In [1]:
```
#Property 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/property new.csv')
#df['l_ipc'] = np.log(df['ipc_cr'])
df.head()
```

Out[3]:

	s.no.	districts	year	type	property_crimes	pop_in_lak	property_cr	avg_temp	
0	1	ariyalur	2011	property crimes	142	7.52	18.9	28.312353	110
1	1	ariyalur	2012	property crimes	182	7.63	23.9	28.777312	97
2	1	ariyalur	2013	property crime	157	7.76	20.2	28.730311	87
3	1	ariyalur	2014	property crime	96	7.88	12.2	28.536042	109
4	1	ariyalur	2015	property crime	88	8.00	11.0	28.565911	150

In [4]:
```
df = df.set_index(['districts','year'])
y = df['property_cr']
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:            property_cr   R-squared:                        0.0346
Estimator:                   PanelOLS   R-squared (Between):              0.0805
No. Observations:                 384   R-squared (Within):              -0.0165
Date:                Wed, Nov 12 2025   R-squared (Overall):              0.0346
Time:                        18:23:03   Log-likelihood                   -1481.6
Cov. Estimator:            Unadjusted
                                        F-statistic:                      6.8335
Entities:                          32   P-value                           0.0012
Avg Obs:                       12.000   Distribution:                   F(2,381)
Min Obs:                       12.000
Max Obs:                       12.000   F-statistic (robust):             6.8335
                                        P-value                           0.0012
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          1.9113     5.8446     0.3270     0.7438     -9.5803      13.403
avg_temp       0.6633     0.1858     3.5706     0.0004      0.2980      1.0285
tot_rf         0.0032     0.0014     2.3250     0.0206      0.0005      0.0059
==============================================================================
```

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:            property_cr   R-squared:                        0.0001
Estimator:                   PanelOLS   R-squared (Between):             -0.0212
No. Observations:                 384   R-squared (Within):               0.0001
Date:                Wed, Nov 12 2025   R-squared (Overall):             -0.0111
Time:                        18:23:03   Log-likelihood                   -1344.7
Cov. Estimator:            Unadjusted
                                        F-statistic:                      0.0261
Entities:                          32   P-value                           0.9742
Avg Obs:                       12.000   Distribution:                   F(2,350)
Min Obs:                       12.000
Max Obs:                       12.000   F-statistic (robust):             0.0261
                                        P-value                           0.9742
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          26.487     20.337     1.3024     0.1936     -13.512      66.486
avg_temp      -0.1181     0.7283    -0.1622     0.8713     -1.5505      1.3143
tot_rf      9.313e-05     0.0013     0.0728     0.9420     -0.0024      0.0026
==============================================================================

F-test for Poolability: 11.748
P-value: 0.0000
Distribution: F(31,350)

Included effects: Entity
```

In [7]:

```
#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:            property_cr   R-squared:                        0.0029
Estimator:              RandomEffects   R-squared (Between):              0.0483
No. Observations:                 384   R-squared (Within):              -0.0014
Date:                Wed, Nov 12 2025   R-squared (Overall):              0.0248
Time:                        18:23:03   Log-likelihood                   -1361.3
Cov. Estimator:            Unadjusted
                                        F-statistic:                      0.5481
Entities:                          32   P-value                           0.5785
Avg Obs:                       12.000   Distribution:                   F(2,381)
Min Obs:                       12.000
Max Obs:                       12.000   F-statistic (robust):             0.5481
                                        P-value                           0.5785
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          12.060     11.065     1.0900     0.2764     -9.6954      33.816
avg_temp       0.3901     0.3865     1.0093     0.3135     -0.3699      1.1501
tot_rf         0.0007     0.0012     0.6068     0.5443     -0.0016      0.0031
==============================================================================
```

In [8]:

```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: 2.662
Degrees of Freedom: 2
p-value: 0.2642
```

In [9]:
```
#Diagnostic Checks
from statsmodels.stats.diagnostic import het_breuschpagan, het_white
from statsmodels.stats.stattools import durbin_watson
```

In [10]:
```
#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(4.938224307667397), 'p-value': np.float
64(0.08465999069211277), 'f-value': np.float64(2.4817372548114314), 'f p-value': n
p.float64(0.08494843213523738)}

