frac{\rho + 1}{\frac{2\rho}{1-\rho} - \frac{T-1}{1-\rho^{T-1}}} \approx \frac{-(1+\rho)}{T-1}$$

The good news here is that this bias is decreasing in T. The bad news, is that it can be fairly large even for decent sizes of T. For example, if $\rho = .25$ and $T = 25$, then we still have an asymptotic bias on $\hat{\rho}$ of -0.05 (in other words a 20% shrinkage). Whether that's enough for us to be concerned about depends on the application and the parameters of interest.

Moving back to the case with additional regressors

$$y_{it} = \rho y_{it-1} + \beta' x_{it} + \alpha_i + \varepsilon_{it},$$

and applying the within transformation

$$\dot{y}_{it} = \rho \dot{y}_{it-1} + \beta' \dot{x}_{it} + \dot{\varepsilon}_{it},$$

we can apply Frisch–Waugh–Lovell to get

$$\hat{\beta} - \beta = \left(\dot{\mathbf{X}}'\dot{\mathbf{X}}\right)^{-1} \dot{\mathbf{X}}'\dot{\varepsilon} - \left(\dot{\mathbf{X}}'\dot{\mathbf{X}}\right)^{-1} \dot{\mathbf{X}}'\dot{y}_{\ell}(\hat{\rho} - \rho),$$

where $\dot{y}_{\ell}$ is the within-transformed lagged DV in matrix form. As $N \to \infty$ we get

$$\hat{\beta} - \beta \xrightarrow{p} 0 - \mathrm{E}\left[\left(\dot{\mathbf{X}}'\dot{\mathbf{X}}\right)^{-1} \dot{\mathbf{X}}'\dot{y}_{\ell}\right] \mathrm{E}\left[(\hat{\rho} - \rho)\right].$$

We've talked about $\mathrm{E}\left[(\hat{\rho} - \rho)\right]$ above, it changes a little bit with additional exogenous variables, but the signs remain the same. So if $\rho > 0$ then $\mathrm{E}\left[(\hat{\rho} - \rho)\right] < 0$. This means that the large sample bias in $\hat{\beta}$ depends on remaining term. As such the direction of the bias will follow from the direction of the relationship of a regression of y_{it-1} on x_{it}^k. If this returns a positive coefficient, then $\hat{\beta}^k$ will be biased upwards and vice versa.

So to recap:

1. Nickell bias is term to refer to Nickell's (1981) result that using the within/LSDV estimator with a lagged dependent variable will bias both ρ and β.
2. In most cases the asymptotic (in N) bias in ρ will be downward (toward -1), while the asymptotic bias in β is equal to the asymptotic bias in ρ times the regression coefficient of a regression of y_{it-1} and x_{it}. Which could be small (or not).
3. All else equal, the bias in both β and ρ will increase with more covariates, but is decreasing in T
4. There's no magic value of T that makes it safe to do this, but the smaller you think ρ and relationship between y_{it-1} and x_{it} are, the better off you'll be.

Ok so where does this persistent bias come from? The main source of the bias is this fact

that y_{it-1} is endogenous. It depends on the entire history of the unit. In terms of the IV approach recall that the within transformation is equivalent to instrumenting each variable with its within-transformed version. For the exogenous covariates, this is a perfectly valid approach. However for y_{it-1}, the within transformation is not a valid instrument. This is also true for the FD transformation. For the within transformation, it gets better with larger T as the more information that goes into the within-unit mean, the less weight is placed on the specific endogenous part of the series. It is however still present and should concern us, particularly with small T panels.

So what can we do? Well the primary issue is that we have a bad instrument, so what can we do to get a better instrument? A lot of ink has been spilled on this. We'll start with the main two approachs.

2.3.2 IV approaches to the dynamic panel model

The first is the **Anderson-Hsiao** (AH) estimator. They start with the FD representation of the problem

$$\Delta y_{it} = \beta'(\Delta x_{it}) + \rho(\Delta y_{it-1}) + \Delta\varepsilon_{it},$$

where x_{it} may contain time dummies. As we know, this induces correlation between Δy_{it-1} and $\Delta\varepsilon_{it}$ with $\mathrm{E}[\Delta y_{it-1}\Delta\varepsilon_{it}] = -\sigma^2_\varepsilon$. Their proposal to solve this endogeneity? Use another instrument! Specifically, what if we instrument for Δy_{it-1} with y_{it-2}? Recall that with weak exogeneity in differences we assume that $\mathrm{E}[\Delta y_{it-s}\Delta\varepsilon_{it}] = 0$ for $s > 1$ Now we have

$$\mathrm{E}[y_{it-2}\Delta\varepsilon_{it}] = \mathrm{E}[y_{it-2}\varepsilon_{it}] - \mathrm{E}[y_{it-2}\varepsilon_{it-1}] = 0$$

This hinges on the model being AR(1). If the true model is AR(2) then a further back lack will be required as a valid instrument (i.e., y_{it-3}).
Like some time series cases, we need the dynamic components to be specified to the point that we can assume ε_{it} is iid. Likewise, in order for the instrument to be relevant, we need a strong relationship between Δy_{it} and y_{it-2}, which is not guaranteed.

Continuing this point, note that when considering the residuals of the fitted model, you'll be looking at $\Delta\varepsilon_{it}$. If ε_{it} are iid and homoskedasitic then these differences will be correlated. We showed this before when we said that the within-unit covariance matrix of $\Delta\varepsilon_{it}$ will be

$$H_i\sigma^2_\varepsilon = \sigma^2_\varepsilon\Delta_i\Delta_i'$$

where H_i is $T_i \times T_i$ with 2s on the diagonal and -1 on the first off diagonals. So the residuals

from AH *should* be AR(1) with a **negative** coefficient and we can check that. There should **not** be any relation between $\Delta\hat{\varepsilon}_{it}$ and $\Delta\hat{\varepsilon}_{it-2}$ however, which we can also check.

The second approach is known as the Arellano-Bond (AB) estimator. They start by saying something like, "hey that AH estimator is a good idea, but they're leaving a lot of money on the table." Specifically, if y_{it-2} is a good instrument, then so is y_{it-3} probably, and if they're both good, then why not include all the lags?

Every lag from $y_{i1}, \ldots, y_{iT-2}$ is a potentially good instrument. So let's start with the FD estimator again

$$\Delta y_{it} = \beta'(\Delta x_{it}) + \rho(\Delta y_{it-1}) + \Delta\varepsilon_{it}.$$

Now we can consider the instrument matrix for this. For unit i, lets then the instrument matrix for the AB estimator for group i is formed by using all the available lags of y_{it}

$$Z_i = \begin{bmatrix} y_{i1} & 0 & 0 & 0 & \ldots & 0 & | & \Delta x_{i3} & \Delta t_3 \\ 0 & y_{i1} & y_{i2} & 0 & \ldots & 0 & | & \Delta x_{i4} & \Delta t_4 \\ 0 & 0 & 0 & y_{i1:3} & 0 & \vdots & | & \Delta x_{i5} & \Delta t_5 \\ \vdots & \vdots & \vdots & \ldots & \ddots & \vdots & | & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & y_{i,1:T-2} & | & \Delta x_{iT} & \Delta T \end{bmatrix}.$$

Now we form full Z matrix by stacking these up

$$Z = [Z_i]_{i=1}^{N}.$$

For ease, let's combine the independent variables to be

$$\Delta\mathbf{x}_{it} = [\Delta y_{it-1}, \Delta x_{it}, \Delta\Delta_T],$$

and stack those into

$$\mathbf{X} = \left[[\Delta\mathbf{x}_{it}]_{t=3}^{T}\right]_{i=1}^{N}.$$

A few notes before we proceed.

1. Creating the instrument matrix is the hardest part of this. This was designed for situations where T is relatively small in comparison to N, however it will work in larger T cases (but the LSDV isn't so bad in those cases so think about the trade-offs)
2. You don't have to use all the lags. For medium to large T, these further lags will be weaker and weaker instruments, while also increasing the size and computational burden

of working with Z. So, you don't have to use them all. We'll talk about specification tests in a moment that may help with these decisions. If you lose all lags, you're looking at $1 + 2 + \ldots + T - 2 + \mathbf{k}$ columns. For $T = 50$ that's 1,176 instruments for that one lagged outcome variable. But if you only use 2 lags (i.e., $y_{it-2:3}$), you're down to $1 + 2 * T + \mathbf{k}$ or only 97 when $T = 50$. You'll want to try and keep the size of Z less than N for the simple fact that if you want to cluster your standard errors (say for either your results or for constructing an F test on the first stage regressors) and $N < \text{ncol}(Z)$ then the meat matrix will not have full rank.

3. The above is also not not the only version of this. A so-called "collapsed" version exists as well where

$$Z_i = \begin{bmatrix} y_{i1} & 0 & 0 & \ldots & | & \Delta x_{i3} & \Delta t_3 & \\ y_{i2} & y_{i1} & 0 & \ldots & | & \Delta x_{i4} & \Delta t_4 & \\ y_{i3} & y_{i2} & y_{i3} & \ldots & | & \Delta x_{i5} & & \Delta t_5 \\ \vdots & \vdots & \vdots & \ldots & | & \vdots & & \end{bmatrix}.$$

   This is nice from both the computational perspective, but also from the perspective that using hundreds of instruments is generally not a great strategy. Some simulation evidence exists to suggest that the collapsed version may perform better when the instruments are overall weak.

4. When building the instrument matrix, there are a few issues to be aware of. First, with unbalanced panels, you'll want to balance it first. By this I mean create fake observations to fill in the panel until all units are observed for the same time periods.

- Do not impute or fill in any of the independent variables.
- Do fill in the dependent variable with 0s to create the lags. Keep track of which observations are fake as you'll need to remove them when it becomes time to fit the model.

Once we have this instrument matrix built, we'll need to decide on a weighting matrix Ω. The efficient weighting matrix will be $\text{Var}(Z\Delta\varepsilon)$, which under our assumptions will be

$$\Omega = \sum_{i=1}^{N} Z_i' H_i Z_i,$$

where as before $H_i = \Delta_i \Delta_i'$. Now we can build the efficient GMM as

$$\hat{\theta}_{AB} = (\mathbf{X}' Z' \Omega^{-1} Z' \mathbf{X})^{-1} (\mathbf{X}' Z' \Omega^{-1} Z' \Delta y).$$

This estimator is sometimes referred to as the one-step AB estimator. Under our assumptions, the variance of this estimator will be

$$\widehat{\text{Var}_0}(\hat{\theta}_{AB} = (\mathbf{X}'Z'\Omega^{-1}Z'\mathbf{X})^{-1}\hat{\sigma}_\varepsilon^2,$$

where we can estimate $\hat{\sigma}_\varepsilon^2$ using the residuals as usual.

If we were worried that we violated some of these assumptions, there is a robust version

$$\widehat{\text{Var}_1}(\hat{\theta}_{AB} = (\mathbf{X}'Z'\Omega^{-1}Z'\mathbf{X})^{-1}(\mathbf{X}'Z'\Omega^{-1}\widehat{\Omega}_2\Omega^{-1}Z'\mathbf{X})(\mathbf{X}'Z'\Omega^{-1}Z'\mathbf{X})^{-1},$$

where

$$\widehat{\Omega}_2 = \sum_{i=1}^{N} Z_i'\Delta\hat{\varepsilon}_i\Delta\hat{\varepsilon}_i'Z_i.$$

Note that this is a cluster-robust weighting matrix. If we need this because we're worried about heteroskedasticity that's not a problem. If we need this because we're worried about residual autocorrelation that could be an issue, because we want to have modeled away that autocorrelation. However, some small amounts may remain, so this version of the matrix isn't the worst idea.

We can also use this matrix to make a two-step estimator

$$\hat{\theta}_{AB}^2 = (\mathbf{X}'Z\widehat{\Omega}_2^{-1}Z'\mathbf{X})^{-1}(\mathbf{X}'Z'\widehat{\Omega}_2^{-1}Z'\Delta y),$$

with variance matrix

$$\widehat{\text{Var}}(\hat{\theta}_{AB}^2) = (\mathbf{X}'Z'\widehat{\Omega}_2^{-1}Z'\mathbf{X})^{-1}.$$

However, note that the one-step weighting matrix is just a function of observables and does not include the residuals at all. This may be a good reason to favor it as it includes less randomness. Some simulation work suggests that the one-step may be better in smaller samples for this reason. Another related point against two-step GMM is that the estimated variances, while asymptotically correct, tend to be too small in practice. Windmeijer (2005)

proposes a correction that is cumbersome to look at, but not too bad to implement.

$$
\begin{aligned}
G &= t(Z)\Delta\hat{\varepsilon}_2 \quad \text{Sample moments, summed} \\
g &= -Z'X \qquad D_\theta G \\
\omega_j &= Gg'_{[j]} + G_{[j]}g' \\
W_j &= \left[g'\widehat{\Omega}_2^{-1}g\right]^{-1}\left[g'(\widehat{\Omega}_2^{-1}\omega_j\widehat{\Omega}_2^{-1})G\right] \\
W &= \begin{bmatrix} W_1 & \dots & W_k \end{bmatrix}/N \\
\widehat{\text{Var}}(\hat{\theta}^2_{AB})_{\text{Corrected}} &= \widehat{\text{Var}}(\hat{\theta}^2_{AB}) + W\widehat{\text{Var}}(\hat{\theta}^2_{AB}) + \widehat{\text{Var}}(\hat{\theta}^2_{AB})W' + W\widehat{\text{Var}}_1(\hat{\theta}_{AB})W'
\end{aligned}
$$

As with the AH estimator, we can consider some post-estimation exercises. First, we can again consider the first and second lags of the differenced residuals. The first, again, should be negatively correlated and the second should be unrelated. Second, we should think about the first-stage relationships. Third, Sargan tests for over-identification for the one- and two-step, respectively are

$$
\begin{aligned}
s_1 &= \Delta\hat{\varepsilon}_1{}'Z\Omega_1^{-1}Z'\Delta\hat{\varepsilon}_1 \\
s_2 &= \Delta\hat{\varepsilon}_2{}'Z\widehat{\Omega}_2^{-1}Z'\Delta\hat{\varepsilon}_2.
\end{aligned}
$$

The main advantage of AB over AH is that it takes more advantage of the available data, which should improve the efficiency of the estimator. The cost of this is the burden of building the instrument matrix and possibility of many weak instruments.

Before moving into applications, let's also consider what effects of interest we can extract from dynamic models. Suppose we have an AR(p) model of the form

$$
y_{it} = \sum_{j=1}^{p}\rho_j y_{it-j} + \beta_1 x_{it,1} + \beta' x_{it} + \alpha_i + \varepsilon_{it},
$$

which we can fit using the AB estimator. The effect of interest is the effect of x_1 on y. The "short-run" effect is, as usual, β_1. This is the instantaneous effect of a 1 unit change in x_1 on

y. We can consider a variety of long-run effects however:

$$\begin{aligned}
y_{it+1} &= \rho_1 y_{it} + \sum_{j=1}^{p-1} \rho_{j+1} y_{it-j} + \beta_1 x_{it+1,1} + \beta' x_1 + \alpha_i + \varepsilon_{it+1} \\
&= \rho_1 \left(\sum_{j=1}^{p} \rho_j y_{it-j} + \beta_1 x_{it,1} + \beta' x_{it} + \alpha_i + \varepsilon_{it} \right) \\
&\quad + \sum_{j=1}^{p-1} \rho_{j+1} y_{it-j} + \beta_1 x_{it+1,1} + \beta' x_{it+1} + \alpha_i + \varepsilon_{it+1} \\
\frac{\partial y_{it+1}}{\partial x_1} &= \beta_1 + \rho_1 \left(\frac{\partial y_{it}}{\partial x_1} \right) = \beta_1 + \rho_1 \beta_1 \\
\frac{\partial y_{it+2}}{\partial x_1} &= \beta_1 + \rho_1 \left(\frac{\partial y_{it+1}}{\partial x_{i1}} \right) + \rho_2 \left(\frac{\partial y_{it}}{\partial x_1} \right) \\
&\vdots \\
\frac{\partial y_{it+s}}{\partial x_1} &= \beta_1 + \sum_{j=1}^{\min\{p,s\}} \rho_j \left(\frac{\partial y_{it+s-j}}{\partial x_1} \right).
\end{aligned}$$

The sequential aspect of this, means that that easiest way to find these these for larger values of s is to use a loop. You could work it out, but the terms explode quickly. Note that this means you'll need to loop through for standard errors too, as in

$$\begin{aligned}
\operatorname{Var}\left(\widehat{\frac{\partial y_{it+1}}{\partial x_1}} \right) &= D_\theta \left[\hat{\beta}_1 + \hat{\rho}_1 \left(\widehat{\frac{\partial y_{it}}{\partial x_1}} \right) \right]' V(\hat{\theta}) D_\theta \left[\hat{\beta}_1 + \hat{\rho}_1 \left(\widehat{\frac{\partial y_{it}}{\partial x_1}} \right) \right] \\
&= \left[1 + \hat{\rho}_1,\ \hat{\beta}_1 \right] V(\hat{\beta}_1, \hat{\rho}_1) \left[1 + \hat{\rho}_1,\ \hat{\beta}_1 \right]' \\
\operatorname{Var}\left(\widehat{\frac{\partial y_{it+s}}{\partial x_1}} \right) &= D \operatorname{Var}(\hat{\beta}, \hat{\rho}) D' \\
D &= \left[1 + \sum_{j=1}^{\min\{s,p\}} \hat{\rho}_j \widehat{\frac{\partial y_{it+s-j}}{\partial x_1 \partial \beta_1}},\ \ \hat{\rho}_j \widehat{\frac{\partial y_{it+s-j}}{\partial x_1 \partial \rho_j}} + \widehat{\frac{\partial y_{it+s-j}}{\partial x_1}} \right].
\end{aligned}$$

Again, the sequential approach means you'll want to use a loop here that and compute the estimate and standard errors simultaneous as you go.

We can also consider the effect of a permanent shift in x_1 at time t. This would be

$$\frac{\beta_1}{1 - \sum_{j=1}^{p} \rho_j}.$$

Finally, maybe we want to know the total "persistence" of y $\sum_{j=1}^{p} \rho_j$, for an AR(1) model

that is just the estimated ρ.

2.3.3 Application

```
library(car)
library(dplyr)
library(fixest)
library(ivreg)
library(sandwich)
library(readstata13)
library(lmtest)
library(Matrix)
rm(list=ls())



data.use <- read.dta13( "Rcode/corruption_terrorism/subsample.dta")
data.use <- data.use %>%
  mutate(nattack = log(sinh(nattack)+1)) %>%
  mutate(L.nattack = lag(nattack,1), .by=id)




within.ldv <- feols(nattack ~ v2x_corr +sp_pop_totl
                    + ny_gdp_pcap_kd
                    +kg_democracy +statefailure
                    + L.nattack|id+year,
                    data=data.use)
summary(within.ldv)
```

```
## OLS estimation, Dep. Var.: nattack
## Observations: 6,709
## Fixed-effects: id: 170,  year: 47
## Standard-errors: Clustered (id)
##                Estimate Std. Error  t value   Pr(>|t|)
## v2x_corr       0.250015   0.128936  1.93906 5.4159e-02 .
## sp_pop_totl    0.568555   0.105899  5.36884 2.5857e-07 ***
```

```
## ny_gdp_pcap_kd 0.127048   0.059541  2.13379 3.4302e-02 *
## kg_democracy   0.062332   0.059411  1.04917 2.9560e-01
## statefailure   0.067291   0.013039  5.16073 6.8401e-07 ***
## L.nattack      0.720108   0.021052 34.20581  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 0.671287     Adj. R2: 0.823752
##                  Within R2: 0.579001
```

```
##effects? Let's focus on the average within-unit standard deviation
## increase in corruption. What is that?
data.use %>%
  summarize(s=sd(v2x_corr, na.rm=TRUE), .by=id) %>%
  summarize(mean(s))
```

```
##      mean(s)
## 1 0.07276422
```

```
## about 0.07

## Short-run effect of corruption on terrorism?
## Approximation: A 1.7% increase  in terrorist attacks
## for every within standard deviation  (all else equal)
deltaMethod(within.ldv, "0.07*v2x_corr")
```

```
##                    Estimate          SE       2.5 % 97.5 %
## 0.07 * v2x_corr  0.01750102  0.00902549 -0.00018862 0.0352
```

```
## 1 period later  (about a 3% increase)
deltaMethod(within.ldv, "0.07*(v2x_corr*L.nattack+v2x_corr)")
```

```
##                                            Estimate          SE       2.5 %
## 0.07 * (v2x_corr * L.nattack + v2x_corr)  0.03010365  0.01551634 -0.00030781
##                                             97.5 %
## 0.07 * (v2x_corr * L.nattack + v2x_corr) 0.0605
```

```
## permanent increase? (about a 6% increase)
deltaMethod(within.ldv, "0.07*(v2x_corr/(1-L.nattack))")
```

```
##                                      Estimate         SE      2.5 % 97.5 %
```

```
## 0.07 * (v2x_corr/(1 - L.nattack))  0.0625279  0.0324247 -0.0010234 0.1261
```

```
### Anderson  Hsiao
col.names <- c("id", "year", "nattack",
               "v2x_corr", "sp_pop_totl",
               "ny_gdp_pcap_kd", "kg_democracy",
               "statefailure", "L.nattack")

## we lose 2 time dummies to differencing and colinearity
time.dummies <- model.matrix(~ factor(year)-1,
                             data=data.use)[,-c(1:2)]
colnames(time.dummies) <- paste0("year", min(data.use$year):max(data.use$year))[-c(1:2)]
data.use <- cbind(data.use, time.dummies)

## Difference the variables
var.names <- c(col.names[-c(1:2)], colnames(time.dummies))
data.use <- data.use %>%
  group_by(id) %>%
  mutate(across(all_of(var.names), \(x){x-lag(x)}, .names="D.{col}"),
         L2.nattack = lag(L.nattack)) %>%
  ungroup()
AH <- ivreg(D.nattack ~
              D.v2x_corr + D.sp_pop_totl + D.ny_gdp_pcap_kd +
              D.kg_democracy + D.statefailure +
              D.L.nattack+
              D.year1973 + D.year1974 + D.year1975 +
              D.year1976 + D.year1977 + D.year1978 +
              D.year1979 + D.year1980 + D.year1981 +
              D.year1982 + D.year1983 + D.year1984 +
              D.year1985 + D.year1986 + D.year1987 +
              D.year1988 + D.year1989 + D.year1990 +
              D.year1991 + D.year1992 + D.year1993 +
              D.year1994 + D.year1995 + D.year1996 +
              D.year1997 + D.year1998 + D.year1999 +
              D.year2000 + D.year2001 + D.year2002 +
              D.year2003 + D.year2004 + D.year2005 +
              D.year2006 + D.year2007 + D.year2008 +
```

```
              D.year2009 + D.year2010 + D.year2011 +
              D.year2012 + D.year2013 + D.year2014 +
              D.year2015 + D.year2016 + D.year2017 +
              D.year2018-1 |
              D.v2x_corr + D.sp_pop_totl + D.ny_gdp_pcap_kd +
              D.kg_democracy + D.statefailure +
              L2.nattack+
              D.year1973 + D.year1974 + D.year1975 +
              D.year1976 + D.year1977 + D.year1978 +
              D.year1979 + D.year1980 + D.year1981 +
              D.year1982 + D.year1983 + D.year1984 +
              D.year1985 + D.year1986 + D.year1987 +
              D.year1988 + D.year1989 + D.year1990 +
              D.year1991 + D.year1992 + D.year1993 +
              D.year1994 + D.year1995 + D.year1996 +
              D.year1997 + D.year1998 + D.year1999 +
              D.year2000 + D.year2001 + D.year2002 +
              D.year2003 + D.year2004 + D.year2005 +
              D.year2006 + D.year2007 + D.year2008 +
              D.year2009 + D.year2010 + D.year2011 +
              D.year2012 + D.year2013 + D.year2014 +
              D.year2015 + D.year2016 + D.year2017 +
              D.year2018-1,
            data=data.use,
            x=TRUE)
V.AH <- vcovCL(AH, cluster=data.use$id)
coeftest(AH, V.AH)[1:6,]
```

```
##                    Estimate Std. Error    t value     Pr(>|t|)
## D.v2x_corr        0.19863954 0.31046378  0.6398155 5.223152e-01
## D.sp_pop_totl     0.71325980 0.29372677  2.4283105 1.519636e-02
## D.ny_gdp_pcap_kd  0.05232954 0.21029570  0.2488379 8.034941e-01
## D.kg_democracy    0.09208301 0.10636850  0.8656981 3.866877e-01
## D.statefailure   -0.02186092 0.01658289 -1.3182821 1.874558e-01
## D.L.nattack       0.34127304 0.03938706  8.6645981 5.653460e-18
```

```r
resid.dat <- data.use %>%
  select(id,year,D.nattack) %>%
  mutate(AH.resid = predict(AH, newdata=data.use)- D.nattack) %>%
  mutate(Lah.resid = lag(AH.resid),
         L2.ah.resid = lag(AH.resid,2),
         .by=id)


## first one should be strong and negative (check)
summary(lm(AH.resid~Lah.resid-1, data=resid.dat))
```

```
##
## Call:
## lm(formula = AH.resid ~ Lah.resid - 1, data = resid.dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6639 -0.3048  0.0402  0.3302  3.2331
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## Lah.resid -0.50497    0.01089  -46.38   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7225 on 6366 degrees of freedom
##   (1247 observations deleted due to missingness)
## Multiple R-squared:  0.2526, Adjusted R-squared:  0.2525
## F-statistic:  2151 on 1 and 6366 DF,  p-value: < 2.2e-16
```

```r
## second should be very zero (could be better but definitely small)
summary(lm(AH.resid~L2.ah.resid-1,  data=resid.dat))
```

```
##
## Call:
## lm(formula = AH.resid ~ L2.ah.resid - 1, data = resid.dat)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0177 -0.3634  0.0069  0.3546  4.0144
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## L2.ah.resid  0.05818    0.01281   4.542 5.68e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8383 on 6195 degrees of freedom
##   (1418 observations deleted due to missingness)
## Multiple R-squared:  0.003319,   Adjusted R-squared:  0.003158
## F-statistic: 20.63 on 1 and 6195 DF,  p-value: 5.675e-06
```

```
## Short-run effect of corruption on terrorism?
## Roughly a 1.4% increase  in terrorist attacks
## for every within standard deviation
deltaMethod(AH, "0.07*D.v2x_corr", vcov=V.AH)
```

```
##                    Estimate        SE     2.5 % 97.5 %
## 0.07 * D.v2x_corr  0.013905  0.021732 -0.028690 0.0565
```

```
## 1 period later  (about a 2% increase)
deltaMethod(AH, "0.07*(D.v2x_corr*D.L.nattack+D.v2x_corr)", vcov=V.AH)
```

```
##                                                 Estimate        SE     2.5 %
## 0.07 * (D.v2x_corr * D.L.nattack + D.v2x_corr)  0.018650  0.029153 -0.038489
##                                                  97.5 %
## 0.07 * (D.v2x_corr * D.L.nattack + D.v2x_corr) 0.0758
```

```
## permanent increase? (about a 2% increase)
deltaMethod(AH, "0.07*(D.v2x_corr/(1-D.L.nattack))", vcov=V.AH)
```

```
##                                       Estimate        SE     2.5 % 97.5 %
## 0.07 * (D.v2x_corr/(1 - D.L.nattack))  0.021109  0.033013 -0.043595 0.0858
```

```
#### Arellano and Bond ####

## step 1 create a balanced the panel using fake data
```

```r
data.use$real.data <- 1
pseudo.data <- data.frame(id=rep(unique(data.use$id), each=48),
                          year=rep(1971:2018))
data.ab <- merge(data.use, pseudo.data, by=c("id",  "year"), all=TRUE)
data.ab$real.data[is.na(data.ab$real.data)] <- 0
dropU <- data.ab %>%
  summarize(across(starts_with("D."),
                   \(x){all(is.na(x))},
                   .names="{.col}"),
            .by=id) %>%
  mutate(id=NULL) %>%
  apply(1, any) %>%
  which()

data.ab <- data.ab %>%
  filter(!id %in% unique(id)[dropU])

## step 2 pull out the data of interest
X <- data.ab %>%
  filter(year >= 1973) %>%
  select(starts_with("D."))

keep <- (1-apply(X, 1, anyNA)) ## create a real measure of sample size
X[apply(X, 1, anyNA),] <- 0 ## make the dropped rows all 0
y <- X$D.nattack
Ly <- X$D.L.nattack
X$D.nattack <- NULL
X$D.L.nattack <- NULL
X <- as.matrix(X)



N <- length(unique(data.ab[data.ab$year>=1973 ,][keep==1,]$id))
print(N)
```

```
## [1] 170
```

```
## useful function for creating multiple lags
## with tidy
lag_multiple <- function(x, n_vec){
  Mat <- sapply(n_vec, lag, x = x)
  colnames(Mat) <- paste0("lag", n_vec)
  as_tibble(Mat)
}

## fill in NA lagged attacks with 0s to create
## the instrument matrix
data.ab$nattack[data.ab$real.data==0] <- 0
lags <- subset(data.ab, select=c("id", "year", "nattack"))
## generate all the lags
lagMat <- lags %>% mutate(lag_multiple(nattack,47:2), .by=id)
lagMat <- lagMat[data.ab$year >= 1973,]
lagMat <- cbind(lagMat, X, keep) #bind in the things that instrument for themselves and
lagMat <- lagMat %>% select(!c(year, nattack))
X <- cbind(Ly, X) ## put the lag back into X

## I suspect there's a better way,
## but this is what I have for now
Z.list <- tapply(lagMat,
                 INDEX=lagMat$id,
                 \(x){
                   zout <- cbind(do.call(bdiag, ## parts of lagMat that are lags of y
                                         apply(x[,2:47],1,
                                               \(y){
                                                 y <- matrix(na.omit(y), nrow=1)
                                                 return(y)
                                               }
                                         )
                   ),
                   as.matrix(x[,48:(ncol(x)-1)])) ## parts that are self-instruments
                   ## Do not ask me why that needs to be a matrix, but it does
                   zout[as.matrix(x[,ncol(x)]==0),] <- 0  ##make sure our dropped rows
```

```
                    return(zout)
                  })
Z <- do.call(rbind, Z.list)


Omega1.hat.inv <- solve(Reduce(`+`,
                               lapply(Z.list,
                                      \(z){
                                        if(nrow(z)>0){
                                          Di <- -cbind(Diagonal(nrow(z)),0) +
                                            cbind(0,Diagonal(nrow(z)))
                                          H <- Di %*% t(Di)
                                          t(z) %*% H %*% z
                                        }else{
                                          0}
                                      })))
ab1.hat <- solve(t(X) %*% Z %*%  Omega1.hat.inv %*% t(Z) %*% X) %*%
  (t(X) %*% Z %*%  Omega1.hat.inv %*% t(Z) %*% y)
ab1.hat[1:6,]
```

```
##                Ly       D.v2x_corr    D.sp_pop_totl D.ny_gdp_pcap_kd
##     0.6180923901     0.4967736838     0.4345174945    -0.0006811801
##   D.kg_democracy   D.statefailure
##     0.0909438585     0.0524024429
```

```
e.ab1.hat <- y - X %*% ab1.hat
sigma2.hat <- crossprod(e.ab1.hat)/ (sum(keep)) ## consistent
V.ab0 <- solve(t(X) %*% Z %*% Omega1.hat.inv %*% t(Z) %*% X)*drop(sigma2.hat)
se.ab0 <- sqrt(diag(V.ab0))
cbind(ab1.hat, se.ab0, ab1.hat/se.ab0)[1:6,]
```

```
## 6 x 3 Matrix of class "dgeMatrix"
##                                    se.ab0
## Ly                0.6180923901 0.02090033 29.573338677
## D.v2x_corr        0.4967736838 0.36232687  1.371064972
## D.sp_pop_totl     0.4345174945 0.22897438  1.897668639
## D.ny_gdp_pcap_kd -0.0006811801 0.18164738 -0.003750013
```

```
## D.kg_democracy    0.0909438585 0.13123024  0.693009897
## D.statefailure    0.0524024429 0.01921219  2.727562776
```

```
## Robust standard errors
Ze <- Z*drop(e.ab1.hat)
Omega2.hat <- Reduce(`+`,
                     lapply(unique(data.ab$id),
                            \(i){
                              Zi <- Ze[data.ab[data.ab$year>=1973,]$id==i,]
                              return(tcrossprod(colSums(Zi)))
                            }
                     )
)

V.ab1 <- solve(t(X) %*% Z %*% Omega1.hat.inv %*% t(Z) %*% X) %*%
  (t(X) %*% Z %*% Omega1.hat.inv %*%Omega2.hat
   %*% Omega1.hat.inv %*% t(Z) %*% X) %*%
  solve(t(X) %*% Z %*% Omega1.hat.inv %*% t(Z) %*% X)
se.ab1 <- sqrt(diag(V.ab1))
cbind(ab1.hat, se.ab0, se.ab1)[1:6,]
```

```
## 6 x 3 Matrix of class "dgeMatrix"
##                                    se.ab0     se.ab1
## Ly                0.6180923901 0.02090033 0.03117205
## D.v2x_corr        0.4967736838 0.36232687 0.38192182
## D.sp_pop_totl     0.4345174945 0.22897438 0.29625159
## D.ny_gdp_pcap_kd -0.0006811801 0.18164738 0.21712866
## D.kg_democracy    0.0909438585 0.13123024 0.11535991
## D.statefailure    0.0524024429 0.01921219 0.02134814
```

```
## AR tests
ar.dat <- data.ab %>%
  filter(year>=1973) %>%
  select(c(id, year)) %>%
  mutate(ehat =drop(e.ab1.hat)) %>%
  mutate(ehat = ifelse(ehat==0, NA, ehat)) %>%
  group_by(id) %>%
  mutate(L.ehat = lag(ehat),
```

```
        L2.ehat = lag(ehat,2)) %>%
  ungroup()
summary(lm(ehat~L.ehat, data=ar.dat))
```

```
##
## Call:
## lm(formula = ehat ~ L.ehat, data = ar.dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0352 -0.3465 -0.0217  0.3635  4.6763
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.003253   0.009877   0.329    0.742
## L.ehat      -0.579686   0.010286 -56.355   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7881 on 6365 degrees of freedom
##   (1453 observations deleted due to missingness)
## Multiple R-squared:  0.3329, Adjusted R-squared:  0.3328
## F-statistic:  3176 on 1 and 6365 DF,  p-value: < 2.2e-16
```

```
summary(lm(ehat~L2.ehat, data=ar.dat))
```

```
##
## Call:
## lm(formula = ehat ~ L2.ehat, data = ar.dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5855 -0.4029  0.0115  0.4571  5.1552
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.003618   0.012257   0.295    0.768
```

```
## L2.ehat     0.090884   0.012775   7.114 1.25e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9648 on 6194 degrees of freedom
##   (1624 observations deleted due to missingness)
## Multiple R-squared:  0.008105,   Adjusted R-squared:  0.007945
## F-statistic: 50.61 on 1 and 6194 DF,  p-value: 1.253e-12
```

```
## Sargan on the first step
sarganJ <- (t(e.ab1.hat) %*% Z %*% Omega1.hat.inv %*%
              t(Z) %*% e.ab1.hat)/sigma2.hat
sarganJ
```

```
## 1 x 1 Matrix of class "dgeMatrix"
##          [,1]
## [1,] 827.3293
```

```
pchisq(drop(sarganJ), df=ncol(Z)-ncol(X), lower=FALSE) ## good
```

```
## [1] 1
```

```
##### strength#####
## We won't be able to invert A V_Cluster A' with this many instruments, so
## we'll use regular robust SEs
g.hat <- solve(crossprod(Z)) %*% t(Z) %*% X[,"Ly"]
Ze <- Z*drop(e.ab1.hat)
bread <- solve(crossprod(Z))
meat <- crossprod(Ze)
V1.white <- bread %*% meat %*% bread

A <- cbind(Diagonal(ncol(Z)-ncol(X)+1),
           Matrix(0, nrow=ncol(Z)-ncol(X)+1, ncol=ncol(X)-1))
Fstat.white <- t(A %*% g.hat ) %*%
  solve(A %*% V1.white %*% t(A)) %*%
  (A %*% g.hat ) / nrow(A)
Fstat.white
```

```
## 1 x 1 Matrix of class "dgeMatrix"
```

```
##           [,1]
## [1,] 1.311756
```

```
## Not great



#### effects ####

## short run: about a 3.5% increase
c(0.07*ab1.hat["D.v2x_corr",], 0.07*sqrt(se.ab1["D.v2x_corr"]^2))
```

```
## D.v2x_corr D.v2x_corr
## 0.03477416 0.02673453
```

```
## One year later about a 5.6% increase
D1 <- c(1+ab1.hat["Ly",], ab1.hat["D.v2x_corr",])
c(0.07*(ab1.hat["D.v2x_corr",]+ab1.hat["D.v2x_corr",]*ab1.hat["Ly",]),
        0.07*drop(sqrt(D1 %*% V.ab1[2:1, 2:1] %*% D1)))
```

```
## D.v2x_corr
## 0.05626780 0.04308581
```

```
## Permanent increase in corruption: about a 7% increase in terrorism
Dlr <- c(1/(1-ab1.hat["Ly",]), ab1.hat["D.v2x_corr",]/(1-ab1.hat["Ly",])^2)
c(0.07*(ab1.hat["D.v2x_corr",]/(1-ab1.hat["D.v2x_corr",])),
        0.07*drop(sqrt(Dlr %*% V.ab1[2:1, 2:1] %*% Dlr)))
```

```
## D.v2x_corr
## 0.06910242 0.06911378
```

We could move on to the two-step, but we'd run in to trouble with inverting $\hat{\Omega}_2$. We would either have to accept the one-step or use fewer instruments (i.e., limit ourselves to just 2 lags per period).

```
#### Fewer lags but with the two-step ####
Z.list <- tapply(lagMat,
                 INDEX=lagMat$id,
                 \(x){
                   zout <- cbind(
                     do.call(bdiag,
```

```
                              apply(x[,2:47],1,
                                    \(y){
                                      y <- matrix(na.omit(y), nrow=1)
                                      if(ncol(y)>2){
                                        y <- y[,(ncol(y)-1):ncol(y),drop=FALSE]
                                      }
                                      return(y)
                                    }
                              )
                        ),
                        as.matrix(x[,48:(ncol(x)-1)]))
                    zout[as.matrix(x[,ncol(x)]==0),] <- 0
                    ## Do not ask me why that needs to be a matrix, but it does
                    return(zout)
                  })
Z <- do.call(rbind, Z.list)

Omega1.hat.inv <- solve(Reduce(`+`,
                               lapply(Z.list,
                                      \(z){
                                        Di <- -cbind(Diagonal(nrow(z)),0) +
                                          cbind(0,Diagonal(nrow(z)))
                                        H <- Di %*% t(Di)
                                        t(z) %*% H %*% z
                                      })))
ab1.hat <- solve(t(X) %*% Z %*%  Omega1.hat.inv %*% t(Z) %*% X) %*%
  (t(X) %*% Z %*%  Omega1.hat.inv %*% t(Z) %*% y)
ab1.hat[1:6,]
```

```
##                Ly       D.v2x_corr    D.sp_pop_totl D.ny_gdp_pcap_kd
##        0.41225665        0.44955541       0.67656300       0.03441324
##   D.kg_democracy   D.statefailure
##        0.15763490       -0.01229968
```

```
e.ab1.hat <- y - X %*% ab1.hat
sigma2.hat <- drop(crossprod(e.ab1.hat))/ (sum(keep)) ## consistent
```

```
V.ab0 <- solve(t(X) %*% Z %*% (Omega1.hat.inv/sigma2.hat) %*% t(Z) %*% X)
se.ab0 <- sqrt(diag(V.ab0))
cbind(ab1.hat, se.ab0, ab1.hat/se.ab0)[1:6,]
```

```
## 6 x 3 Matrix of class "dgeMatrix"
##                                   se.ab0
## Ly                0.41225665 0.02949930 13.9751326
## D.v2x_corr        0.44955541 0.46348750  0.9699407
## D.sp_pop_totl     0.67656300 0.29609473  2.2849545
## D.ny_gdp_pcap_kd  0.03441324 0.27065955  0.1271459
## D.kg_democracy    0.15763490 0.14487474  1.0880772
## D.statefailure   -0.01229968 0.02153233 -0.5712192
```

```
Ze <- Z*drop(e.ab1.hat)
Omega2.hat <- Reduce(`+`,
                     lapply(unique(data.ab$id),
                            \(i){
                              Zi <- Ze[data.ab[data.ab$year>=1973,]$id==i,]
                              return(tcrossprod(colSums(Zi)))
                            }
                     )
)



bread <- solve(t(X) %*% Z %*% Omega1.hat.inv %*% t(Z) %*% X)
meat <-  (t(X) %*% Z %*% Omega1.hat.inv %*%Omega2.hat %*%
     Omega1.hat.inv %*% t(Z) %*% X)
V.ab1 <- bread  %*% meat %*% bread
se.ab1 <- sqrt(diag(V.ab1))
cbind(ab1.hat, se.ab0, se.ab1)[1:6,]
```

```
## 6 x 3 Matrix of class "dgeMatrix"
##                                   se.ab0     se.ab1
## Ly                0.41225665 0.02949930 0.03716272
## D.v2x_corr        0.44955541 0.46348750 0.31907013
## D.sp_pop_totl     0.67656300 0.29609473 0.27480576
## D.ny_gdp_pcap_kd  0.03441324 0.27065955 0.20964542
```

```
## D.kg_democracy    0.15763490 0.14487474 0.10859522
## D.statefailure   -0.01229968 0.02153233 0.01705089
```

```
## AR tests
ar.dat <- data.ab %>%
  filter(year>=1973) %>%
  select(c(id, year)) %>%
  mutate(ehat =drop(e.ab1.hat)) %>%
  mutate(ehat = ifelse(ehat==0, NA, ehat)) %>%
  group_by(id) %>%
  mutate(L.ehat = lag(ehat),
         L2.ehat = lag(ehat,2)) %>%
  ungroup()
summary(lm(ehat~L.ehat, data=ar.dat))
```

```
##
## Call:
## lm(formula = ehat ~ L.ehat, data = ar.dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3728 -0.3268 -0.0384  0.3184  4.6609
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0004013  0.0092095   0.044    0.965
## L.ehat      -0.5310371  0.0106914 -49.670   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7349 on 6365 degrees of freedom
##   (1453 observations deleted due to missingness)
## Multiple R-squared:  0.2793, Adjusted R-squared:  0.2792
## F-statistic:  2467 on 1 and 6365 DF,  p-value: < 2.2e-16
```

```
summary(lm(ehat~L2.ehat, data=ar.dat))
```

```
##
```

```
## Call:
## lm(formula = ehat ~ L2.ehat, data = ar.dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1608 -0.3686 -0.0002  0.3829  5.0448
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.00161    0.01102   0.146    0.884
## L2.ehat      0.06994    0.01280   5.464 4.85e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8675 on 6194 degrees of freedom
##   (1624 observations deleted due to missingness)
## Multiple R-squared:  0.004796,   Adjusted R-squared:  0.004636
## F-statistic: 29.85 on 1 and 6194 DF,  p-value: 4.847e-08
```

```
## Sargan on the first step -- Not as good as before
sarganJ <- t(e.ab1.hat) %*% Z %*%
  (Omega1.hat.inv/sigma2.hat) %*%
  t(Z) %*% e.ab1.hat
sarganJ
```

```
## 1 x 1 Matrix of class "dgeMatrix"
##          [,1]
## [1,] 136.7253
```

```
pchisq(drop(sarganJ), df=ncol(Z)-ncol(X), lower=FALSE)
```

```
## [1] 0.001096007
```

```
## The two step
Omega2.hat.inv <- solve(Omega2.hat)
ab2.hat <- solve(t(X) %*% Z %*%  Omega2.hat.inv %*% t(Z) %*% X) %*%
  (t(X) %*% Z %*%  Omega2.hat.inv %*% t(Z) %*% y)
ab2.hat[1:6]
```

```
## [1]  0.390162170  0.439056522  0.622045171 -0.073917902  0.085366556
## [6]  0.001898386
```

```
e.ab2.hat <- y - X %*% ab2.hat

V.ab2 <- solve(t(X) %*% Z %*% Omega2.hat.inv %*% t(Z) %*% X)
se.ab2 <- sqrt(diag(V.ab2))
cbind(ab2.hat, se.ab2, ab2.hat/se.ab2)[1:6,]
```

```
## 6 x 3 Matrix of class "dgeMatrix"
##                                    se.ab2
## Ly                 0.390162170 0.01736289 22.4710433
## D.v2x_corr         0.439056522 0.16601539  2.6446737
## D.sp_pop_totl      0.622045171 0.18333750  3.3928965
## D.ny_gdp_pcap_kd -0.073917902 0.12618508 -0.5857896
## D.kg_democracy     0.085366556 0.06932483  1.2313995
## D.statefailure     0.001898386 0.00791760  0.2397679
```

```
## two step with correction
Dgi.dtheta <- -t(Z) %*% X
Gn <- Dgi.dtheta
gn <- (t(Z) %*% e.ab2.hat)

D <- matrix(0, ncol(X),ncol(X))
for(j in 1:ncol(X)){
  GAMMA.j <- (t(Z)%*%drop(e.ab2.hat) %*% t(Dgi.dtheta[,j,drop=FALSE]))
  dOmega.j <- (GAMMA.j + t(GAMMA.j))
  D[,j] <- drop(solve(t(Gn) %*% Omega2.hat.inv %*% Gn) %*%
                  t(Gn) %*%
                  (Omega2.hat.inv  %*% dOmega.j %*% Omega2.hat.inv)
                %*% gn )/N
}

V2.corrected <- (V.ab2) +
  D %*% (V.ab2) +
  (V.ab2) %*% t(D) +
  D %*% (V.ab1) %*% t(D)
```

```
se.ab2.c <- sqrt(diag(V2.corrected))
cbind(ab2.hat, se.ab2, se.ab2.c)[1:6,]
```

```
## 6 x 3 Matrix of class "dgeMatrix"
##                                    se.ab2   se.ab2.c
## Ly                0.390162170 0.01736289 0.03669105
## D.v2x_corr        0.439056522 0.16601539 0.33439425
## D.sp_pop_totl     0.622045171 0.18333750 0.33761271
## D.ny_gdp_pcap_kd -0.073917902 0.12618508 0.24033040
## D.kg_democracy    0.085366556 0.06932483 0.12944239
## D.statefailure    0.001898386 0.00791760 0.01678014
```

```
## AR tests
ar.dat <- data.ab %>%
  filter(year>=1973) %>%
  select(id, year) %>%
  mutate(ehat = drop(e.ab2.hat)) %>%
  mutate(L.ehat = lag(ehat),
         L2.ehat = lag(ehat,2),
         .by=id)
summary(lm(ehat~L.ehat, data=ar.dat))
```

```
##
## Call:
## lm(formula = ehat ~ L.ehat, data = ar.dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3364 -0.2427 -0.0016  0.2150  4.6389
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.001648   0.007658   0.215     0.83
## L.ehat      -0.523454   0.009836 -53.220   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.6698 on 7648 degrees of freedom
##   (170 observations deleted due to missingness)
## Multiple R-squared:  0.2703, Adjusted R-squared:  0.2702
## F-statistic:  2832 on 1 and 7648 DF,  p-value: < 2.2e-16
```

```
summary(lm(ehat~L2.ehat, data=ar.dat))
```

```
##
## Call:
## lm(formula = ehat ~ L2.ehat, data = ar.dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1324 -0.2364 -0.0010  0.2534  4.9987
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.001034   0.009093   0.114    0.909
## L2.ehat     0.065957   0.011710   5.632 1.84e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7864 on 7478 degrees of freedom
##   (340 observations deleted due to missingness)
## Multiple R-squared:  0.004224,   Adjusted R-squared:  0.004091
## F-statistic: 31.72 on 1 and 7478 DF,  p-value: 1.842e-08
```

```
## Sargan on the second step
sarganJ <- t(e.ab2.hat) %*% Z %*% Omega2.hat.inv %*% t(Z) %*% e.ab2.hat
sarganJ
```

```
## 1 x 1 Matrix of class "dgeMatrix"
##          [,1]
## [1,] 115.3366
```

```
pchisq(drop(sarganJ), df=ncol(Z)-ncol(X), lower=FALSE)
```

```
## [1] 0.0371866
```

```
## better




## strength?
g.hat <- solve(crossprod(Z)) %*% t(Z) %*% X[,"Ly"]
Ze <- Z*drop(e.ab2.hat)
bread <- solve(crossprod(Z))
meat <- Reduce(`+`,
                     lapply(unique(data.ab$id),
                            \(i){
                              Zi <- Ze[data.ab[data.ab$year>=1973,]$id==i,]
                              return(tcrossprod(colSums(Zi)))
                            }
                     )
)
Vcl <- bread %*% meat %*% bread



A <- cbind(Diagonal(ncol(Z)-ncol(X)+1),
           Matrix(0, nrow=ncol(Z)-ncol(X)+1, ncol=ncol(X)-1))
Fstat.cl<- (t(A %*% g.hat ) %*%
  solve(A %*% Vcl %*% t(A)) %*%
  (A %*% g.hat )) / nrow(A)
Fstat.cl
```

```
## 1 x 1 Matrix of class "dgeMatrix"
##          [,1]
## [1,] 36.45341
```

```
## Not bad
```

This seems like a lot of work, so you're probably, wisely, asking, is their a good canned version? The answer is a little disappointing. Honestly, this is one of those cases where you're better off with Stata. The functions `xtabond` and `xtabond2` are very good. The packaged R versions are iffy.

This is not to crap on anyone specifically, but to demonstrate the value of learning how to do these things so you can verify that a package does what you think it should be doing.

2.3.4 Weak instruments and Blundell and Bond

Weak instruments can be an issue in dynamic panels to see this consider fitting the simplest AR(1) with the AH estimator. The model is

$$\begin{aligned} y_{it} &= \rho y_{it-1} + \alpha_i + \varepsilon_{it} \\ \Delta y_{it} &= \rho \Delta y_{it-1} + \Delta \varepsilon_{it}, \end{aligned}$$

and the reduced-form first-stage relation is

$$\Delta y_{it-1} = \gamma y_{it-2} + u_{it}.$$

In this case, the "true" γ takes the standard form for simple regression (covariance over variance), such that

$$\begin{aligned} \gamma &= \frac{\mathrm{E}[y_{it-2}\Delta y_{it-1}]}{\mathrm{E}[y_{it-2}^2]} \\ &= \frac{\mathrm{E}[y_{it-2}\left((\rho-1)y_{it-2} + \alpha_i + \varepsilon_{it-1}\right)]}{\mathrm{E}[y_{it-2}^2]} \\ &= \frac{\mathrm{E}[y_{it-2}^2(\rho-1) + \alpha_i y_{it-2} + y_{it-2}\varepsilon_{it-1}]}{\mathrm{E}[y_{it-2}^2]} \\ &= (\rho-1) + \frac{\mathrm{E}[\alpha_i y_{it-2}]}{\mathrm{E}[y_{it-2}^2]}. \end{aligned}$$

We will assume that y_i is stationary with $y_{i0} = \alpha_i$ such that we can rewrite the above as

$$y_{it-2} = \sum_{s=0}^{\infty}(u_i + \varepsilon_{it-2-s})\rho^s.$$

Then we can find

$$\begin{aligned}
\mathrm{E}[\alpha_i y_{it-2}] &= \mathrm{E}\left[\alpha_i \sum_{s=0}^{\infty}(\alpha_i + \varepsilon_{it-2-s})\rho^s\right] \\
&= \mathrm{E}\left[\alpha_i \frac{\alpha_i}{1-\rho}\right] + \mathrm{E}\left[\sum_{s=0}^{\infty}(\varepsilon_{it-2-s})\rho^s\right] \\
&= \frac{\sigma_\alpha^2}{1-\rho} \\
\mathrm{E}[y_{it-2}^2] &= \mathrm{E}\left[\left(\sum_{s=0}^{\infty}(\alpha_i + \varepsilon_{it-2-s})\rho^s\right)^2\right] \\
&= \frac{\sigma_\alpha^2}{(1-\rho)^2} + \frac{\sigma_\varepsilon^2}{1-\rho^2}
\end{aligned}$$

The ratio then becomes

$$\gamma = (\rho - 1)\frac{(1-\rho)(1+\rho)}{(1-\rho)(1+\rho) + \sigma_\alpha^2/\sigma_\varepsilon^2}.$$

So when will this be a weak instrument?

1. If ρ is close to one, then $\gamma \to 0$ (i.e., we the data are close to non-stationary)
2. If $\sigma_\alpha^2/\sigma_\varepsilon^2$ gets very large (the unit-specific effects swamp the noise)

We can't really know if either of these conditions is true ahead of time. However, we can do things like check the estimated γ (first stage F). Checking the estimated ρ is a big more challenging, but panel unit root tests are a possibility (more on this later).

Blundell and Bond look to reduce the issues with weak instruments by flipping AH and AB around. Rather than starting with an FD model and looking for instruments that are uncorrelated with $\Delta\varepsilon_{it}$, they leave the model as is and ask, can we find instruments uncorrelated with $\alpha_i + \varepsilon_{it}$.

To be precise we start with

$$y_{it} = \rho y_{it-1} + \beta' x_{it} + \alpha_i + \varepsilon_{it},$$

and now we're going to ask, can we use Δy_{it-1} as the instrument?

1. Relevance: Δy_{it-1} and y_{it-1} are for sure related
2. Validity: Does $\mathrm{E}[\Delta y_{it-1}\alpha_i] + \mathrm{E}[\Delta y_{it-1}\varepsilon_{it}] = 0$ The latter term is 0 if ε_{it} are iid. So the key is whether the former is true.

A sufficient condition for validity, according to BB, is

$$\mathrm{E}\left[\left(y_{i1} - \frac{\alpha_i}{1-\rho}\right)\alpha_i\right] = 0.$$

Basically, what this says is that if the difference between the initially observed y_{i1} and its conditional mean $\frac{\alpha_i}{1-\rho}$ is uncorrelated with $\frac{\alpha_i}{1-\rho}$ then we're good to go. This condition will hold if the data are actually stationary. That's good news, so we will now write the BB moment conditions as

$$\mathrm{E}[\Delta y_{it-1}(y_{it} - \rho y_{it-1} - \beta' x_{it})] = 0\,\mathrm{E}[\Delta x_{it-1}(y_{it} - \rho y_{it-1} - \beta' x_{it})] \quad = 0.$$

With AB we used levels to instrument for differences, now we're using differences to instrument for levels, so for unit i the instrument matrix is

$$Z_i = \begin{bmatrix} \Delta y_{i2} & 0 & 0 & \dots & | & \Delta x_{i3} \\ 0 & \Delta y_{i3} & 0 & \dots & | & \Delta x_{i4} \\ 0 & 0 & \ddots & \dots & | & \Delta x_{i5} \\ \vdots & \vdots & \vdots & \dots & | & \vdots \end{bmatrix}.$$

But wait, there's more! BB aren't satisfied with just saying using this approach. They want to combine it with AB. This is a case where they really do just shout YOLO and say let's use all the information from both, because we think our instruments will be strong enough and they have all those other instruments to build in efficiency.

So they build what's called a "system estimator" for this model where they have

$$\begin{aligned} \mathbf{y}_i &= (\Delta y_i', y_i')' \\ \mathbf{x}_i^* &= (\Delta \mathbf{x}_i', \mathbf{x}_i')' \\ \mathbf{Z}_i &= \begin{bmatrix} Z_i^{AB} & 0 \\ 0 & Z_i \end{bmatrix}, \end{aligned}$$

where Z_i^{AB} is the AB estimator instrument matrix. The moment conditions are now

$$\mathrm{E}[\mathbf{Z}_i'(\mathbf{y}_i - \mathbf{x}_i^*\theta)] = 0.$$

We can then fit this model with GMM.

The first stage weighting matrix still assumes iid errors and so we get

$$\Omega_1 = \sum_{i=1}^{N} \mathbf{Z}_i' \mathbf{H} \mathbf{Z}_i$$
$$\mathbf{H} = \begin{bmatrix} H & 0 \\ 0 & I_{T-2} \end{bmatrix}.$$

As in the AB one-step, this first stage matrix only depends on data, no estimates. The one-step BB is then given as

$$\hat{\theta}_{BB} = (\mathbf{X}^{*'} \mathbf{Z} \Omega^{-1} \mathbf{Z}' \mathbf{X}^*)^{-1} (\mathbf{X}^{*'} \mathbf{Z} \Omega^{-1} \mathbf{Z}' \mathbf{y}).$$

with robust variance matrix

$$\widehat{\text{Var}}_1(\hat{\theta}_{BB}) = (\mathbf{X}^{*'} \mathbf{Z} \Omega^{-1} \mathbf{Z}' \mathbf{X}^*)^{-1} (\mathbf{X}^{*'} \mathbf{Z} \Omega^{-1} \widehat{\Omega}_2 \Omega^{-1} \mathbf{Z}' \mathbf{X}^*) (\mathbf{X}^{*'} \mathbf{Z} \Omega^{-1} \mathbf{Z}' \mathbf{X}^*)^{-1},$$

where

$$\widehat{\Omega}_2 = \sum_{i=1}^{N} \mathbf{Z}_i' \hat{\varepsilon}_i \hat{\varepsilon}_i' \mathbf{Z}_i.$$

As before, we can build a two-step estimator

$$\hat{\theta}^2_{BB} = (\mathbf{X}^{*'} \mathbf{Z} \widehat{\Omega}_2^{-1} \mathbf{Z}' \mathbf{X}^*)^{-1} (\mathbf{X}^{*'} \mathbf{Z}' \widehat{\Omega}_2^{-1} \mathbf{Z}' \mathbf{y}),$$

with variance matrix

$$\widehat{\text{Var}}(\hat{\theta}^2_{BB}) = (\mathbf{X}^{*'} \mathbf{Z}' \widehat{\Omega}_2^{-1} \mathbf{Z}' \mathbf{X}^*)^{-1},$$

which we may want to correct as with the AB.

2.3.5 A short note on non-stationary panels

Everything we've done so far as relied on stationarity within each unit. So it makes sense that we would want to (a) be able to test for that, and (b) know what we can do if that fails.

Regarding testing, there have been many attempts to extend what we know from the single time series case to panels. If you don't remember what these tests look like, we can review the Dickey-Fuller framework. For a single time-series we have something like

$$y_t = \rho y_{t-1} + \varepsilon_t,$$

and we want to know if $\rho = 1$ (non-stationary). However, if it does then we have a problem

because OLS may not be a good choice under the null of non-stationary data (absent some other structure). We will subtract y_{t-1} from both sides to get

$$\Delta y_t = (\rho - 1)y_{t-1} + \varepsilon_t.$$

We can find this model with OLS and test the hypothesis that $\rho - 1 = 0$, but again because of the unit-root if the null is true, we have to use different tests statistic values derived by scholars past.

Many of the extensions to the panel setting require a balanced panel and assume a common ρ across all panels. We can use these if we restrict ourselves to just the balanced sample or we can restrict our attention to the tests that don't require strong balance (and also don't require a common ρ). Of this latter group, three jump to mind as potentially useful to us.

First, we'll describe the **Levin-Lin-Chu** (LLC) test as it's probably the most commonly used of these tests. This is best used for balanced panels, but as you'll see in your next problem set, sometimes people shoe-horn it in for unbalanced cases. The asymptotics are derived for $N/T \to 0$, so both dimensions can be large but N should be larger. We start with a Dickey-Fuller looking setup with time-demeaned y_{it}:

$$\Delta y_{it} = \alpha_i + \phi_i y_{it-1} + \sum_{\ell=1}^{L} \theta_{i\ell} \Delta y_{it-\ell} + \tau_i t + \varepsilon_{it}. \tag{2.1}$$

The time demeaning is not technically required, but is generally recommended by seemingly all the papers in this field. The null hypothesis is that a common $\phi = 0$ (unit root) against the alternative of stationarity. As in regular DF case, the test statistics are not distributed t and are not centered at 0. So like with Dickey-Fuller, we will have to turn to tables to assess critical t values. Before starting the test you need to make decisions about a) constant terms (set $\alpha_i = 0$?), b) lag length (set $L = 0, 1, \ldots$), and c) time trend (set $\tau_i = 0$).
All of these imply different kinds of unit roots. Theory should guide this, although lag length is probably a good thing to experiment with. Generally, a good start is to include constants, set $L = 0$, and not include the trend, but try things and think about it. The residual terms should be unserially correlated, so increase lags to get that.

For each unit,

1. Fit Eq. 2.1 and save $\theta_{i\ell}$. If you're setting $L = 0$ (common) you can skip this step.
2. Regress Δy_{it} on 1, $\Delta y_{it-\ell}$, and t (if using), save the residuals, $\hat{e}_{it}$.
3. Regress y_{t-1} on 1, $\Delta y_{it-\ell}$, and t (if using), save the residuals, $\hat{v}_{it-1}$
4. Regress $\hat{e}_{it}$ on v_{it-1}, save the variance of the residuals $\hat{\sigma}^2_{\varepsilon,i}$. Create $\tilde{e}_{it} = \hat{e}_{it}/\hat{\sigma}_{\varepsilon,i}$ and

$\tilde{v}_{it-1} = \hat{v}_{it-1}/\hat{\sigma}_{\varepsilon,i}$

5. Estimate the ratio of long-run to short-run standard deviations

$$\hat{s}_i = \hat{\sigma}_{y,i}/\hat{\sigma}_{\varepsilon,i} = \frac{1}{\left|1 - \sum_\ell \hat{\theta}_{i\ell}\right|}.$$

   If not including any lags this becomes 1.

We can now move to the full panel again,

1. Regress $\tilde{e}_{it}$ on $\tilde{v}_{it-1}$. The resulting estimate is an estimate of the overall ϕ. Save the standard error of this estimate $s.e.(\hat{\phi})$ and the naive t statistic $t_0 = \hat{\phi}/s.e.(\hat{\phi})$.
2. Create the adjusted test statistic:

$$t_{llc} = \frac{t_0 - \bar{s}(NT)s.e.(\hat{\phi})\mu^*(T)/\hat{\sigma}^2_{\tilde{\varepsilon}}}{\sigma^*(T)},$$

   where

- $\bar{s} = N^{-1}\sum_i \hat{s}_i$
- $\mu^*(T)$ and $\sigma^*(T)$ come from LLC's paper and change with T and modeling choices (i.e., whether we include constants in the main regression or a time trend).
- $\hat{\sigma}^2_{\tilde{\varepsilon}}$ is the estimated variance of the residuals from the regression in step (4) above.

Under the null hypothesis t_{llc} will be standard normal. Larger, negative values are evidence against he null hypothesis.

Second, we have the **Im-Pesaran-Shin** (IPS) test. This test is valid under $N \to \infty$, fixed T asymptotics. We again start with the Dickey-Fuller from Eq. 2.1, but with no additional lags (trend can be used). The null hypothesis will be that $\phi_i = 0$ for all i, which is to say that **every** unit has a unit root problem. The alternative is that some units are stationary. So the way this works is

1. Estimate ϕ_i for each unit. This can be done sequentially or as a system approach
2. Save the regular t statistic from the hypothesis that $\phi_i = 0$, call it t_i
3. Save an alternative t statistic where you use the sample variance of Δy_{it} to estimate σ^2_ε instead. Call this statistic $\tilde{t}_i$
4. The paper provides critical values for $\bar{t} = \frac{1}{N}\sum_{i=1}^N t_i$
5. The statistic

$$z = \frac{\tilde{t}_i - \mathrm{E}[\tilde{t}]}{\sqrt{\mathrm{Var}(\tilde{t})/N}} \overset{asy}{\sim} N(0,1)$$

They also provide $\mathrm{E}[\tilde{t}]$ and $\mathrm{Var}[\tilde{t}]$ for different values of T and N in the paper. For an unbalanced panel you can use the average T_i or the smallest.

If we want to add additional lags here (i.e., if we think that there is a serial correlation in the main regression), then they recommend a different test statatic.

$$W = \frac{\bar{t} - \frac{1}{N}\sum_{i=1}^{N} \mathrm{E}[t_i(p_i, T_i)]}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} \mathrm{Var}(t_i(p_i, T_i))/N}} \overset{asy}{\sim} N(0, 1).$$

Note that if the same number of lags are used for each panel, this simplifies in balanced panels. These expected values and variances are found in IPS Table 3.

Another appproach is based on meta analysis. In this test (called a Fisher-type) test we exploit a result from the R. A. Fisher to consider p-values from multiple subtests. Specifically, we can test each unit with an augmented Dickey-Fuller or other single time-series test and treat each one as a study in a meta-analysis.

The intuition here is that under the null hypothesis, p-values from the individual cases are uniform. Because we're relying on good individual tests, these are valid for $T \to \infty$. The first test very simple

$$-2\sum_{i=1}^{N} \log(p_i) \sim \chi^2_{2N}.$$

This follows directly from the relationship between the uniform and the χ^2. Likewise, we can exploit inverse uniform sampling to get another test

$$\frac{1}{\sqrt{N}}\sum_{i=1}^{N} \Phi^{-1}(p_i) \sim N(0, 1).$$

Now, the issue here is that as above the p values do not come from a standard t distribution. They come from a distribution specific to the test and we only know so many based on simulation. We can either simulate specifically, or more commonly, use linear interpolation to come up with p values based on the known ones. It's not ideal, but we play the hand we're given.

For an example of these tests, we'll go back to the data on corruption and terrorism. Here we will look at whether there is a unit root in (transformed) terrorist attacks. We will not include additional covariates, additional lags, or a time trend $\beta = 0$

```
library(readstata13)
library(dplyr)
```

```r
library(Matrix)
library(fUnitRoots)
source("panelFunctions.r")

terror <- read.dta13("Rcode/corruption_terrorism/subsample.dta")

#### LLC test ####
data.llc <- terror %>%
  select(id, year, nattack) %>%
  mutate(nattack= nattack-mean(nattack,na.rm=TRUE), .by=year) %>%
  mutate(y=nattack,
         l.y = lag(y),
         D.y = y-l.y,
         .by=id) %>%
  mutate(Ti = length(na.omit(D.y)), .by=id)



N <- length(unique(data.llc$id))
data.llc %>% summarize(mean(Ti), .by=id) %>% summary()
```

```
##        id            mean(Ti)
##  Min.   :  1.00   Min.   : 6.00
##  1st Qu.: 49.50   1st Qu.:46.00
##  Median : 95.00   Median :47.00
##  Mean   : 96.08   Mean   :42.51
##  3rd Qu.:142.50   3rd Qu.:47.00
##  Max.   :192.00   Max.   :47.00
```

```r
e.tilde <- v.tilde <- list()
s<- rep(0, N)
corrs <- rep(0, N)
for(i in 1:N){
  data.i <- data.llc %>% filter(id==unique(data.llc$id)[i])
  fit0 <- lm(D.y~l.y, data=data.i)
  corrs[i] <- lm(fit0$residuals[-length(fit0$residuals)]~
                   fit0$residuals[-1]-1)$coef
```

```
  fit1 <- lm(D.y~1, data=data.i)
  e.i <- predict(fit1, data.i)- data.i$D.y
  fit2 <- lm(l.y~ 1, data=data.i)
  v.i <- predict(fit2, data.i)- data.i$l.y
  fit3 <- lm(e.i~v.i-1)
  s.eps.i <- summary(fit3)$sigma
  e.tilde[[i]] <- e.i/s.eps.i
  v.tilde[[i]] <- v.i/s.eps.i
  lag.coef <- fit0$coefficients[grep(names(fit0$coefficients), pattern="L[1-4].D.y")]
  if(is.null(lag.coef)){lag.coef <- 0}
  s[i] <- 1/abs(1-sum(lag.coef))

}
S <- mean(s)
e.tilde <- unlist(e.tilde)
v.tilde <- unlist(v.tilde)
mean(corrs)
```

```
## [1] -0.06901681
```

```
fit4 <- lm(e.tilde~v.tilde-1)
NT <-  length(fit4$residuals)
t.stat <- summary(fit4)$coef["v.tilde", "t value"]
se <-  summary(fit4)$coef["v.tilde", "Std. Error"]
s2.eps.tilde <- summary(fit4)$sigma^2

### values for mu and sigma star from LLC table 2
### For "model 2" and average T of about 45

t.adj <-( t.stat -  (NT*S*se*(-0.533))/s2.eps.tilde)/0.837
t.adj
```

```
## [1] -6.771681
```

```
pnorm(t.adj)
```

```
## [1] 6.364712e-12
```

```r
## reject the null of unit root



### IPS test ####
data.ips <- terror %>%
  select(id, year, nattack) %>%
  mutate(Ti = length(na.omit(nattack)), .by=id) %>%
  filter(Ti > 10) %>%
  mutate(y=nattack,
         y= y- mean(y,na.rm=TRUE),
         .by=year)%>%
  mutate(l.y = lag(y),
         D.y = y-l.y,
         .by=id)
data.ips.clean <- na.omit(data.ips)
N <- length(unique(data.ips$id))

## System estimator
DeltaN <- sparse.model.matrix(~factor(id) -1 ,data=data.ips.clean)
y <- data.ips.clean$D.y
X <- cbind(data.ips.clean$l.y* DeltaN, DeltaN)
colnames(X)[1:N] <- paste0("phi",1:N)
XX <- solve(crossprod(X))
ests <- XX %*% t(X) %*% y



## t-normal
si <- lapply(split.matrix(y-X%*%ests, data.ips.clean$id),
             \(x){drop(crossprod(x)/(length(x)-2))})
se <- mapply(\(x,y){sqrt(diag(y*solve(crossprod(x))))[2]},
             x=split.matrix(cbind(1,data.ips.clean$l.y), data.ips.clean$id),
             y=si)
phi <- ests[1:N]
t1 <- phi/se
mean(t1) ## critical value for tbar (based on Table 2) -1.67 and the average Ti
```

```
## [1] -3.236714
```

```
## t-tilde
st <- mapply(\(x,y){sqrt(diag(var(y)*solve(crossprod(x))))[2]},
             x=split.matrix(cbind(1,data.ips.clean$l.y), data.ips.clean$id),
             y=split(data.ips.clean$D.y, data.ips.clean$id))
t.tilde <- phi/st
mean(t.tilde)
```

```
## [1] -2.794029
```

```
## Approximating using table 1 in IPS
E.ti.tilde <- -1.47
V.ti.tilde <- 0.6475
z.stat <- (mean(t.tilde) -  E.ti.tilde)/sqrt(V.ti.tilde/N)
pnorm(z.stat)
```

```
## [1] 9.278393e-105
```

```
## Reject the null



ttp <- matrix(0, nrow=N, ncol=3)
## Or regression by regression approach
for(i in 1:N){
  fit.i <- lm(D.y~l.y, data=data.ips,
              subset=id==unique(data.ips$id)[i],
              x=TRUE, y=TRUE)
  t.stat <- summary(fit.i)$coeff[,"t value"]["l.y"]
  V.tilde <- var(fit.i$y) * solve(crossprod(fit.i$x))
  t.til <- fit.i$coefficients["l.y"]/sqrt(diag(V.tilde)["l.y"])

  ## Let's do the Fisher test while we're in here
  pi <- suppressWarnings(adfTest(data.ips[data.ips$id==unique(data.ips$id)[i], ]$y,
                                 lags=0, type="c"))
  pi <- pi@test$p.value
```

```
  ttp[i,] <- c(t.stat, t.til, pi)
}
summary(ttp)
```

```
##        V1                 V2                  V3
##  Min.   :-7.2137   Min.   :-4.96668   Min.   :0.01000
##  1st Qu.:-4.2195   1st Qu.:-3.56783   1st Qu.:0.01000
##  Median :-3.1301   Median :-2.79774   Median :0.03456
##  Mean   :-3.2367   Mean   :-2.79403   Mean   :0.15643
##  3rd Qu.:-2.2733   3rd Qu.:-2.12610   3rd Qu.:0.22561
##  Max.   :-0.0199   Max.   :-0.02012   Max.   :0.95050
```

```
FisherP <- -2*sum(log(ttp[,3]))
FisherZ <-sum(qnorm(ttp[,3]))/sqrt(N)
pchisq(FisherP, lower.tail = FALSE, df=2*N)
```

```
## [1] 1.870097e-70
```

```
pnorm(FisherZ)
```

```
## [1] 7.666104e-82
```

Should we find ourselves in a situation where we have non-stationary data we should be careful about what comes next. Note that as $T \to \infty$, non-stationary y and x can lead to a spurious regression problem wherein we find relationships that are not real, but appear to be because of a common time trend in two variables.

The main thing you want to remember is that we can difference non-stationary variables to produce stationary ones (it may take more than one differencing). There are more complicated/interesting things we can consider like cointegration or error-correction techniques, however, in the interest of covering other things we'll set that aside and refer you to Baltagi for more.[2]

2.4 Attrition and missing data in panel models

So far we've talked about balanced and unbalanced panels as just a "you get what you get," and in many cases that's not too far off. We can't always control when data are present or not, particularly in purely observational settings. In many applied situations we implicitly

[2]https://www.amazon.com/Nonstationary-Cointegration-Dynamic-Advances-Econometrics/dp/0762306882

treat the missing data as missing completely at random (MCAR) and just delete observations where we do not have complete data. MCAR means we assume that process that determines whether observation it is missing is completely independent of the observables. Because the process is independent, we don't need to worry about it as our sample will be representative of the "complete" data set.

To formalize that a bit, let S be a random variable that denotes the missingness of an observation from data $Y = \{Y_{\text{obs}}, Y_{\text{miss}}\}$. The MCAR assumption says that

$$\Pr(S|Y) = \Pr(S).$$

MCAR is a convenient, if often unconvincing, lie we often tell ourselves.
If we want to tell a more convincing lie to ourselves we can consider just the idea of data that are missing at random (MAR). MAR means that the missing processes is independent of the data once we **conditional on the observables.** For example, if Republicans are less likely to answer a question about who they're going to vote for than Democrats (i.e., a "shy-Trump voter" effect) then the data are MAR so long as we have the data on party ID or similar things that predict the missingness from data. Formally, this becomes

$$\Pr(S|Y) = \Pr(S|Y_{\text{obs}}).$$

This is where multiple imputation comes into play as a field.

The third type of missing data is called non-ignorable (NI). Here, the process determining missingness is not independent conditional on the observables so $\Pr(S|Y)$ does not reduce. One example of this that is common to survey panels is selection/attrition bias. In these cases, we lose whole observations because some unobserved process leads people to drop out. For example, if we have multiple waves of standardized testing on a random sample of students, and those who think they're going to do poorly on future rounds opt out of waves 2+, then we have a case where individuals are removing themselves based explicitly on the unobserved data.

So what can we do here? Well first we want to know if the attrition is actually a problem or not. Verbeek and Nijman (1992) propose tests for random (good) versus non-random (bad) attrition. Start with a standard panel model:

$$y_{it} = \beta' x_{it} + \alpha_i + \varepsilon_{it},$$

and define s_{it} as an indicator that denotes whether we observe it. The "complete" data are

$$y = \{y_{it}s_{it}, (1 - s_{it})y_{it}\}$$

Let $c_i = \prod s_{it}$, this will reflect those units that are fully observed. For the within transformation we will have

$$\dot{x}_{it} = \begin{cases} x_{it} - \frac{1}{\sum_t s_{it}} \sum_{t=1}^{T} x_{it}s_{it} & \sum_t s_{it} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Now consider two different within (or other consistent) estimators for β

$$\hat{\beta}_U^w = \left(\sum_{i=1}^{N} \sum_{t=1}^{T} \dot{x}_{it}\dot{x}_{it}' s_{it} \right)^{-1} \sum_{i=1}^{N} \sum_{t=1}^{T} \dot{x}_{it}\dot{y}_{it}s_{it}$$
$$\hat{\beta}_R^w = \left(\sum_{i=1}^{N} c_i \sum_{t=1}^{T} \dot{x}_{it}\dot{x}_{it}' \right)^{-1} \sum_{i=1}^{N} c_i \sum_{t=1}^{T} \dot{x}_{it}\dot{y}_{it}c_i.$$

The first is an unrestricted estimator that considers the full, observed sample. The second is a restricted estimator that based on the balanced sub-sample of complete cases (i.e., where $c_i = 1$). We are interested in when these two estimators will be consistent.

To flip that question, when won't they? For the unrestricted estimator a sufficient condition for this is if $\mathrm{E}[\varepsilon_{it}|s_i, X_i] = 0$. This means that the errors within each unit are fully independent of the missingness process of that unit. The restricted estimator will be consistent due to the iid nature of the units. V&N propose a Hausman test since under the null hypothesis that selection isn't an issue we have two consistent estimators and the unrestricted is more efficient. In this case we get

$$(\hat{\beta}_U^w - \hat{\beta}_R^w)'(\mathrm{Var}(\hat{\beta}_R^w) - \mathrm{Var}(\hat{\beta}_U^w))^{-1}(\hat{\beta}_U^w - \hat{\beta}_R^w) \sim \chi_k^2.$$

An alternative test, by the same authors, proposes fitting the model

$$y_{it} = \beta' x_{it} + \phi s_{it-1} + \alpha_i + \varepsilon_{it}$$

with the within estimator. Under the null hypothesis $(s_{i1}, \ldots, s_{iT})$ is uncorrelated with ε_{it} and so selection in the previous period should say nothing about today's outcomes. Using s_{it+1} instead also works for this reason.

Having identified attrition bias, what can we do about it? Wooldridge describes two ap-

proaches. In the first, we're going to assume that attrition is an absorbing state, such that once someone leaves they do not return. This means that $s_{it} = 1$ implies that $s_{it-1} = 1$. Now let's consider the model in differences such that

$$\Delta y_{it} = \Delta x_{it}'\beta + \delta\varepsilon_{it}$$

and let's consider the selection process in reduced form:

$$s_{it} = \mathbb{I}(\gamma' w_{it} + u_{it})$$

where $u_{it}|\Delta x_{it}, w_{it}, s_{it-1} \sim N(0,1)$. The vector w_{it} should contain variables observed at t for all units where $s_{it-1} = 1$, these can include

1. x_{it-1}
2. Any variable in x_{it} that are observed only if $s_{it-1} = 1$. This would include any lags in x_{it} and any variables with deterministic changes (age, time elapsed, etc)

Note we're being a little loose with notation there so let's be clear that x_{it} includes whatever we're choosing to contemporaneous model y_{it} even if they aren't strictly observed at t. If x_{it} are strictly exogenous then we're just looking at using $w_{it} = x_{it-1}$.

Suppose that Δx_{it} are strictly exogeneity and selection is independent of Δx_{it} once we condition on w_{it}. Additionally, assume that $\Delta\varepsilon_{it}$ and u_{it} are bivariate normal with some correlation. Then by standard properties of the bivariate normal we get

$$\mathrm{E}[\Delta\varepsilon_{it}|\Delta x_{it}, w_{it}, u_{it}, s_{it-1} = 1] = \mathrm{E}[\Delta\varepsilon_{it}|u_{it}, s_{it-1} = 1] = \psi_t u_{it}$$

and from that we get to

$$\begin{aligned}
\mathrm{E}[\Delta y_{it}|\Delta x_{it}, w_{it}, s_{it} = 1] &= \Delta x_{it} + \psi_t \,\mathrm{E}[u_{it}|s_{it} = 1] \\
&= \Delta x_{it} + \psi_t \,\mathrm{E}[u_{it}|u_{it} > -\gamma' w_{it}] \\
&= \Delta x_{it} + \psi_t \frac{\phi(-\gamma' w_{it})}{1 - \Phi(-\gamma' w_{it})} \\
&= \Delta x_{it} + \psi_t \frac{\phi(\gamma' w_{it})}{\Phi(\gamma' w_{it})}.
\end{aligned}$$

This fraction term is known as the **inverse Mill's ratio**.

Now I think we have everything need to move forward.

1. Fit $T - 1$ cross-sectional probit regressions using data $(s_{it}, w_{it})_{t=2}^{T}$. Use these to create

the variable

$$q_{it} = \frac{\phi(\hat{\gamma}' w_{it})}{\Phi(\hat{\gamma}' w_{it})}$$

. If this is infeasible for some reason, you can fit a single probit with some kind of flexible time trend (e.g., polynomials or splines).

2. Fit the regression

$$\begin{aligned}\Delta y_{it} &= \Delta x_{it} + \psi_t q_{it} + \Delta \varepsilon_{it} \\ &= \Delta x_{it} + \sum_{s=2}^{T} \psi_s \mathbb{I}(s = t) q_{is} + \Delta \varepsilon_{it}\end{aligned}$$

Note that the term $\sum_{s=2}^{T} \psi_s \mathbb{I}(s = t) q_{is}$ is an interaction of q_{it} with time dummies.

Having controlled for the endogenous component, everything is now exogenous and the estimator will be consistent for β under these assumptions. This also gives us a post estimation test of whether attrition is a problem by considering the linear hypothesis that $\psi = (\psi_t)_{t=2}^{T} = 0$. Overall, this method is nice for the kinds of linear panel models we've been talking about.

2.4.1 IPW

Another, perhaps more common, alternative is inverse probability weighting (IPW). This presents a nice general framework that actually extends to non-linear panel models too. As before we are looking at a situation where we have non-random attrition that shrinks our overall sample. Likewise, we take as given that the sample at $t = 1$ is a random sample for the initial population.

The main additional assumption we will make is that the observables at $t = 1$ contain all the information we need for selection,

$$\Pr(s_{it} | y_{it}, x_{it}, w_{i1}) = \Pr(s_{it} | w_{i1}).$$

This assumption is called **selection on observables**.

We now proceed in two steps:

1. Estimate the probability of remaining in the sample at time $t > 2$ by regressing $(s_{i2}, \ldots, s_{iT})$ on w_{i1} using a logit or probit. Note that we use the same cross-section of units (those observed at $t = 1$) for each period.
2. In the linear context, we then fit a weighted regression

$$\hat{\beta}_{IPW} = (X' S X)^{-1} X' S y,$$

where $S = \text{diag}(s_{it}/\hat{p}_{it})$, $\hat{p}_{it}$ are the fitted probabilities using the fitted model in step 1, and $\hat{p}_{i1} = 1$. For non-linear models, we weight observation log-likelihoods by these values.

Why does this work? Let β_0 be the value that minimizes the population-level regression problem

$$\beta_0 = \underset{\beta}{\text{argmin}} \sum_{t=1}^{T} \text{E}[(y_{it} - \beta' x_{it})^2]$$

Under standard MLE results $\hat{p}_{it} \xrightarrow{p} p^0_{it}$, so we can work with the true value of p^0_{it}.

Once we have the p's fixed, we need to know is if β_0 also minimizes

$$\sum_{t=1}^{T} \text{E}[(s_{it}/p^0_{it})(y_{it} - \beta' x_{it})^2].$$

Let's apply iterated expectations with respect to y_{it}, x_{it} and w_{i1}

$$\begin{aligned}
\text{E}[(s_{it}/p^0_{it})(y_{it} - \beta' x_{it})^2] &= \text{E}\left[\text{E}[(s_{it}/p^0_{it})(y_{it} - \beta' x_{it})^2 \mid y_{it}, x_{it}, w_{i1}]\right] \\
&= \text{E}\left[\left(\text{E}[s_{it} \mid y_{it}, x_{it}, w_{i1}]/p^0_{it}\right)(y_{it} - \beta' x_{it})^2\right] \\
&= \text{E}\left[\left(\Pr(s_{it} = 1|w_{i1})/p^0_{it}\right)(y_{it} - \beta' x_{it})^2\right] \\
&= \text{E}\left[(y_{it} - \beta' x_{it})^2\right].
\end{aligned}$$

It becomes the same optimization problem applies to the full sample. In other words, the limit of the weighted objective function is identical to the unweighted function from a world without any attrition.

We can loosen this up a little more though. Perhaps it is unsatisfying that we are only using information from one time period to predict when units will exit the sample. So why not use one probit for each period t where we only use units and observations that appear in period $t-1$? Such an approach would depend on a much more believeable assumption about the selection process:

$$\Pr(s_{it}|y_{it}, x_{it}, s_{it-1} = 1, y_{it-1}, x_{it-1}, \ldots, y_{i1}, x_{i1}) = \Pr(s_{it}|s_{it-1} = 1, y_{it-1}, x_{it-1}, \ldots, y_{i1}, x_{i1}).$$

However, the problem here is that these will not be representative samples at each period. Instead, if we continue to assume that attrition is absorbing we can see that we actually need a sequential probability of making it to $t-1$ and then continuing to participate. So this proposal becomes

1. For $t \geq 2$, fit a probit of s_{it} on the sample of individuals still present at $t-1$ with predictors $y_{it-1}, x_{it-1}, \ldots, y_{i1}, x_{i1}$. Save the fitted probabilities as $\hat{\pi}_{it}$
2. Build weights $\hat{p}_{it} = \prod_{s=2}^{t} \hat{\pi}_{is}$.
3. Fit the weighted regression as before.

The main assumption for this procedure to be consistent is that

$$\Pr(s_{it}|y_i, x_i, s_{it-1} = 1) = \Pr(s_{it}|s_{it-1} = 1, y_{it-1}, x_{it-1}, \ldots, y_{i1}, x_{i1}).$$

The difference here is that we now assume that selection is independent of both current and future values of the observables. This is a stronger assumption, but it gets us where we want to go.

```
library(dplyr)
library(lmtest)
library(sandwich)
library(tidyr)
library(car)
mig.dt <- read.csv("Rcode/datasets/migration/mig.long.csv")
mig.dt <- mig.dt %>%
  mutate(complete  = prod(wave.comp), .by=UniqueID)



unrestricted <- lm(mg.asp ~ treat.dum +educ + age + male + bs.exp.now + bs.exp.past + So
                     factor(Region) +factor(wave), data=mig.dt)
coeftest(unrestricted, vcovCL(unrestricted, ~UniqueID))
```

```
##
## t test of coefficients:
##
##                                Estimate Std. Error t value  Pr(>|t|)
## (Intercept)                   4.3866789  0.2671979 16.4173 < 2.2e-16 ***
## treat.dum                    -0.1367273  0.0571137 -2.3940  0.016760 *
## educ                          0.0279660  0.0647949  0.4316  0.666073
## age                          -0.0148757  0.0067756 -2.1955  0.028244 *
## male                         -0.1401403  0.0584771 -2.3965  0.016644 *
## bs.exp.now                   -0.1463530  0.0652047 -2.2445  0.024908 *
## bs.exp.past                   0.0156524  0.0927510  0.1688  0.866005
```

```
## Score                              -0.0029341  0.0026356 -1.1132  0.265738
## factor(Region)Lower River Region -0.0647400  0.1125412 -0.5753  0.565183
## factor(Region)Upper River Region -0.2310350  0.0894570 -2.5826  0.009875 **
## factor(Region)West Coast Region  -0.0768771  0.0694079 -1.1076  0.268163
## factor(wave)2                    -0.1517554  0.0516014 -2.9409  0.003310 **
## factor(wave)3                    -0.1692377  0.0559883 -3.0227  0.002537 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
restricted <- lm(mg.asp ~ treat.dum +educ + age + male + bs.exp.now + bs.exp.past + Scor
                   factor(Region) +factor(wave), data=mig.dt, subset=complete==1)
coeftest(restricted, vcovCL(restricted, ~UniqueID))
```

```
##
## t test of coefficients:
##
##                                    Estimate Std. Error t value  Pr(>|t|)
## (Intercept)                         4.5477949  0.3145546 14.4579 < 2.2e-16 ***
## treat.dum                          -0.1569167  0.0658445 -2.3831  0.017281 *
## educ                               -0.0085585  0.0786876 -0.1088  0.913402
## age                                -0.0159881  0.0079570 -2.0093  0.044671 *
## male                               -0.1334234  0.0675890 -1.9740  0.048548 *
## bs.exp.now                         -0.1431907  0.0750771 -1.9072  0.056666 .
## bs.exp.past                         0.0420951  0.1092260  0.3854  0.699996
## Score                              -0.0040448  0.0030582 -1.3226  0.186154
## factor(Region)Lower River Region -0.0966800  0.1349617 -0.7164  0.473878
## factor(Region)Upper River Region -0.2790431  0.1029782 -2.7097  0.006805 **
## factor(Region)West Coast Region  -0.1027102  0.0793874 -1.2938  0.195925
## factor(wave)2                      -0.1530108  0.0600888 -2.5464  0.010976 *
## factor(wave)3                      -0.1369042  0.0595187 -2.3002  0.021565 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
length(unrestricted$resid)
```

```
## [1] 2009
```

```r
length(restricted$resid)
```

```
## [1] 1621
```

```r
## V&N Hasuman
H.stat <- c(unrestricted$coef-restricted$coef) %*%
  solve(vcov(restricted)- vcov(unrestricted)) %*%
  c(unrestricted$coef-restricted$coef)
H.stat
```

```
##          [,1]
## [1,] 8.696609
```

```r
pchisq(H.stat, df=length(unrestricted$coef), lower=FALSE)
```

```
##           [,1]
## [1,] 0.7954518
```

```r
## that's a good thing

##  But let's keep going to be safe
probit.w2 <- glm(wave.comp~ treat.dum +treat.dum +educ + age + male + bs.exp.now
                 + bs.exp.past + Score+
                   factor(Region), data=mig.dt,
                 subset=wave==2, family=binomial("probit"))
probit.w3 <- glm(wave.comp~ treat.dum +treat.dum +educ + age + male + bs.exp.now
                 + bs.exp.past + Score+
                   factor(Region), data=mig.dt,
                 subset=wave==3, family=binomial("probit"))
lp2 <- predict(probit.w2, newdata=mig.dt[mig.dt$wave==2,])
lp3 <- predict(probit.w3, newdata=mig.dt[mig.dt$wave==3,])
mig.dt <- mig.dt %>%
  mutate(inv.mills2 =  ifelse(wave==2,
                              dnorm(lp2)/pnorm(lp2),
                              0),
         inv.mills3 = ifelse(wave==3,
                             dnorm(lp3)/pnorm(lp3),
                             0))
```

```
attrition.corr <- lm(mg.asp ~ treat.dum + inv.mills2+inv.mills3 +educ +
                       age + male + bs.exp.now + bs.exp.past + Score+
                       factor(Region) +factor(wave), data=mig.dt)
coeftest(attrition.corr, vcovCL(attrition.corr, ~UniqueID))
```

```
##
## t test of coefficients:
##
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       4.4061787  0.2712822 16.2421  < 2e-16 ***
## treat.dum                        -0.1387042  0.0576036 -2.4079  0.01614 *
## inv.mills2                       -0.1721851  0.6759230 -0.2547  0.79895
## inv.mills3                        0.0249273  0.4763873  0.0523  0.95827
## educ                              0.0319183  0.0653488  0.4884  0.62530
## age                              -0.0157290  0.0068577 -2.2936  0.02192 *
## male                             -0.1458591  0.0589857 -2.4728  0.01349 *
## bs.exp.now                       -0.1482837  0.0657858 -2.2540  0.02430 *
## bs.exp.past                       0.0301062  0.0937215  0.3212  0.74807
## Score                            -0.0029733  0.0026565 -1.1193  0.26316
## factor(Region)Lower River Region -0.0551186  0.1150797 -0.4790  0.63202
## factor(Region)Upper River Region -0.2320938  0.0903510 -2.5688  0.01028 *
## factor(Region)West Coast Region  -0.0732967  0.0700322 -1.0466  0.29541
## factor(wave)2                    -0.1272960  0.1258045 -1.0119  0.31173
## factor(wave)3                    -0.1726381  0.2383381 -0.7243  0.46894
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## Wald test on the interactions
linearHypothesis(attrition.corr,
                 c("inv.mills2", "inv.mills3"),
                 vcov=vcovCL(attrition.corr, ~UniqueID))
```

```
## Linear hypothesis test
##
## Hypothesis:
```

```
## inv.mills2 = 0
## inv.mills3 = 0
##
## Model 1: restricted model
## Model 2: mg.asp ~ treat.dum + inv.mills2 + inv.mills3 + educ + age + male +
##     bs.exp.now + bs.exp.past + Score + factor(Region) + factor(wave)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F Pr(>F)
## 1   1959
## 2   1957  2 0.0347 0.9659
```

```
## is this strong or weak evidence against the idea that there is not an attrition pro
```

```
## IPW approach

### we're lucky in the model that none of the covariates actually change over time
### so we can fit this in one step
selection.mod <- glm(wave.comp~ treat.dum +treat.dum +educ + age + male + bs.exp.now
                     + bs.exp.past + Score+
                       factor(Region), data=mig.dt, family=binomial("probit"))
mig.dt$p.hat <- predict(selection.mod, newdata=mig.dt, type="response")
mig.dt$S <- mig.dt$wave.comp/mig.dt$p.hat

ipw1 <- lm(mg.asp ~ treat.dum + educ +
             age + male + bs.exp.now + bs.exp.past + Score+
             factor(Region) +factor(wave), data=mig.dt, weights=sqrt(S),
           x=TRUE, y=TRUE)
coeftest(ipw1, vcovCL(ipw1, ~UniqueID))
```

```
##
## t test of coefficients:
```

```
##
##                                   Estimate Std. Error t value  Pr(>|t|)
## (Intercept)                      4.3832605  0.2684785 16.3263 < 2.2e-16 ***
## treat.dum                       -0.1368929  0.0571050 -2.3972  0.016612 *
## educ                             0.0276062  0.0647733  0.4262  0.670010
## age                             -0.0148303  0.0067764 -2.1885  0.028748 *
## male                            -0.1402109  0.0585009 -2.3967  0.016634 *
## bs.exp.now                      -0.1474301  0.0651487 -2.2630  0.023744 *
## bs.exp.past                      0.0171859  0.0925034  0.1858  0.852631
## Score                           -0.0028948  0.0026428 -1.0953  0.273508
## factor(Region)Lower River Region -0.0622442  0.1123424 -0.5541  0.579601
## factor(Region)Upper River Region -0.2304554  0.0892556 -2.5820  0.009894 **
## factor(Region)West Coast Region  -0.0771618  0.0694211 -1.1115  0.266485
## factor(wave)2                    -0.1491411  0.0515724 -2.8919  0.003871 **
## factor(wave)3                    -0.1692527  0.0558822 -3.0287  0.002487 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## IPW alternative
mig.dt <- mig.dt %>% mutate(L.mg.asp=lag(mg.asp),
                            L2.mg.asp=lag(mg.asp,2),
                            .by=UniqueID)
probit.w2 <- glm(wave.comp~ treat.dum +treat.dum +educ + age + male + bs.exp.now
                 + bs.exp.past + Score+L.mg.asp
                 +  factor(Region), data=mig.dt,
                 subset=wave==2, family=binomial("probit"))
probit.w3 <- glm(wave.comp~ treat.dum +treat.dum +educ + age + male + bs.exp.now
                 + bs.exp.past + Score+ L.mg.asp+ L2.mg.asp
                 + factor(Region), data=mig.dt,
                 subset=wave==3, family=binomial("probit"))

pi2 <- predict(probit.w2, newdata=mig.dt[mig.dt$wave==2,], type="response")
pi3 <- predict(probit.w3, newdata=mig.dt[mig.dt$wave==3,], type="response")

pi.df <- data.frame(p1=1, p2= pi2,  p3=pi2*pi3) %>%
  pivot_longer(cols=everything(),
               names_to="wave", values_to="p.hat.ipw") %>%
```

```
  mutate(UniqueID=mig.dt$UniqueID,
         wave=mig.dt$wave)
mig.dt <- mig.dt %>%
  merge(pi.df) %>%
  mutate(S2 = wave.comp/p.hat.ipw)

ipw2 <- lm(mg.asp ~ treat.dum + educ +
             age + male + bs.exp.now + bs.exp.past + Score+
             factor(Region) +factor(wave), data=mig.dt, weights=sqrt(S2),
           x=TRUE, y=TRUE)
coeftest(ipw2, vcovCL(ipw2, ~UniqueID))
```

```
##
## t test of coefficients:
##
##                                   Estimate Std. Error t value  Pr(>|t|)
## (Intercept)                      4.3228731  0.2739539 15.7796 < 2.2e-16 ***
## treat.dum                       -0.1343176  0.0580774 -2.3127  0.020840 *
## educ                             0.0414000  0.0655155  0.6319  0.527518
## age                             -0.0138185  0.0069399 -1.9912  0.046600 *
## male                            -0.1299166  0.0594159 -2.1866  0.028891 *
## bs.exp.now                      -0.1443654  0.0660005 -2.1873  0.028835 *
## bs.exp.past                      0.0368778  0.0933036  0.3952  0.692705
## Score                           -0.0030022  0.0026776 -1.1212  0.262329
## factor(Region)Lower River Region -0.0861102  0.1156282 -0.7447  0.456532
## factor(Region)Upper River Region -0.2335770  0.0912875 -2.5587  0.010581 *
## factor(Region)West Coast Region  -0.0775988  0.0703155 -1.1036  0.269909
## factor(wave)2                   -0.1538228  0.0517267 -2.9738  0.002977 **
## factor(wave)3                   -0.1574010  0.0565113 -2.7853  0.005399 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2.4.2 A note on two-step estimators

Several of the above are two-step estimators where we use the output of a first stage model as either covariates or weights in a second stage.

In these cases we should account for first stage estimation error to the extent that it makes a

difference.

We don't need to worry about this when

1. The derivatives of the second stage optimization function (i.e., least squares or a likelihood function) doesn't change with respect to the first stage estimates in expectation. This is true in many cases of weighted least squares. This includes the random effects GLS.
2. Only the dependent variable is affected by the first-stage in a mean-0 kind of way, then we just have an extra joint error term (linear models only) that can be addressed through regular standard error corrections like clustering

Otherwise, we need to think about how to adjust the standard errors to reflect our additional uncertainty. Common situations involve like what we have above with a first-stage MLE and then a second-stage M estimator of any kind. An M estimator is one that looks like this

$$\hat{\theta}_M = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} m(y_i; \theta).$$

This clear encompasses MLE (m is the log likelihood), GMM and OLS (m is -1 times the fitting criteria).

For the two-step M, where the second step is least squares we can adapt the above to write

$$\hat{\beta}_M = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} m(\beta; y_i, \hat{\gamma}),$$

where m is the weighted OLS objective function times -1, and $\hat{\gamma}$ is the first stage estimate given by maximizing the log-likelihood L, such that

$$\hat{\gamma} \operatorname{argmax}_{\gamma} \sum_{i=1}^{N} L(\gamma; s_i).$$

The variance of the two-step then becomes

$$Var(\hat{\beta}) = \mathrm{E}[D_{\beta\beta'} m(\beta; y_i, \hat{\gamma})]^{-1} \, \mathrm{E}[g(\theta, \gamma) g(\theta, \gamma)'] \, \mathrm{E}[D_{\beta\beta'} m(\beta; y_i, \hat{\gamma}))]^{-1},$$

where

$$g(\theta, \gamma) = \underbrace{D_{\beta} m(\beta; y_i, \hat{\gamma})}_{\text{Score of } m} - \mathrm{E}[\underbrace{D_{\beta\gamma'} m(\beta; y_i, \hat{\gamma})}_{\text{Jacobian}}] \, \mathrm{E}[\underbrace{D_{\gamma\gamma'} L(\gamma; s_i)}_{\text{Hessian/OPG}}]^{-1} \underbrace{D_{\gamma} L(\gamma; s_i)}_{\text{Score/gradient of } L}.$$

Going back to our first IPW example we can think about what these functions would be.

$$
\begin{aligned}
m(\beta; y_i, \hat{\gamma}) &= \frac{s_{it}}{\Phi(z_{it}'\gamma)}(y_{it} - \beta' x_{it})^2 \\
D_\beta m(\beta; y_i, \hat{\gamma}) &= x_{it}\frac{-2s_{it}(y_{it} - x_{it}\beta)}{\Phi(z_{it}'\gamma)} \\
D_{\beta\beta'} m(\beta; y_i, \hat{\gamma}) &= x_{it}x_{it}'\frac{2s_{it}}{\Phi(z_{it}'\gamma)} \\
D_{\beta\gamma'} m(\beta; y_i, \hat{\gamma}) &= x_{it}z_{it}'\frac{2s_{it}(y_{it} - x_{it}'\beta)\phi(z_{it}'\gamma)}{\Phi(z_{it}'\gamma)^2} \\
L(\gamma; s_i) &= \log(\Phi((2s_{it} - 1)z_{it}'\gamma)) \\
D_\gamma L(\gamma; s_i) &= z_i\frac{(2s_{it} - 1)\phi((2s_{it} - 1)z_{it}'\gamma)}{\Phi((2s_{it} - 1)z_{it}'\gamma)} \\
\mathrm{E}[D_{\gamma\gamma'} L(\gamma; s_i)] &= -\mathrm{E}[D_\gamma L(\gamma; s_i) D_\gamma L(\gamma; s_i)']
\end{aligned}
$$

```
selection.mod <- glm(wave.comp~ treat.dum +treat.dum +educ + age + male + bs.exp.now
                     + bs.exp.past + Score+
                       factor(Region), data=mig.dt, family=binomial("probit"),
                     y=TRUE, x=TRUE)
mig.dt$p.hat <- predict(selection.mod, newdata=mig.dt, type="response")
mig.dt$S <- mig.dt$wave.comp/mig.dt$p.hat

ipw1 <- lm(mg.asp ~ treat.dum + educ +
             age + male + bs.exp.now + bs.exp.past + Score+
             factor(Region) +factor(wave), data=mig.dt, weights=sqrt(S),
           x=TRUE, y=TRUE)
coeftest(ipw1, vcovCL(ipw1, ~UniqueID))
```

```
##
## t test of coefficients:
##
##                                Estimate Std. Error t value  Pr(>|t|)
## (Intercept)                   4.3832605  0.2684785 16.3263 < 2.2e-16 ***
## treat.dum                    -0.1368929  0.0571050 -2.3972  0.016612 *
## educ                          0.0276062  0.0647733  0.4262  0.670010
## age                          -0.0148303  0.0067764 -2.1885  0.028748 *
## male                         -0.1402109  0.0585009 -2.3967  0.016634 *
## bs.exp.now                   -0.1474301  0.0651487 -2.2630  0.023744 *
```

```
## bs.exp.past                          0.0171859  0.0925034  0.1858  0.852631
## Score                               -0.0028948  0.0026428 -1.0953  0.273508
## factor(Region)Lower River Region    -0.0622442  0.1123424 -0.5541  0.579601
## factor(Region)Upper River Region    -0.2304554  0.0892556 -2.5820  0.009894 **
## factor(Region)West Coast Region     -0.0771618  0.0694211 -1.1115  0.266485
## factor(wave)2                       -0.1491411  0.0515724 -2.8919  0.003871 **
## factor(wave)3                       -0.1692527  0.0558822 -3.0287  0.002487 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
mig.dt2 <- mig.dt %>% filter(!is.na(educ))
s <- mig.dt2$wave.comp
y <- mig.dt2$mg.asp
X <- with(mig.dt2, cbind(1, treat.dum, educ, age, male, bs.exp.now,
                         bs.exp.past, Score, Region=="Lower River Region",
                         Region == "Upper River Region",
                         Region == "West Coast Region",
                         wave ==2,
                         wave==3))
Z <- X[,1:11]

## A few weird cases of is.na(y) & s==1. let's just
## make those irrelevant
X[is.na(y),] <- 0
y[is.na(y)] <- 0

gamma=selection.mod$coef
beta <- ipw1$coefficients

score <- function(beta, gamma, y, X, Z, s){
  ## D_beta m
  weights <- s/pnorm(drop(Z %*% gamma))
  obj <- drop(y- X%*%beta)* weights *(X)
  return( 2*obj)
}
```

```
d <- function(gamma, s, Z){
  ## D_gamma L
  Z <- (2*s-1)*Z
  ZG <- drop(Z %*% gamma)
  return(Z*dnorm(ZG)/pnorm(ZG))
}


Dscore.b <-  function(gamma, X, Z, s){
  #D_betabeta m
  weights <- s/pnorm(drop(Z %*% gamma))
  mid <-  sqrt(weights)
  obj <- crossprod(X*mid)
  return(-2*obj)
}

Dscore.g <-  function(beta, gamma, y, X, Z, s){
  #D_betagamma m

  ZG <- drop(Z%*%gamma)
  top <- s*drop(y-X%*%beta)*dnorm(ZG)
  signs <- sign(top)
  mid <- sqrt(abs(top))/pnorm(ZG)
  -2* (t(X*mid) %*%  (Z*mid*signs))
}


bread <- solve(Dscore.b(gamma, X,Z,s))


F.hat <- Dscore.g(beta, gamma,y, X,Z,s)
r.hat <- solve(-crossprod(d(gamma, s,Z))) %*% t(d(gamma, s,Z))
D <- score(beta, gamma, y, X, Z,s) - t(F.hat %*% r.hat)
DD <- crossprod(D)

cbind(sqrt(diag(bread %*% (DD) %*% bread)),
```

```
      sqrt(diag( vcovCL(ipw1, ~UniqueID))))
```

```
##                    [,1]        [,2]
##             0.225840410 0.268478517
## treat.dum   0.048305451 0.057105028
## educ        0.054320078 0.064773319
## age         0.005797822 0.006776444
## male        0.049265050 0.058500923
## bs.exp.now  0.054626198 0.065148719
## bs.exp.past 0.076380827 0.092503388
## Score       0.002166975 0.002642842
##             0.098765025 0.112342412
##             0.076078177 0.089255589
##             0.058512617 0.069421052
##             0.055585545 0.051572410
##             0.059915906 0.055882182
```

An appropriate bootstrap would also be acceptable, but be sure to bootstrap **the whole** process

```
library(matrixStats)
boot.sample <- mig.dt %>%
  slice(as.numeric(names(selection.mod$residuals)))

B <- 100
## draw all individuals ahead of time to save a little time
ids <- unique(boot.sample$UniqueID)
idx <- matrix(sample(unique(ids), replace=TRUE, size=length(ids)*B), ncol=B)
data.base <- split(boot.sample, boot.sample$UniqueID)
out <- matrix(0, nrow=B, ncol=length(ipw1$coefficients))

for(b in 1:B){
  boot.dat <- do.call(rbind, data.base[as.character(idx[,b])])
  step1.bs <- glm(wave.comp~ treat.dum +treat.dum +educ + age + male + bs.exp.now
                  + bs.exp.past + Score+
                    factor(Region), data=boot.dat,
                  family=binomial("probit"))
```

```
  boot.dat$p.hat <- predict(step1.bs, newdata=boot.dat, type="response")
  boot.dat$S <- boot.dat$wave.comp/boot.dat$p.hat
  step2 <- lm(mg.asp ~ treat.dum + educ +
                age + male + bs.exp.now + bs.exp.past + Score+
                factor(Region) +factor(wave), data=boot.dat, weights=sqrt(S))
  out[b,] <- step2$coef
}
cbind(colSds(out),
      sqrt(diag(bread %*% (DD) %*% bread)),
      sqrt(diag( vcovCL(ipw1, ~UniqueID))))
```

```
##                    [,1]        [,2]        [,3]
##             0.248441268 0.225840410 0.268478517
## treat.dum   0.059122022 0.048305451 0.057105028
## educ        0.066606942 0.054320078 0.064773319
## age         0.006728027 0.005797822 0.006776444
## male        0.063547790 0.049265050 0.058500923
## bs.exp.now  0.059489047 0.054626198 0.065148719
## bs.exp.past 0.088716565 0.076380827 0.092503388
## Score       0.002834829 0.002166975 0.002642842
##             0.108209217 0.098765025 0.112342412
##             0.092312010 0.076078177 0.089255589
##             0.064029024 0.058512617 0.069421052
##             0.047772500 0.055585545 0.051572410
##             0.059594777 0.059915906 0.055882182
```

2.4.3 Incidental truncation

Note that attrition is not the only form of missingness that might appear. Another form is known as **incidental truncation** where

$$y_{it} \begin{cases} \beta' x_{it} + \alpha_i + \varepsilon_{it} & s_{it} = 1 \\ 0 & \text{otherwise.} \end{cases}$$

This is a "classic" selection problem (a la Heckman–you've seen these before right?), we only observe the "real" outcome if some underlying choice problem is satisfied. In the classic example, we only observe non-zero wages when someone decides to enter the labor market.

In a political science setting, we (maybe) only observe armed conflict after a decision has been made to start a dispute.

As before, we consider a selection equation. Here it will be one that contains some superset of $x_{it}^* \supseteq x_{it}$

$$s_{it} = \mathbb{I}(\gamma' x_{it}^* + u_{it} > 0).$$

where $u_{it}|x_{it}^* \sim N(0,1)$. We want this to be as flexible as possible and allow for unobserved heterogeneity. One way would be to make it a Mundlak specification

$$x_{it}^* = (x_{it}, \bar{x}_{it})$$

. An even more flexible alternative is what's called a Chamberlain approach

$$x_{it}^* = (x_{i1}, x_{i2}, \ldots, x_{iT})$$

.

Under the null hypothesis that there is no selection problem, we can build a single inverse Mills ratio built using just the probit of s_{it} on x_{it}^*. Under the null hypothesis, including this in the main regression will not produce a significant result.

If we reject this null, however, then we need to do something about it. The obvious idea would be to just use this model, it is the Heckman two-step after all. However, we run into a problem when we start thinking about how unobserved heterogeneity actually enters these processes. For example, suppose that

$$s_{it} = \mathbb{I}(x_{it}'\psi + \kappa_i + u_{it}), \quad v_{it}|x_i, \kappa_i, \alpha_i \sim N(0,1),$$

and we write this in Chamberlain form such that

$$s_{it} = \mathbb{I}\left(\sum_{s=1}^{T_i} x_{is}'\gamma_s + u_{it}\right).$$

The error term u_{it} now contains v_{it} and whatever parts of κ_i are not captured by the specification. We might assume then that $\mathrm{E}[\varepsilon_{it} \mid x_i^*, \alpha_i, \kappa_i, u_i] = \delta_i + \rho u_{it}$, which holds if v_{it} and ε_{it} are conditionally independent. This would be a best case scenario. Now what would

we have if we plopped in the inverse Mills ratio like a normal Heckman-model?

$$\begin{aligned} y_{it} &= x_{it}'\beta + \alpha_i + \varepsilon_{it} \\ &= x_{it}'\beta + \alpha_i + \mathrm{E}[\varepsilon_{it} \mid x_i^*, \alpha_i, \kappa_i, u_i] + e_{it}, \\ &= x_{it}'\beta + \alpha_i + \delta_i + \rho u_{it} + e_{it}, \\ &= x_{it}'\beta + (\alpha_i + \delta_i) + \rho\,\mathrm{E}[u_{it}|x_i^*, s_i] + e_{it} + \rho(u_{it} - \mathrm{E}[u_{it}|x_i^*, s_i]) \end{aligned}.$$

The good news is that the joint error term

$$e_{it} + \rho(u_{it} - \mathrm{E}[u_{it}|x_i^*, s_i])$$

is exogenous wrt to x^* and s_{it}. The bad news is that the inverse Mills Ratio that makes sense is based on $\mathrm{E}[u_{it}|x_i^*, s_i]$, which is a function of s at every time period. This is very hard to deal with as these are many nonlinear functions.

To make this easier, we will impose some linearity assumptions, specifically:

1. $\mathrm{E}[\varepsilon_{it}|x_i^*, u_{it}] = \rho_t u_{it}$. This will allow us to drop κ in favor of estimating a different probit for each time period.
2. $\mathrm{E}[\alpha_i|x^*, u_{it}] = x_{it}^*\pi + \psi_t u_{it}$ This will allow us to use the Chamberlain specification for the fixed effects without having to deal with an expectation that depends on every value of s.

Plugging this into the above gives us

$$\begin{aligned} \mathrm{E}[y_{it}|x_i^*, u_{it}] &= \beta' x_{it} + \pi' x_i^* + \rho_t u_{it} \\ &= \beta' x_{it} + \pi' x_i^* + \tau_t u_{it} \\ \mathrm{E}[y_{it}|x_i^*, s_{it} = 1] &= \beta' x_{it} + \pi' x_i^* + \tau_t \frac{\phi(\gamma_t' x_i^*)}{\Phi(\gamma_t' x_i^*)}. \end{aligned}$$

Which can be summarized as:

1. Fit T probits of s_{it} on x_i^*, generate the inverse Mills ratio from each one
2. Use pooled OLS of y_{it} on x_{it}, x_i^*, and the inverse Mills ratio interacted with time dummies on just the selected sample (i.e., just where $s_{it} = 1$.)

2.4.4 Application

How do autocrats prepare for the kind of media attention that comes with hosting major international sporting events? This is the question addressed by Scharpf, Gl aßel, and Edwards

(2023, *APSR*). They compile a panel of department-day level in authoritarian Argentina that covers March 1 - June 25, 1978. This is 3 months before and the 25 days that Argentina hosted the World Cup. For clarity, a department is roughly the equivalent of a U.S. county.

The dependent variable is a count of the number of repressive actions on that department day. Their main hypothesis is that there will be more repression in the run-up to the World Cup than during and that this will be more pronounced in areas with World Cup venues.

```
library(car)
library(dplyr)
library(tidyr)
library(readstata13)
library(sandwich)
library(lmtest)
library(fixest)
library(ggplot2)

source("panelFunctions.r")
fifa <- read.dta13("Rcode/datasets/main_data.dta")
length(unique(fifa$id)) ## number of department = 499
```

```
## [1] 499
```

```
summary(as.numeric(table(fifa$id))) ## each obsevered for 268 days. that's too many
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     268     268     268     268     268     268
```

```
fifa <- fifa %>%
  filter(date >= as.Date("1978-03-01") & date <= as.Date("1978-06-25"))



length(unique(fifa$id)) ## still 499
```

```
## [1] 499
```

```
summary(as.numeric(table(fifa$id))) ## 117 days, better
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     117     117     117     117     117     117
```

```
m1 <- glm(repression ~ hostcitytime +
              hostcitytime2 +
              hostcity +
            time +
            time2 +
            lnpop_1970 +
            vote_frejuli +
            literacy_avg +
            lnrepression70_77 +
           factor(prov)-1,  ## we haven't covered this yet, but trust me for now
          data=fifa,
          x=TRUE,
          family="poisson")
V1 <- vcovCL(m1, ~id) ## we haven't covered this yet, but trust me for now
coeftest(m1,V1)[1:9,]
```

```
##                      Estimate Std. Error   z value     Pr(>|z|)
## hostcitytime       7.40047841 1.55371332  4.763091 1.906495e-06
## hostcitytime2     -5.65656410 1.25093823 -4.521857 6.129940e-06
## hostcity          -0.99293400 0.41254958 -2.406823 1.609195e-02
## time              -1.90282652 1.39466378 -1.364362 1.724536e-01
## time2              1.38744238 1.10917735  1.250875 2.109800e-01
## lnpop_1970         0.39543692 0.39541260  1.000062 3.172807e-01
## vote_frejuli       0.05608371 0.01880945  2.981678 2.866734e-03
## literacy_avg       2.10676818 5.01919806  0.419742 6.746739e-01
## lnrepression70_77  0.69598487 0.30997364  2.245303 2.474866e-02
```

```
m2 <- feols(lnrepression ~ hostcitytime +
              hostcitytime2 +
              hostcity +
              time +
              time2 +
              lnpop_1970 +
              vote_frejuli +
              literacy_avg +
              lnrepression70_77|prov,
            data=fifa, cluster=~id)
```

```r
summary(m2)
```

```
## OLS estimation, Dep. Var.: lnrepression
## Observations: 56,394
## Fixed-effects: prov: 24
## Standard-errors: Clustered (id)
##                    Estimate Std. Error   t value   Pr(>|t|)
## hostcitytime       0.401445   0.176964  2.268508 2.3741e-02 *
## hostcitytime2     -0.323106   0.133777 -2.415263 1.6096e-02 *
## hostcity          -0.070292   0.043959 -1.599034 1.1047e-01
## time              -0.003046   0.002789 -1.092348 2.7523e-01
## time2              0.002169   0.002182  0.994086 3.2068e-01
## lnpop_1970         0.001501   0.000607  2.471370 1.3805e-02 *
## vote_frejuli       0.000089   0.000061  1.457998 1.4549e-01
## literacy_avg      -0.016108   0.005333 -3.020393 2.6588e-03 **
## lnrepression70_77  0.003095   0.000702  4.412159 1.2642e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 0.049707     Adj. R2: 0.136319
##                  Within R2: 0.01711
```

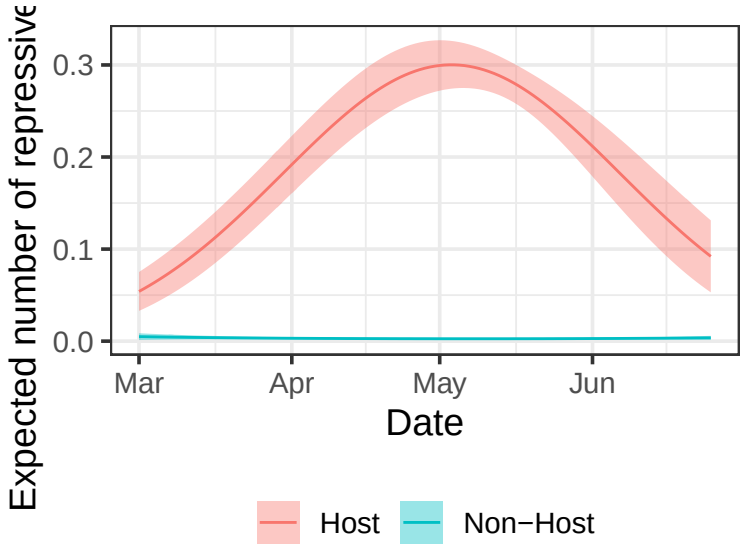
```r
## main effects
Ey <- exp(m1$x %*% m1$coefficients)
results <- data.frame(Ey=Ey,
                      Host=m1$x[,"hostcity"],
                      time=m1$x[,"time"]) %>%
  group_by(Host,time) %>%
  summarize(Ey=mean(Ey))



Db <- m1$x * drop(Ey)
Db1 <- sapply(split.matrix(Db[m1$x[,"hostcity"]==1, ],
                           m1$x[m1$x[,"hostcity"]==1, "time"]),
              colMeans)
Db0 <-  sapply(split.matrix(Db[m1$x[,"hostcity"]==0, ],
                            m1$x[m1$x[,"hostcity"]==0, "time"]),
```

```
                colMeans)
Db <- cbind(Db0, Db1)
results$SE <- sqrt(diag(t(Db) %*% V1 %*% Db))
results$Date <- rep(seq(as.Date("1978-03-01"), as.Date("1978-06-25"), "days"),2)
results$Host <- recode(results$Host, "0"="Non-Host", "1"="Host")

ggplot(results) +
  geom_line(aes(x=Date,y=Ey, color=Host)) +
  geom_ribbon(aes(x=Date, ymin=Ey-1.96*SE, ymax=Ey+1.96*SE,  fill=Host), alpha=.4)+
  ylab("Expected number of repressive actions")+
  theme_bw(14)+
  theme(legend.position = "bottom", legend.title = element_blank())
```




```
### The estimated maximum repression is at
### (-beta_time-beta_time*host)/ (2 (beta_time2 + beta_time2*host))

deltaMethod(m1, "(-time-hostcitytime)/(2*(time2+hostcitytime2))", vcov=V1)
```

```
##                                                    Estimate       SE    2.5 %
## (-time - hostcitytime)/(2 * (time2 + hostcitytime2)) 0.643886 0.031459 0.582228
##                                                    97.5 %
## (-time - hostcitytime)/(2 * (time2 + hostcitytime2)) 0.7055
```

```
## time = 0.64, which corresponds to
unique(fifa$date[round(fifa$time,2)==0.64])
```

```
## [1] "1978-05-03"
```

```
## May 3rd 1978. About a whole month before the cup began
```

Note that the Poisson was easier to work with because with the Poisson we know that

$$\mathrm{E}[y_{it}|X] = \exp(x_{it}'\beta),$$

with the logged dependent variable (+1) OLS this is less clear as

$$\mathrm{E}[\log(y_{it}+1)|X_i] = x_{it}'\beta,$$

which doesn't help us with finding expected levels of attacks as $\exp(\mathrm{E}[\log(y)]) \leq \mathrm{E}[y]$ by Jensen's inequality. To easily see this consider a standard uniform random variable U, $\exp(E[\log(U)])) = \exp(-1) \approx 0.37$ while $\mathrm{E}[\exp(\log(U))] = 0.5$. Ok, what if we work it the other way with

$$\mathrm{E}[y_{it}|X_i] = \mathrm{E}[\exp(x_{it}'\beta)\exp(\varepsilon_{it})|X_i] - 1,$$

If ε_{it} is iid, homoskedastic, and normal, then $\exp(\varepsilon_{it}) \sim LN(0, \sigma_\varepsilon^2)$, where LN refers to the log-normal distribution. This means that

$$\begin{aligned}\mathrm{E}[y_{it}|X_i] &= \exp(x_{it}'\beta)\,\mathrm{E}[\exp(\varepsilon_{it})|X_i] - 1 \\ &= \exp(x_{it}'\beta)\exp(\sigma_\varepsilon^2/2) - 1.\end{aligned}$$

Perhaps not ideal, but estimable.

```
## main effects from the linear model
bonus.term <- exp(m2$sigma2/2)
bonus.term
```

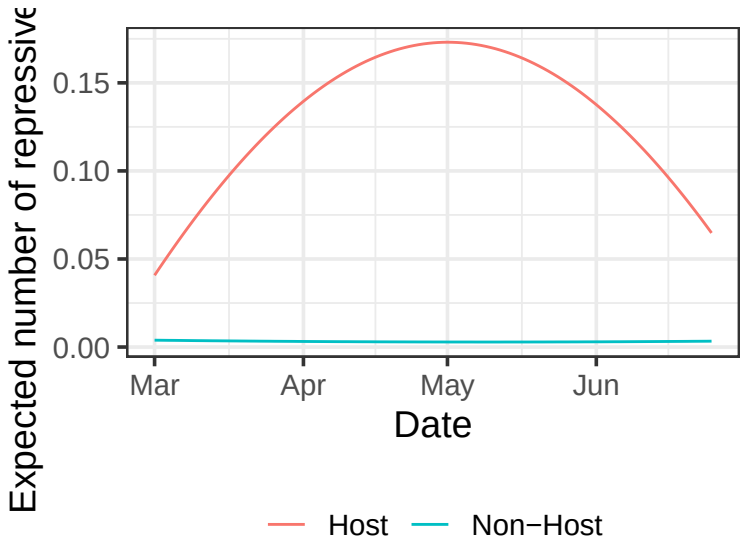
```
## [1] 1.001237
```

```
Ey <- exp(m2$fitted.values) * bonus.term-1
results <- data.frame(Ey=Ey,
                      Host=m1$x[,"hostcity"],
                      time=m1$x[,"time"]) %>%
  group_by(Host,time) %>%
  summarize(Ey=mean(Ey))

results$Date <- rep(seq(as.Date("1978-03-01"), as.Date("1978-06-25"), "days"),2)
```

```r
results$Host <- recode(results$Host, "0"="Non-Host", "1"="Host")

ggplot(results) +
  geom_line(aes(x=Date,y=Ey, color=Host)) +
  ylab("Expected number of repressive actions")+
  theme_bw(14)+
  theme(legend.position = "bottom", legend.title = element_blank())
```




```r
## similar, but about 1/2 the magnitude at the peak
```

Ok but we were interested in selection. So part of the deal here is that host cities are not randomly assigned. Maybe it's the case that 0 repression in the non-host areas is incidental truncation (i.e., we only observe "true" repression in the host-city areas).

Following their lead we will rewrite the main model

$$\log(\text{repression}_{it+1}) = t\beta_1 + t^2\beta_2 + \alpha_i + \varepsilon_{it}, \quad \text{If host city}_{it} = 1$$
$$\Pr(\text{Host city}_{it}) = \mathbb{I}\left[z_i'\gamma + u_{it}\right], \quad u_{it} \sim N(0,1)$$

```r
## Considering the case with selection bias
## Note that we only have time invariant predictors for
## hostcity here and host city is itself invariant within id
## makes our life that much easier as we only
## have one selection model
## we don't have to worry about
## doing this for each time period and interacting.
```

```r
## In this case it actually simplifies to the regular
## Heckman-twostep

heck.sam <- fifa %>%
  filter(!((is.na(lnrepression)| is.na(time) | is.na(time2)) & hostcity==1)) %>%
  select(id,prov,
         time, time2, hostcity,
         vote_frejuli,literacy_avg,lnpop_1970,
         lnrepression) %>%
  filter(!(is.na(literacy_avg) | is.na(vote_frejuli)| is.na(lnpop_1970)))

## We'll do it with a Mundlak at the province level

heck.sam <- heck.sam %>%
  mutate(lnpop_1970.bar =mean(lnpop_1970,na.rm=TRUE),
         vote_frejuli.bar=mean(vote_frejuli,na.rm=TRUE),
         .by=prov)

s1 <- glm(hostcity ~  vote_frejuli+lnpop_1970+
           vote_frejuli.bar+lnpop_1970.bar,
    family=binomial("probit"), data=heck.sam, x=TRUE, y=TRUE)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(dnorm(predict(s1, newdata=heck.sam))[heck.sam$hostcity==1])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1553  0.2255  0.2183  0.3351  0.3758
```

```r
summary(pnorm(predict(s1, newdata=heck.sam))[heck.sam$hostcity==1])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2774  0.3649  0.8573  0.6829  0.9152  1.0000
```

```r
heck.sam$mills <- dnorm(predict(s1, newdata=heck.sam))/predict(s1, newdata=heck.sam, typ
summary(heck.sam$mills[heck.sam$hostcity==1])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1697  0.2631  0.5342  1.0300  1.2081
```

```
m3 <- lm(lnrepression ~
              time +
              time2 +
           vote_frejuli.bar+
           # literacy_avg.bar+
           lnpop_1970.bar+
              mills,
            data=heck.sam,
            subset=hostcity==1, x=TRUE, y=TRUE)
V3 <- vcovCL(m3, ~id)
round(coeftest(m3, V3), 4) ## these SEs are wrong, but a fine start
```

```
##
## t test of coefficients:
##
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -0.4869     0.1730 -2.8141   0.0051 **
## time               0.3984     0.1984  2.0079   0.0451 *
## time2             -0.3209     0.1500 -2.1397   0.0328 *
## vote_frejuli.bar  -0.0071     0.0014 -5.2708   <2e-16 ***
## lnpop_1970.bar     0.0742     0.0086  8.6216   <2e-16 ***
## mills              0.0404     0.0274  1.4759   0.1405
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

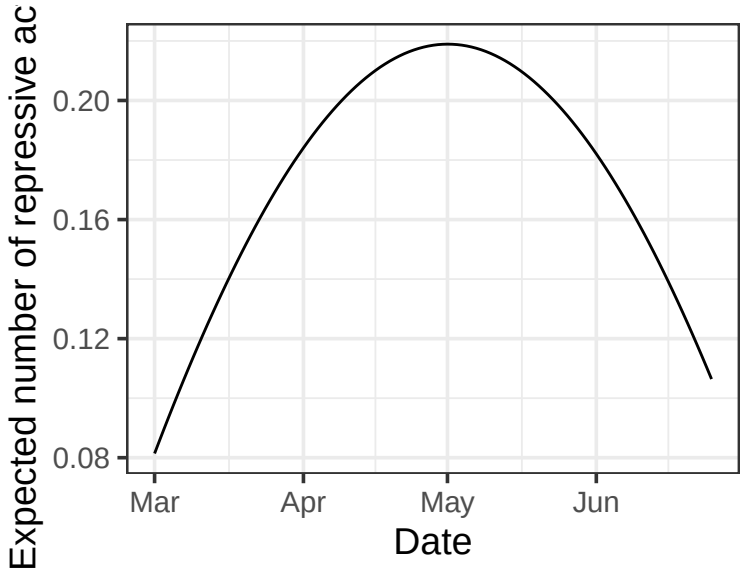
```
bonus.term <- exp(summary(m3)$sigma^2/2)
bonus.term
```

```
## [1] 1.040511
```

```
Ey <- exp(m3$fitted.values) * bonus.term-1
results <- data.frame(Ey=Ey,
                      time=m3$x[,"time"]) %>%
  group_by(time) %>%
  summarize(Ey=mean(Ey))

results$Date <- seq(as.Date("1978-03-01"), as.Date("1978-06-25"), "days")
```

```
ggplot(results) +
  geom_line(aes(x=Date,y=Ey)) +
  ylab("Expected number of repressive actions")+
  theme_bw(14)
```




```
gamma=s1$coef
beta <- m3$coefficients
X <- model.matrix(~
               time +
               time2 +
            vote_frejuli.bar+
            lnpop_1970.bar,
             data=heck.sam)
Z <- s1$x
y <- heck.sam$lnrepression
s <- heck.sam$hostcity

score <- function(beta, gamma, y, X, Z, s){
  ## D_beta m
  p <- pnorm(Z %*% gamma)
  p[p<.Machine$double.eps] <- .Machine$double.eps
  X2 <- cbind(X, dnorm(Z%*%gamma)/p)
  obj <- drop(y- X2%*%beta)* s *(X2)
```

```
  return( obj)
}



d <- function(gamma, s, Z){
  ## D_gamma L
  Z <- (2*s-1)*Z
  ZG <- drop(Z%*%gamma)

  p <- pnorm(ZG)
  dz <- dnorm(ZG)
  p[p<.Machine$double.eps] <- .Machine$double.eps

  return(Z*dz/p)
}



Dscore.b <-  function(gamma, X, Z, s){
  #D_betabeta m
  p <- pnorm(Z %*% gamma)
  p[p<.Machine$double.eps] <- .Machine$double.eps
  X2 <- cbind(X, dnorm(Z%*%gamma)/p) * sqrt(s)
  obj <- (crossprod(X2))
  return(obj)
}

Dscore.g <-  function(beta, gamma, y, X, Z, s){
  #D_betagamma m
  ZG <- drop(Z%*%gamma)
  p <- pnorm(ZG)
  dz <- dnorm(ZG)
  p[p<.Machine$double.eps] <- .Machine$double.eps
  mills <- dz/p
  rho.hat <- beta[length(beta)]
  b <- beta[-length(beta)]
  e <- drop(y-X%*%b-rho.hat*mills)
```

```
  top <-  rho.hat*s*(ZG*p*dz + dz^2)
  signs <- sign(top)
  mid <- sqrt(abs(top))/p

  dBeta <- (t(X*mid) %*% (Z*mid*signs)   )

  dRho <- s*Z*(e)*dz*(-ZG)/p -
    (s*Z*(e)*dz^2)/p^2 +
    (s*rho.hat*Z*dz/p)*(dz^2/p + dz*(ZG))/p
  return(rbind(dBeta, colSums(dRho)))
}

# Dscore.g(beta, gamma, y, X, Z, s)
# numDeriv::jacobian(\(x)score(beta=beta, gamma=x, y=y,X=X,Z=Z,s=s),gamma)

bread <- solve(Dscore.b(gamma, X,Z,s))



F.hat <- Dscore.g(beta, gamma,y, X,Z,s)
r.hat <- solve(-crossprod(d(gamma, s,Z))) %*% t(d(gamma, s,Z))
D <- score(beta, gamma, y, X, Z,s) - t(F.hat %*% r.hat)
DD <- crossprod(D)

cbind(sqrt(diag(bread %*% (DD) %*% bread)),
      sqrt(diag(V3)))
```

```
##                          [,1]        [,2]
## (Intercept)      0.270689544 0.173020131
## time             0.123637460 0.198417315
## time2            0.097883661 0.149993040
## vote_frejuli.bar 0.002982297 0.001355798
## lnpop_1970.bar   0.013938259 0.008610465
##                  0.020819761 0.027398885
```