Panel Data

Today's Outline

- Panel Data models
    - Organization
    - First differences
    - Fixed effects
    - Random effects
    - Hausman test

Assignment due tonight (November 30th!)

A panel Data Model

- Panel data can be boiled down to data that takes multiple observations for the same individuals over different time periods
- We can index every observation by the individual who was observed (i) and the time the observation was recorded (t)
- A generic panel model takes the following form:

$$y_{it} = \beta_0 + \beta_1 x_{1it} + \beta_2 x_{2it} + \ldots + \beta_k x_{kit} + a_i + u_{it}$$

Panel data Example

- Suppose we have three individuals (Larry, Dean, Sarah) observed once a year from 2000 to 2009
- This is an example of a panel data set

```python
# We have three individuals
i = ["Larry", "Dean", "Sarah"]

# We take observations once a year
t = np.arange(2000, 2010)

# each individual is observed once a year
panel_example = pd.DataFrame(itertools.product(t,i), columns = ["year", "person"])

panel_example.head()
```

	year	person
0	2000	Larry
1	2000	Dean
2	2000	Sarah
3	2001	Larry
4	2001	Dean

Organizing Panel Data

- Some synthetic Y values are recorded for each observation
- To work with most panel modelling functions, we need to set the index of our panel dataset as a dual index with the (i) and (t) variables
- The df.set_index() function accepts a list of columns as an arguments, where the columns are the indices

```
panel_example["outcome"] = np.random.normal(0,1,30)
panel_example.head()
```

	year	person	outcome
0	2000	Larry	-0.281639
1	2000	Dean	2.065740
2	2000	Sarah	-1.777229
3	2001	Larry	0.987535
4	2001	Dean	0.464998

```
panel_example = panel_example.set_index(["year", "person"])
panel_example.head(10)
```

		outcome
year	**person**	
2000	**Larry**	-0.281639
	Dean	2.065740
	Sarah	-1.777229
2001	**Larry**	0.987535
	Dean	0.464998
	Sarah	0.517912
2002	**Larry**	0.726195
	Dean	0.028095
	Sarah	-0.870143

Balanced Panel

- Since we have an observation for each individual for all periods we call this a "balanced panel"
- If for any reason this wasn't the case then the panel would be unbalanced

Balanced

```
panel_example = panel_example.set_index(["year", "person"])
panel_example.head(10)
```

year	person	outcome
2000	Larry	-0.281639
	Dean	2.065740
	Sarah	-1.777229
2001	Larry	0.987535
	Dean	0.464998
	Sarah	0.517912
2002	Larry	0.726195
	Dean	0.028095
	Sarah	-0.870143

Unbalanced

```
panel_example.drop([(2008, "Larry"), (2000, "Sarah")])
```

year	person	outcome
2000	Larry	1.235603
	Dean	-2.123426
2001	Larry	0.369016
	Dean	1.278201
	Sarah	-0.065678
2002	Larry	-1.091034
	Dean	0.789812
	Sarah	-1.297215
2003	Larry	1.363545

A panel Data Model

- A generic panel model takes the following form:

$$y_{it} = \beta_0 + \beta_1 x_{1it} + \beta_2 x_{2it} + \ldots + \beta_k x_{kit} + \boxed{a_i + u_{it}}$$

- Note that we have broken out our usual error term into a component that is dependent on time and one that is not

$$v_{it} = a_i + u_{it}$$

- Unfortunately **endogeneity** is often a problem in panel data. Specifically, when the part of the errors for an individual that don't change across time are correlated with X and Y

Panel Data Models

- Endogeneity is a problem we have encountered before
- One form of endogeneity will occur when our regressors (x) are correlated with the unobserved error term (a_i)
- For example, suppose we are studying the effect of union membership on wages
  - Question: Do workers earn more as a result of union membership or are there factors that would cause them to earn more anyways? (more skilled, experienced, etc).
- WIth proper techniques (fixed effects), panel data allows us to control for individual confounders that don't change (or change *very slowly*) over time:
  - Sex, IQ, ethnicity, location, etc.
- We can also adjust our standard errors for serial correlation that naturally occurs when we observe the same people over time (using random effects)

$$log(wage)_{it} = \beta_0 + \beta_1 Union_{it} + a_i + u_{it}$$

First Differences

- Suppose we are trying to answer the question of whether unemployment causes an increase in the crime rate

$$crmrte_{it} = \beta_0 + \beta_1 unem_{it} + ai + u_{it}$$

- We have a panel data set that observes different areas over two time periods

```
# There are two time periods
crime2 = woo.data('crime2')
crime2[["year", "area", "crmrte", "unem"]].head()
```

[66]:

	year	area	crmrte	unem
0	82	44.599998	74.657562	8.2
1	87	44.599998	70.117294	3.7
2	82	375.000000	92.934868	8.1
3	87	375.000000	89.972214	5.4
4	82	49.799999	83.611130	9.0

```
crime2.year.unique()
```

[67]: `array([82, 87], dtype=int64)`

```
# create a time dummy
crime2['t'] = (crime2.year == 87)*1
```

```
# There are 46 different areas observed (i)
crime2['ids'] = crime2.area
crime2.ids
```

```
0      44.599998
1      44.599998
2     375.000000
3     375.000000
4      49.799999
         ...
87     53.000000
88    255.899994
89    255.899994
90     95.800003
91     95.800003
Name: ids, Length: 92, dtype: float64
```

- If we only have two time periods, we can take advantage of panel data simply by taking the difference between our variables between the first and second observation for each individual

$$crmrte_{i2} - crmrte_{i1} = (\beta_0 - \beta_0) + \beta_1(unem_{i2} - unem_{i1}) + (ai - ai) + u_{i2} - u_{i1})$$

```
crime2["crmrte_diff"] = crime2.groupby("ids")["crmrte"].diff()
crime2["unem_diff"] = crime2.groupby("ids")["unem"].diff()
```

- This will eliminate any confounding variables that don't change over time

$$\Delta crmrte_{it} = \beta_1 \Delta unem_{it} + \Delta u_{it}$$

First Differences vs OLS

- We can see very large changes between the pooled estimator and the first differences estimator

```
model1 = smf.ols("crmrte ~ unem", crime2).fit()
model1.summary()
```

OLS Regression Results

Dep. Variable:	crmrte	R-squared:	0.001
Model:	OLS	Adj. R-squared:	-0.010
Method:	Least Squares	F-statistic:	0.1090
Date:	Thu, 01 Dec 2022	Prob (F-statistic):	0.742
Time:	13:46:14	Log-Likelihood:	-442.41
No. Observations:	92	AIC:	888.8
Df Residuals:	90	BIC:	893.9
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	103.2434	8.059	12.811	0.000	87.233	119.253
unem	-0.3077	0.932	-0.330	0.742	-2.159	1.543

Omnibus:	9.658	Durbin-Watson:	1.207
Prob(Omnibus):	0.008	Jarque-Bera (JB):	10.302
Skew:	0.820	Prob(JB):	0.00579
Kurtosis:	3.014	Cond. No.	22.5

```
fdiff = smf.ols("crmrte_diff ~ unem_diff", crime2).fit()
fdiff.summary()
```

OLS Regression Results

Dep. Variable:	crmrte_diff	R-squared:	0.127
Model:	OLS	Adj. R-squared:	0.107
Method:	Least Squares	F-statistic:	6.384
Date:	Thu, 01 Dec 2022	Prob (F-statistic):	0.0152
Time:	13:46:14	Log-Likelihood:	-202.17
No. Observations:	46	AIC:	408.3
Df Residuals:	44	BIC:	412.0
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	15.4022	4.702	3.276	0.002	5.926	24.879
unem_diff	2.2180	0.878	2.527	0.015	0.449	3.987

Omnibus:	2.636	Durbin-Watson:	1.146
Prob(Omnibus):	0.268	Jarque-Bera (JB):	2.255
Skew:	0.539	Prob(JB):	0.324
Kurtosis:	2.883	Cond. No.	8.70

First Differences in linearmodels

- The linearmodels package will contain all of our panel estimators
- Each of these linearmodels functions *require* that our panel data is indexed correctly before estimation
- Note that "t" here functions as an intercept after differencing and represents the time trend

```
crime2[["t","crmrte_diff", "unem_diff"]].head()
```

	t	crmrte_diff	unem_diff
0	0	NaN	NaN
1	1	-4.540268	-4.5
2	0	NaN	NaN
3	1	-2.962654	-2.7
4	0	NaN	NaN

↓

```
crime2[["t","crmrte_diff", "unem_diff"]].dropna().head()
```

	t	crmrte_diff	unem_diff
1	1	-4.540268	-4.500000
3	1	-2.962654	-2.700000
5	1	-6.416374	-3.100000
7	1	-4.901543	-6.900001
9	1	-4.608994	-5.200000

```
import linearmodels as plm
```

```
crime2 = crime2.set_index(['ids', 'year'])
```

```
plm.FirstDifferenceOLS.from_formula(formula = 'crmrte ~ t + unem', data = crime2).fit()
```

```
C:\Users\kunzn\anaconda3\lib\site-packages\linearmodels\shared\utility.py:187: FutureWarning:
all arguments of MultiIndex.set_levels except for the argument 'levels' will be keyword-only
  df.index = df.index.set_levels(final_levels, [0, 1])
```

FirstDifferenceOLS Estimation Summary

Dep. Variable:	crmrte	R-squared:	0.1961
Estimator:	FirstDifferenceOLS	R-squared (Between):	0.4064
No. Observations:	46	R-squared (Within):	0.1961
Date:	Thu, Dec 01 2022	R-squared (Overall):	0.4041
Time:	13:50:35	Log-likelihood	-202.17
Cov. Estimator:	Unadjusted		
		F-statistic:	5.3653
Entities:	46	P-value	0.0082
Avg Obs:	2.0000	Distribution:	F(2,44)
Min Obs:	2.0000		
Max Obs:	2.0000	F-statistic (robust):	5.3653
		P-value	0.0082
Time periods:	2	Distribution:	F(2,44)
Avg Obs:	46.000		
Min Obs:	46.000		
Max Obs:	46.000		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
t	15.402	4.7021	3.2756	0.0021	5.9257	24.879
unem	2.2180	0.8779	2.5266	0.0152	0.4488	3.9872

Fixed Effects

- Fixed effects is another, similar, method of controlling for the unobserved time-invariant effects
- For two periods, the estimates obtained by fixed effects are the same as the first differences estimator
- The fixed effects model can be estimated equivalently in two ways, first by demeaning each observation by for the individuals:

$$
\begin{aligned}
y_{it} &= \beta_0 + \beta_1 x_{it1} + \cdots + \beta_k x_{itk} + a_i + u_{it}; \qquad t = 1, \ldots, T; \qquad i = 1, \ldots, n, \\
\bar{y}_i &= \beta_0 + \beta_1 \bar{x}_{i1} + \cdots + \beta_k \bar{x}_{ik} + a_i + \bar{u}_i \\
\ddot{y}_{it} = y_{it} - \bar{y}_i &= \beta_1 \ddot{x}_{it1} + \cdots + \beta_k \ddot{x}_{itk} + \ddot{u}_{it},
\end{aligned}
$$

- Note that doing this manually will lead to incorrect standard errors
- We can also just include a dummy in our regression for each individual (this is computationally less efficient, and creates an uglier output, but will give the same result)

```
smf.ols('Y ~ x1 + C(person)-1', data = data).fit().summary()
```

Within Estimator

- We can also use differencing across many time periods
- The *within* estimator subtracts the mean of our variables from each period
- The smf.ols() function adds an intercept automatically that we can remove with “-1”

```
crime2["crmrte_demmean"] = crime2.crmrte - crime2.groupby("ids", axis=0).transform('mean')["crmrte"]
crime2["unem_demmean"] = crime2.unem - crime2.groupby("ids", axis=0).transform('mean')["unem"]
```

```
mod = smf.ols('crmrte ~ t+unem-1', data=demeaned_data).fit()
mod.summary()
```

1]:

OLS Regression Results

Dep. Variable:	crmrte	**R-squared (uncentered):**	0.196
Model:	OLS	**Adj. R-squared (uncentered):**	0.178
Method:	Least Squares	**F-statistic:**	10.97
Date:	Thu, 01 Dec 2022	**Prob (F-statistic):**	5.43e-05
Time:	15:00:42	**Log-Likelihood:**	-340.57
No. Observations:	92	**AIC:**	685.1
Df Residuals:	90	**BIC:**	690.2
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
t	15.4022	3.288	4.685	0.000	8.871	21.934
unem	2.2180	0.614	3.614	0.000	0.999	3.437

Omnibus:	0.005	**Durbin-Watson:**	2.890
Prob(Omnibus):	0.998	**Jarque-Bera (JB):**	0.053
Skew:	0.000	**Prob(JB):**	0.974
Kurtosis:	2.883	**Cond. No.**	8.70

$$y_{it} = \beta_0 + \beta_1 x_{it1} + \cdots + \beta_k x_{itk} + a_i + u_{it}; \qquad t = 1, \ldots, T; \qquad i = 1, \ldots, n,$$

$$\bar{y}_i = \beta_0 + \beta_1 \bar{x}_{i1} + \cdots + \beta_k \bar{x}_{ik} + a_i + \bar{u}_i$$

$$\ddot{y}_{it} = y_{it} - \bar{y}_i = \beta_1 \ddot{x}_{it1} + \cdots + \beta_k \ddot{x}_{itk} + \ddot{u}_{it},$$

Within Estimator (linearmodels)

- The PanelOLS.from_formula() function automatically can implement fixed effects
- Add EntityEffects to the formula
- PanelOLS does not automatically add an intercept
- drop_absorbed will remove variables from the regression that do not change over time

```
crime2 = crime2.set_index(['ids', 'year'], drop = False)
```

```
# Drop absorbed drops any variables that do not change over time (ethnicity would be an example)
fe_fit = plm.PanelOLS.from_formula(formula = 'crmrte ~ t+ unem + EntityEffects',
                                   data = crime2, drop_absorbed = True).fit()
fe_fit
```

PanelOLS Estimation Summary

Dep. Variable:	crmrte	R-squared:	0.1961
Estimator:	PanelOLS	R-squared (Between):	0.4064
No. Observations:	92	R-squared (Within):	0.1961
Date:	Thu, Dec 01 2022	R-squared (Overall):	0.4041
Time:	15:11:26	Log-likelihood	-340.57
Cov. Estimator:	Unadjusted		
		F-statistic:	5.3653
Entities:	46	P-value	0.0082
Avg Obs:	2.0000	Distribution:	F(2,44)
Min Obs:	2.0000		
Max Obs:	2.0000	F-statistic (robust):	5.3653
		P-value	0.0082
Time periods:	2	Distribution:	F(2,44)
Avg Obs:	46.000		
Min Obs:	46.000		
Max Obs:	46.000		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
t	15.402	4.7021	3.2756	0.0021	5.9257	24.879
unem	2.2180	0.8779	2.5266	0.0152	0.4488	3.9872

Dummy Variable Regression

- Fixed effects can be estimated equivalently by including a dummy variable for each (less one) (i) in the sample
- We can test if fixed effects are necessary by running an F-test on the individual dummy variables

```
1  smf.ols(formula = 'crmrte ~ t+ unem + C(ids)', data = crime2).fit().summary()
```

OLS Regression Results

Dep. Variable:	crmrte	R-squared:	0.891
Model:	OLS	Adj. R-squared:	0.774
Method:	Least Squares	F-statistic:	7.642
Date:	Thu, 01 Dec 2022	Prob (F-statistic):	1.70e-10
Time:	11:44:37	Log-Likelihood:	-340.57
No. Observations:	92	AIC:	777.1
Df Residuals:	44	BIC:	898.2
Df Model:	47		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	35.7417	15.515	2.304	0.026	4.473	67.010
C(ids)[T.17.799999237060547]	110.7026	14.992	7.384	0.000	80.489	140.917
C(ids)[T.18.899999618530277]	47.7586	14.545	3.283	0.002	18.444	77.073
C(ids)[T.20.799999237060547]	88.4714	15.235	5.807	0.000	57.768	119.175
C(ids)[T.21.899999618530277]	13.6771	14.779	0.925	0.360	-16.107	43.461
C(ids)[T.24.100000381469727]	51.7858	14.231	3.639	0.001	23.105	80.467
C(ids)[T.24.200000762939453]	24.2242	15.006	1.614	0.114	-6.019	54.467
C(ids)[T.25.299999237060547]	48.2313	14.978	3.220	0.002	18.046	78.417
C(ids)[T.27.399999618530277]	30.3498	14.791	2.052	0.046	0.540	60.159
C(ids)[T.34.20000076293945]	43.7282	14.545	3.006	0.004	14.414	73.042
C(ids)[T.604.0]	52.6707	15.050	3.500	0.001	22.340	83.002
t	15.4022	4.702	3.276	0.002	5.926	24.879
unem	2.2180	0.878	2.527	0.015	0.449	3.987

Omnibus:	0.005	Durbin-Watson:	3.413
Prob(Omnibus):	0.998	Jarque-Bera (JB):	0.053
Skew:	-0.000	Prob(JB):	0.974
Kurtosis:	2.883	Cond. No.	425.

Random Effects

- Random Effects is another panel data method that is only appropriate when:

$$\mathrm{Cov}(x_{itj}, a_i) = 0, \quad t = 1, 2, \ldots, T; j = 1, 2, \ldots, k.$$

- That means we have no endogeneity!

- However the errors in our model will be serially correlated across time, since *even if* u_it is uncorrelated with itself and a_i, the covariance of a_i with itself is non-zero

- More precisely:

$$Cov(a_i + u_{it}, a_i + u_{is}) = Var(a_i)^2$$

- This violates our basic assumption for OLS estimators for that the error is not correlated with itself

Random Effects Original Regression

- Here we fit the model again using OLS to compare with the random effects model
- Note that we can include an intercept since we do not need to eliminate time invariant parts of the model
- We can include a time dummy to account for changes between each period as well
- Note that neither are statistically significant
  - Direction of unem makes sense with t included
  - Standard errors are large

```
: mod = smf.ols('crmrte ~ t+ unem-1+1', data=crime2).fit()
mod.summary()
```

OLS Regression Results

Dep. Variable:	crmrte	**R-squared:**	0.012
Model:	OLS	**Adj. R-squared:**	-0.010
Method:	Least Squares	**F-statistic:**	0.5501
Date:	Wed, 06 Dec 2023	**Prob (F-statistic):**	0.579
Time:	16:51:17	**Log-Likelihood:**	-441.90
No. Observations:	92	**AIC:**	889.8
Df Residuals:	89	**BIC:**	897.4
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	93.4202	12.739	7.333	0.000	68.107	118.733
t	7.9404	7.975	0.996	0.322	-7.906	23.787
unem	0.4265	1.188	0.359	0.720	-1.935	2.788

Omnibus:	8.350	**Durbin-Watson:**	1.157
Prob(Omnibus):	0.015	**Jarque-Bera (JB):**	8.771
Skew:	0.756	**Prob(JB):**	0.0125
Kurtosis:	2.935	**Cond. No.**	40.1

Random Effects

- Random effects estimates the structure of our variance using a type of FGLS
- This procedure corrects our standard errors
- As long as our assumption about the **covariance between a and x being 0 holds,** then Random Effects will produce better results than OLS or Fixed Effects
  - This can be an admittedly strong assumption
- Random Effects may also be estimated in using plm

```
reg_re = plm.RandomEffects.from_formula(formula = 'crmrte ~ 1+ t  + unem', data = crime2).fit()
reg_re
```

RandomEffects Estimation Summary

Dep. Variable:	crmrte	R-squared:	0.0927
Estimator:	RandomEffects	R-squared (Between):	-0.0320
No. Observations:	92	R-squared (Within):	0.1911
Date:	Thu, Dec 01 2022	R-squared (Overall):	-0.0017
Time:	11:56:44	Log-likelihood	-372.87
Cov. Estimator:	Unadjusted		
		F-statistic:	4.5472
Entities:	46	P-value	0.0132
Avg Obs:	2.0000	Distribution:	F(2,89)
Min Obs:	2.0000		
Max Obs:	2.0000	F-statistic (robust):	4.5472
		P-value	0.0132
Time periods:	2	Distribution:	F(2,89)
Avg Obs:	46.000		
Min Obs:	46.000		
Max Obs:	46.000		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
Intercept	80.031	9.2597	8.6429	0.0000	61.632	98.430
t	13.487	4.4767	3.0128	0.0034	4.5922	22.382
unem	1.7583	0.8077	2.1767	0.0321	0.1533	3.3632

id: 0x198c37e36a0

Hausman Test

- The Hausman Test helps us determine whether we have a problem with endogeneity
- The null hypothesis is that the individual effects are exogenous
- If we reject the null then individual effects are endogenous and we should use Fixed effects
- The Hausman simply tests whether there is a statistically significant difference between the coefficients estimated by RE and FE
- Including important omitted variables will change our estimated coefficients, so if the individual effects are important then the coefficients on RE and FE should be very different

```
def PnlHausman(fe_fit, re_fit):
    # pull out the variances and parameters for test
    Dcov = fe_fit.cov - re_fit.cov.iloc[1:, 1:]
    dparams = fe_fit.params - re_fit.params[1:]
    # get the test statistic
    Chi2 = dparams.dot(np.linalg.inv(Dcov)).dot(dparams)
    # calculate the degrees of freedom
    dof = re_fit.params.size - 1
    # calculate the p-value
    pvalue = stats.chi2(dof).sf(Chi2)
    return(Chi2, dof, pvalue)
```

```
# Takes in a fitted fixed effects model and a fitted random effects model
# The null hypthesis is that the individual effects are exogenous
PnlHausman(fe_fit, reg_re)
```

```
(0.08940229523166991, 1, 0.7649383929396042)
```

Simulated Data Demonstration

Panel Exercise

- Download the **MURDER** dataset from wooldridge
- Estimate the following pooled ols model

$$mrdrte_{it} = \boxed{\eta_t} + \beta_1 exec + \beta_2 unem + a_i + u_{it}$$

Year dummies

- Estimate the same model again and include fixed effects then random effects
- (Optional) Run the hausman test for endogeneity