ECON272A - Problem Set #1

Christopher Saw

1 February 2022

Code

All code is attached at the end of this document in pdf. Python scripts are provided in separate attachments.

1.1 Conceptual Questions

Question 1

Assumption	**Example Question**	**Ruled out because...**
A1(a)	Does the elasticity of output with respect to inputs (capital and labor) change as output increases?	Output elasticities (β_k, β_l) are constant for all levels of output.
A1(b)	Are there fixed material costs incurred from production?	Materials are in fixed proportion to capital and labor.
A1(c)	How does investment risk affect the capital accumulation of the firm?	Capital accumulation is deterministic.
A1(d)	Can firm investments lead to endogenous productivity improvements in the same period?	Productivity is modeled as an AR(1) process; ω_{it} depends only on $\omega_{i,t-1}$.
A1(e)	Is productivity growth factor-augmenting for the firm?	$\omega_{it} \in \mathbb{R}$ and only captures factor-neutral productivity.

Question 2

Assumption	**Example Question**	**Ruled out because...**
A2(a)	Do firms produce differentiated varieties?	All firms produce the same output and belong to the same market.
A2(b)	Do different firms have different values for future profits?	Firms discount the future with a common factor $(1 - r)$.
A2(c)	Does the firm have monopsony power in the labor market?	Wages are determined by an AR(1) process.
A2(d)	How do changes in investment cost over time affect firms?	The cost of investment varies only across firms but not over time.

Question 3

- Assumptions A1(b),(e) imply that productivity shocks are Hicks-neutral.
- Assumptions A1(d),(e) imply that the correlation coefficient ρ in (2) is a scalar value.
- Assumptions A2(a),(b) imply that any two or more firms with the same state variables (k_{it}, ω_{it}) will face the same profit maximization problem and behave identically.
- Assumption A2(d) implies that the firm chooses the same level of optimal investment ($I_{it} = I^*$) in every period. This, with Assumption A1(c), implies one of the following 3 cases for the capital stock of the firm: (i) K grows at a constant rate (if $I^* > \delta K_{it}$), (ii) K has a steady-state (if $I^* = \delta K_{it}$) or (iii) K eventually falls to 0 (if $I^* < \delta K_{it}$).

Question 4

Equation (2) imposes the structural assumption that $\omega_{it} = \rho\omega_{i,t-1} + \xi_{it}$ where ξ_{it} is a mean zero i.i.d. random shock.

- To justify the use of OLS to estimate (1), we have to assume that $\omega_{it} = 0$ for all i and for all t. This is equivalent to assuming $\rho = 0$ in (2).
- To justify the use of FE to estimate (1), we have to assume that $\omega_{it} = \omega_i$ for all t. This is equivalent to assuming $\rho = 1$ in (2).

Therefore, the assumptions required for OLS and FE can be interpreted as special cases of the model that specify the value of ρ in equation (2).

1.2 Simulation

Question 2a

Let $A_{it} = \exp(\beta_0)\exp(\omega_{it})\exp(\varepsilon_{it})$. The production function is Cobb-Douglas in capital and labor, and can be written as:

$$Q_{it} = F(K_{it}, L_{it}) = A_{it}K_{it}^{\beta_k}L_{it}^{\beta_l}$$

Which gives the first-order condition (FOC) with respect to labor:

$$P_{it}^{l} = A_{it}K_{it}^{\beta_k}\beta_l L_{it}^{\beta_l - 1}$$

Re-arranging, the optimal labor input is:

$$L_{it} = \Big(\frac{A_{it}K_{it}^{\beta_k}\beta_l}{P_{it}^{l}}\Big)^{\frac{1}{1-\beta_l}}$$

Question 2b

Capital evolves according to:

$$K_{i,t+1} = (1-\delta)K_{it} + I_{it}$$

with convex investment costs:

$$C(I_{it};\gamma_i) = \frac{1}{2}\gamma_i I_{it}^2$$

Next-period production can thus be expressed as:

$$Q_{i,t+1} = A_{i,t+1}K_{i,t+1}^{\beta_k}L_{i,t+1}^{\beta_l} = A_{i,t+1}[(1-\delta)K_{it} + I_{it}]^{\beta_k}L_{i,t+1}^{\beta_l}$$

Using the labor FOC, the FOC with respect to investment-today:

$$\begin{aligned}
\gamma_i I_{it} &= A_{i,t+1}L_{i,t+1}^{\beta_l}\beta_k[(1-\delta)K_{it} + I_{it}]^{\beta_k - 1} \\
&= A_{i,t+1}\Big(\frac{A_{i,t+1}K_{i,t+1}^{\beta_k}\beta_l}{P_{i,t+1}^l}\Big)^{\frac{\beta_l}{1-\beta_l}}\beta_k[K_{i,t+1}]^{\beta_k - 1} \\
&= A_{i,t+1}\Big(\frac{A_{i,t+1}\beta_l}{P_{i,t+1}^l}\Big)^{\frac{\beta_l}{1-\beta_l}}\beta_k[K_{i,t+1}]^{\frac{\beta_k\beta_l}{1-\beta_l}+\beta_k-1} \\
&= A_{i,t+1}\Big(\frac{A_{i,t+1}\beta_l}{P_{i,t+1}^l}\Big)^{\frac{\beta_l}{1-\beta_l}}\beta_k
\end{aligned}$$

which is independent of the level of capital because if $\beta_k + \beta_l = 1$, then

$$\frac{\beta_k\beta_l}{1-\beta_l} + \beta_k - 1 = \frac{\beta_k\beta_l + \beta_k(1-\beta_l) - 1 + \beta_l}{1-\beta_l} = \frac{(\beta_k+\beta_l)-1}{1-\beta_l} = 0$$