White Test
{'LM stat': np.float64(16.287402217667818), 'LM p-value': np.float64(0.0060696790030
693335), 'F p-value': np.float64(3.3486141488809498)}
```

In [11]:
```
#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:  1.208
```

In [12]:
```
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('Pesaran 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: 1148.184
p-value: 0.0000

Pearson CD Test
Pesaran CD statistic: 5.713
p-value: 0.0000
```

In [13]:

```
#Re-estimate RE Model
```

In [14]:

```
#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:            property_cr   R-squared:                        0.0029
Estimator:              RandomEffects   R-squared (Between):              0.0483
No. Observations:                 384   R-squared (Within):              -0.0014
Date:                Wed, Nov 12 2025   R-squared (Overall):              0.0248
Time:                        18:23:04   Log-likelihood                   -1361.3
Cov. Estimator:             Clustered
                                        F-statistic:                      0.5481
Entities:                          32   P-value                           0.5785
Avg Obs:                       12.000   Distribution:                   F(2,381)
Min Obs:                       12.000
Max Obs:                       12.000   F-statistic (robust):             0.7805
                                        P-value                           0.4589
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          12.060     9.0051     1.3393     0.1813     -5.6458      29.766
avg_temp       0.3901     0.3130     1.2464     0.2134     -0.2253      1.0055
tot_rf         0.0007     0.0016     0.4508     0.6524     -0.0024      0.0039
==============================================================================
```

In [15]:

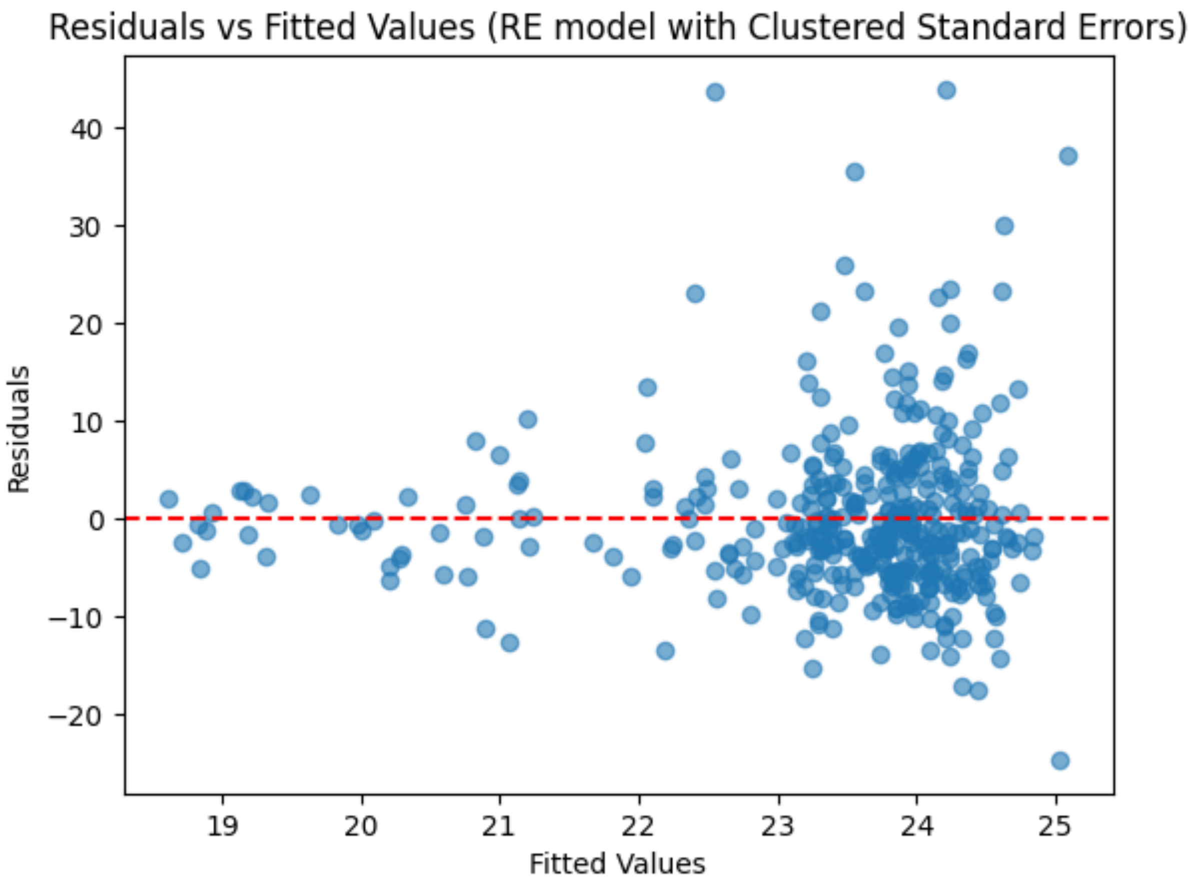
```
#RE with cov.type 'kernel' (Driscoll-Kraay Method)
re_model_robust2 = REmodel.fit(cov_type='kernel')
print(re_model_robust2.summary)
```

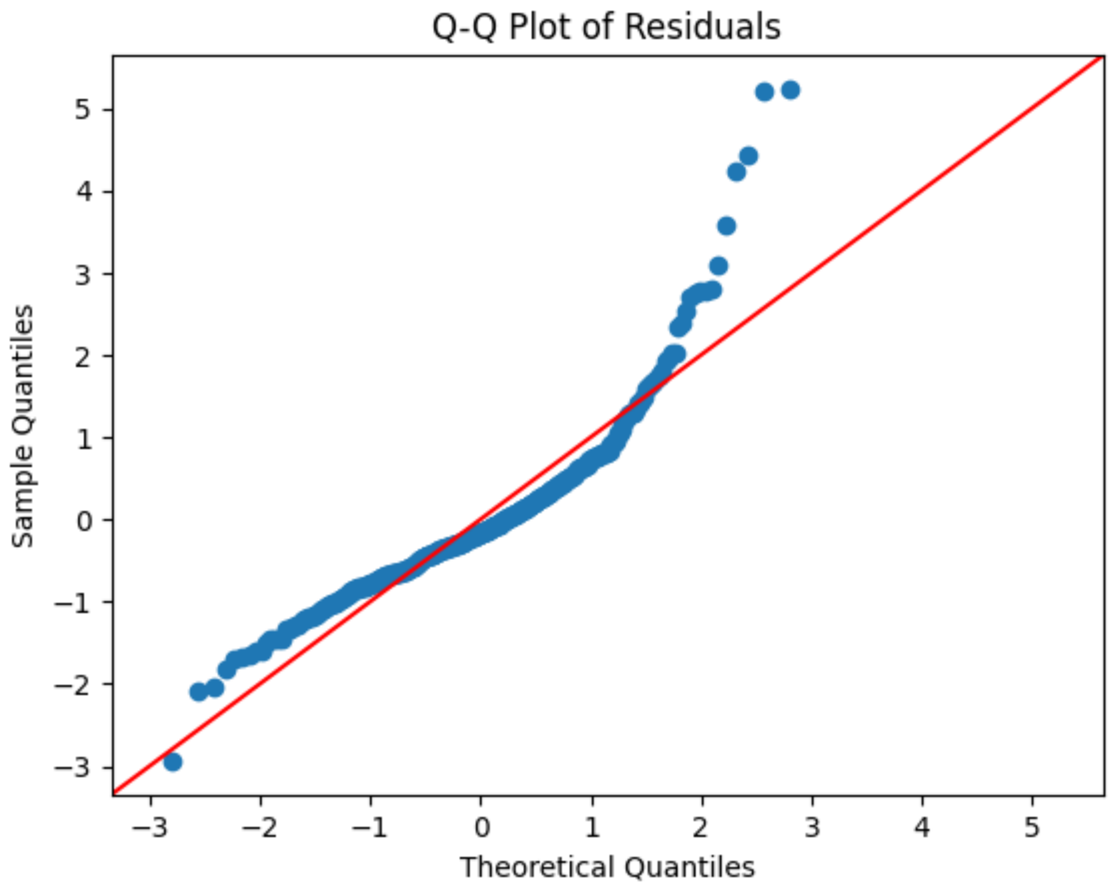
```
                          RandomEffects Estimation Summary
================================================================================
Dep. Variable:            property_cr   R-squared:                        0.0029
Estimator:              RandomEffects   R-squared (Between):              0.0483
No. Observations:                 384   R-squared (Within):              -0.0014
Date:                Wed, Nov 12 2025   R-squared (Overall):              0.0248
Time:                        18:23:04   Log-likelihood                   -1361.3
Cov. Estimator:         Driscoll-Kraay
                                        F-statistic:                      0.5481
Entities:                          32   P-value                           0.5785
Avg Obs:                       12.000   Distribution:                    F(2,381)
Min Obs:                       12.000
Max Obs:                       12.000   F-statistic (robust):             0.6792
                                        P-value                           0.5076
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          12.060     11.453     1.0531     0.2930     -10.458      34.578
avg_temp       0.3901     0.3412     1.1432     0.2537     -0.2808      1.0610
tot_rf         0.0007     0.0019     0.3838     0.7013     -0.0030      0.0044
==============================================================================
```

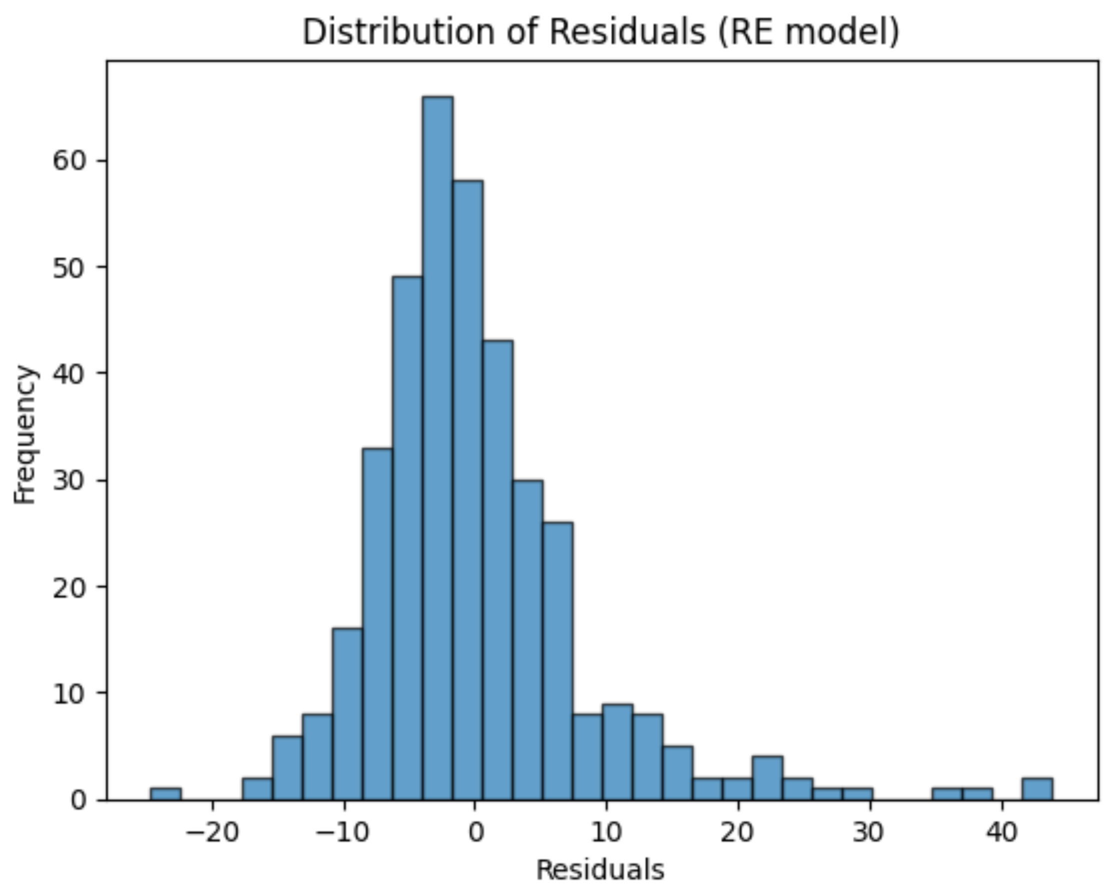
In [16]:

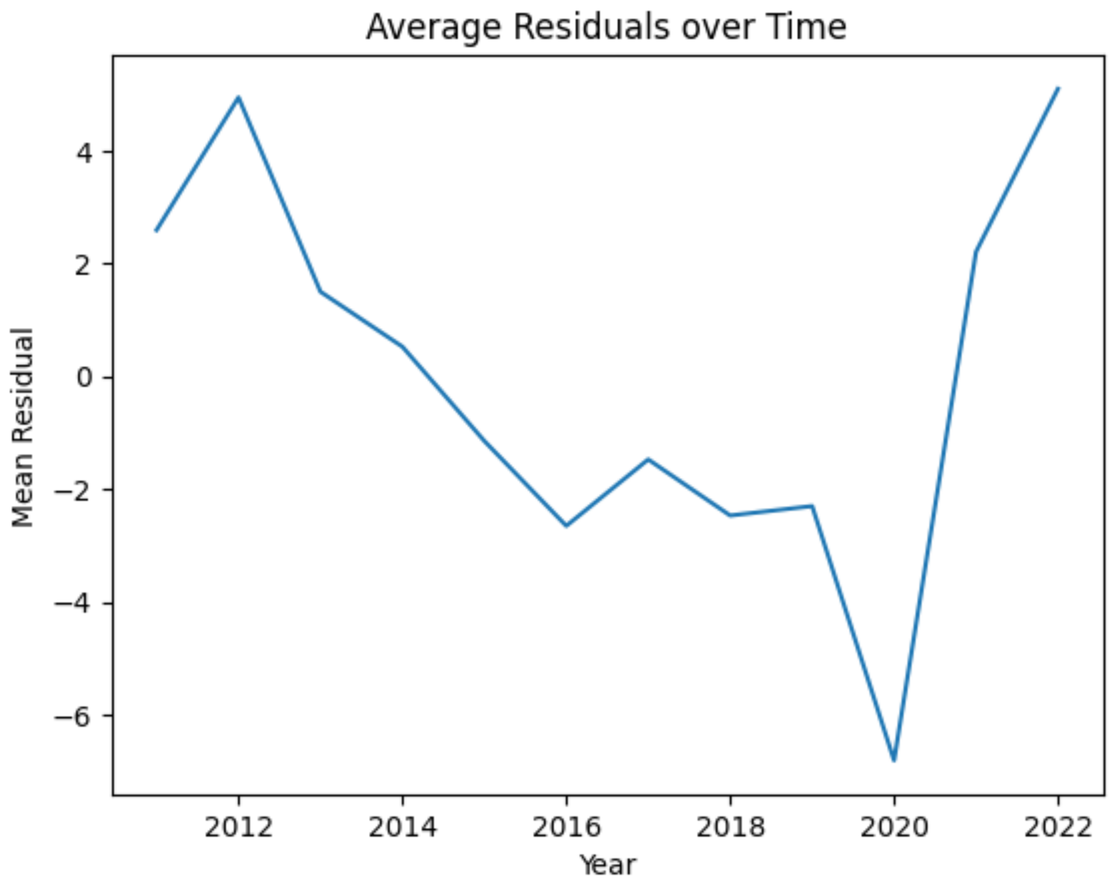
```
# Check residuals and fitted values
df['residuals1'] = re_model_robust1.resids
df['fitted1'] = re_model_robust1.fitted_values

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
−3
−3
−2
−1
0
1
2
3
4
5


Distribution of Residuals (RE model)
Frequency
Residuals
60
50
40
30
20
10
0
−20
−10
0
10
20
30
40

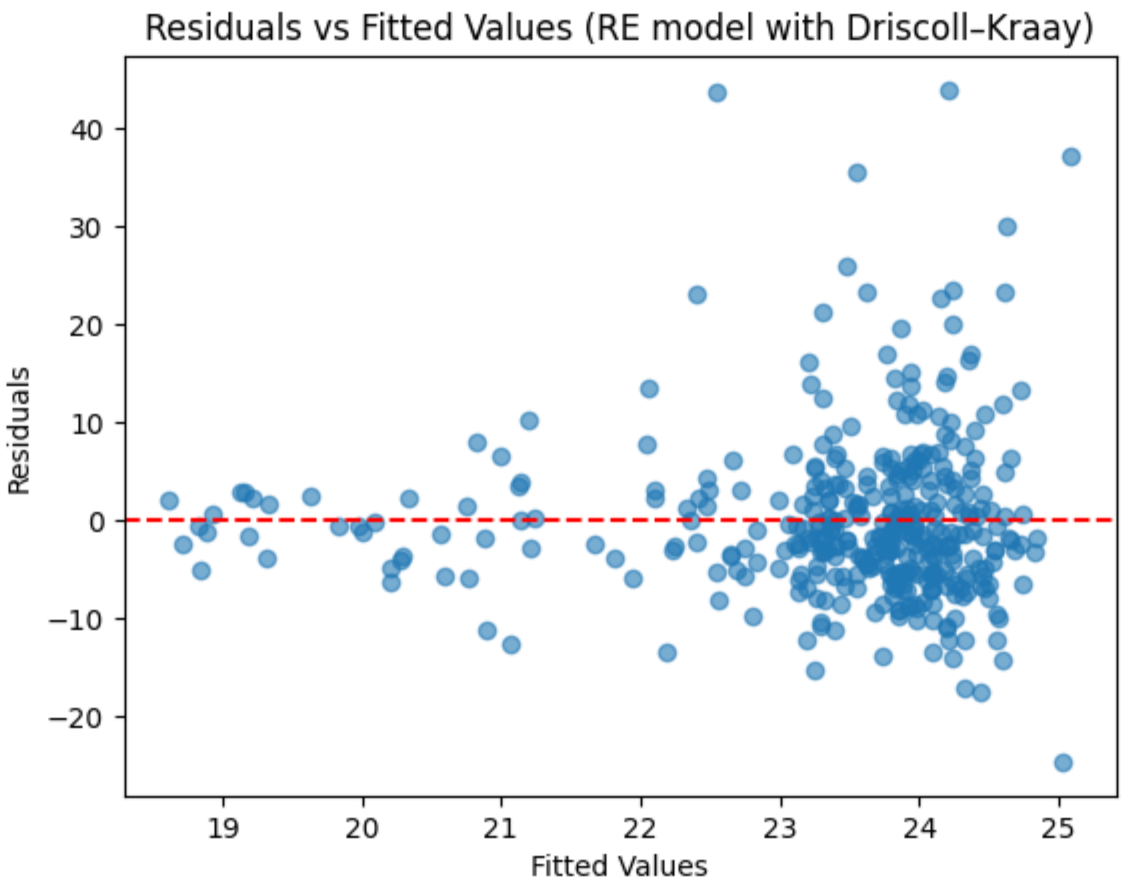


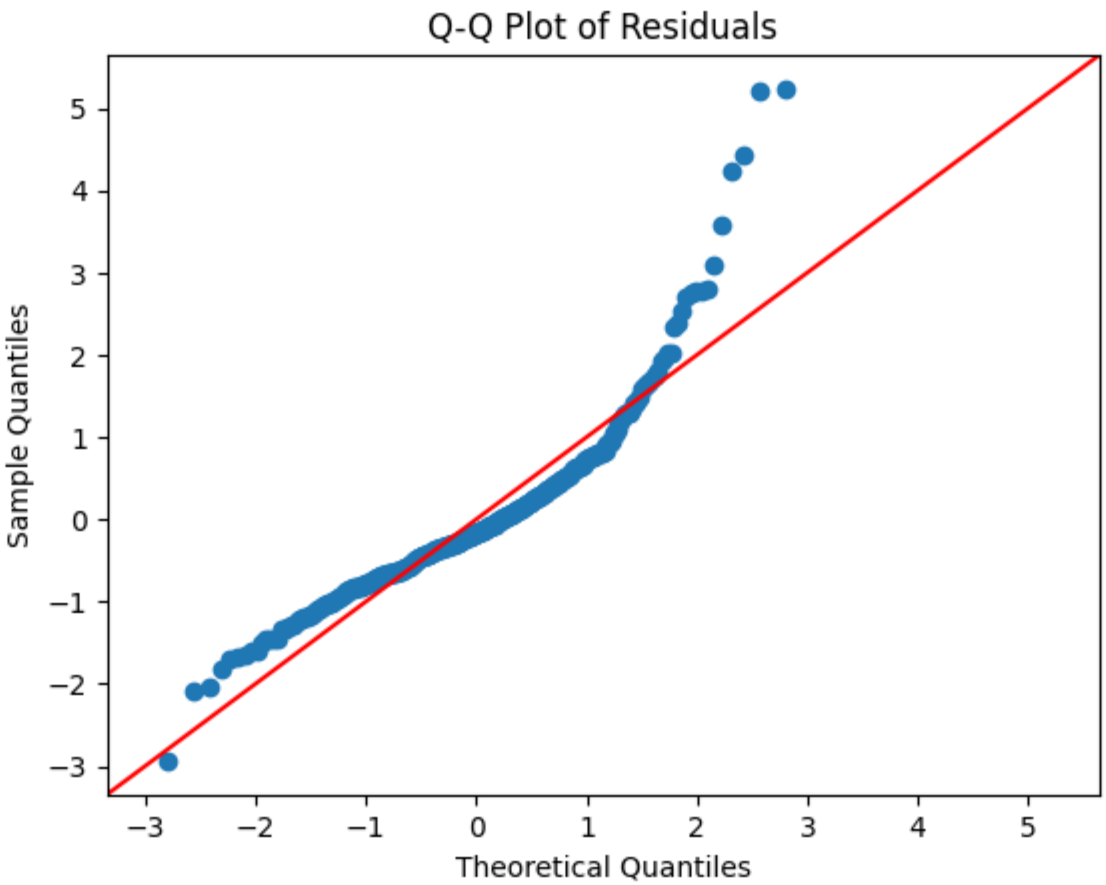

```
Shapiro-Wilk Test: Statistic=0.886, p-value=0.0000
```

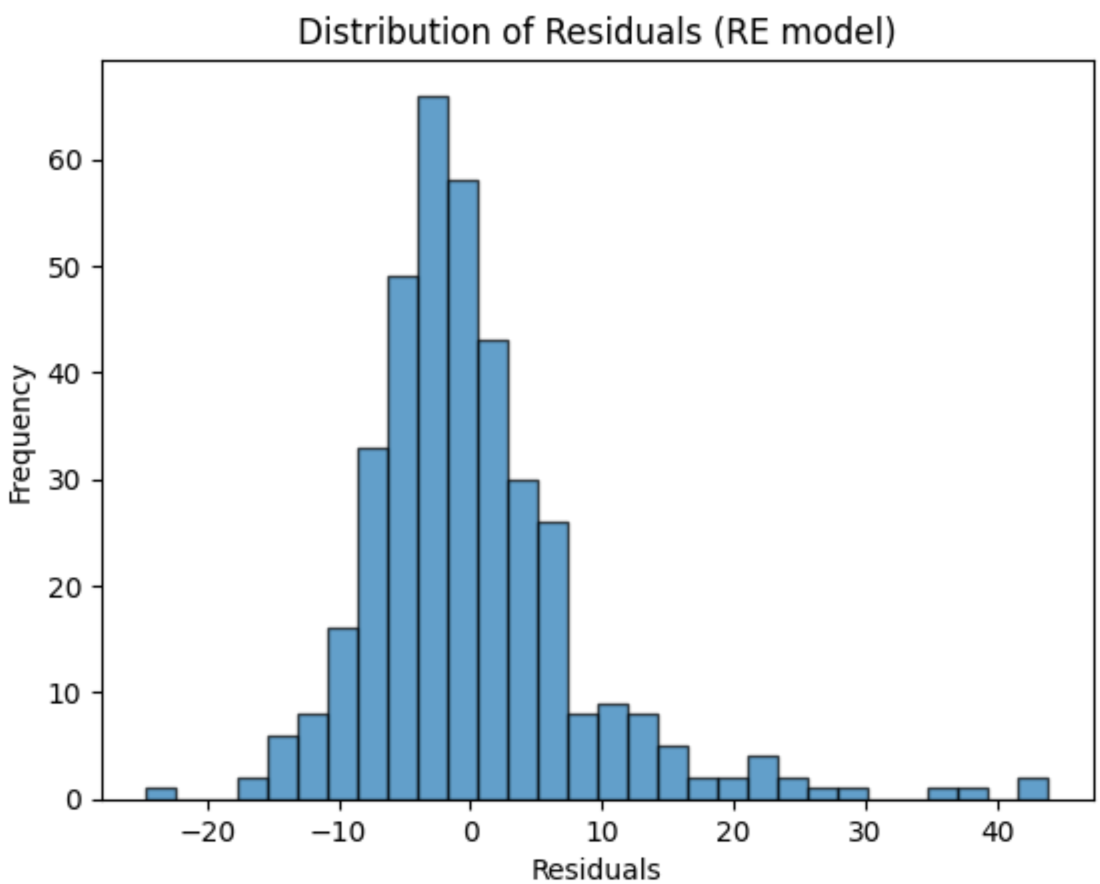
In [17]:

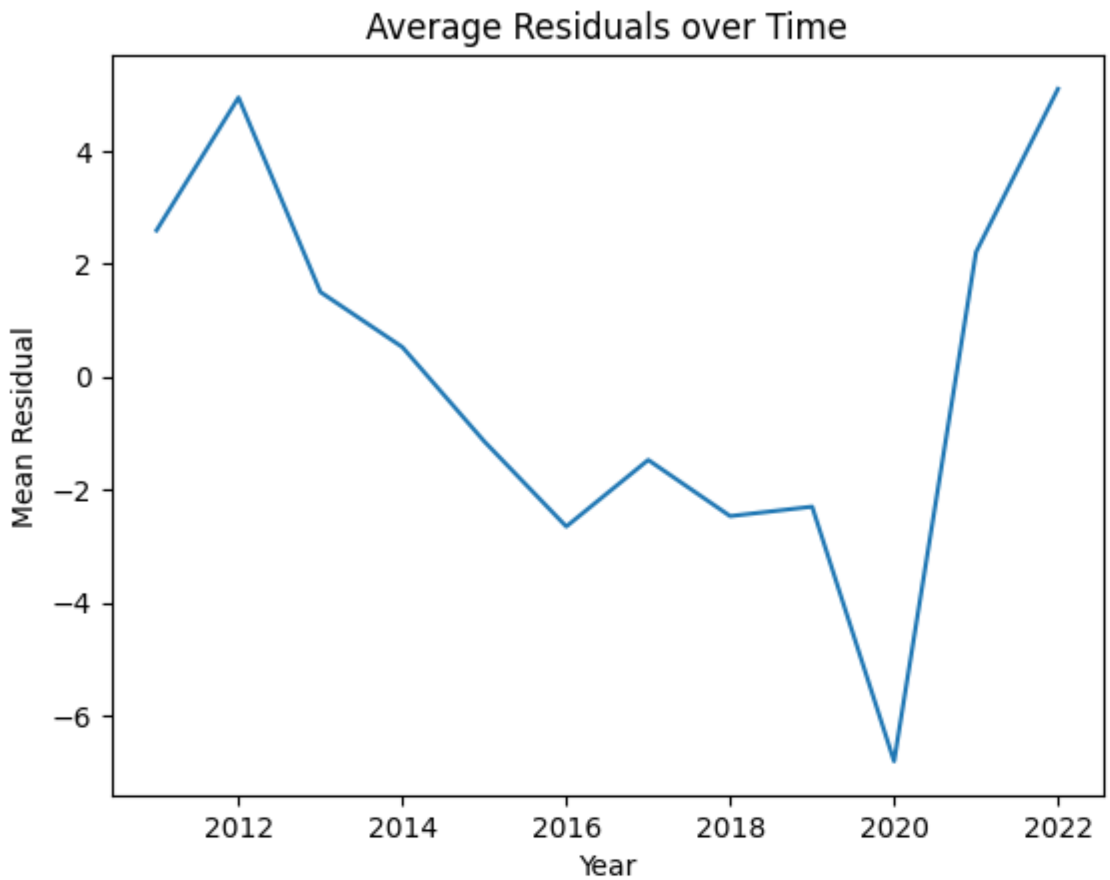
```
# Check residuals and fitted values
df['residuals2'] = re_model_robust2.resids
df['fitted2'] = re_model_robust2.fitted_values

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')
```

```python
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
−3
−3
−2
−1
0
1
2
3
4
5


Distribution of Residuals (RE model)
Frequency
Residuals
60
50
40
30
20
10
0
−20
−10
0
10
20
30
40




```
Shapiro-Wilk Test: Statistic=0.886, p-value=0.0000
```

In [ ]: