In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
from scipy import stats
from matplotlib.lines import Line2D
import seaborn as sns
from sklearn.linear_model import LinearRegression
from IPython.display import Image
import statsmodels.api as sm
import statsmodels.stats.api as sms
import statsmodels.formula.api as smf
import linearmodels as plm
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
df = pd.read_csv('desktop/productivity.csv')
```

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 816 entries, 0 to 815
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   STATE   816 non-null    object 
 1   YR      816 non-null    int64  
 2   P_CAP   816 non-null    float64
 3   HWY     816 non-null    float64
 4   WATER   816 non-null    float64
 5   UTIL    816 non-null    float64
 6   PC      816 non-null    float64
 7   GSP     816 non-null    int64  
 8   EMP     816 non-null    float64
 9   UNEMP   816 non-null    float64
dtypes: float64(7), int64(2), object(1)
memory usage: 63.9+ KB
```

In [4]:

```
df.head(3)
```

Out[4]:

	STATE	YR	P_CAP	HWY	WATER	UTIL	PC	GSP	EMP	UNEMP
0	ALABAMA	1970	15032.67	7325.80	1655.68	6051.20	35793.80	28418	1010.5	4.7
1	ALABAMA	1971	15501.94	7525.94	1721.02	6254.98	37299.91	29375	1021.9	5.2
2	ALABAMA	1972	15972.41	7765.42	1764.75	6442.23	38670.30	31303	1072.3	4.7

In [6]:
```
Model = smf.ols(formula='np.log(GSP) ~ np.log(P_CAP) + np.log(PC) + np.log(
results = Model.fit()
results.summary()
```

Out[6]:

OLS Regression Results

Dep. Variable:	np.log(GSP)	**R-squared:**	0.993
Model:	OLS	**Adj. R-squared:**	0.993
Method:	Least Squares	**F-statistic:**	2.717e+04
Date:	Sun, 28 Nov 2021	**Prob (F-statistic):**	0.00
Time:	21:00:53	**Log-Likelihood:**	826.98
No. Observations:	816	**AIC:**	-1644.
Df Residuals:	811	**BIC:**	-1620.
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.6433	0.058	28.536	0.000	1.530	1.756
np.log(P_CAP)	0.1550	0.017	9.036	0.000	0.121	0.189
np.log(PC)	0.3092	0.010	30.100	0.000	0.289	0.329
np.log(EMP)	0.5939	0.014	43.203	0.000	0.567	0.621
UNEMP	-0.0067	0.001	-4.754	0.000	-0.010	-0.004

Omnibus:	23.719	**Durbin-Watson:**	0.180
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	28.801
Skew:	0.333	**Prob(JB):**	5.57e-07
Kurtosis:	3.635	**Cond. No.**	336.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [7]:
```
df = df.set_index(['STATE', 'YR'], drop=False)
```

In [9]:
```
reg_fe = plm.PanelOLS.from_formula(
    formula='np.log(GSP) ~ np.log(P_CAP) + np.log(PC) + np.log(EMP) + UNEMP
    data=df, drop_absorbed=True)

results_fe = reg_fe.fit()
results_fe
```

Out[9]:

PanelOLS Estimation Summary

Dep. Variable:	np.log(GSP)	**R-squared:**	0.9413
Estimator:	PanelOLS	**R-squared (Between):**	0.9503
No. Observations:	816	**R-squared (Within):**	0.9413
Date:	Sun, Nov 28 2021	**R-squared (Overall):**	0.9503
Time:	21:01:38	**Log-likelihood**	1534.5
Cov. Estimator:	Unadjusted		
		F-statistic:	3064.8
Entities:	48	**P-value**	0.0000
Avg Obs:	17.000	**Distribution:**	F(4,764)
Min Obs:	17.000		
Max Obs:	17.000	**F-statistic (robust):**	3064.8
		P-value	0.0000
Time periods:	17	**Distribution:**	F(4,764)
Avg Obs:	48.000		
Min Obs:	48.000		
Max Obs:	48.000		

Parameter Estimates

	Parameter	**Std. Err.**	**T-stat**	**P-value**	**Lower CI**	**Upper CI**
np.log(P_CAP)	-0.0261	0.0290	-0.9017	0.3675	-0.0831	0.0308
np.log(PC)	0.2920	0.0251	11.625	0.0000	0.2427	0.3413
np.log(EMP)	0.7682	0.0301	25.527	0.0000	0.7091	0.8272
UNEMP	-0.0053	0.0010	-5.3582	0.0000	-0.0072	-0.0034

F-test for Poolability: 75.820
P-value: 0.0000
Distribution: F(47,764)

Included effects: Entity
id: 0x7fe891e78b20

```
Looking at the OLS model, I conclude that Public Capital has a statistical
significant effect on Production.
Looking at the Fixed Effect Model, I conclude that Public Capital does not
have a statistical significant effect on Production.
```

In [11]:
```
reg_re = plm.RandomEffects.from_formula(
    formula='np.log(GSP) ~ np.log(P_CAP) + np.log(PC) + np.log(EMP) + UNEMP

results_re = reg_re.fit()
results_re
```

Out[11]:

RandomEffects Estimation Summary

Dep. Variable:	np.log(GSP)	**R-squared:**	0.9974
Estimator:	RandomEffects	**R-squared (Between):**	0.9998
No. Observations:	816	**R-squared (Within):**	0.9274
Date:	Sun, Nov 28 2021	**R-squared (Overall):**	0.9998
Time:	21:06:08	**Log-likelihood**	1420.2
Cov. Estimator:	Unadjusted		
		F-statistic:	7.855e+04
Entities:	48	**P-value**	0.0000
Avg Obs:	17.000	**Distribution:**	F(4,812)
Min Obs:	17.000		
Max Obs:	17.000	**F-statistic (robust):**	7.855e+04
		P-value	0.0000
Time periods:	17	**Distribution:**	F(4,812)
Avg Obs:	48.000		
Min Obs:	48.000		
Max Obs:	48.000		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
np.log(P_CAP)	0.2732	0.0187	14.603	0.0000	0.2365	0.3100
np.log(PC)	0.4401	0.0221	19.949	0.0000	0.3968	0.4834
np.log(EMP)	0.4698	0.0219	21.468	0.0000	0.4268	0.5127
UNEMP	-0.0138	0.0009	-15.961	0.0000	-0.0154	-0.0121

id: 0x7fe891eb1bb0

In [13]:
```
b_fe = results_fe.params
b_fe_cov = results_fe.cov
```

In [14]:
```
b_re = results_re.params
b_re_cov = results_re.cov
```

In [15]:
```
common_coef = set(results_fe.params.index).intersection(results_re.params.i


b_diff = np.array(results_fe.params[common_coef] - results_re.params[common
df = len(b_diff)
b_diff.reshape((df, 1))
b_cov_diff = np.array(b_fe_cov.loc[common_coef, common_coef] -
                      b_re_cov.loc[common_coef, common_coef])
b_cov_diff.reshape((df, df))


stat = abs(np.transpose(b_diff) @ np.linalg.inv(b_cov_diff) @ b_diff)
pval = 1 - stats.chi2.cdf(stat, df)

print(f'stat: {stat}\n')
print(f'pval: {pval}\n')
```

```
stat: 201.7683088005258

pval: 0.0
```

After running a Hausman test, we reject the null hypothesis that the Random Effect Model is preferred over the Fixed Effect model, therefore, we will choose to stick the the alternative hypothesis that the Fixed Effect Model is preferrable.