Question 2c

Given that $\beta_0 = 0$ and $\mathbb{E}[\exp(\varepsilon_{it})] = 1$,

$$\mathbb{E}[A_{it}] = \mathbb{E}[\exp(0)\exp(\omega_{it})\exp(\varepsilon_{it})] = \exp(\omega_{it})\mathbb{E}[\exp(\varepsilon_{it})] = \exp(\omega_{it})$$

Where $\exp(\omega_{it})$ is outside the expectation operator because of Assumption A2(b) (each firm takes as given their ω_{it} in each period t). Now we can write the optimal investment rule as:

$$\begin{aligned}
I_{it} &= \frac{1}{\gamma_i}\mathbb{E}[A_{i,t+1}]\Big(\frac{\mathbb{E}[A_{i,t+1}]\beta_l}{P_{i,t+1}^l}\Big)^{\frac{\beta_l}{1-\beta_l}}\beta_k \\
&= \frac{1}{\gamma_i}\exp(\omega_{i,t+1})\Big(\frac{\exp(\omega_{i,t+1})\beta_l}{P_{i,t+1}^l}\Big)^{\frac{\beta_l}{1-\beta_l}}\beta_k
\end{aligned}$$

The investment policy equation is the present discounted value of future investment flows (net of depreciation) at $t = 0$:

$$\begin{aligned}
\kappa(\{\omega_{it}, P_{it}^l\}_{t=1}^{\infty}) &= \sum_{t=1}^{\infty}(1-\delta)^t(1-r)^t I_{it} \\
&= \sum_{t=1}^{\infty}(1-\delta)^t(1-r)^t\frac{1}{\gamma_i}\exp(\omega_{it})\Big(\frac{\exp(\omega_{it})\beta_l}{P_{it}^l}\Big)^{\frac{\beta_l}{1-\beta_l}}\beta_k \\
&= \frac{1}{\gamma_i}\cdot\beta_k\sum_{t=1}^{\infty}(1-\delta)^t(1-r)^t\exp(\omega_{it})\Big(\frac{\exp(\omega_{it})\beta_l}{P_{it}^l}\Big)^{\frac{\beta_l}{1-\beta_l}}
\end{aligned}$$

1.3 Estimation

Questions 1, 2

The estimates of β are shown in Table 1. Standard errors are reported in parentheses, and standard errors in the FE model are clustered at the firm-level.

	OLS	FE
	(1)	(2)
β_0	0.392**	0.408**
	(0.006)	(0.033)
β_k	0.079**	0.077**
	(0.004)	(0.022)
β_l	0.930**	0.895**
	(0.003)	(0.004)
Obs	20000	20000
R^2	0.935	0.852

** p<0.01, * p<0.05, + p<0.10

Table 1: **OLS and FE estimates**

Question 3d

ACF instruments $(k_{it}, l_{i,t-1})$ satisfy the exclusion restriction $\mathbb{E}[Z_{it}\xi_{it}|\hat{\omega}_{i,t-1}] = 0$ if the timing assumption in A2(b) holds and firms' capital input are determined according to A1(c). At $t-1$, firm i observes $\omega_{i,t-1}$ before it chooses $(l_{i,t-1}, i_{i,t-1})$. Firms' labor and investment decisions in period $t-1$ cannot be correlated with the unobserved productivity innovation ξ_{it} in period t. With A1(c), $K_{it} = (1-\delta)K_{i,t-1} + I_{i,t-1}$ so the capital input is determined in period $t-1$ (via $I_{i,t-1}$ given $\omega_{i,t-1}$) and also cannot be correlated with ξ_{it}. Therefore we have $\mathbb{E}[Z_{it}\xi_{it}|\hat{\omega}_{i,t-1}] = 0$ for $Z_{it} = (1, k_{it}, l_{i,t-1})$.

It is straightforward to see that k_{it} is the capital input in period t.

$l_{i,t-1}$ is a relevant shifter of period-t inputs if assumption A2(c) holds. If wages evolve according to:

$$P^l_{it} = \theta P^l_{i,t-1} + \nu_{i,t-1}$$

And the first-order condition which determines firm i's labor choice:

$$P^l_{it} = A_{it} K^{\beta_k}_{it} \beta_l L^{\beta_l - 1}_{it}$$

It can be shown that

$$A_{it} K^{\beta_k}_{it} \beta_l L^{\beta_l - 1}_{it} = \theta A_{i,t-1} K^{\beta_k}_{i,t-1} \beta_l L^{\beta_l - 1}_{i,t-1} + \nu_{i,t-1}$$

or in other words, firm i's labor choice in period t is correlated with its labor choice in $t-1$.

Question 3h

The ACF estimates for (β_k, β_l) and 90% confidence intervals are shown in Table 2.

	LB	ACF estimate	UB
β_k	0.016	0.033	0.046
β_l	0.954	0.968	0.978
GMM value	-	6.48e-06	-
Iterations	-	61	-

Table 2: **ACF estimates with 90% CI**

From CLT we have:

$$\sqrt{n}(\hat{\beta} - \beta) \xrightarrow{d} N(0, \sigma_\beta^2)$$

$$\frac{\sqrt{n}}{\sigma_\beta}(\hat{\beta} - \beta) \xrightarrow{d} N(0, 1)$$

Use the standard normal distribution to get:

$$Pr\Big(-1.645 < \frac{\sqrt{n}}{\sigma_\beta}(\hat{\beta} - \beta) < 1.645 \Big) = 0.90$$

$$Pr\Big(\beta - \frac{1.645\sigma_\beta}{\sqrt{n}} < \hat{\beta} < \beta + \frac{1.645\sigma_\beta}{\sqrt{n}} \Big) = 0.90$$

Use the bootstrap estimates to construct a confidence interval for $\hat{\beta}$:

$$Pr\Big(\bar{\beta}_K - \frac{1.645 s_\beta}{\sqrt{K}} < \hat{\beta} < \bar{\beta}_K + \frac{1.645 s_\beta}{\sqrt{K}} \Big) = 0.90$$

where

$$\bar{\beta}_K = \frac{1}{K} \sum_{k=1}^{K} \hat{\beta}_k$$

$$s_\beta^2 = \frac{1}{K-1} \sum_{k=1}^{K} (\hat{\beta}_k - \bar{\beta}_K)^2$$

Question 5

	OLS	FE	ACF
βk	0.079	0.077	0.033
βl	0.930	0.895	0.968

Table 3: **Production function coefficients**

1.4 Counterfactuals

Question 1

To plot the empirical distribution of productivity at T (see Figure 1),

- Use ACF estimates of β and the functions from 3(a), 3(b) and 3(c) to calculate ω_{it} for all t
- Keep only observations in period T and draw a kernel density plot over $\exp(\omega_{iT})$

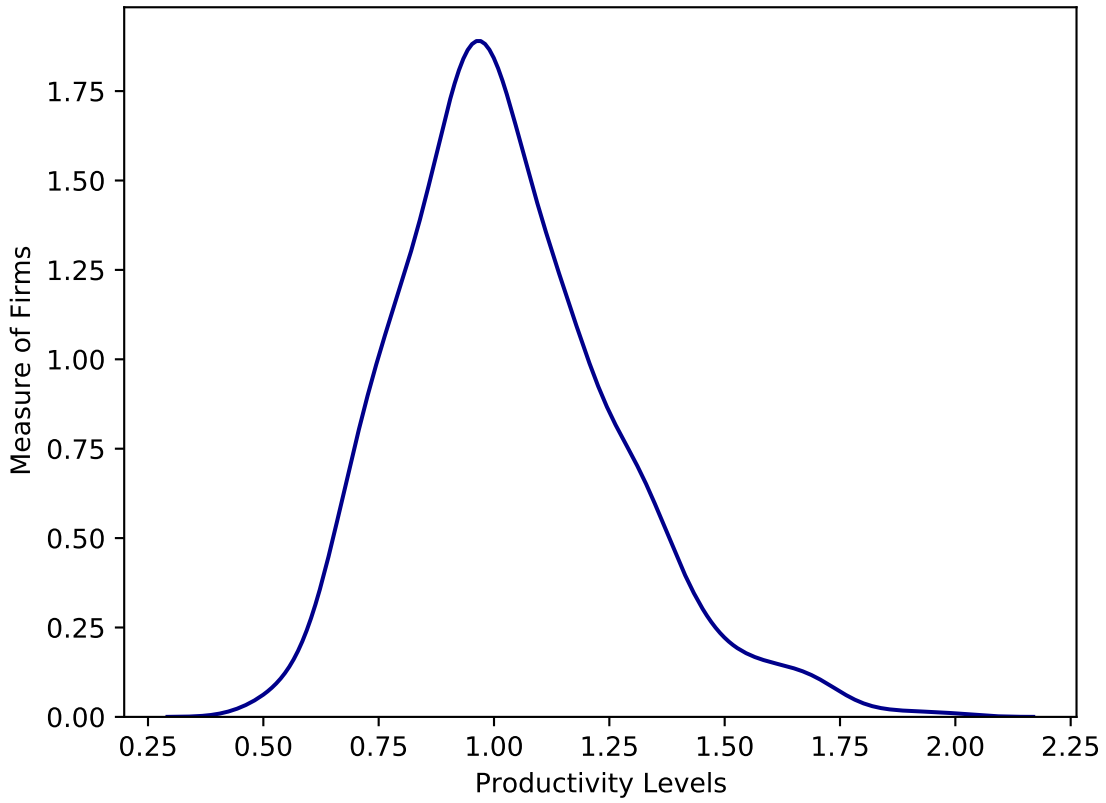

Figure 1: **Kernel Density Plot of Productivity** ($t = T$)

Question 2a

The empirical distribution of MPK at T is shown in Figure 2, and is calculated from the data as follows:

$$MPK = \exp(\hat{\omega}_{it}) \frac{\partial F(K_{it}, L_{it}; \beta)}{\partial K} = \exp(\hat{\omega}_{it}) \beta_k [\exp(k_{it})]^{\beta_k - 1} [\exp(l_{it})]^{\beta_l}$$

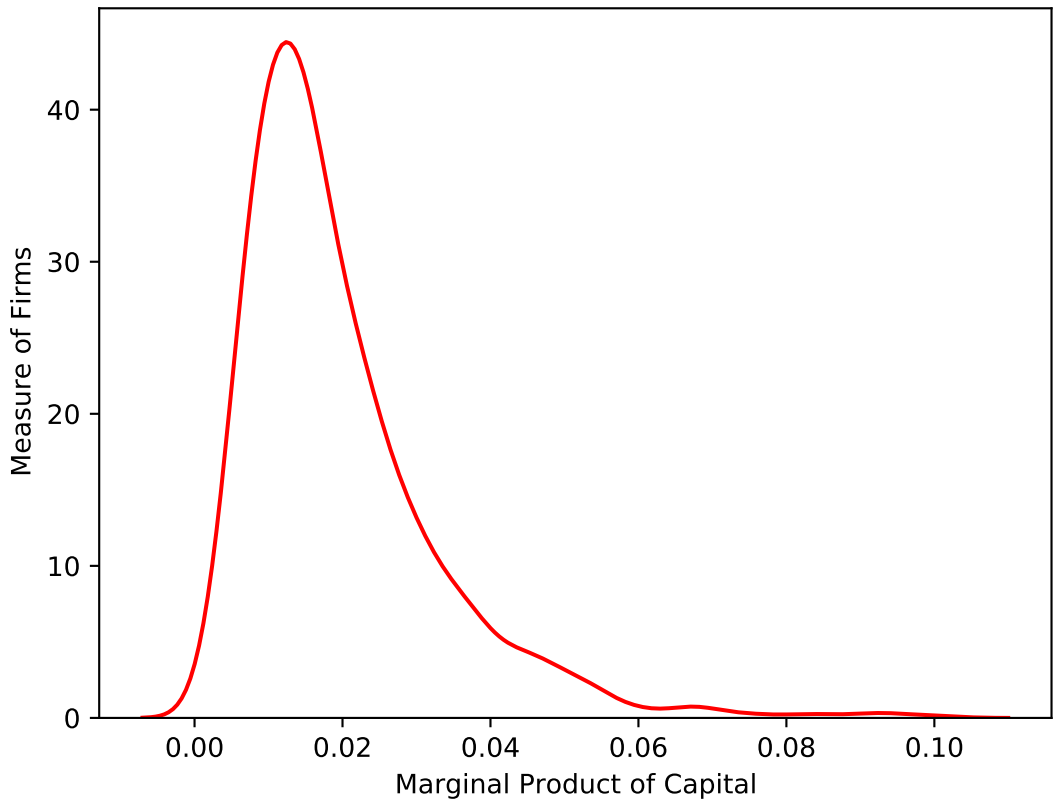

Figure 2: **Kernel Density Plot of Marginal Product of Capital** ($t = T$)

Questions 2b & 2c

	MPK
Average	0.020
First Decile	0.006
Ninth Decile	0.031

Table 4: **Summary Statistics for** MPK

Question 3a

$$K_T = \sum_{i=1}^{N} K_{iT} = 5816$$

Question 3c

	MPK
min	0.003
mean	0.042
max	0.679

Table 5: **Summary Statistics for MPK (Counterfactual)**

Question 3d

	Output
Data	3653
Counterfactual	3649

Table 6: **Total Output (Data vs. Counterfactual)**

Output under the counterfactual capital allocation is almost the same as the total output from the original capital allocation.

Python code for simulation exercise

ps1_simulation.py

```
import pandas as pd
import numpy as np

# (1a) Set seed and parameters
np.random.seed(123456789)
N=1000
T=50
beta_zero=0
beta_k=0.4
beta_l=0.6
rho=0.7
theta=0.3
delta=0.2
r=0.05

# (1b) Initial conditions for productivity and wages
sigma_omega=np.sqrt(0.3)
sigma_logwage=np.sqrt(0.1)
omega_zero=np.random.normal(0,sigma_omega,(N,1)) # omega_0 ~ N(0,0.3)
wage_zero=np.exp(np.random.normal(0,sigma_logwage,(N,1))) # log wage_0 ~ N(0,0.1)

# (1b) Shocks to productivity and wages (i.i.d. across firms and time)
sigma_xi=np.sqrt(0.214)
sigma_nu=np.sqrt(0.1)
xi=np.random.normal(0,sigma_xi,(N,T)) # xi ~ N(0,0.214)
nu=np.exp(np.random.normal(0,sigma_nu,(N,T))) # log nu ~ N(0,0.1)

# (1c) Paths for productivity and wages
M=np.zeros((N,T))
omega=np.concatenate((omega_zero,M),axis=1)
wage=np.concatenate((wage_zero,M),axis=1)

for t in range(T):
        omega[:,t+1]=rho*omega[:,t]+xi[:,t] # Productivity path from t=0 to t=50
        wage[:,t+1]=theta*wage[:,t]+nu[:,t] # Wage path from t=0 to t=50


# (1c) Generate time-invariant investment costs for firms
sigma_g=np.sqrt(0.6)
g=np.random.normal(0,sigma_g,(N,1)) # g = log(1/gamma)~N(0,0.6)
gamma=1/np.exp(g) # gamma = 1/exp(g)

# (2d) Path for investments from t=0 to t=49 (use FOC for investments)
inv=np.zeros((N,T))
```

```
for i in range(N):
        for t in range(T):
                inv[i,t]=1/gamma[i]*beta_k*np.exp(omega[i,t+1])*\
                np.power((np.exp(omega[i,t+1])*beta_l/wage[i,t+1]),(beta_l/(1-beta_l)))


# (2d) Path for capital from t=0 to t=50 (use equation for capital growth)
# Assume that every firm begins with 1 unit of capital at t=0
capital=np.ones((N,T+1))
for t in range(T):
        capital[:,t+1]=(1-delta)*capital[:,t]+inv[:,t]


log_capital=np.log(capital)

# (2e) Path for labor from t=0 to t=50 (use FOC for labor)
labor=np.power((np.exp(omega)*np.power(capital,beta_k)*beta_l)/wage,1/(1-beta_l))
log_labor=np.log(labor)

# (2f) Simulate measurement error
sigma_err=np.sqrt(0.1)
err=np.random.normal(0,sigma_err,(N,T+1))

# (2f) Path for output (use equation 1)
log_output=beta_zero+beta_k*log_capital+beta_l*log_labor+omega+err
output=np.exp(log_output)

# (2g) Path for materials (output prior to measurement error)
log_materials=log_output-omega-err
materials=np.exp(log_materials)

# (3) Dataframe for q, k, l, m
# Index for firm_ids and time periods
firms=pd.DataFrame(np.linspace(1,N,num=N), columns=["firm_id"])
firm_id=pd.concat([firms]*51,ignore_index=True)
time=pd.Series(np.linspace(0,T,num=T+1)).repeat(1000)
time=pd.DataFrame(time, columns=["time"]).reset_index()
# Reshape arrays and assemble dataframe
q_it=pd.DataFrame(log_output.flatten('F'), columns=["log_output"])
k_it=pd.DataFrame(log_capital.flatten('F'), columns=["log_capital"])
l_it=pd.DataFrame(log_labor.flatten('F'), columns=["log_labor"])
m_it=pd.DataFrame(log_materials.flatten('F'), columns=["log_materials"])

# (3) Assemble dataframe
dataframe=firm_id.join(time['time']).join(q_it).join(k_it).join(l_it).join(m_it)
```

Python code for GMM estimation

ps1_estimation.py

```
import os
os.chdir("/Users/christophersaw/Desktop/PS1")
import pyreadr
import pandas as pd
import numpy as np
from numpy.random import randint
import linearmodels
from linearmodels import OLS
from linearmodels.panel import PanelOLS
import statsmodels.api as sm
from pystout import pystout
import scipy as sp
from sklearn.preprocessing import PolynomialFeatures
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore", category=FutureWarning)

# Set seed
np.random.seed(123456789)

# Load data
robj=pyreadr.read_r('acf_sim_V2.RData')
print(robj.keys()) # retrieve object name
df=robj['acf_sim'] # convert to dataframe
n_obs=int(len(df))
n_firms=int(max(df['firm_id']))
n_periods=int(max(df['year']))
panel_df=df.set_index(['firm_id', 'year']) # set entity and time indices

# (1) OLS and (2) FE regressions
dep_var=panel_df['q']
lin_var=sm.add_constant(panel_df[['k','labor']])
ols=OLS(dep_var, lin_var).fit()
fe=PanelOLS(dep_var, lin_var, entity_effects=True)\
.fit(cov_type="clustered", cluster_entity=True)
# Save results
pystout(models=[ols,fe],
        file='ols_fe.tex',
        digits=3,
        mgroups={'OLS':1,'FE':2},
        varlabels={'const':'Constant','k':'Capital','labor':'Labor'},
        modstat={'nobs':'Obs','rsquared':'R\sym{2}'},
        addnotes=['Standard errors in parentheses',
        '\sym{**} p\sym{_<}0.01, * p\sym{_<}0.05, \sym{+} p\sym{_<}0.10']
```

```
        )

# (3a) Non-parameteric regression (polynomial approximation)
poly=PolynomialFeatures(degree=3) # set degree of polynomial
poly_var=panel_df[['k','labor','m']] # endogenous variables for polynominal regression
poly_var=pd.DataFrame(poly.fit_transform(poly_var)) # calculate terms up to third degree
poly_reg=OLS(dep_var,poly_var).fit() # Regression of q on phi(k,l,m)
phi_it=np.array(poly_reg.predict()) # Calculate fitted values

# (3b) Construct function f(k,l; b) = b0 + bl*l + bk*k
X=np.array(sm.add_constant(panel_df[['k','labor']]))
def f(beta):
        f_beta=np.zeros((n_obs,1))
        for i in range(n_obs):
                        f_beta[i]=np.sum(X[i,j]*beta[j] for j in range(0,3))
        return f_beta

# (3c) Construct functions for omega and xi
def omega(beta):
        omega_beta=phi_it-f(beta) # use results from (3a),(3b)
        return omega_beta

def xi(beta):
        W1=np.array(omega(beta))[n_firms:] # remove first n_firm observations in year 1
        W0=np.array(omega(beta))[0:n_obs-n_firms] # remove last n_firm observations in year T
        rho=np.linalg.inv(W0.transpose() @ W0) @ W0.transpose() @ W1 # OLS
        xi_beta=W1-rho*W0 # Calculate residuals xi
        return xi_beta

# (3d) Assemble instruments Z=[1 k l_1]
Z=np.array(sm.add_constant(panel_df[['k','labor_1']]))
Z=Z[n_firms:,0:] # remove first n_firm observations in year 1

# (3e) GMM objective function
W=np.linalg.inv(Z.transpose() @ Z) # optimal weighting matrix for GMM
def gmmobjfn(beta):
        gmmobjfn=xi(beta).transpose() @ Z @ W @ Z.transpose() @ xi(beta)
        return gmmobjfn

# (3f) Solve for beta
beta_init=np.array([[0.4083],[0.0766],[0.8952]]) # use beta_FE
results=sp.optimize.minimize(gmmobjfn,beta_init,\
        method='Nelder-Mead',options={'maxiter':1000})
 # final_simplex: (array([[0.45077117, 0.03287014, 0.96821934],
 #       [0.45084014, 0.03284315, 0.96825184],
 #       [0.45086032, 0.03280441, 0.9682736 ],
 #       [0.45082943, 0.03283342, 0.96822781]]),
 #                 array([6.48237678e-06, 6.86708258e-06, 7.77195868e-06, 8.16875677e-06]))
```

```
 #            fun: 6.4823767804439994e-06
 #        message: 'Optimization terminated successfully.'
 #           nfev: 109
 #            nit: 61
 #         status: 0
 #        success: True
 #              x: array([0.45077117, 0.03287014, 0.96821934])
beta_acf=results.x
gmm_value=results.fun

# (5) Tabulate and export results
beta_ols=pd.DataFrame(ols.params).reset_index()
beta_fe=pd.DataFrame(fe.params).reset_index()
beta_acf=pd.DataFrame(beta_acf.transpose())
coeffs=beta_ols.merge(beta_fe,on='index').join(beta_acf)\
.rename(columns={'index':'','parameter_x':'OLS','parameter_y':'FE',0:'ACF'})
coeffs.to_csv('coeffs.csv',index=False)
# coeffs
#                 OLS        FE       ACF
# 0  const  0.391694  0.408290  0.450771
# 1      k  0.079261  0.076602  0.032870
# 2  labor  0.930155  0.895151  0.968219
```

Python code for bootstraps

ps1_bootstrap.py

```
import os
os.chdir("/Users/christophersaw/Desktop/PS1")
import pyreadr
import pandas as pd
import numpy as np
from numpy.random import randint
import linearmodels
from linearmodels import OLS
from linearmodels.panel import PanelOLS
import statsmodels.api as sm
from pystout import pystout
import scipy as sp
from sklearn.preprocessing import PolynomialFeatures
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore", category=FutureWarning)

# Set seed
np.random.seed(123456789)

# Load data
robj=pyreadr.read_r('acf_sim_V2.RData')
print(robj.keys()) # retrieve object name
df=robj['acf_sim'] # convert to dataframe

# (3g) Construct K bootstrap draws of the dataset

# NOTES:
# Want to construct K subsamples, each with Nk observations 'Nk x K'
# Since the dataset is a panel, we should take draws of firm_id with replacement
# Total number of draws = Nk x K / n_periods
# Place all firm x time draws into 'dfb', dfb contains all 'Nk x K' bootstrap obs
# Then divide 'dfb' into 40 subsamples
# Each subsample consists of 50 firms, each observed for 20 periods

# Parameters for bootstrap
n_obs=int(len(df)) # 20,000
n_firms=int(max(df['firm_id'])) # 1000
n_periods=int(max(df['year'])) # 20
K=40
Nk=1000
n_draws=int(Nk*K/n_periods) # 2,000
nb_firms=int(Nk/n_periods) # no. of firms in each subsample
```

```
# Draw firms with replacement
draws=pd.DataFrame(randint(1,1+n_firms,(n_draws,1)))\
.rename(columns={0:'firm_id'}) # Draw 2,000 firms
dfb=pd.merge(draws,df,on='firm_id',how ='inner',validate="m:m")

# Divide into subsamples and create an id for each subsample
n_subsample=pd.Series(np.linspace(1,K,num=K)) # create 40 subsamples
newcol=pd.DataFrame(n_subsample.repeat(Nk))\
.rename(columns={0:'subsample'}).reset_index() # each with Nk obs

# Join subsample ids onto dfb
dfb=pd.merge(dfb,newcol['subsample'],left_index=True, right_index=True)

# (3h) Recover K estimates of beta

# Functions for GMM (dimensions adjusted for subsample)
def f(beta):
        f_beta=np.zeros((Nk,1))
        for i in range(Nk):
                        f_beta[i]=np.sum(X[i,j]*beta[j] for j in range(0,3))
        return f_beta

def omega(beta):
        omega_beta=phi_it-f(beta) # use results from (3a),(3b)
        return omega_beta

def xi(beta):
        W1=np.array(omega(beta))[nb_firms:]
        W0=np.array(omega(beta))[0:Nk-nb_firms]
        rho=np.linalg.inv(W0.transpose() @ W0) @ W0.transpose() @ W1
        xi_beta=W1-rho*W0
        return xi_beta

def gmmobjfn(beta):
        gmmobjfn=xi(beta).transpose() @ Z @ W @ Z.transpose() @ xi(beta)
        return gmmobjfn

# Use same initial guess for all subsamples
beta_init=np.array([[0.4083],[0.0766],[0.8952]])

# Create a new firm_id that is unique within each subsample
new_firm_id=pd.Series(np.linspace(1,nb_firms,num=nb_firms))
newcol2=pd.DataFrame(new_firm_id.repeat(n_periods))\
.rename(columns={0:'new_firm_id'}).reset_index()

# Create a box to store our bootstrap estimates of beta
bootstrap_box=np.zeros((K,3))
```

```
# Bootstrap estimates of beta, loop over k
# Within each loop, construct X, Z, W
# and estimate phi, f, omega, xi
for k in range(K):
        dfk=dfb.loc[dfb['subsample']==k+1].reset_index() # keep only observations in k
        dfk=pd.merge(dfk,newcol2['new_firm_id'],\
        left_index=True, right_index=True) # use new_firm_id
        dfk=dfk.sort_values(by=['year','new_firm_id']) # sort data to set panel
        dep_var=dfk['q'] # dependent variable
        poly=PolynomialFeatures(degree=3) # set degree of polynomial
        poly_var=dfk[['k','labor','m']] # endogenous variables for polynominal regression
        poly_var=pd.DataFrame(poly.fit_transform(poly_var))
        poly_reg=OLS(dep_var,poly_var).fit()
        phi_it=np.array(poly_reg.predict())
        X=np.array(sm.add_constant(dfk[['k','labor']])) # endogenous variables
        Z=np.array(sm.add_constant(dfk[['k','labor_1']])) #instruments
        Z=Z[nb_firms:,0:] # remove first nb_firm observations in year 1
        W=np.linalg.inv(Z.transpose() @ Z) # optimal weighting matrix for GMM
        results=sp.optimize.minimize(gmmobjfn,beta_init,\
                method='Nelder-Mead',options={'maxiter':1000})
        beta_acf=results.x
        bootstrap_box[k,:]=beta_acf


# (3h) Use bootstrap estimates to calculate 90% CI
c=1.645
bootstraps=pd.DataFrame(bootstrap_box).rename(columns={0:'beta_0',1:'beta_k',2:'beta_l'})
bk_bar=bootstraps.beta_k.mean() # mean of bootstrap estimates of beta_k
bk_s=bootstraps.beta_k.std() # std dev of bootstrap estimates
beta_k_LB=bk_bar-(c*bk_s/np.sqrt(K)) # LB for 90% CI
beta_k_UB=bk_bar+(c*bk_s/np.sqrt(K)) # UB for 90% CI
bl_bar=bootstraps.beta_l.mean() # repeat above for beta_l
bl_s=bootstraps.beta_l.std()
beta_l_LB=bl_bar-(c*bl_s/np.sqrt(K))
beta_l_UB=bl_bar+(c*bl_s/np.sqrt(K))

# Export estimates to csv
bootstraps.to_csv('bootstraps.csv',index=False)
pd.DataFrame([beta_k_LB,beta_k_UB])\
.rename({0:'CI beta_k'},axis=1).rename({0:'LB',1:'UB'},axis=0)\
.to_csv('beta_k_CI.csv',index=False)
pd.DataFrame([beta_l_LB,beta_l_UB]).rename({0:'CI beta_l'},axis=1)\
.rename({0:'LB',1:'UB'},axis=0)\
.to_csv('beta_l_CI.csv',index=False)
```

Python code for Counterfactual Exercise

ps1_counterfactual.py

```
import os
os.chdir("/Users/christophersaw/Desktop/PS1")
import pyreadr
import pandas as pd
import numpy as np
from numpy.random import randint
import linearmodels
from linearmodels import OLS
from linearmodels.panel import PanelOLS
import statsmodels.api as sm
from pystout import pystout
import scipy as sp
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore", category=FutureWarning)

# Set seed
np.random.seed(123456789)

# Load data
robj=pyreadr.read_r('acf_sim_V2.RData')
print(robj.keys()) # retrieve object name
df=robj['acf_sim'] # convert to dataframe
n_obs=int(len(df))
n_firms=int(max(df['firm_id']))
n_periods=int(max(df['year']))
panel_df=df.set_index(['firm_id', 'year']) # set entity and time indices

# Load estimates
coeffs=pd.read_csv('coeffs.csv')
beta=np.array(coeffs)[:,3]

# Repeat steps (3a) to (3c) with ACF estimates
dep_var=panel_df['q']

# (3a) Non-parameteric regression (polynomial approximation)
poly=PolynomialFeatures(degree=3) # set degree of polynomial
poly_var=panel_df[['k','labor','m']] # endogenous variables for polynominal regression
poly_var=pd.DataFrame(poly.fit_transform(poly_var)) # calculate terms up to third degree
poly_reg=OLS(dep_var,poly_var).fit() # Regression of q on phi(k,l,m)
phi_it=np.array(poly_reg.predict()) # Calculate fitted values
```

```
# (3b) Construct function f(k,l; b) = b0 + bl*l + bk*k
X=np.array(sm.add_constant(panel_df[['k','labor']]))
f_beta=np.zeros((n_obs,1))
for i in range(n_obs):
        f_beta[i]=np.sum(X[i,j]*beta[j] for j in range(0,3))


# (3c) Calculate productiivty omega and merge onto dataframe
omega=pd.DataFrame(phi_it-f_beta).rename(columns={0:'omega'})
df=pd.merge(df,omega,left_index=True, right_index=True)

# Keep only observations at T
T=int(max(df['year']))
dft=df.loc[df['year']==T].reset_index()

# (1) Plot empirical distribution of productivity at T
dft['W']=np.exp(dft['omega'])
sns.distplot(dft['W'], hist=False, kde=True, bins=int(20), color='darkblue')
plt.xlabel('Productivity Levels')
plt.ylabel('Measure of Firms')
plt.savefig('productivity(levels).pdf')
plt.close('all')

# Convert q, k, l from logs to levels
dft['Q']=np.exp(dft['q'])
dft['K']=np.exp(dft['k'])
dft['L']=np.exp(dft['labor'])

# Calculate A = exp(b0)*exp(w_it)*exp(e_it)
dft['epsilon']=dft['q']-beta[0]-beta[1]*dft['k']-beta[2]*dft['labor']-dft['omega']
dft['A']=np.exp(beta[0])*np.exp(dft['omega'])*np.exp(dft['epsilon'])

# (2a) Plot marginal product of capital at T
dft['MPK']=dft['A']*beta[1]*np.power(dft['K'],(beta[1]-1))*np.power(dft['L'],beta[2])
sns.distplot(dft['MPK'], hist=False, kde=True, bins=int(20), color='red')
plt.xlabel('Marginal Product of Capital')
plt.ylabel('Measure of Firms')
plt.savefig('MPK.pdf')
plt.close('all')

# (2b) Average MPK across firms at T
avg_mpk=dft['MPK'].mean()

# (2c) First and ninth deciles of MPK distribution
dft['MPK_decile']=pd.qcut(dft['MPK'], 10, labels=False)
decile_1_mpk=dft.loc[dft['MPK_decile']==0,'MPK'].mean()
decile_9_mpk=dft.loc[dft['MPK_decile']==8,'MPK'].mean()
```

```
# Save results
mpk_stats=pd.DataFrame(np.array([[avg_mpk],[decile_1_mpk],[decile_9_mpk]]))\
.rename({0:'MPK'},axis=1).rename({0:'Average',1:'First Decile',2:'Ninth Decile'},axis=0)
mpk_stats.to_csv('MPK.csv')
#                       MPK
# Average        0.019624
# First Decile   0.005749
# Ninth Decile   0.031467

# (3a) Total economy capital stock and output
total_Q=dft['Q'].sum()
total_K=dft['K'].sum()
print(total_K)

# (3b) Capital reallocation
# Create arrays and parameters for capital reallocation
MPK=np.array(dft['MPK'])
K=np.array(dft['K'])
L=np.array(dft['L'])
A=np.array(dft['A'])
new_MPK=np.zeros((n_firms,1))
max_MPK=np.max(MPK)
step=1
# Suppose initial capital allocation for every firm is set to K min
new_K=np.ones((n_firms,1))*np.min(K)
new_total_K=np.sum(new_K)
# Iterate
while new_total_K < total_K:
        i=np.where(MPK==max_MPK)
        new_K[i]=K[i]+step
        new_MPK[i]=A[i]*beta[1]*np.power(new_K[i],(beta[1]-1))*np.power(L[i],beta[2])
        K[i]=new_K[i]
        MPK[i]=new_MPK[i]
        max_MPK=np.max(MPK)
        new_total_K=np.sum(new_K)
        new_total_K


# (3c) Marginal product of capital
for i in range(n_firms):
        new_MPK[i]=A[i]*beta[1]*np.power(new_K[i],(beta[1]-1))*np.power(L[i],beta[2])


MPK_stats=np.array([np.min(new_MPK),np.mean(new_MPK),np.max(new_MPK)])
MPK_stats=pd.DataFrame(MPK_stats)\
.rename({0:'MPK'},axis=1)\
.rename({0:'min',1:'mean',2:'max'},axis=0)
MPK_stats.to_csv('Counterfactual_MPK.csv',index=False)
```

```
#             MPK
# min    0.003446
# mean   0.041816
# max    0.679497

# Calculate counterfactual output
dft['new_K']=new_K # Join new capital allocation to dataframe
dft['new_Q']=dft['A']*np.power(dft['new_K'],beta[1])*np.power(dft['L'],beta[2])
new_total_Q=dft['new_Q'].sum()
output_compare=np.array([total_Q,new_total_Q])
output_compare=pd.DataFrame(output_compare)\
.rename({0:'Total Output'},axis=1)\
.rename({0:'Data',1:'Counterfactual'})
output_compare.to_csv('Counteractual_Output.csv',index=False)
#                  Total Output
# Data              3652.512807
# Counterfactual    3648.744031
```