回归分析

Table of Contents

0.1 建立计量模型 1
0.2 OLS 估计量 1
0.3 IV 估计 11

0.1 建立计量模型

作者认为，一个国家的生活水平取决于该国与其他国家的国际贸易、该国的国内贸易以及其他因素。从一个简单的计量模型出发：

$$\ln Y_i = \alpha + \beta T_i + \gamma W_i + \epsilon_i$$

其中，**被解释变量**是人均收入 Y_i 的对数；**解释变量** T_i 表示国际贸易，W_i 表示国内贸易，ϵ_i 为**误差项**，表示其他对收入影响的因素。

预期国际贸易和

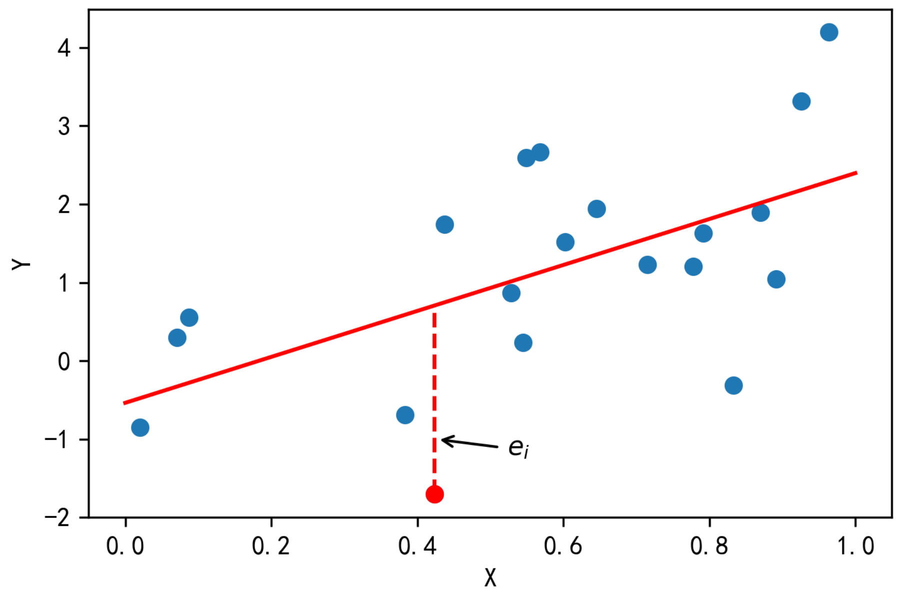
0.1.1 最小二乘回归方法

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams['font.family']='SimHei'
plt.rcParams['axes.unicode_minus'] = False
import statsmodels.api as sm
```

0.2 OLS 估计量

```
# Generate random data
np.random.seed(0)
x = np.random.rand(20)
y = 2 * x + np.random.randn(20)

# Fit a line using OLS
coefficients = np.polyfit(x, y, 1)
poly = np.poly1d(coefficients)
```

```
x_fit = np.linspace(0, 1, 100)
y_fit = poly(x_fit)

# Calculate residuals
residuals = y - poly(x)
# Plot scatter plot and fitted line
plt.scatter(x, y, label='Data')
plt.plot(x_fit, y_fit, color='red', label='Fitted Line')
plt.scatter(x[4], y[4], color='red', label='Selected Residual')
plt.plot([x[4], x[4]], [y[4], poly(x[4])], color='red', linestyle='--')

# Add annotation of the residual
plt.annotate(f'$e_i $ ',
             xy=(x[4], -1),
             xytext=(x[4]+0.1, y[4]+0.5),
             arrowprops=dict(facecolor='black', arrowstyle='->'))

plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```



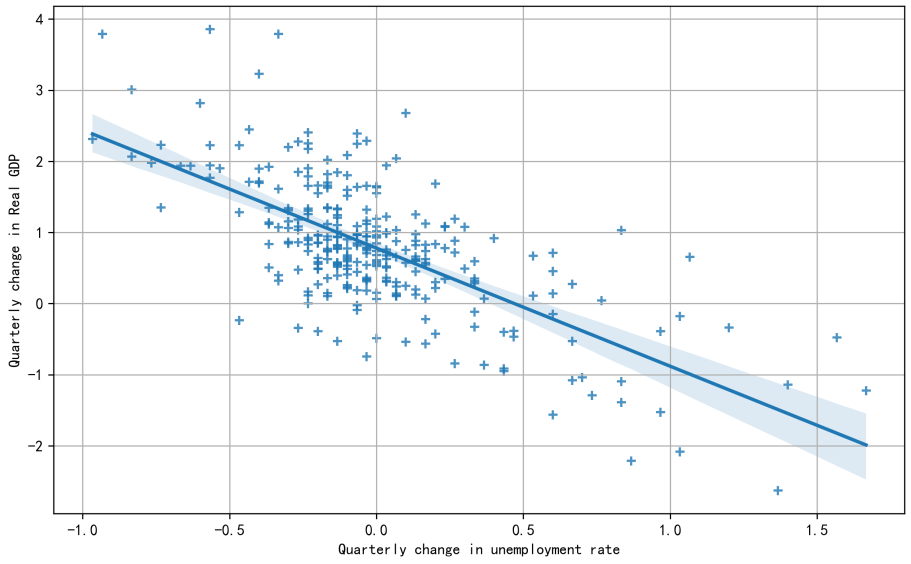
0.2.1 Okun's law

```
import pandas_datareader.data as web
start_date = '1947-01-01'
end_date = '2019-12-31'

data = web.DataReader(['UNRATE','GDPC1'], 'fred', start_date, end_date)
df = data.resample('Q').mean()
df['gdp_growth_rate'] = np.log(df['GDPC1']/df['GDPC1'].shift(1))*100
df['unemp_changed'] = df['UNRATE'].diff()
df.dropna(inplace=True)
fig,ax=plt.subplots(figsize=(10,6))
sns.regplot(df, y='gdp_growth_rate',x='unemp_changed',ax=ax,marker='+')
ax.set_xlabel('Quarterly change in unemployment rate')
ax.set_ylabel('Quarterly change in Real GDP')
ax.grid()
model = sm.OLS(df['gdp_growth_rate'], sm.add_constant(df['unemp_changed'])).fit()
```

```
print(model.summary())
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_6616\1037386578.py:6: FutureWarning: '
  df = data.resample('Q').mean()
```

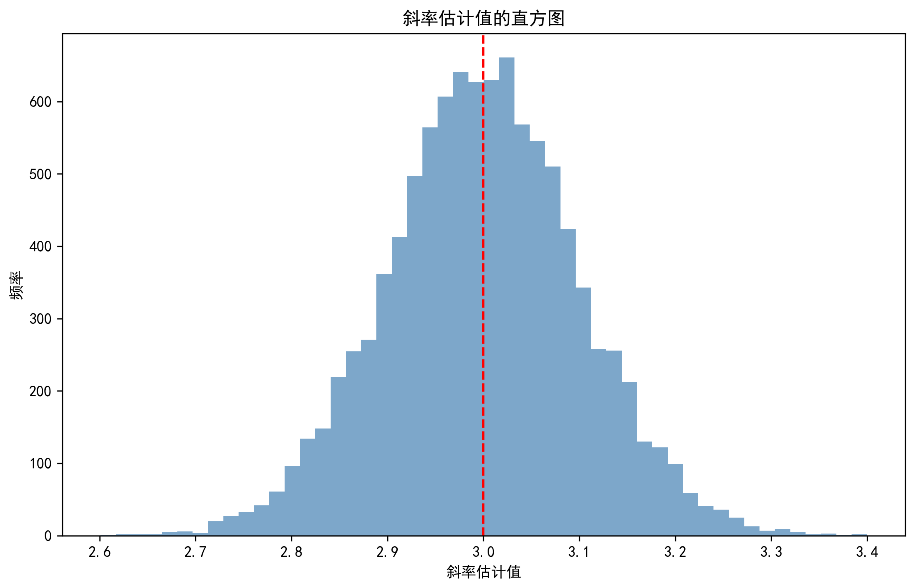
```
                            OLS Regression Results
==============================================================================
Dep. Variable:        gdp_growth_rate   R-squared:                       0.466
Model:                            OLS   Adj. R-squared:                  0.464
Method:                 Least Squares   F-statistic:                     248.4
Date:                Fri, 08 Aug 2025   Prob (F-statistic):           1.11e-40
Time:                        09:54:55   Log-Likelihood:                -295.55
No. Observations:                 287   AIC:                             595.1
Df Residuals:                     285   BIC:                             602.4
Df Model:                           1
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const             0.7788      0.040     19.403      0.000       0.700       0.858
unemp_changed    -1.6609      0.105    -15.762      0.000      -1.868      -1.453
==============================================================================
Omnibus:                       11.629   Durbin-Watson:                   1.912
Prob(Omnibus):                  0.003   Jarque-Bera (JB):               13.361
Skew:                           0.383   Prob(JB):                      0.00125
Kurtosis:                       3.729   Cond. No.                         2.63
==============================================================================

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



0.2.2 无偏性

```
np.random.seed(12345)
true_intercept = 2.0
true_slope = 3.0
sample_size = 100
num_simulations = 10000

estimated_slopes = np.zeros(num_simulations)
for i in range(num_simulations):
    x = np.random.randn(sample_size)
    y = true_intercept + true_slope * x + np.random.randn(sample_size)
    X = sm.add_constant(x)
    model = sm.OLS(y, X).fit()
    estimated_slopes[i] = model.params[1]
average_slope = np.mean(estimated_slopes)

print(f" 斜率真实值: {true_slope}")
print(f" 斜率估计值的平均值: {average_slope}")

fig, ax = plt.subplots(figsize=(10,6))
ax.hist(estimated_slopes, bins=50, color='steelblue', alpha=0.7)
ax.axvline(x=3.0, color='red', linestyle='--')
ax.set_xlabel('斜率估计值')
ax.set_ylabel('频率')
ax.set_title('斜率估计值的直方图')
plt.show()
```

```
斜率真实值: 3.0
斜率估计值的平均值: 2.9993457145927493
```



0.2.3 置信区间

```
### ci
lower, upper = (1.0941-1.96*0.029, 1.0941+1.96*0.029)
np.round((lower, upper),3)
```

```
array([1.037, 1.151])
```

0.2.4 联合显著性检验

0.2.5 解释回归结果

0.2.6 函数形式变化

0.2.7 回归诊断分析

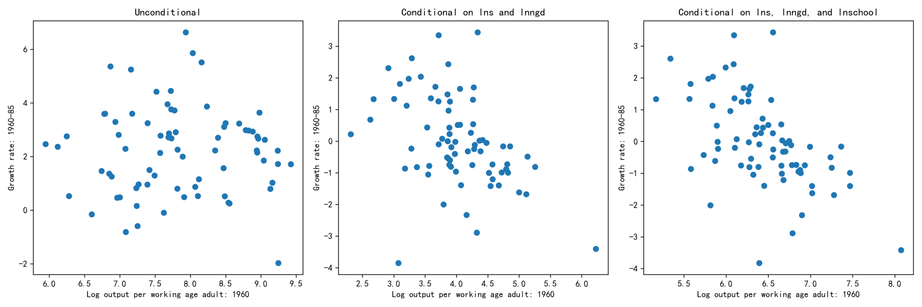
0.2.8 应用：Mankiw, Romer, Weil (1992)

这一部分以 @mankiw1992contribution 为例，阐释最小二乘法的应用。

```python
import pandas as pd
import numpy as np

import statsmodels.api as sm
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt

# Load data
data = pd.read_csv("datasets/mrw.csv")

# Generate variables
data['lnY85'] = np.log(data['rgdpw85'])
data['lnY60'] = np.log(data['rgdpw60'])
data['lns'] = np.log(data['i_y'] / 100)
data['lnngd'] = np.log(data['popgrowth'] / 100 + 0.05)
data['lnschool'] = np.log(data['school'] / 100)
data['growth'] = data['lnY85'] - data['lnY60']
data['growth_annu'] = 100 * data['growth'] / 25
data['lns_lnngd'] = data['lns'] - data['lnngd']
```

```
data['lnschool_lnngd'] = data['lnschool'] - data['lnngd']

# Table I
samples = {'n': data['n'] == 1, 'i': data['i'] == 1, 'o': data['o'] == 1}
results_table1 = {}
for sample, condition in samples.items():
    model = smf.ols('lnY85 ~ lns + lnngd', data=data[condition]).fit()
    results_table1[sample] = model.summary()

# Estimate alpha
model_i = smf.ols('lnY85 ~ lns + lnngd', data=data[data['i'] == 1]).fit()
alpha = model_i.params['lns'] / (1 + model_i.params['lns'])

# Restricted regression
results_restricted = {}
for sample, condition in samples.items():
    model = smf.ols('lnY85 ~ lns_lnngd', data=data[condition]).fit()
    results_restricted[sample] = model.summary()

# Table II
correlation = data.loc[data['i'] == 1, ['school', 'i_y', 'popgrowth']].corr()
results_table2 = {}
for sample, condition in samples.items():
    model = smf.ols('lnY85 ~ lns + lnngd + lnschool', data=data[condition]).fit()
    results_table2[sample] = model.summary()

# Test coefficients
test_model = smf.ols('lnY85 ~ lns + lnngd + lnschool', data=data[data['i'] == 1]).
test1 = test_model.t_test('lns + lnngd + lnschool = 0')
test2 = test_model.t_test('lns = lnschool')

# Implicit alpha
alpha_lns = test_model.params['lns'] / (1 + test_model.params['lns'] + test_model.
alpha_lnschool = test_model.params['lnschool'] / (1 + test_model.params['lns'] + test_model.

# Restricted regression with school
results_restricted_school = {}
for sample, condition in samples.items():
    model = smf.ols('lnY85 ~ lns_lnngd + lnschool_lnngd', data=data[condition]).fit()
    results_restricted_school[sample] = model.summary()

# Table III
results_table3 = {}
for sample, condition in samples.items():
    model = smf.ols('growth ~ lnY60', data=data[condition]).fit()
    results_table3[sample] = model.summary()
    implied_lambda = -np.log(1 + model.params['lnY60']) / (85 - 60)

# Table IV
results_table4 = {}
for sample, condition in samples.items():
    model = smf.ols('growth ~ lnY60 + lns + lnngd', data=data[condition]).fit()
    results_table4[sample] = model.summary()
    implied_lambda = -np.log(1 + model.params['lnY60']) / (85 - 60)

# Table V
results_table5 = {}
for sample, condition in samples.items():
    model = smf.ols('growth ~ lnY60 + lns + lnngd + lnschool', data=data[condition]).fit()
    results_table5[sample] = model.summary()
    implied_lambda = -np.log(1 + model.params['lnY60']) / (85 - 60)

# Figure I
plt.figure(figsize=(15, 5))

# Unconditional scatter plot
```

```
plt.subplot(1, 3, 1)
plt.scatter(data.loc[data['i'] == 1, 'lnY60'], data.loc[data['i'] == 1, 'growth_an
plt.xlabel("Log output per working age adult: 1960")
plt.ylabel("Growth rate: 1960-85")
plt.title("Unconditional")

# Partial out lns and lnngd
residual_model1 = smf.ols('lnY60 ~ lns + lnngd', data=data[data['i'] == 1]).fit()
data['lnY60_residual1'] = residual_model1.resid + residual_model1.params['Intercep
residual_model2 = smf.ols('growth_annu ~ lns + lnngd', data=data[data['i'] == 1]).
data['growth_annu_residual1'] = residual_model2.resid
plt.subplot(1, 3, 2)
plt.scatter(data['lnY60_residual1'], data['growth_annu_residual1'])
plt.xlabel("Log output per working age adult: 1960")
plt.ylabel("Growth rate: 1960-85")
plt.title("Conditional on lns and lnngd")

# Partial out lns, lnngd, and lnschool
residual_model3 = smf.ols('lnY60 ~ lns + lnngd + lnschool', data=data[data['i'] ==
data['lnY60_residual2'] = residual_model3.resid + residual_model3.params['Intercep
residual_model4 = smf.ols('growth_annu ~ lns + lnngd + lnschool', data=data[data['
data['growth_annu_residual2'] = residual_model4.resid
plt.subplot(1, 3, 3)
plt.scatter(data['lnY60_residual2'], data['growth_annu_residual2'])
plt.xlabel("Log output per working age adult: 1960")
plt.ylabel("Growth rate: 1960-85")
plt.title("Conditional on lns, lnngd, and lnschool")

plt.tight_layout()
plt.show()
```



0.3 IV 估计

0.3.1 应用：开放与增长

这一部分中，用 Frankel and Romer [1] 的经典论文，阐释 OLS 和 IV 估计的方法，包括：

- 如何从理论观点中确定一个统计模型；
- 如何理解回归分析的逻辑；
- 如何为统计模型中的变量准备数据；
- 如何估计统计模型；
- 如何解释模型估计的结果；
- 如何根据模型估计值进行统计推断；
- 如何解释模型整体拟合度。

0.3.2 应用：制度与增长

Bibliography

[1] Jeffrey A. Frankel and David H. Romer. “Does Trade Cause Growth?” In: *American Economic Review* 89.3 (June 1999), pp. 379–399. DOI: 10.1257/aer.89.3.379. URL: https://www.aeaweb.org/articles?id=10.1257/aer.89.3.379.