In [1]:
```
#Overall IPC Crime Category
```

In [2]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from linearmodels.panel import PanelOLS
from linearmodels.panel import compare
```

In [3]:
```
df = pd.read_csv('panel_data/new data.csv')
df['l_ipc'] = np.log(df['ipc_cr'])
df.head()
```

Out[3]:

	s.no.	districts	year	ipc_crimes	pop_in_lak	ipc_cr	avg_temp	tot_rf	l_ipc
0	1	ariyalur	2011	1953	7.52	259.54	28.312353	1103.207404	5.558911
1	1	ariyalur	2012	2480	7.63	324.83	28.777312	973.207972	5.783302
2	1	ariyalur	2013	1991	7.76	256.53	28.730311	870.158045	5.547246
3	1	ariyalur	2014	1615	7.88	204.88	28.536042	1090.802339	5.322424
4	1	ariyalur	2015	1386	8.00	173.00	28.565911	1501.644532	5.153292

In [4]:
```
df = df.set_index(['districts','year'])
y = df['l_ipc']
X = df[['avg_temp','tot_rf']]
```

In [5]:
```
#PooledOLS Estimation
X = sm.add_constant(X)
pols = PanelOLS(y,X)
pols_result = pols.fit()
print(pols_result.summary)
```

PanelOLS Estimation Summary

Dep. Variable:	l_ipc	R-squared:	0.0457
Estimator:	PanelOLS	R-squared (Between):	0.0408
No. Observations:	384	R-squared (Within):	0.0466
Date:	Tue, Nov 11 2025	R-squared (Overall):	0.0457
Time:	15:29:21	Log-likelihood	-265.57
Cov. Estimator:	Unadjusted		
		F-statistic:	9.1189
Entities:	32	P-value	0.0001
Avg Obs:	12.000	Distribution:	F(2,381)
Min Obs:	12.000		
Max Obs:	12.000	F-statistic (robust):	9.1189
		P-value	0.0001
Time periods:	12	Distribution:	F(2,381)
Avg Obs:	32.000		
Min Obs:	32.000		
Max Obs:	32.000		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	4.8315	0.2463	19.618	0.0000	4.3473	5.3158
avg_temp	0.0187	0.0078	2.3838	0.0176	0.0033	0.0341
tot_rf	0.0002	5.757e-05	4.2059	0.0000	0.0001	0.0004

In [6]:
```
#FE Model Estimation
X = sm.add_constant(X)
FEmodel = PanelOLS(y,X,entity_effects=True)
feresult = FEmodel.fit()
print(feresult.summary)
```

```
                          PanelOLS Estimation Summary
================================================================================
Dep. Variable:                  l_ipc   R-squared:                        0.0474
Estimator:                   PanelOLS   R-squared (Between):              0.0275
No. Observations:                 384   R-squared (Within):               0.0474
Date:                Tue, Nov 11 2025   R-squared (Overall):              0.0444
Time:                        15:29:21   Log-likelihood                   -233.43
Cov. Estimator:            Unadjusted
                                        F-statistic:                      8.7101
Entities:                          32   P-value                           0.0002
Avg Obs:                       12.000   Distribution:                  F(2,350)
Min Obs:                       12.000
Max Obs:                       12.000   F-statistic (robust):             8.7101
                                        P-value                           0.0002
Time periods:                      12   Distribution:                  F(2,350)
Avg Obs:                       32.000
Min Obs:                       32.000
Max Obs:                       32.000

                             Parameter Estimates
================================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
--------------------------------------------------------------------------------
const          4.6751     1.1259     4.1523     0.0000      2.4607      6.8895
avg_temp       0.0229     0.0403     0.5669     0.5712     -0.0564      0.1022
tot_rf         0.0003  7.078e-05     3.9602     0.0001      0.0001      0.0004
================================================================================

F-test for Poolability: 2.0570
P-value: 0.0010
Distribution: F(31,350)

Included effects: Entity
```

In [9]:

```
#RE Model Estimation
from linearmodels.panel import RandomEffects
import statsmodels.api as sm
X = sm.add_constant(X)
REmodel = RandomEffects(y,X)
reresult = REmodel.fit()
print(reresult.summary)
```

```
                          RandomEffects Estimation Summary
================================================================================
Dep. Variable:                  l_ipc   R-squared:                        0.0463
Estimator:              RandomEffects   R-squared (Between):              0.0362
No. Observations:                 384   R-squared (Within):               0.0471
Date:                Tue, Nov 11 2025   R-squared (Overall):              0.0455
Time:                        15:29:22   Log-likelihood                   -249.01
Cov. Estimator:            Unadjusted
                                        F-statistic:                      9.2495
Entities:                          32   P-value                           0.0001
Avg Obs:                       12.000   Distribution:                  F(2,381)
Min Obs:                       12.000
Max Obs:                       12.000   F-statistic (robust):             9.2495
                                        P-value                           0.0001
Time periods:                      12   Distribution:                  F(2,381)
Avg Obs:                       32.000
Min Obs:                       32.000
Max Obs:                       32.000

                             Parameter Estimates
================================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
--------------------------------------------------------------------------------
const          4.7907     0.3184     15.046     0.0000      4.1646      5.4168
avg_temp       0.0195     0.0106     1.8415     0.0663     -0.0013      0.0403
tot_rf         0.0003   6.03e-05     4.2891     0.0000      0.0001      0.0004
================================================================================
```

In [10]:

```python
#Hausman Test
from numpy.linalg import inv
from scipy.stats import chi2

b_FE = feresult.params
b_RE = reresult.params

common_coef = list(set(b_FE.index) & set(b_RE.index))

if 'const' in common_coef:
    common_coef.remove('const')

b_FE = b_FE[common_coef]
b_RE = b_RE[common_coef]

V_FE = feresult.cov
V_RE = reresult.cov

diff = b_FE - b_RE
diff_var = V_FE.loc[common_coef, common_coef] - V_RE.loc[common_coef, common_coef]

hausman_stat = np.dot(np.dot(diff.T, inv(diff_var)), diff)

df_h = len(diff)
p_value = 1 - chi2.cdf(hausman_stat, df_h)

print("Hausman Test Statistic:", round(hausman_stat, 3))
print("Degrees of Freedom:", df_h)
```

```
print("p-value:", round(p_value, 4))
```

```
Hausman Test Statistic: 0.567
Degrees of Freedom: 2
p-value: 0.753
```

In [11]:
```
#Diagnostic Checks
from statsmodels.stats.diagnostic import het_breuschpagan, het_white
from statsmodels.stats.stattools import durbin_watson
```

In [12]:
```
#Test for Heteroskedasticity

#H0: No heteroskedasticity
#H1: Heteroskedasticity exists

#p-value <= 0.05 ---> Heteroskedasticity; p-value > 0.05 ---> Homoskedasticity

print('Breusch-Pagan Test')
residuals = reresult.resids
bp_test = het_breuschpagan(residuals, X)
bp_labels = ['Lagrange multiplier statistic','p-value','f-value','f p-value']
print(dict(zip(bp_labels, bp_test)))
print()
print('White Test')
white_test = het_white(residuals,X)
white_labels = ['LM stat','LM p-value','F p-value']
print(dict(zip(white_labels,white_test)))
```

```
Breusch-Pagan Test
{'Lagrange multiplier statistic': np.float64(9.98334328950044), 'p-value': np.float6
4(0.006794297342475483), 'f-value': np.float64(5.084872190924652), 'f p-value': np.f
loat64(0.006616473096637813)}

White Test
{'LM stat': np.float64(13.545411746797555), 'LM p-value': np.float64(0.0187700415342
66983), 'F p-value': np.float64(2.7642608852180715)}
```

In [13]:
```
#Test for serial correlation (autocorrelation)

#Durbin-Watson statistic ranges between 0 to 4

#DW statistic = 2 ---> No autocorrelation
#DW statistic < 2 ---> Positive autocorrelation
#DW statistic > 2 ---> Negative autocorrelation

print('Durbin-Watson Test')
dw_value = durbin_watson(residuals)
print("Durbin-Watson statistic: ", round(dw_value,3))
```

```
Durbin-Watson Test
Durbin-Watson statistic:  1.701
```

In [14]:
```
from scipy import stats

#Test for cross-section dependency

#H0: No cross-section dependency
```

```
#H1: Cross-section dependency exists

print('Breusch-Pagan LM Test')
resid_df = residuals.unstack(level=0)
T = resid_df.shape[0]
N = resid_df.shape[1]

rho = resid_df.corr().values
upper_tri_idx = np.triu_indices(N, k=1)
rho_upper = rho[upper_tri_idx]
LM_stat = T * np.sum(rho_upper**2)
p_value = 1 - stats.chi2.cdf(LM_stat, N*(N-1)/2)

print(f"Breusch-Pagan LM statistic: {LM_stat:.3f}")
print(f"p-value: {p_value:.4f}")
print()

print('Pearson CD Test')
CD_stat = np.sqrt(2 / (N*(N-1))) * np.sum(rho_upper)
p_value_cd = 2 * (1 - stats.norm.cdf(abs(CD_stat)))

print(f"Pesaran CD statistic: {CD_stat:.3f}")
print(f"p-value: {p_value_cd:.4f}")
```

```
Breusch-Pagan LM Test
Breusch-Pagan LM statistic: 3117.798
p-value: 0.0000

Pearson CD Test
Pesaran CD statistic: 14.825
p-value: 0.0000
```

In [15]:
```
#Re-estimate RE Model
```

In [16]:
```
#RE with cov.type 'clustered'
re_model_robust1 = REmodel.fit(cov_type='clustered', cluster_entity=True)
print(re_model_robust1.summary)
```

```
                          RandomEffects Estimation Summary
================================================================================
Dep. Variable:                  l_ipc   R-squared:                        0.0463
Estimator:                RandomEffects   R-squared (Between):              0.0362
No. Observations:                 384   R-squared (Within):               0.0471
Date:                Tue, Nov 11 2025   R-squared (Overall):              0.0455
Time:                        15:29:22   Log-likelihood                   -249.01
Cov. Estimator:            Clustered
                                        F-statistic:                      9.2495
Entities:                          32   P-value                           0.0001
Avg Obs:                       12.000   Distribution:                  F(2,381)
Min Obs:                       12.000
Max Obs:                       12.000   F-statistic (robust):             15.015
                                        P-value                           0.0000
Time periods:                      12   Distribution:                  F(2,381)
Avg Obs:                       32.000
Min Obs:                       32.000
Max Obs:                       32.000

                             Parameter Estimates
================================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
--------------------------------------------------------------------------------
const          4.7907     0.4626     10.357     0.0000      3.8812      5.7002
avg_temp       0.0195     0.0160     1.2148     0.2252     -0.0121      0.0510
tot_rf         0.0003  4.723e-05     5.4757     0.0000      0.0002      0.0004
================================================================================
```

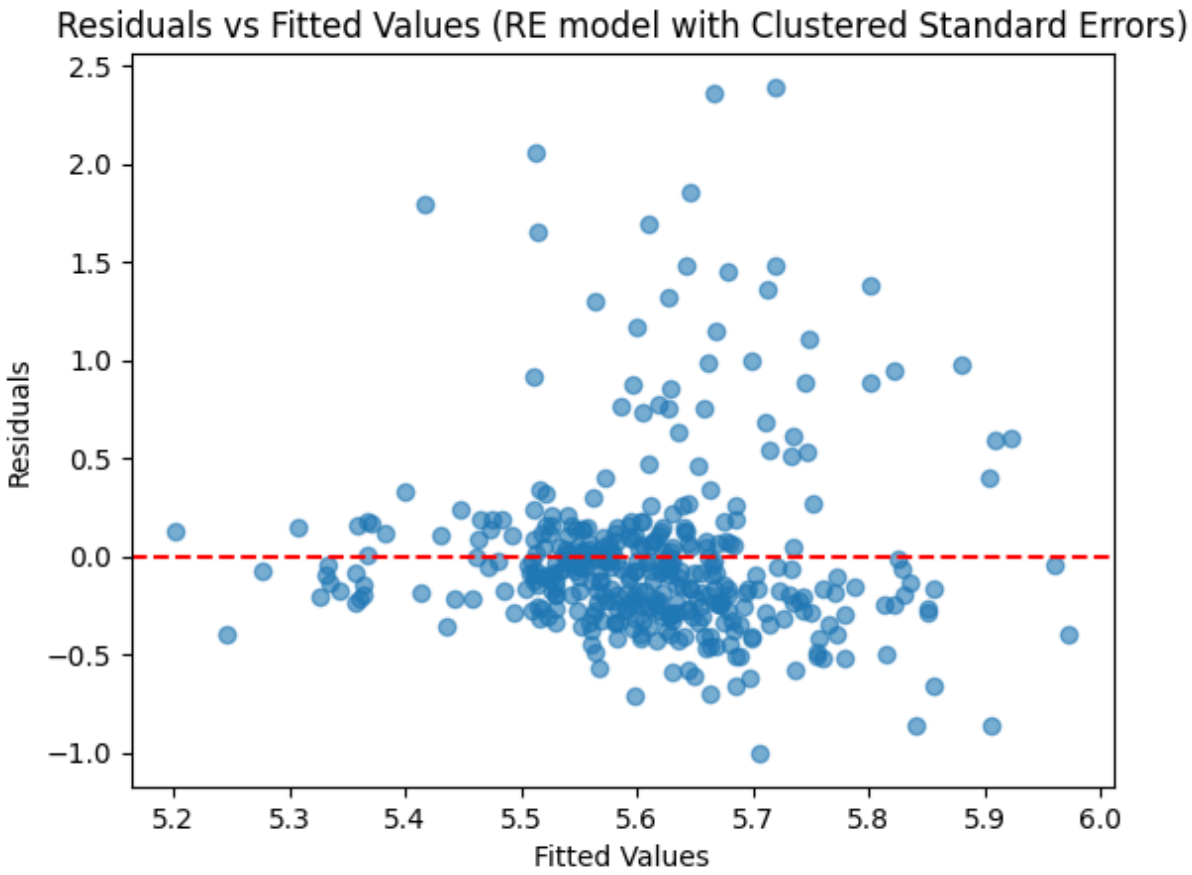
In [17]:
```
#RE with cov.type 'kernel' (Driscoll-Kraay Method)
re_model_robust2 = REmodel.fit(cov_type='kernel')
print(re_model_robust2.summary)
```

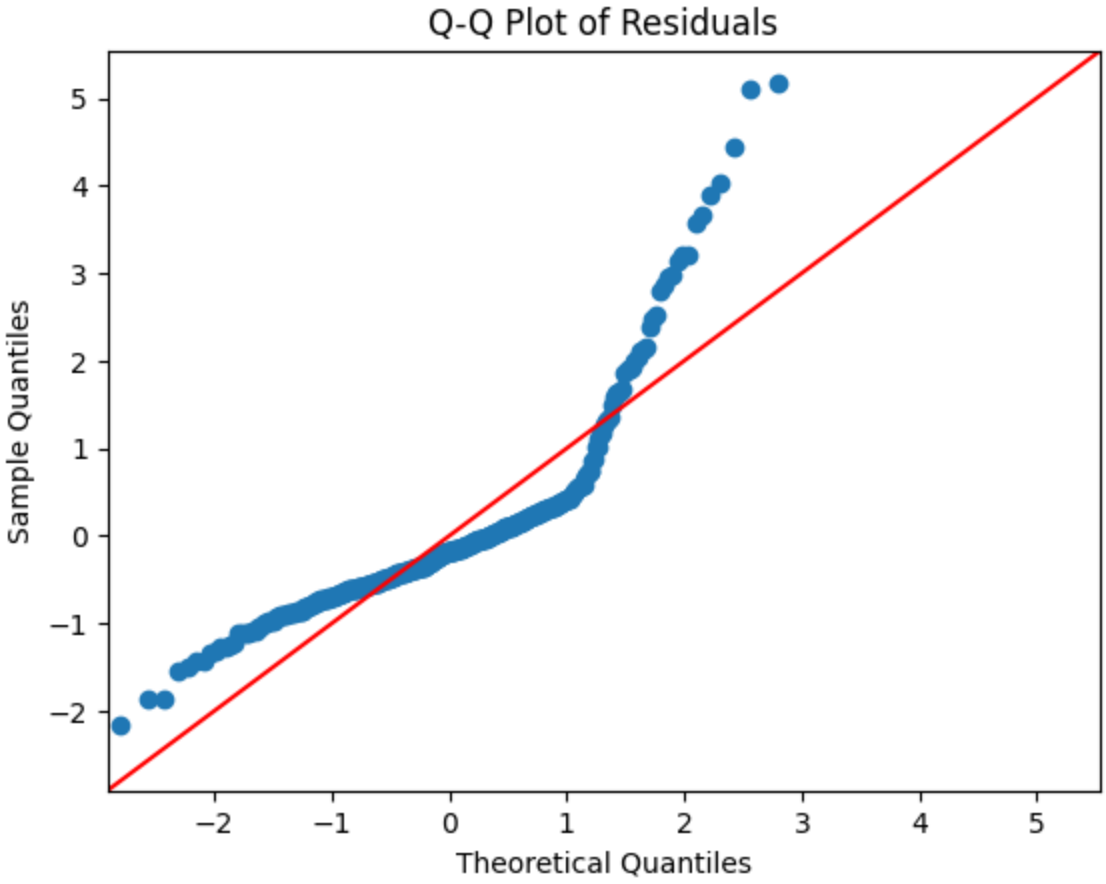
```
                          RandomEffects Estimation Summary
================================================================================
Dep. Variable:                  l_ipc   R-squared:                        0.0463
Estimator:              RandomEffects   R-squared (Between):              0.0362
No. Observations:                 384   R-squared (Within):               0.0471
Date:                Tue, Nov 11 2025   R-squared (Overall):              0.0455
Time:                        15:29:22   Log-likelihood                   -249.01
Cov. Estimator:        Driscoll-Kraay
                                        F-statistic:                      9.2495
Entities:                         32   P-value                           0.0001
Avg Obs:                      12.000   Distribution:                   F(2,381)
Min Obs:                      12.000
Max Obs:                      12.000   F-statistic (robust):             4.6172
                                        P-value                           0.0104
Time periods:                     12   Distribution:                   F(2,381)
Avg Obs:                      32.000
Min Obs:                      32.000
Max Obs:                      32.000

                             Parameter Estimates
================================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
--------------------------------------------------------------------------------
const          4.7907     0.2095     22.865     0.0000      4.3787      5.2027
avg_temp       0.0195     0.0064     3.0323     0.0026      0.0068      0.0321
tot_rf         0.0003     0.0001     2.1170     0.0349   1.842e-05      0.0005
================================================================================
```

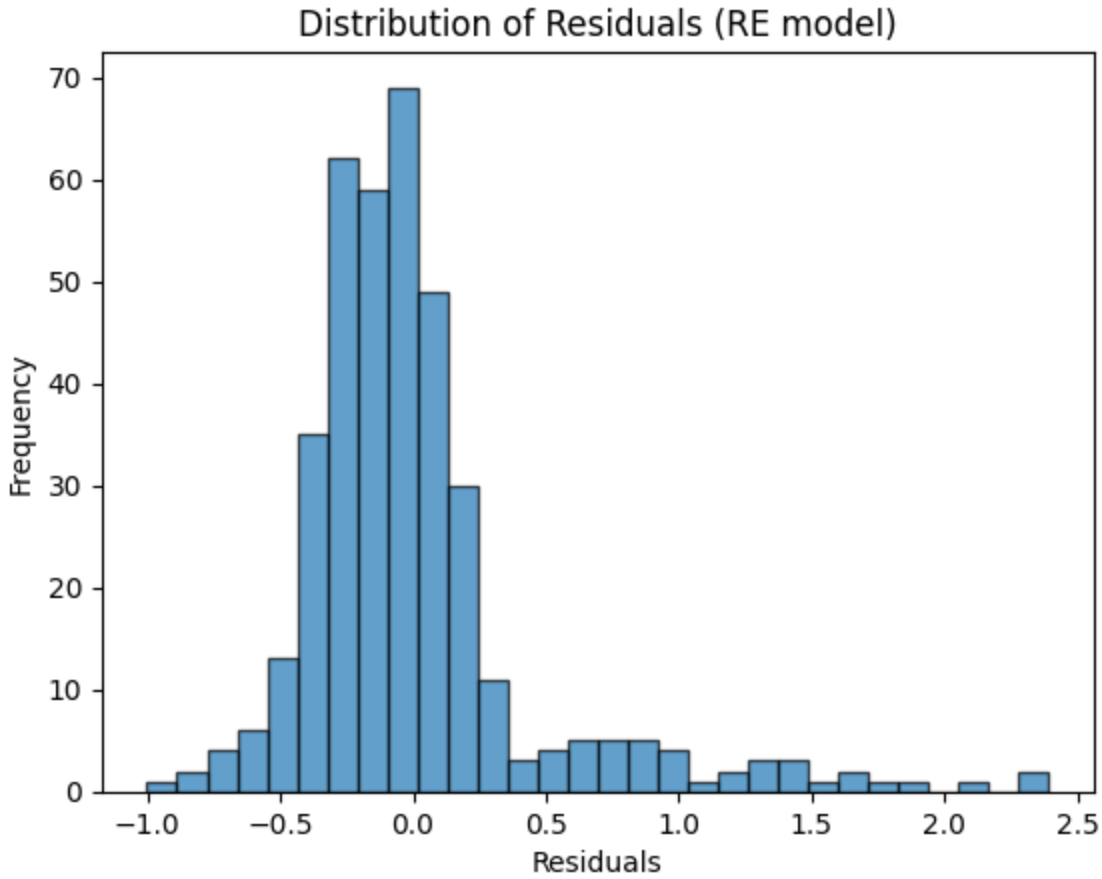
In [18]:

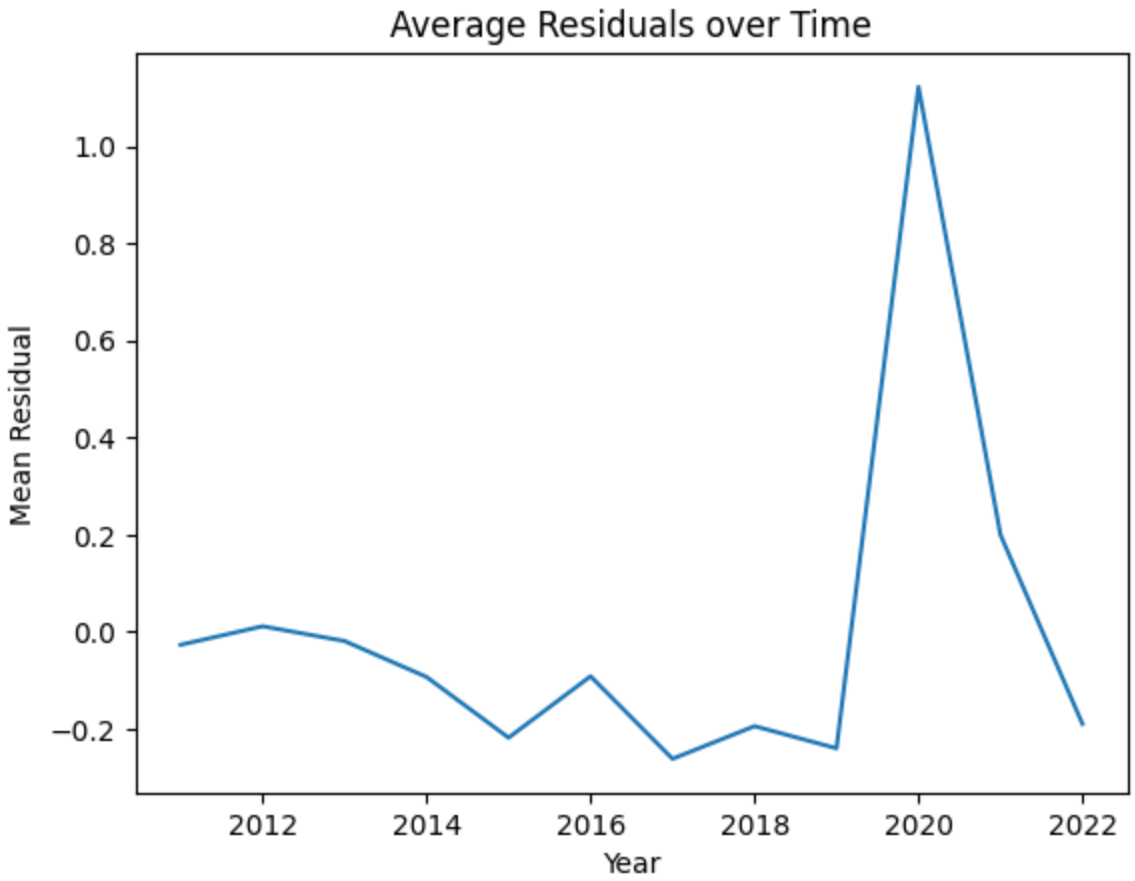
```python
# Check residuals and fitted values
df['residuals1'] = re_model_robust1.resids
df['fitted1'] = re_model_robust1.fitted_values

import matplotlib.pyplot as plt

plt.scatter(df['fitted1'], df['residuals1'], alpha=0.6)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted Values (RE model with Clustered Standard Errors)')
plt.show()

sm.qqplot(df['residuals1'], line='45', fit=True)
plt.title('Q-Q Plot of Residuals')
plt.show()

plt.hist(df['residuals1'], bins=30, edgecolor='black', alpha=0.7)
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.title('Distribution of Residuals (RE model)')
plt.show()

resid_df = df['residuals1'].unstack(level=0)
plt.plot(resid_df.mean(axis=1))
plt.title('Average Residuals over Time')
plt.xlabel('Year')
plt.ylabel('Mean Residual')
```

```
plt.show()

from scipy.stats import shapiro

#Test for normality

stat, p = shapiro(df['residuals1'])
print(f"Shapiro-Wilk Test: Statistic={stat:.3f}, p-value={p:.4f}")
```




Q-Q Plot of Residuals
Sample Quantiles
Theoretical Quantiles
5
4
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4
5

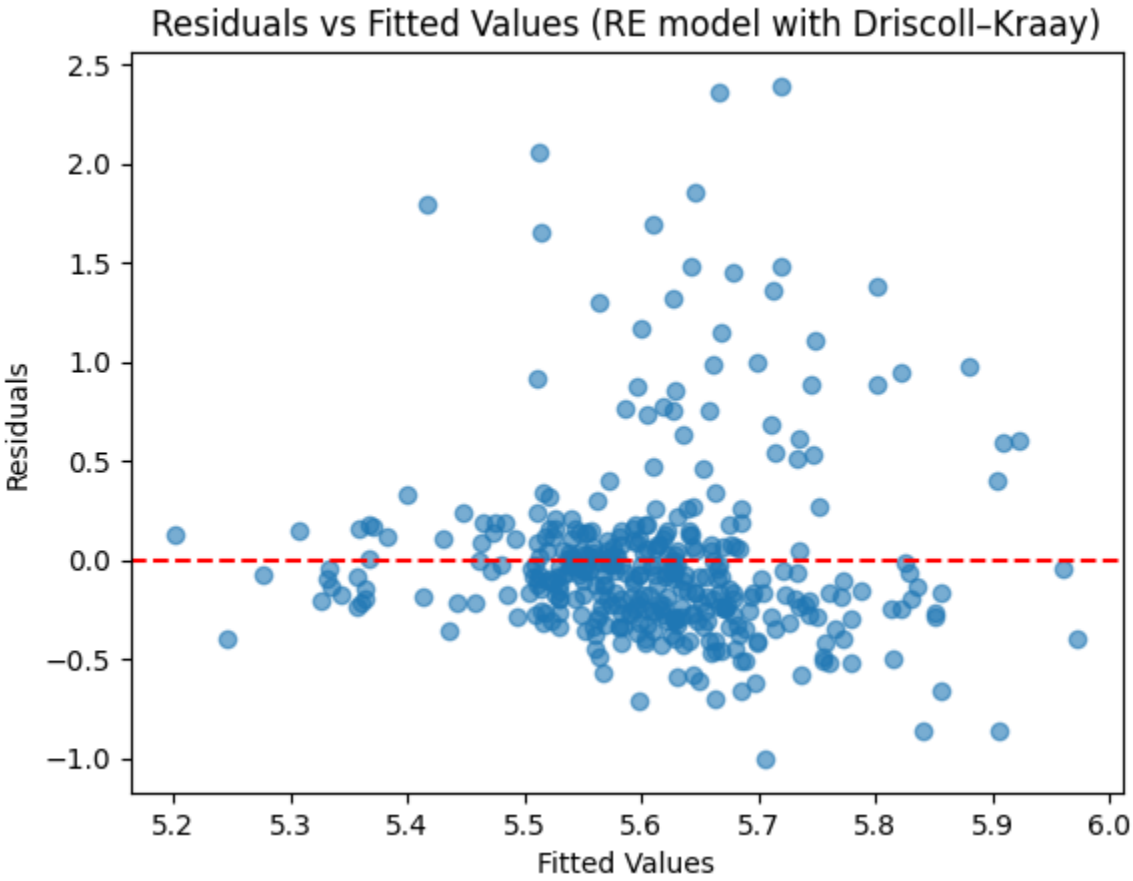

Distribution of Residuals (RE model)
Frequency
Residuals
70
60
50
40
30
20
10
0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5

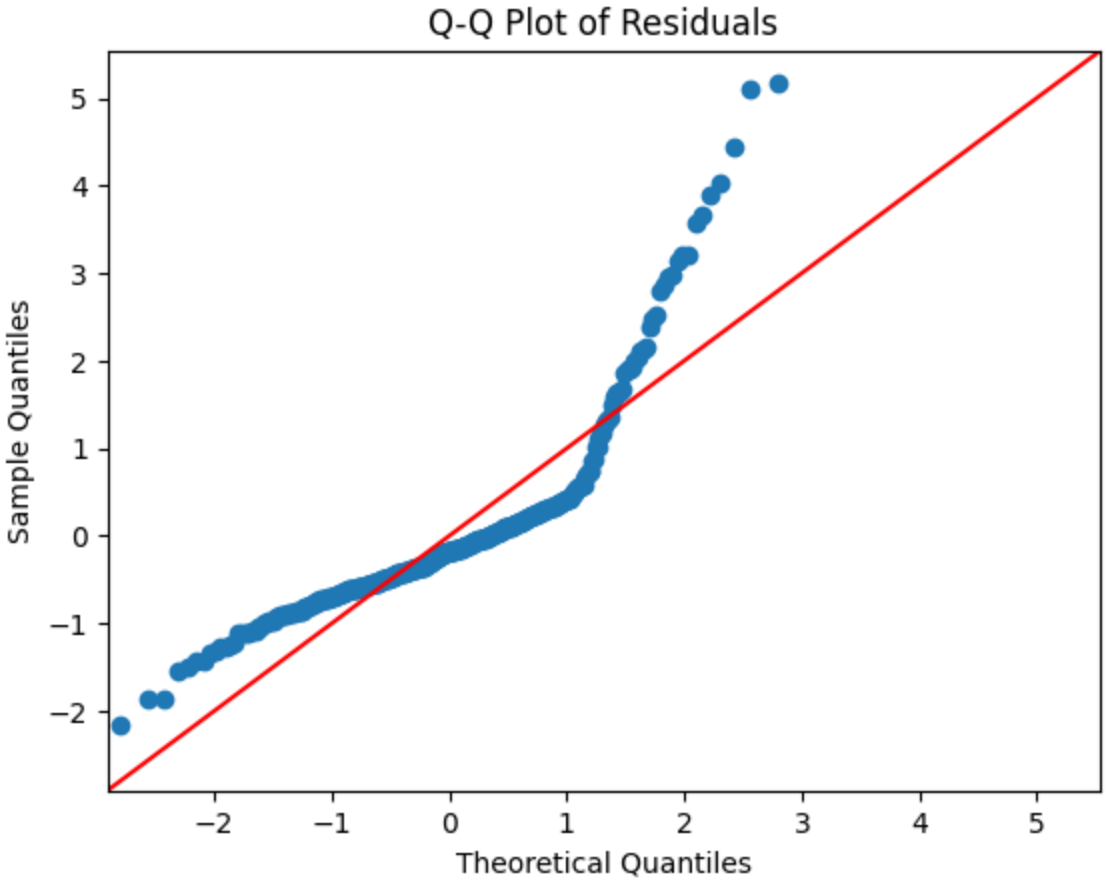


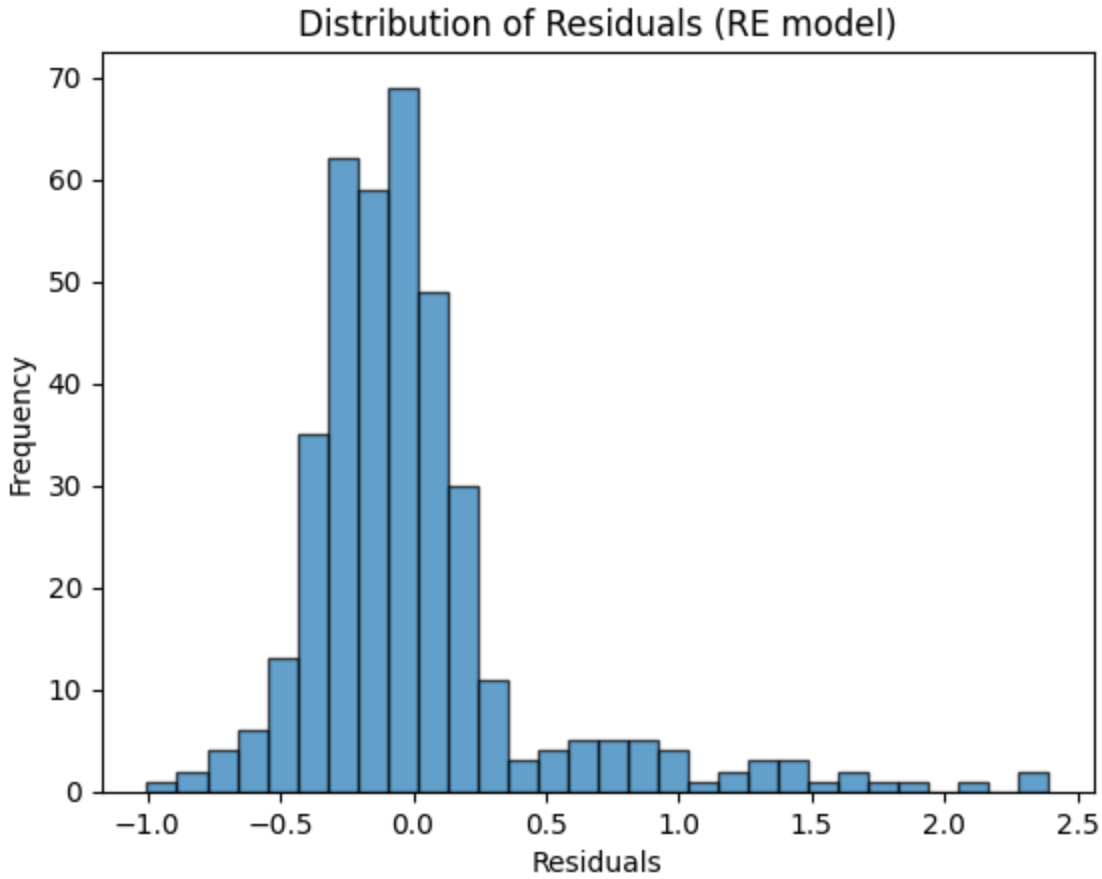

```
Shapiro-Wilk Test: Statistic=0.797, p-value=0.0000
```

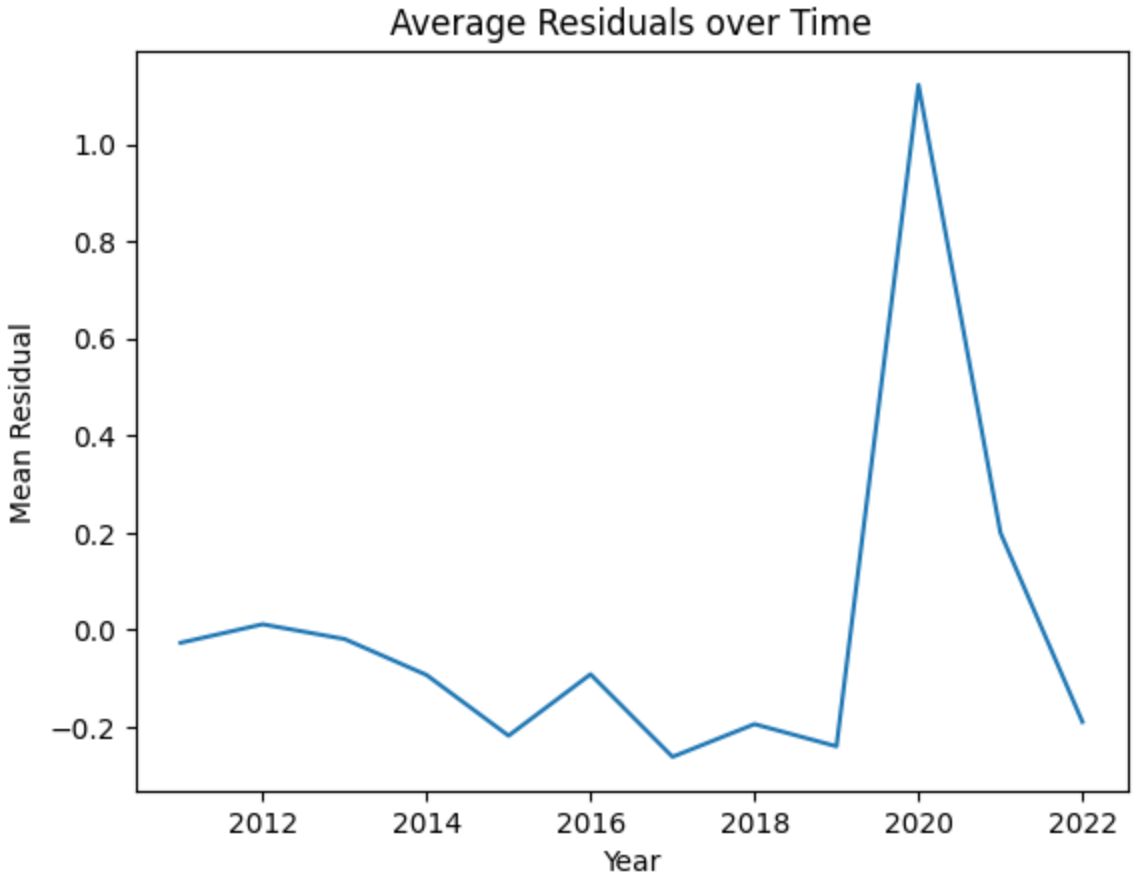
In [19]:
```
# Check residuals and fitted values
df['residuals2'] = re_model_robust2.resids
df['fitted2'] = re_model_robust2.fitted_values

import matplotlib.pyplot as plt

plt.scatter(df['fitted2'], df['residuals2'], alpha=0.6)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted Values (RE model with Driscoll-Kraay)')
plt.show()

sm.qqplot(df['residuals2'], line='45', fit=True)
plt.title('Q-Q Plot of Residuals')
plt.show()

plt.hist(df['residuals2'], bins=30, edgecolor='black', alpha=0.7)
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.title('Distribution of Residuals (RE model)')
plt.show()

resid_df = df['residuals2'].unstack(level=0)
plt.plot(resid_df.mean(axis=1))
plt.title('Average Residuals over Time')
plt.xlabel('Year')
```

```
plt.ylabel('Mean Residual')
plt.show()

from scipy.stats import shapiro

#Test for normality

stat, p = shapiro(df['residuals2'])
print(f"Shapiro-Wilk Test: Statistic={stat:.3f}, p-value={p:.4f}")
```




Q-Q Plot of Residuals
Sample Quantiles
Theoretical Quantiles
5
4
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4
5


Distribution of Residuals (RE model)
Frequency
Residuals
70
60
50
40
30
20
10
0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5




```
Shapiro-Wilk Test: Statistic=0.797, p-value=0.0000
```

In [ ]: