In [38]:
```
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 wooldridge as woo
import quantecon as qe
from quantecon import MarkovChain
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.model_selection import cross_val_score
```

1a) When there is a big variation in the independent variable given different values of x. Taking the log could help decrease the variation. If there is a big range of values of our independent variable, taking the log would convert it to percentile change and help us manage our range.

1b) Having our residuals randomly distributed around 0 implies that our model is a good fit. It implies that out estimates are very close to the actual value were looking for.

1c) It is a good method to test for the robustness of out estimates. It can help us construct confidence intervals and find accurate statistics such as standard errors.

1d) If one of our explanatory variable is equal to 0.

2) Integrals represents areas under the curve, using monte carlo simulations, we can take different points for different values of x, and estimate the area under the curve for each value of x. We can then average the areas under the curve for each value of x, and find the total area under the curve.

3) MLE

4)

5a) The slope equals 1.9803. this means that 1% increase in education, would raise hourly wages by $0.019803.

b) elasticity = B2 x 1/y. 1.9803 x 1/20.6 = 0.096.

c) 32.8975 - 13.0927 = $19.803

6a) The lowest value for age combined with a highets value for sqft would reflect the highest selling prices age =1 , sqrft = 100

6b)

7)

8) it would be approximately 9% change.

9) 169.81

10) 50

In [15]:
```
df = woo.dataWoo('bwght')
```

In [16]:
```
df
```

Out[16]:

	faminc	cigtax	cigprice	bwght	fatheduc	motheduc	parity	male	white	cigs	lbwght
0	13.5	16.5	122.300003	109	12.0	12.0	1	1	1	0	4.691348
1	7.5	16.5	122.300003	133	6.0	12.0	2	1	0	0	4.890349
2	0.5	16.5	122.300003	129	NaN	12.0	2	0	0	0	4.859812
3	15.5	16.5	122.300003	126	12.0	12.0	2	1	0	0	4.836282
4	27.5	16.5	122.300003	134	14.0	12.0	2	1	1	0	4.897840
...	...	...	...	...	...	...	...	...	...	...	...
1383	27.5	30.0	138.300003	110	12.0	12.0	4	1	1	0	4.700480
1384	5.5	30.0	138.300003	146	NaN	16.0	2	1	1	0	4.983607
1385	65.0	8.0	118.599998	135	18.0	16.0	2	0	1	0	4.905275
1386	27.5	8.0	118.599998	118	NaN	14.0	2	0	1	0	4.770685
1387	37.5	8.0	118.599998	111	16.0	13.0	2	0	1	0	4.709530

1388 rows × 14 columns

In [17]:
```
X = df[['faminc']].copy()
X["cigs"] = df["cigs"]
X = sm.add_constant(X)
Y = df['bwght']
result1 = sm.OLS(Y, X).fit()
result1.summary()
```

Out[17]:

OLS Regression Results

Dep. Variable:	bwght	**R-squared:**	0.030
Model:	OLS	**Adj. R-squared:**	0.028
Method:	Least Squares	**F-statistic:**	21.27
Date:	Fri, 29 Oct 2021	**Prob (F-statistic):**	7.94e-10
Time:	17:22:59	**Log-Likelihood:**	-6130.4
No. Observations:	1388	**AIC:**	1.227e+04
Df Residuals:	1385	**BIC:**	1.228e+04
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	116.9741	1.049	111.512	0.000	114.916	119.032
faminc	0.0928	0.029	3.178	0.002	0.036	0.150
cigs	-0.4634	0.092	-5.060	0.000	-0.643	-0.284

Omnibus:	116.751	**Durbin-Watson:**	1.922
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	619.781
Skew:	-0.154	**Prob(JB):**	2.61e-135
Kurtosis:	6.259	**Cond. No.**	67.4

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

In [34]:
```
boot_slopes = []
boot_interc = []
boot_adjR2 = []
n_boots = 100
n_points = df.shape[0]
plt.figure()
for _ in range(n_boots):

    sample_df = df.sample(n=n_points, replace=True)

    OLS_boot = sm.OLS(formula = 'bwght ~ faminc , cigs', data=sample_df)
    results_boot = OLS_boot.fit()

    boot_interc.append(results_boot.params[0])
    boot_slopes.append(results_boot.params[1])
    boot_adjR2.append(results_boot.rsquared_adj)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-34-02cc520f5553> in <module>
      9     sample_df = df.sample(n=n_points, replace=True)
     10 
---> 11     OLS_boot = sm.OLS(formula = 'bwght ~ faminc , cigs', data=sample_df)
     12     results_boot = OLS_boot.fit()
     13 

TypeError: __init__() missing 1 required positional argument: 'endog'
```

```
<Figure size 432x288 with 0 Axes>
```

In [32]:
```
P = [[1/2, 1/4, 1/4],
     [1/3, 1/3, 1/3],
     [0 , 1/3,  3/4]]
mc = qe.MarkovChain(P)
X = mc.simulate(ts_length=10000)

np.mean(X == 0)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-32-2ce038272b7a> in <module>
      2      [1/3, 1/3, 1/3],
      3      [0 , 1/3,  3/4]]
----> 4 mc = qe.MarkovChain(P)
      5 X = mc.simulate(ts_length=10000)
      6

~/opt/anaconda3/lib/python3.8/site-packages/quantecon/markov/core.py in __init__(self, P, state_values)
    195             row_sums = row_sums.getA1()
    196         if not np.allclose(row_sums, np.ones(self.n)):
--> 197             raise ValueError('The rows of P must sum to 1')
    198
    199         # Call the setter method

ValueError: The rows of P must sum to 1
```

In [36]:
```
crime = woo.dataWoo('crime1')
```

In [37]:
```
crime
```

Out[37]:

	narr86	nfarr86	nparr86	pcnv	avgsen	tottime	ptime86	qemp86	inc86	durat	b
0	0	0	0	0.38	17.600000	35.200001	12	0.0	0.000000	0.0	
1	2	2	0	0.44	0.000000	0.000000	0	1.0	0.800000	0.0	
2	1	1	0	0.33	22.799999	22.799999	0	0.0	0.000000	11.0	
3	2	2	1	0.25	0.000000	0.000000	5	2.0	8.800000	0.0	
4	1	1	0	0.00	0.000000	0.000000	0	2.0	8.100000	1.0	
...	...	...	...	...	...	...	...	...	...	...	
2720	1	1	0	0.00	0.000000	0.000000	0	0.0	0.000000	3.0	
2721	0	0	0	0.00	0.000000	0.000000	0	3.0	11.500000	1.0	
2722	0	0	0	0.00	0.000000	0.000000	0	1.0	1.900000	1.0	
2723	1	1	0	0.00	0.000000	0.000000	0	0.0	0.000000	19.0	
2724	0	0	0	0.00	0.000000	0.000000	0	4.0	191.300003	0.0	

2725 rows × 16 columns

In [41]:
```
x = crime['pcnv']
y = crime['narr86']

regr = LinearRegression()
model = regr.fit(x,y)
regr.coef_
regr.intercept_
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, ra

regr = LinearRegression()
regr.fit(x_train, y_train)

y_pred = regr.predict(x_test)

print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))


regr = linear_model.LinearRegression()
scores = cross_val_score(regr, x, y, cv=5, scoring='neg_root_mean_squared_e
print('5-Fold CV MSE Scores:', scores)
```

```
---------------------------------------------------------------------------
--
ValueError                                Traceback (most recent call las
t)
<ipython-input-41-62e3b5d032fc> in <module>
      3
      4 regr = LinearRegression()
----> 5 model = regr.fit(x,y)
      6 regr.coef_
      7 regr.intercept_

~/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_base.py
in fit(self, X, y, sample_weight)
    516         accept_sparse = False if self.positive else ['csr', 'csc'
, 'coo']
    517
--> 518         X, y = self._validate_data(X, y, accept_sparse=accept_spa
rse,
    519                                    y_numeric=True, multi_output=T
rue)
    520

~/opt/anaconda3/lib/python3.8/site-packages/sklearn/base.py in _validate_
data(self, X, y, reset, validate_separately, **check_params)
    431                 y = check_array(y, **check_y_params)
    432             else:
--> 433                 X, y = check_X_y(X, y, **check_params)
    434             out = X, y
    435

~/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py i
n inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
```

```
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

~/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py i
n check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order, copy,
force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples,
 ensure_min_features, y_numeric, estimator)
    812         raise ValueError("y cannot be None")
    813
--> 814     X = check_array(X, accept_sparse=accept_sparse,
    815                     accept_large_sparse=accept_large_sparse,
    816                     dtype=dtype, order=order, copy=copy,

~/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py i
n inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

~/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py i
n check_array(array, accept_sparse, accept_large_sparse, dtype, order, co
py, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min
_features, estimator)
    635             # If input is 1D raise error
    636             if array.ndim == 1:
--> 637                 raise ValueError(
    638                     "Expected 2D array, got 1D array instead:\nar
ray={}.\n"
    639                     "Reshape your data either using array.reshape
(-1, 1) if "

ValueError: Expected 2D array, got 1D array instead:
array=[0.38       0.44       0.33000001 ... 0.         0.         0.
].
Reshape your data either using array.reshape(-1, 1) if your data has a si
ngle feature or array.reshape(1, -1) if it contains a single sample.
```

In [42]:
```
ceosal = woo.dataWoo('ceosal1')
```

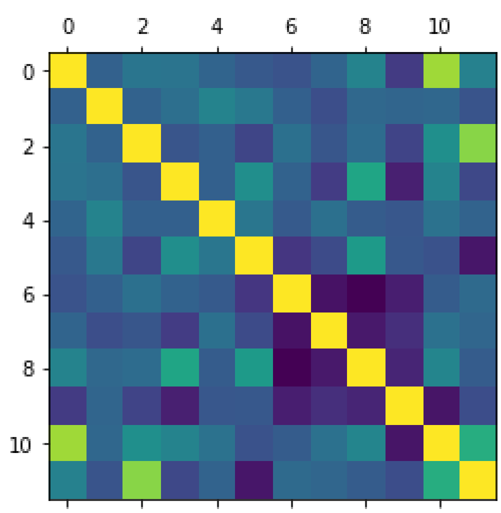
In [43]:
```
ceosal
```

Out[43]:

	salary	pcsalary	sales	roe	pcroe	ros	indus	finance	consprod	utility	lsal
0	1095	20	27595.000000	14.1	106.400002	191	1	0	0	0	6.998
1	1001	32	9958.000000	10.9	-30.600000	13	1	0	0	0	6.908
2	1122	9	6125.899902	23.5	-16.299999	14	1	0	0	0	7.022
3	578	-9	16246.000000	5.9	-25.700001	-21	1	0	0	0	6.359
4	1368	7	21783.199219	13.8	-3.000000	56	1	0	0	0	7.221
...	...	...	...	...	...	...	...	...	...	...	
204	930	10	1509.099976	9.0	20.500000	131	0	0	0	1	6.835
205	525	3	1097.099976	15.5	20.100000	72	0	0	0	1	6.263
206	658	32	4542.600098	12.1	-7.800000	68	0	0	0	1	6.489
207	555	6	2023.000000	13.7	-14.600000	60	0	0	0	1	6.318
208	626	0	1442.500000	14.4	-10.200000	62	0	0	0	1	6.439

209 rows × 12 columns

In [44]:
```
plt.matshow(ceosal.corr())
plt.show()
```



In [45]:
```
sav = woo.dataWoo('saving')
```

In [46]:
```
sav
```

Out[46]:

	sav	inc	size	educ	age	black	cons
0	30	1920	4	2	40	1	1890
1	874	12403	4	9	33	0	11529
2	370	6396	2	17	31	0	6026
3	1200	7005	3	9	50	0	5805
4	275	6990	4	12	28	0	6715
...	...	...	...	...	...	...	...
95	1800	32080	2	16	54	0	30280
96	1684	9260	5	12	31	0	7576
97	1475	10450	2	18	27	0	8975
98	566	9138	5	12	40	0	8572
99	25405	12350	6	18	34	0	-13055

100 rows × 7 columns

In [48]:
```
X = sav['inc']
X = sm.add_constant(X)
Y = sav['sav']
OLS_mod = sm.OLS(Y, X).fit()
OLS_mod.summary()
```

Out[48]:

OLS Regression Results

Dep. Variable:	sav	**R-squared:**	0.062
Model:	OLS	**Adj. R-squared:**	0.053
Method:	Least Squares	**F-statistic:**	6.492
Date:	Fri, 29 Oct 2021	**Prob (F-statistic):**	0.0124
Time:	18:10:39	**Log-Likelihood:**	-947.89
No. Observations:	100	**AIC:**	1900.
Df Residuals:	98	**BIC:**	1905.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	124.8424	655.393	0.190	0.849	-1175.764	1425.449
inc	0.1466	0.058	2.548	0.012	0.032	0.261

Omnibus:	126.825	**Durbin-Watson:**	1.536
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3750.981
Skew:	4.206	**Prob(JB):**	0.00
Kurtosis:	31.801	**Cond. No.**	2.33e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.33e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [ ]:
```

```