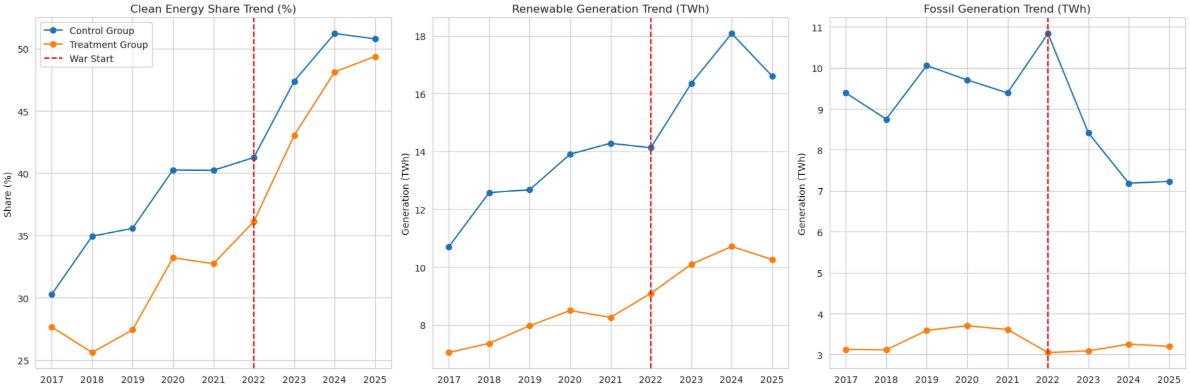
In [104…

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from linearmodels import PanelOLS
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
import warnings
warnings.filterwarnings('ignore')

# ================================
# 1. 核心配置：更新处理组/对照组（用户指定）
# ================================
treatment_countries = ['Bulgaria', 'Estonia', 'Germany', 'Hungary', 'Poland', 'Lithu
control_countries = ['Belgium', 'France', 'Spain', 'Ireland', 'Italy', 'Greece', 'Po
all_countries = treatment_countries + control_countries  # 16国完整列表

# ================================
# 2. 数据加载（完全匹配用户提供的CSV文件）
# ================================
# 基础能源数据
energy_share_df = pd.read_csv("D:\\大修\\16国清洁能源发电量占比.csv")  # 清洁能源占
nuclear_share_df = pd.read_csv("D:\\大修\\16国核能发电量占比.csv")  # 核电占比
renewable_gen_df = pd.read_csv("D:\\大修\\16国清洁发电量Gwh.csv")  # 清洁发电量（GWh
fossil_gen_df = pd.read_csv("D:\\大修\\16国天然气发电量Gwh.csv")  # 化石发电量（GWh）

# 宏观经济数据
gdp_df = pd.read_csv("D:\\大修\\16国GDP现行价格百万欧元.csv")  # GDP（季度）
electricity_price_df = pd.read_csv("D:\\大修\\16国电价（每千瓦时欧元）.csv")  # 电价

# 政策与基础设施数据
repowerEU_df = pd.read_csv("D:\\大修\\16国REPowerEU 框架下可再生能源投资规模（亿欧元
infra_df = pd.read_csv("D:\\大修\\16国可再生能源基础设施存量即累计装机容量 (GW).csv"

# 对俄依赖度数据
oil_dep_df = pd.read_csv("D:\\大修\\16国对俄罗斯石油依赖程度.csv")  # 俄油依赖度
gas_dep_df = pd.read_csv("D:\\大修\\16国对俄罗斯天然气依赖程度.csv")  # 俄气依赖度

# 天气数据（用户提供的太阳辐射、降水、风速）
ghi_df = pd.read_csv("D:\\大修\\GHI_16countries_2017_2025_annual_long.csv")  # 太阳辐
precip_df = pd.read_csv("D:\\大修\\Precipitation_16countries_2017_2025.csv")  # 降水
windspeed_df = pd.read_csv("D:\\大修\\WindSpeed_16countries_2017_2025_annual_long.cs

# ================================
# 3. 数据预处理（统一时间频率：季度，适配不同格式）
# ================================
def preprocess_time_month(df, date_col='month', freq='Q'):
    """处理月度数据→季度聚合（发电量、核电占比等）"""
    df['date'] = pd.to_datetime(df[date_col])
    df['quarter'] = df['date'].dt.to_period(freq)
    df['year'] = df['date'].dt.year
    return df

def preprocess_time_halfyear(df, date_col='time', freq='Q'):
    """处理半年数据→季度插值（电价：2017-S1→拆分为Q1/Q2）"""
    df['year'] = df[date_col].str.split('-').str[0].astype(int)
    df['halfyear'] = df[date_col].str.split('-').str[1].map({'S1': 1, 'S2': 2})
    # 生成季度映射（S1→Q1/Q2，S2→Q3/Q4）
    quarter_map = {1: ['Q1', 'Q2'], 2: ['Q3', 'Q4']}
    df_expanded = []
    for _, row in df.iterrows():
        for q in quarter_map[row['halfyear']]:
                new_row = row.copy()
                new_row['quarter'] = pd.Period(f"{row['year']}-{q}")
                df_expanded.append(new_row)
    # 1. 将列表转换为 DataFrame
    df_expanded = pd.DataFrame(df_expanded)
    # 2. 按国家和季度排序，确保插值顺序正确
    df_expanded = df_expanded.sort_values(['country', 'quarter'])
    # 3. 对每个国家的价格进行线性插值
    df_expanded['price'] = df_expanded.groupby('country')['price'].transform(lambda

    return df_expanded[['country', 'quarter', 'price']]


def preprocess_time_annual(df, date_col='year', freq='Q'):
    """处理年度数据→季度插值（对俄依赖度、天气、REPowerEU投资）"""
    df['year'] = df[date_col].astype(int)
    # 生成年度内4个季度
    df_expanded = []
    for _, row in df.iterrows():
        for q in ['Q1', 'Q2', 'Q3', 'Q4']:
            new_row = row.copy()
            new_row['quarter'] = pd.Period(f"{row['year']}-{q}")
            df_expanded.append(new_row)
    df_expanded = pd.DataFrame(df_expanded)
    # 按国家-季度插值补全
    df_expanded = df_expanded.sort_values(['country', 'quarter'])
    return df_expanded

# 3.1 处理月度数据（发电量、核电占比）
renewable_gen_df = preprocess_time_month(renewable_gen_df, date_col='month')
fossil_gen_df = preprocess_time_month(fossil_gen_df, date_col='month')
nuclear_share_df = preprocess_time_month(nuclear_share_df, date_col='month')
energy_share_df = preprocess_time_month(energy_share_df, date_col='month')  # 假设清

# 3.2 处理季度数据（GDP）
gdp_df['quarter'] = pd.PeriodIndex(gdp_df['quarter'], freq='Q')

# 3.3 处理半年数据（电价）
electricity_price_df = preprocess_time_halfyear(electricity_price_df, date_col='time

# 3.4 处理年度数据（对俄依赖度、天气、REPowerEU、基础设施）
# 对俄依赖度（合并石油+天然气）
oil_dep_expanded = preprocess_time_annual(oil_dep_df, date_col='year')
gas_dep_expanded = preprocess_time_annual(gas_dep_df, date_col='year')
dependency_df = pd.merge(oil_dep_expanded[['country', 'quarter', 'percentage']].rena
                        gas_dep_expanded[['country', 'quarter', 'percentage']].renam
                        on=['country', 'quarter'], how='inner')

# 天气数据（假设年度数据列名：GHI→solar_irradiance，Precipitation→precipitation，W
# 1. 处理太阳辐射（年度数据）
ghi_expanded = preprocess_time_annual(ghi_df, date_col='year').rename(columns={'GHI'

# 2. 处理降水量（季度数据，直接转换格式）
precip_expanded = precip_df.copy()  # 先定义变量
# 假设你的降水量数据中，时间列名为 'quarter'，如果是其他名字（如 'date'），请修改下面
precip_expanded['quarter'] = pd.PeriodIndex(precip_expanded['quarter'], freq='Q')
precip_expanded = precip_expanded.rename(columns={'Precipitation': 'precipitation'})

# 3. 处理风速（年度数据）
windspeed_expanded = preprocess_time_annual(windspeed_df, date_col='year').rename(co

# 4. 合并天气数据
# 合并太阳辐射和降水
weather_df = pd.merge(ghi_expanded[['country', 'quarter', 'solar_irradiance']],
                      precip_expanded[['country', 'quarter', 'precipitation']],
                      on=['country', 'quarter'], how='inner')

# 再合并风速数据
weather_df = pd.merge(weather_df,
                      windspeed_expanded[['country', 'quarter', 'wind_speed']],
                      on=['country', 'quarter'], how='inner')

# REPowerEU投资（年度→季度插值）
repowerEU_expanded = preprocess_time_annual(repowerEU_df, date_col='year')

# 可再生基础设施存量（年度→季度插值）
infra_expanded = preprocess_time_annual(infra_df, date_col='year').rename(columns={'

# 3.5 聚合月度数据到季度（发电量、核电占比取均值/总和）
def aggregate_to_quarter(df, group_cols=['country', 'quarter'], value_cols=None):
    if value_cols is None:
        value_cols = [col for col in df.columns if col not in group_cols + ['date'
    agg_dict = {}
    for col in value_cols:
        if 'Gwh' in col or 'gen' in col:
            agg_dict[col] = 'sum'  # 发电量求和
        else:
            agg_dict[col] = 'mean'  # 占比取均值
    return df.groupby(group_cols)[value_cols].agg(agg_dict).reset_index()

# 执行聚合
renewable_gen_quarterly = aggregate_to_quarter(renewable_gen_df, value_cols=['Gwh'])
fossil_gen_quarterly = aggregate_to_quarter(fossil_gen_df, value_cols=['Gwh']).renam
nuclear_share_quarterly = aggregate_to_quarter(nuclear_share_df, value_cols=['nuclea
energy_share_quarterly = aggregate_to_quarter(energy_share_df, value_cols=['clean_sh

# ================================
# 4. 合并所有数据集（核心步骤）
# ================================
# 4.1 逐步合并
merged_df = pd.merge(energy_share_quarterly, gdp_df[['country', 'quarter', 'gdp']],
merged_df = pd.merge(merged_df, electricity_price_df, on=['country', 'quarter'], how
merged_df = pd.merge(merged_df, nuclear_share_quarterly, on=['country', 'quarter'],
merged_df = pd.merge(merged_df, renewable_gen_quarterly, on=['country', 'quarter'],
merged_df = pd.merge(merged_df, fossil_gen_quarterly, on=['country', 'quarter'], how
merged_df = pd.merge(merged_df, dependency_df, on=['country', 'quarter'], how='left'
merged_df = pd.merge(merged_df, weather_df, on=['country', 'quarter'], how='left')
merged_df = pd.merge(merged_df, repowerEU_expanded[['country', 'quarter', 'repowereu
merged_df = pd.merge(merged_df, infra_expanded[['country', 'quarter', 'renewable_inf

# 4.2 筛选16国样本
merged_df = merged_df[merged_df['country'].isin(all_countries)]

# 4.3 缺失值处理（前向填充+线性插值）
merged_df = merged_df.sort_values(['country', 'quarter'])
for col in ['gdp', 'price', 'nuclear_share', 'renewable_gen', 'fossil_gen', 'oil_dep
            'solar_irradiance', 'precipitation', 'wind_speed', 'repowereu', 'renewab
    merged_df[col] = merged_df.groupby('country')[col].ffill().interpolate(method='l

# ================================
# 5. 核心变量构建（适配新分组和数据）
# ================================
# 5.1 处理组/时间虚拟变量
merged_df['treat'] = merged_df['country'].isin(treatment_countries).astype(int)
merged_df['post'] = (merged_df['quarter'].dt.year >= 2022).astype(int)  # 2022年为战
merged_df['quarter_timestamp'] = merged_df['quarter'].dt.to_timestamp()  # PanelOLS要

# 5.2 DID交互项（二元+连续型）
```

```
merged_df['treat_post'] = merged_df['treat'] * merged_df['post']
# 连续型对俄依赖度（油+气加权，权重：油0.3，气0.7，因天然气对电力结构影响更大）
merged_df['russia_dep_continuous'] = (merged_df['oil_dep'] * 0.3 + merged_df['gas_de
merged_df['dep_post'] = merged_df['russia_dep_continuous'] * merged_df['post']

# 5.3 政策异质性变量（REPowerEU强度指数+EGD交互项）
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
merged_df['repowereu_std'] = scaler.fit_transform(merged_df[['repowereu']])
merged_df['repowerEU_index'] = merged_df['repowereu_std']  # 单指标标准化后作为强度指
# EGD政策（2020年后）×可再生基础设施存量
merged_df['egd'] = (merged_df['quarter'].dt.year >= 2020).astype(int)
merged_df['egd_infra'] = merged_df['egd'] * merged_df['renewable_infra']

# 5.4 天气控制变量标准化
merged_df[['wind_std', 'solar_std', 'precip_std']] = scaler.fit_transform(merged_df[

# 5.5 效应分解变量（绝对量+占比）
merged_df['total_demand'] = merged_df['renewable_gen'] + merged_df['fossil_gen']  #
merged_df['renewable_gen_share'] = merged_df['renewable_gen'] / merged_df['total_dem
merged_df['fossil_gen_share'] = merged_df['fossil_gen'] / merged_df['total_demand']

# 5.6 设置面板数据结构
merged_df = merged_df.set_index(['country', 'quarter_timestamp'])
# 替代原有的 dropna()，仅删除核心因变量缺失的样本，保留控制变量缺失的样本
merged_df = merged_df.dropna(subset=['clean_share'])  # 只删因变量缺失
# 对控制变量的缺失值，用国家均值填充（而非删除）
for col in ['gdp', 'price', 'repowereu', 'egd_infra', 'wind_std']:
    if col in merged_df.columns:
        merged_df[col] = merged_df.groupby('country')[col].fillna(lambda x: x.mean(

print("最终数据维度：", merged_df.shape)
print("变量列表：", merged_df.columns.tolist())
print("处理组国家样本数：", merged_df[merged_df['treat']==1].index.get_level_values(
print("对照组国家样本数：", merged_df[merged_df['treat']==0].index.get_level_values(

# ================================
# 6. 描述性统计与可视化（保持原有逻辑）
# ================================
desc_stats = merged_df[['clean_share', 'renewable_gen', 'fossil_gen', 'russia_dep_co
print("\n核心变量描述性统计：")
print(desc_stats)

# 趋势图（处理组vs对照组）
trend_data = merged_df.reset_index()
trend_data['year'] = trend_data['quarter_timestamp'].dt.year
yearly_avg = trend_data.groupby(['year', 'treat'])[['clean_share', 'renewable_gen',

fig, axes = plt.subplots(1, 3, figsize=(18, 6))
# 清洁能源占比趋势
for treat in [0, 1]:
    subset = yearly_avg[yearly_avg['treat'] == treat]
    label = 'Treatment Group' if treat == 1 else 'Control Group'
    axes[0].plot(subset['year'], subset['clean_share'], label=label, marker='o')
axes[0].axvline(x=2022, color='r', linestyle='--', label='War Start')
axes[0].set_title('Clean Energy Share Trend (%)')
axes[0].set_ylabel('Share (%)')
axes[0].legend()

# 可再生绝对发电量趋势
for treat in [0, 1]:
    subset = yearly_avg[yearly_avg['treat'] == treat]
    label = 'Treatment Group' if treat == 1 else 'Control Group'
    axes[1].plot(subset['year'], subset['renewable_gen']/1000, label=label, marker='
axes[1].axvline(x=2022, color='r', linestyle='--')
axes[1].set_title('Renewable Generation Trend (TWh)')
axes[1].set_ylabel('Generation (TWh)')

# 化石绝对发电量趋势
for treat in [0, 1]:
    subset = yearly_avg[yearly_avg['treat'] == treat]
    label = 'Treatment Group' if treat == 1 else 'Control Group'
    axes[2].plot(subset['year'], subset['fossil_gen']/1000, label=label, marker='o')
axes[2].axvline(x=2022, color='r', linestyle='--')
axes[2].set_title('Fossil Generation Trend (TWh)')
axes[2].set_ylabel('Generation (TWh)')

plt.tight_layout()
plt.savefig(r"D:\Dissertation 1\trend_analysis_updated.png", dpi=300, bbox_inches='t
plt.show()
```

```
最终数据维度： (576, 29)
变量列表： ['quarter', 'clean_share', 'gdp', 'price', 'nuclear_share', 'renewable_ge
n', 'fossil_gen', 'oil_dep', 'gas_dep', 'solar_irradiance', 'precipitation', 'wind_s
peed', 'repowereu', 'renewable_infra', 'treat', 'post', 'treat_post', 'russia_dep_co
ntinuous', 'dep_post', 'repowereu_std', 'repowerEU_index', 'egd', 'egd_infra', 'wind
_std', 'solar_std', 'precip_std', 'total_demand', 'renewable_gen_share', 'fossil_gen
_share']
处理组国家样本数： 8
对照组国家样本数： 8

核心变量描述性统计：
       clean_share  renewable_gen    fossil_gen  russia_dep_continuous  \
count   576.000000     576.000000    576.000000             576.000000   
mean     38.620843   11586.019196   6149.384971               0.505552   
std      19.518164   14990.547686   8447.342255               0.310939   
min       5.293667     126.200000     26.000000               0.021600   
25%      23.201167    1677.524750    583.737750               0.185960   
50%      36.452167    5131.453000   2951.469000               0.552465   
75%      52.134000   16082.849000   6238.493750               0.816530   
max      97.768000   68879.740000  39677.932000               0.930310   

       repowerEU_index      wind_std
count     5.560000e+02  5.760000e+02
mean      1.597443e-16 -2.467162e-16
std       1.000900e+00  1.000869e+00
min      -1.077120e+00 -1.530466e+00
25%      -7.860992e-01 -7.477382e-01
50%      -4.661412e-01 -2.307947e-01
75%       6.557258e-01  5.118743e-01
max       2.855601e+00  2.739882e+00
```



In [ ]:

```
# 7 基线DID（含天气+政策控制，双聚类标准误）
print("\n=== 基线DID回归结果（清洁能源占比，含完整控制变量）===")
baseline_model = PanelOLS.from_formula(
    '''clean_share ~ treat_post + dep_post + repowerEU_index  +
    gdp + price + wind_std + solar_std +
    EntityEffects + TimeEffects''',
    data=merged_df
)
baseline_results = baseline_model.fit(cov_type='clustered', cluster_entity=True, cl
print(baseline_results.summary)
```

```
=== 基线DID回归结果（清洁能源占比，含完整控制变量）===
                          PanelOLS Estimation Summary
================================================================================
Dep. Variable:            clean_share   R-squared:                        0.1037
Estimator:                   PanelOLS   R-squared (Between):              0.1051
No. Observations:                 556   R-squared (Within):               0.2107
Date:                Sun, Feb 22 2026   R-squared (Overall):              0.1186
Time:                        22:16:15   Log-likelihood                   -1876.1
Cov. Estimator:             Clustered
                                        F-statistic:                      8.2308
Entities:                          16   P-value                           0.0000
Avg Obs:                       34.750   Distribution:                   F(7,498)
Min Obs:                       16.000
Max Obs:                       36.000   F-statistic (robust):            -42.681
                                        P-value                           1.0000
Time periods:                      36   Distribution:                   F(7,498)
Avg Obs:                       15.444
Min Obs:                       15.000
Max Obs:                       16.000

                                 Parameter Estimates
===================================================================================
                 Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
-----------------------------------------------------------------------------------
treat_post          8.7449     4.0011     2.1856     0.0293      0.8837      16.606
dep_post           -2.6623     7.1990    -0.3698     0.7117     -16.806      11.482
repowerEU_index    -1.7392     0.9346    -1.8608     0.0634     -3.5755      0.0971
gdp              5.412e-06  1.786e-05     0.3030     0.7620  -2.968e-05   4.051e-05
price               1.0916     14.212     0.0768     0.9388     -26.832      29.015
wind_std            4.2405     1.7703     2.3953     0.0170      0.7623      7.7187
solar_std           9.1344     2.0044     4.5573     0.0000      5.1964      13.072
===================================================================================

F-test for Poolability: 49.787
P-value: 0.0000
Distribution: F(50,498)

Included effects: Entity, Time
```

In [ ]:

```
# ================================
# 8 稳健性检验
# 8.1 添加降水和egd政策控制变量
print("\n=== 基线DID回归结果（清洁能源占比，含完整控制变量）===")
baseline_model = PanelOLS.from_formula(
    '''clean_share ~ treat_post + dep_post + repowerEU_index + egd_infra +
    gdp + price + wind_std + solar_std + precip_std +
    EntityEffects + TimeEffects''',
    data=merged_df
)
baseline_results = baseline_model.fit(cov_type='clustered', cluster_entity=True, cl
print(baseline_results.summary)
```

```
=== 基线DID回归结果（清洁能源占比，含完整控制变量）===
                          PanelOLS Estimation Summary
================================================================================
Dep. Variable:            clean_share   R-squared:                        0.1040
Estimator:                   PanelOLS   R-squared (Between):              0.0891
No. Observations:                 556   R-squared (Within):               0.2129
Date:                Sun, Feb 22 2026   R-squared (Overall):              0.1033
Time:                        22:20:27   Log-likelihood                   -1876.0
Cov. Estimator:             Clustered
                                        F-statistic:                      6.3986
Entities:                          16   P-value                           0.0000
Avg Obs:                       34.750   Distribution:                   F(9,496)
Min Obs:                       16.000
Max Obs:                       36.000   F-statistic (robust):             7.4003
                                        P-value                           0.0000
Time periods:                      36   Distribution:                   F(9,496)
Avg Obs:                       15.444
Min Obs:                       15.000
Max Obs:                       16.000

                                Parameter Estimates
===================================================================================
                 Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
-----------------------------------------------------------------------------------
treat_post          8.8694     3.9746     2.2315     0.0261      1.0603      16.678
dep_post           -2.9045     7.5259    -0.3859     0.6997     -17.691      11.882
repowerEU_index    -1.8599     1.1814    -1.5743     0.1160     -4.1811      0.4613
egd_infra           0.0054     0.0483     0.1112     0.9115     -0.0896      0.1003
gdp              2.207e-06  1.909e-05     0.1156     0.9080  -3.529e-05   3.971e-05
price               1.2453     14.694     0.0847     0.9325     -27.625      30.115
wind_std            4.2021     1.8679     2.2497     0.0249      0.5322      7.8720
solar_std           9.0262     2.2579     3.9975     0.0001      4.5899      13.463
precip_std          0.2608     1.3276     0.1965     0.8443     -2.3476      2.8692
===================================================================================

F-test for Poolability: 48.491
P-value: 0.0000
Distribution: F(50,496)

Included effects: Entity, Time
```

In [ ]:

```
# 8.2 效应分解回归（绝对量被解释变量）
print("\n=== 效应分解回归（可再生绝对发电量）===")
renewable_model = PanelOLS.from_formula(
    '''renewable_gen ~ treat_post + repowerEU_index + egd_infra +
    gdp + price + wind_std + solar_std + precip_std +
    EntityEffects + TimeEffects''',
    data=merged_df
)
renewable_results = renewable_model.fit(cov_type='clustered', cluster_entity=True, 
print(renewable_results.summary)

print("\n=== 效应分解回归（化石绝对发电量）===")
fossil_model = PanelOLS.from_formula(
    '''fossil_gen ~ treat_post + repowerEU_index + egd_infra +
    gdp + price + wind_std + solar_std + precip_std +
    EntityEffects + TimeEffects''',
    data=merged_df
)
fossil_results = fossil_model.fit(cov_type='clustered', cluster_entity=True, cluste
print(fossil_results.summary)
```

```
=== 效应分解回归（可再生绝对发电量）===
                          PanelOLS Estimation Summary                           
================================================================================
Dep. Variable:          renewable_gen   R-squared:                        0.3110
Estimator:                   PanelOLS   R-squared (Between):              0.5129
No. Observations:                 556   R-squared (Within):               0.4099
Date:                Sun, Feb 22 2026   R-squared (Overall):              0.5104
Time:                        22:16:46   Log-likelihood                   -5105.2
Cov. Estimator:             Clustered                                           
                                        F-statistic:                      28.046
Entities:                          16   P-value                           0.0000
Avg Obs:                       34.750   Distribution:                   F(8,497)
Min Obs:                       16.000                                           
Max Obs:                       36.000   F-statistic (robust):             3.6648
                                        P-value                           0.0004
Time periods:                      36   Distribution:                   F(8,497)
Avg Obs:                       15.444                                           
Min Obs:                       15.000                                           
Max Obs:                       16.000                                           
                                                                                
                                 Parameter Estimates                                 
=====================================================================================
                 Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
-------------------------------------------------------------------------------------
treat_post         -51.551     436.18    -0.1182     0.9060     -908.54      805.44
repowerEU_index     713.91     256.98     2.7781     0.0057      209.01      1218.8
egd_infra           48.983     14.610     3.3528     0.0009      20.279      77.688
gdp                 0.0103     0.0121     0.8495     0.3960     -0.0135      0.0340
price              -1379.5     6677.5    -0.2066     0.8364   -1.45e+04   1.174e+04
wind_std           -237.78     143.29    -1.6594     0.0977     -519.31      43.756
solar_std           526.00     594.49     0.8848     0.3767     -642.03      1694.0
precip_std         -353.34     690.43    -0.5118     0.6090     -1709.9      1003.2
=====================================================================================

F-test for Poolability: 16.785
P-value: 0.0000
Distribution: F(50,497)

Included effects: Entity, Time

=== 效应分解回归（化石绝对发电量）===
                          PanelOLS Estimation Summary                           
================================================================================
Dep. Variable:             fossil_gen   R-squared:                        0.0478
Estimator:                   PanelOLS   R-squared (Between):             -0.1042
No. Observations:                 556   R-squared (Within):               0.0246
Date:                Sun, Feb 22 2026   R-squared (Overall):             -0.1032
Time:                        22:16:46   Log-likelihood                   -4981.7
Cov. Estimator:             Clustered                                           
                                        F-statistic:                      3.1159
Entities:                          16   P-value                           0.0019
Avg Obs:                       34.750   Distribution:                   F(8,497)
Min Obs:                       16.000                                           
Max Obs:                       36.000   F-statistic (robust):             3.7823
                                        P-value                           0.0003
Time periods:                      36   Distribution:                   F(8,497)
Avg Obs:                       15.444                                           
Min Obs:                       15.000                                           
Max Obs:                       16.000                                           
                                                                                
                                 Parameter Estimates                                 
=====================================================================================
                 Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
-------------------------------------------------------------------------------------
```

```
treat_post          404.55     624.47     0.6478     0.5174     -822.37      1631.5
repowerEU_index    -80.847     402.51    -0.2009     0.8409     -871.67      709.98
egd_infra          -7.5756     11.035    -0.6865     0.4927     -29.257      14.106
gdp                -0.0054     0.0081    -0.6628     0.5078     -0.0213      0.0106
price               8760.2     5765.9     1.5193     0.1293     -2568.4   2.009e+04
wind_std           -371.07     419.71    -0.8841     0.3771     -1195.7      453.56
solar_std          -25.232     464.28    -0.0543     0.9567     -937.43      886.97
precip_std          9.0053     483.11     0.0186     0.9851     -940.19      958.20
===================================================================================

F-test for Poolability: 62.082
P-value: 0.0000
Distribution: F(50,497)

Included effects: Entity, Time
```

In [ ]:

```
# 8.3 核电剔除稳健性检验
print("\n=== 稳健性检验：仅可再生能源占比（剔除核电）===")
merged_df['renewable_only_share'] = merged_df['renewable_gen'] / merged_df['total_de
nuclear_excl_model = PanelOLS.from_formula(
    '''renewable_only_share ~ treat_post + repowerEU_index + egd_infra +
    gdp + price + wind_std + solar_std + precip_std +
    EntityEffects + TimeEffects''',
    data=merged_df
)
nuclear_excl_results = nuclear_excl_model.fit(cov_type='clustered', cluster_entity=T
print(nuclear_excl_results.summary)
```

```
=== 稳健性检验：仅可再生能源占比（剔除核电）===
                          PanelOLS Estimation Summary
================================================================================
Dep. Variable:     renewable_only_share   R-squared:                        0.0761
Estimator:                     PanelOLS   R-squared (Between):             -0.2655
No. Observations:                   556   R-squared (Within):               0.0767
Date:                  Sun, Feb 22 2026   R-squared (Overall):             -0.2528
Time:                          22:17:14   Log-likelihood                   -1944.9
Cov. Estimator:               Clustered
                                          F-statistic:                      5.1169
Entities:                            16   P-value                           0.0000
Avg Obs:                         34.750   Distribution:                   F(8,497)
Min Obs:                         16.000
Max Obs:                         36.000   F-statistic (robust):             12.211
                                          P-value                           0.0000
Time periods:                        36   Distribution:                   F(8,497)
Avg Obs:                         15.444
Min Obs:                         15.000
Max Obs:                         16.000

                                Parameter Estimates
=================================================================================
                 Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
---------------------------------------------------------------------------------
treat_post          6.8419     3.6035     1.8987     0.0582     -0.2381      13.922
repowerEU_index     0.1681     1.3398     0.1255     0.9002     -2.4643      2.8005
egd_infra          -0.0024     0.0489    -0.0491     0.9609     -0.0984      0.0936
gdp             -2.438e-05  2.014e-05    -1.2103     0.2267  -6.396e-05    1.52e-05
price              -21.525     26.158    -0.8229     0.4110     -72.919      29.869
wind_std            3.9494     1.4129     2.7953     0.0054      1.1735      6.7254
solar_std           5.6994     2.3390     2.4367     0.0152      1.1038      10.295
precip_std          2.0373     1.6104     1.2650     0.2065     -1.1269      5.2014
=================================================================================

F-test for Poolability: 29.993
P-value: 0.0000
Distribution: F(50,497)

Included effects: Entity, Time
```

In [ ]:

```
# 8.4 安慰剂检验（虚假战争时间：2020年）
print("\n=== 安慰剂检验：虚假战争时间（2020年）===")
placebo_df = merged_df.loc[merged_df.index.get_level_values('quarter_timestamp') < '
placebo_df['placebo_post'] = (placebo_df.index.get_level_values('quarter_timestamp')
placebo_df['placebo_treat_post'] = placebo_df['treat'] * placebo_df['placebo_post']

placebo_model = PanelOLS.from_formula(
    '''clean_share ~ placebo_treat_post + repowerEU_index + egd_infra +
    gdp + price + wind_std + solar_std + precip_std +
    EntityEffects + TimeEffects''',
    data=placebo_df
)
placebo_results = placebo_model.fit(cov_type='clustered', cluster_entity=True, clus
print(placebo_results.summary)
```

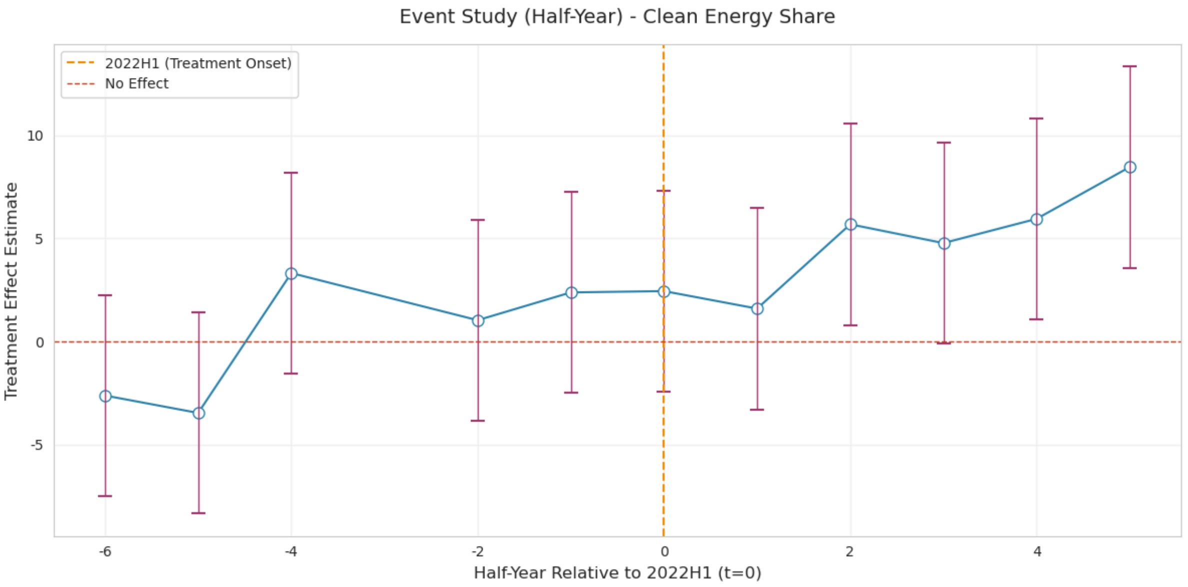
```
=== 安慰剂检验：虚假战争时间（2020年）===
                          PanelOLS Estimation Summary
================================================================================
Dep. Variable:            clean_share   R-squared:                        0.0750
Estimator:                   PanelOLS   R-squared (Between):             -0.8391
No. Observations:                 300   R-squared (Within):               0.0970
Date:                Sun, Feb 22 2026   R-squared (Overall):             -0.7953
Time:                        22:17:30   Log-likelihood                   -999.60
Cov. Estimator:             Clustered
                                        F-statistic:                      2.6148
Entities:                          15   P-value                           0.0091
Avg Obs:                       20.000   Distribution:                    F(8,258)
Min Obs:                       20.000
Max Obs:                       20.000   F-statistic (robust):            -1.8349
                                        P-value                           1.0000
Time periods:                      20   Distribution:                    F(8,258)
Avg Obs:                       15.000
Min Obs:                       15.000
Max Obs:                       15.000

                                Parameter Estimates
====================================================================================
==
                     Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper 
CI
------------------------------------------------------------------------------------
--
placebo_treat_post      2.6373     4.4830     0.5883     0.5569     -6.1907      11.4
65
repowerEU_index         0.2191     1.4658     0.1494     0.8813     -2.6675      3.10
56
egd_infra              -0.0045     0.0445    -0.1009     0.9197     -0.0921      0.08
31
gdp                 -1.438e-05  1.585e-05    -0.9075     0.3650  -4.559e-05  1.683e-
05
price                  -63.103     70.872    -0.8904     0.3741     -202.66      76.4
58
wind_std                5.2042     2.1732     2.3947     0.0173      0.9247      9.48
36
solar_std               12.107     4.0628     2.9801     0.0032      4.1071      20.1
08
precip_std              0.2270     1.3162     0.1725     0.8632     -2.3649      2.81
90
====================================================================================
==

F-test for Poolability: 34.156
P-value: 0.0000
Distribution: F(33,258)

Included effects: Entity, Time
```

In [ ]:

```python
# 8.5 平行趋势检验
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
import numpy as np

# 解决Matplotlib显示问题
%matplotlib inline
plt.rcParams['font.sans-serif'] = ['DejaVu Sans']
plt.rcParams['axes.unicode_minus'] = False

def semi_annual_parallel_trends_test(file_path):
```

```
"""
半年度平行趋势检验（适配2019-2024窗口，基准期2021H2）
核心逻辑：保留原代码的半年度事件研究，补充数据校验、异常处理和统计检验

参数:
    file_path (str): 数据文件路径（CSV格式）
返回:
    model_results: 回归模型结果
    plot_df: 绘图用的系数和置信区间数据
"""
# ==================== 1. 数据加载与预处理 ====================
try:
    # 解决Windows路径反斜杠转义问题
    df = pd.read_csv(file_path)
    print(f"✅ 成功加载数据，数据形状：{df.shape}")
except Exception as e:
    print(f"❌ 数据加载失败：{str(e)}")
    return None, None

# 重命名列（保持原逻辑）
df.rename(columns={
    'country': 'Country',
    'month': 'Date',
    'clean_share': 'CleanEnergyShare'
}, inplace=True)

# 数据类型转换（带校验）
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
df['CleanEnergyShare'] = pd.to_numeric(df['CleanEnergyShare'], errors='coerce')

# 检查缺失值
if df['Date'].isnull().any():
    print(f"⚠️  发现{df['Date'].isnull().sum()}条无效日期，已剔除")
    df = df.dropna(subset=['Date'])
if df['CleanEnergyShare'].isnull().any():
    print(f"⚠️  发现{df['CleanEnergyShare'].isnull().sum()}条无效清洁能源占比，
    df = df.dropna(subset=['CleanEnergyShare'])

# 计算半年度和事件半年度（保持原逻辑）
df['Year'] = df['Date'].dt.year
df['Half'] = np.where(df['Date'].dt.month <= 6, 1, 2)
# EventHalfYear计算规则：2022H1=0，2021H2=-1，2021H1=-2，2022H2=1...
df['EventHalfYear'] = (df['Year'] - 2022) * 2 + (df['Half'] - 1)

# 筛选分析窗口（2019-2024）
df = df[(df['Year'] >= 2019) & (df['Year'] <= 2024)]
if len(df) == 0:
    print("❌ 筛选后数据为空，请检查时间范围")
    return None, None
print(f"✅ 筛选2019-2024数据后，剩余行数：{len(df)}")

# 设置处理组（前5个国家）
treated_countries = df['Country'].unique()[:5]
df['Treat'] = df['Country'].isin(treated_countries).astype(int)
print(f"✅ 处理组国家：{list(treated_countries)}，共{len(treated_countries)}个")

# ==================== 2. 基准期设定（2021H2 = EventHalfYear=-1） ==============
# 确保EventHalfYear为数值型，再设置基准期
df['EventHalfYear'] = df['EventHalfYear'].astype(int)
# 确认基准期存在
if -3 not in df['EventHalfYear'].unique():
    print("❌ 基准期2021H2（EventHalfYear=-1）不存在于数据中")
    return None, None
# 将2021H2（-3）设为基准期
df['EventHalfYear_cat'] = df['EventHalfYear'].astype('category')
# 重新排序类别，将-1设为基准（statsmodels会自动将第一个类别作为基准）
all_cats = sorted(df['EventHalfYear'].unique())
all_cats.remove(-3)  # 移除基准期
all_cats = [-3] + all_cats  # 把基准期放在第一个位置
df['EventHalfYear_cat'] = df['EventHalfYear_cat'].cat.set_categories(all_cats)

# ==================== 3. DID回归模型（半年度） ====================
try:
    # 回归公式：国家固定效应 + 时间固定效应 + 处理组*事件半年度交互项
    formula = "CleanEnergyShare ~ C(Country) + C(EventHalfYear_cat) + Treat*C(Ev
    model = smf.ols(formula, data=df).fit()
    print("\n✅ 回归模型拟合成功")
    print(f"模型R²：{model.rsquared:.4f}")
except Exception as e:
    print(f"❌ 回归模型拟合失败：{str(e)}")
    return None, None

# ==================== 4. 提取交互项系数和置信区间（稳健版） ====================
# 筛选Treat*EventHalfYear_cat的交互项系数
coef_mask = model.params.index.str.contains('Treat:C\(EventHalfYear_cat\)')
event_effects = model.params[coef_mask]
if len(event_effects) == 0:
    print("❌ 未提取到交互项系数，请检查回归公式")
    return model, None

# 提取对应的置信区间
event_conf = model.conf_int().loc[event_effects.index]
# 提取事件半年度数值（从系数名中解析）
event_half = event_effects.index.str.extract(r'Treat:C\(EventHalfYear_cat\)\[T\.

# 整理绘图数据
plot_df = pd.DataFrame({
    'EventHalfYear': event_half,
    'Coefficient': event_effects.values,
    'CI_Lower': event_conf[0].values,
    'CI_Upper': event_conf[1].values
}).sort_values('EventHalfYear').reset_index(drop=True)

# ==================== 5. 平行趋势统计检验（冲击前系数联合显著性） ==============
# 筛选冲击前时期（EventHalfYear < 0，即2022H1之前）
pre_periods = plot_df[plot_df['EventHalfYear'] < 0]['EventHalfYear'].tolist()
if len(pre_periods) > 0:
    # 构建检验公式：所有冲击前交互项系数联合为0
    pre_coef_names = [f"Treat:C(EventHalfYear_cat)[T.{p}]" for p in pre_periods
    if pre_coef_names:
        test_formula = " + ".join(pre_coef_names) + " = 0"
        try:
            f_test = model.f_test(test_formula)
            print("\n" + "="*60)
            print("📊 平行趋势统计检验结果（冲击前系数联合显著性）：")
            print(f"F统计量：{f_test.statistic[0][0]:.4f}")
            print(f"p值：{f_test.pvalue:.4f}")
            print(f"结论：{'✅ 平行趋势成立' if f_test.pvalue > 0.05 else '❌
            print("="*60)
        except Exception as e:
            print(f"⚠️  平行趋势检验失败：{str(e)}")
    else:
        print("⚠️  无冲击前系数可用于平行趋势检验")
else:
    print("⚠️  数据中无冲击前时期（EventHalfYear < 0）")

# ==================== 6. 绘制半年度事件研究图 ====================
plt.figure(figsize=(12, 6))
# 绘制系数折线
plt.plot(plot_df['EventHalfYear'], plot_df['Coefficient'],
         marker='o', linestyle='-', color='#2E86AB',
         markerfacecolor='white', markeredgecolor='#2E86AB', markersize=8)
# 绘制置信区间
plt.errorbar(plot_df['EventHalfYear'], plot_df['Coefficient'],
             yerr=[plot_df['Coefficient'] - plot_df['CI_Lower'],
                   plot_df['CI_Upper'] - plot_df['Coefficient']],
             fmt='none', color='#A23B72', capsize=5, capthick=1.5, elinewidth=1)

# 基准线和冲击点
plt.axvline(x=0, color='#F18F01', linestyle='--', linewidth=1.5, label='2022H1
plt.axhline(y=0, color='#C73E1D', linestyle='--', linewidth=1, label='No Effect'

# 图表美化
plt.title('Event Study (Half-Year) - Clean Energy Share', fontsize=14, pad=15)
plt.xlabel('Half-Year Relative to 2022H1 (t=0)', fontsize=12)
plt.ylabel('Treatment Effect Estimate', fontsize=12)
plt.grid(True, alpha=0.2)
plt.legend(loc='best', fontsize=10)
plt.tight_layout()
plt.show()

return model, plot_df

# ==================== 运行函数（只需修改文件路径） ====================
if __name__ == "__main__":
# 注意：Windows路径用双反斜杠或单斜杠，避免转义错误
file_path = r"D:\Dissertation 1\clean share.csv"  # r前缀表示原始字符串，避免反
model_results, plot_data = semi_annual_parallel_trends_test(file_path)

# 可选：打印系数详情
if plot_data is not None:
    print("\n📋 事件半年度系数详情：")
    print(plot_data.round(4))
```

```
✅ 成功加载数据，数据形状：(1000, 3)
✅ 筛选2019-2024数据后，剩余行数：720
✅ 处理组国家：['Bulgaria', 'Germany', 'Estonia', 'Hungary', 'Poland']，共5个

✅ 回归模型拟合成功
模型R²：0.7582

============================================================
📊 平行趋势统计检验结果（冲击前系数联合显著性）：
⚠️ 平行趋势检验失败：'float' object is not subscriptable
```



```
📋 事件半年度系数详情：
    EventHalfYear  Coefficient  CI_Lower  CI_Upper
0              -6      -2.6118   -7.4873    2.2638
1              -5      -3.4559   -8.3315    1.4196
2              -4       3.3177   -1.5579    8.1932
3              -2       1.0353   -3.8402    5.9109
4              -1       2.3814   -2.4941    7.2569
5               0       2.4451   -2.4304    7.3206
6               1       1.5977   -3.2778    6.4732
7               2       5.6794    0.8039   10.5550
8               3       4.7699   -0.1056    9.6454
9               4       5.9438    1.0682   10.8193
10              5       8.4517    3.5762   13.3272
```

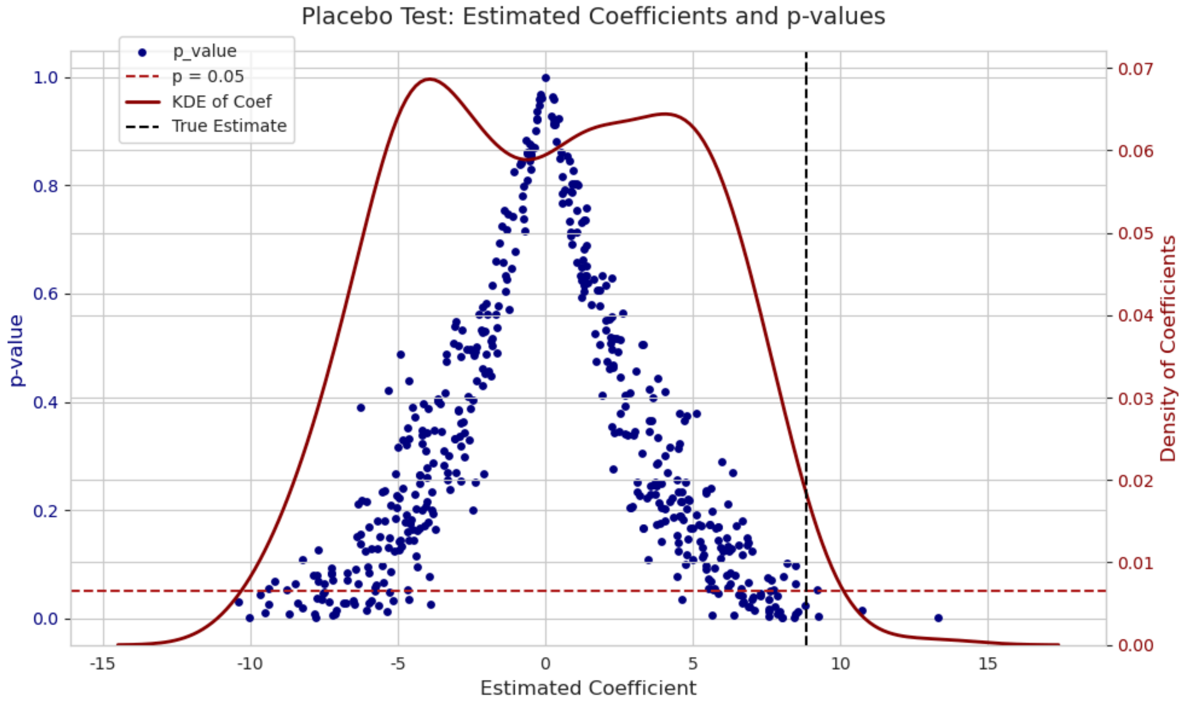
In [ ]:

```
# 8.6 随机处理组
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from linearmodels import PanelOLS
from tqdm import tqdm
import seaborn as sns

np.random.seed(2026)

# ------------------------------
# 1) 国家集合 & 真实处理组数量
# ------------------------------
countries = merged_df.index.get_level_values('country').unique()
n_treated = int(merged_df['treat'].groupby('country').first().sum())

print("Total countries:", len(countries))
print("True treated countries:", n_treated)

# ------------------------------
# 2) 当前主模型（不含核电）
# ------------------------------
true_formula = """
clean_share ~ treat_post + dep_post
+ repowerEU_index + egd_infra
+ gdp + price
+ wind_std + solar_std + precip_std
+ EntityEffects + TimeEffects
"""

true_model = PanelOLS.from_formula(true_formula, data=merged_df, drop_absorbed=True
true_res = true_model.fit(cov_type='clustered', cluster_entity=True)
```

```
true_coef = float(true_res.params.loc['treat_post'])
true_pval = float(true_res.pvalues.loc['treat_post'])

print(f"True treat_post coef: {true_coef:.4f}, p-value: {true_pval:.4f}")

# -----------------------------
# 3) 随机处理组 placebo
# -----------------------------
placebo_coefs = []
placebo_pvalues = []

for _ in tqdm(range(500), desc='Random treatment placebo (no nuclear)'):

    placebo_treated = np.random.choice(countries, size=n_treated, replace=False)

    # 不污染 merged_df
    data = merged_df.copy()

    data['placebo_treat'] = data.index.get_level_values('country').isin(placebo_trea
    data['placebo_treat_post'] = data['placebo_treat'] * data['post']

    placebo_formula = """
    clean_share ~ placebo_treat_post + dep_post
    + repowerEU_index + egd_infra
    + gdp + price
    + wind_std + solar_std + precip_std
    + EntityEffects + TimeEffects
    """

    try:
        model = PanelOLS.from_formula(placebo_formula, data=data, drop_absorbed=Tru
        result = model.fit(cov_type='clustered', cluster_entity=True)

        coef = float(result.params.loc['placebo_treat_post'])
        pval = float(result.pvalues.loc['placebo_treat_post'])

        placebo_coefs.append(coef)
        placebo_pvalues.append(pval)

    except:
        continue

placebo_coefs = np.array(placebo_coefs)
placebo_pvalues = np.array(placebo_pvalues)

print(f"Successful runs: {len(placebo_coefs)} / 500")

fig, ax1 = plt.subplots(figsize=(10, 6))

# 🔵 p-value scatter
ax1.scatter(placebo_coefs, placebo_pvalues,
            color='navy', s=15, label='p_value')

# 🔴 p=0.05 line
ax1.axhline(0.05, color='firebrick',
            linestyle='--', linewidth=1.5, label='p = 0.05')

ax1.set_xlabel('Estimated Coefficient', fontsize=12)
ax1.set_ylabel('p-value', fontsize=12, color='navy')
ax1.tick_params(axis='y', labelcolor='navy')

# 🔴 第二个y轴（density）
ax2 = ax1.twinx()
```

```
sns.kdeplot(placebo_coefs,
            ax=ax2,
            color='darkred',
            linewidth=2,
            label='KDE of Coef')

# ⚫ True estimate line
ax2.axvline(true_coef,
            color='black',
            linestyle='--',
            linewidth=1.5,
            label='True Estimate')

ax2.set_ylabel('Density of Coefficients',
               fontsize=12,
               color='darkred')

ax2.tick_params(axis='y', labelcolor='darkred')

fig.suptitle('Placebo Test: Estimated Coefficients and p-values',
             fontsize=14)

fig.legend(loc='upper left', bbox_to_anchor=(0.1, 0.95))

plt.tight_layout()
plt.savefig(r"D:\大修\placebo_random_no_nuclear.png",
            dpi=300, bbox_inches='tight')
plt.show()
```

```
Total countries: 16
True treated countries: 8
True treat_post coef: 8.8694, p-value: 0.0175
Random treatment placebo (no nuclear): 100%|██████████| 500/500 [00:42<00:00, 11.76it/s]
Successful runs: 500 / 500
```

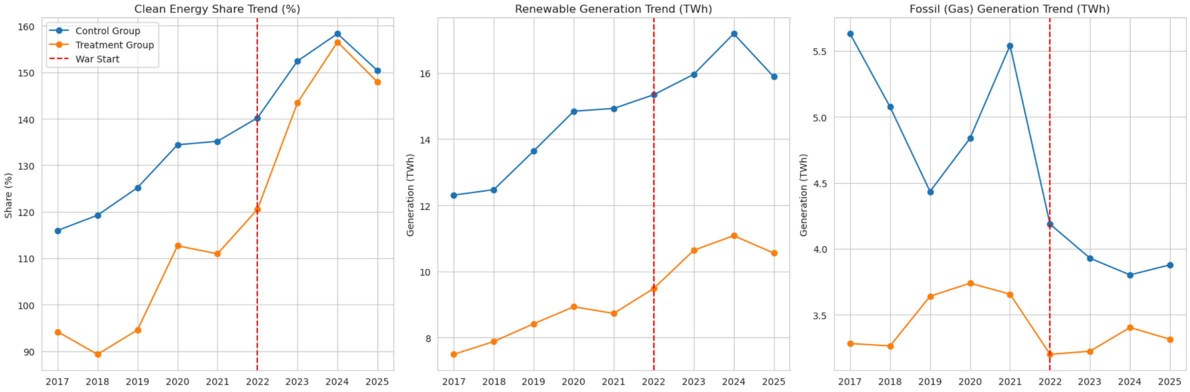


In [ ]:

```
# 8.7 更换对照组
import pandas as pd
import numpy as np
from linearmodels import PanelOLS
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
import warnings
warnings.filterwarnings('ignore')

# ================================
# 1. 核心配置：处理组/对照组（13国）
# ================================
treatment_countries = ['Bulgaria', 'Estonia', 'Germany', 'Hungary', 'Poland', 'Lithu
control_countries = [ 'Slovenia', 'Sweden',  'Turkey',  'Malta', 'Cyprus', 'Netherla
all_countries = treatment_countries + control_countries  # 14国完整列表

# ================================
# 2. 数据加载（换成14国文件）
# ================================
energy_share_df = pd.read_csv(r"D:\新对照组\清洁能源发电量占比.csv")          # 清洁

renewable_gen_df = pd.read_csv(r"D:\新对照组\清洁能源发电量.csv")              # 清
fossil_gen_df = pd.read_csv(r"D:\新对照组\天然气发电量.csv")                # 天然气

gdp_df = pd.read_csv(r"D:\新对照组\GDP.csv")                                  # GDP
electricity_price_df = pd.read_csv(r"D:\新对照组\电价.csv")                   # 电价

repowerEU_df = pd.read_csv(r"D:\新对照组\repowereu.csv")  # 年度
infra_df = pd.read_csv(r"D:\新对照组\可再生能源基础设施.csv")            # 年度

oil_dep_df = pd.read_csv(r"D:\新对照组\石油依赖.csv")                         # 年度
gas_dep_df = pd.read_csv(r"D:\新对照组\天然气依赖.csv")                       # 年度

ghi_df = pd.read_csv(r"D:\新对照组\GHI.csv")      # 年度 long
precip_df = pd.read_csv(r"D:\新对照组\降水.csv")     # 季度
windspeed_df = pd.read_csv(r"D:\新对照组\风速.csv")  # 年度 long

# ================================
# 3. 数据预处理（统一到季度）
# ================================
def preprocess_time_month(df, date_col='month', freq='Q'):
    df = df.copy()
    df['date'] = pd.to_datetime(df[date_col], errors='coerce')
    df = df.dropna(subset=['date'])
    df['quarter'] = df['date'].dt.to_period(freq)
    df['year'] = df['date'].dt.year
    return df

def preprocess_time_halfyear(df, date_col='time'):
    """
    半年 -> 季度展开：S1->Q1/Q2, S2->Q3/Q4
    说明：你原代码做了线性插值；这里保留你的逻辑（仍插值），确保结果与你之前一致。
    """
    df = df.copy()
    df[date_col] = df[date_col].astype(str)

    df['year'] = df[date_col].str.split('-').str[0].astype(int)
    df['halfyear'] = df[date_col].str.split('-').str[1].map({'S1': 1, 'S2': 2})

    # 兼容电价列名：若不是 price，自动找数值列
    if 'price' not in df.columns:
        cand = [c for c in df.columns if c.lower() in ['price', 'value', 'electrici
        if len(cand) == 0:
            cand = df.select_dtypes(include=[np.number]).columns.tolist()
        df = df.rename(columns={cand[0]: 'price'})

    quarter_map = {1: ['Q1', 'Q2'], 2: ['Q3', 'Q4']}
    rows = []
    for _, row in df.iterrows():
        if row['halfyear'] not in quarter_map:
            continue
        for q in quarter_map[row['halfyear']]:
            new_row = row.copy()
            new_row['quarter'] = pd.Period(f"{row['year']}-{q}", freq='Q')
            rows.append(new_row)

    out = pd.DataFrame(rows).sort_values(['country', 'quarter'])
    out['price'] = out.groupby('country')['price'].transform(lambda x: x.interpolat
    return out[['country', 'quarter', 'price']]

def preprocess_time_annual(df, date_col='year'):
    df = df.copy()
    df['year'] = df[date_col].astype(int)
    rows = []
    for _, row in df.iterrows():
        for q in ['Q1', 'Q2', 'Q3', 'Q4']:
            new_row = row.copy()
            new_row['quarter'] = pd.Period(f"{row['year']}-{q}", freq='Q')
            rows.append(new_row)
    out = pd.DataFrame(rows).sort_values(['country', 'quarter'])
    return out

def aggregate_to_quarter(df, group_cols=('country', 'quarter'), value_col=None, how
    out = df.groupby(list(group_cols))[value_col].agg(how).reset_index()
    return out

# ---- 3.1 月度 -> 季度（发电量/占比）
renewable_gen_df = preprocess_time_month(renewable_gen_df, date_col='month')
fossil_gen_df = preprocess_time_month(fossil_gen_df, date_col='month')
nuclear_share_df = preprocess_time_month(nuclear_share_df, date_col='month')
energy_share_df = preprocess_time_month(energy_share_df, date_col='month')

# 兼容列名
# 发电量列默认 Gwh
if 'Gwh' not in renewable_gen_df.columns:
    cand = [c for c in renewable_gen_df.columns if 'gwh' in c.lower()]
    renewable_gen_df = renewable_gen_df.rename(columns={cand[0]: 'Gwh'})
if 'Gwh' not in fossil_gen_df.columns:
    cand = [c for c in fossil_gen_df.columns if 'gwh' in c.lower()]
    fossil_gen_df = fossil_gen_df.rename(columns={cand[0]: 'Gwh'})

# 核电占比列名
if 'nuclear_share' not in nuclear_share_df.columns:
    cand = [c for c in nuclear_share_df.columns if 'nuclear' in c.lower() or 'shar
    if len(cand) == 0:
        cand = nuclear_share_df.select_dtypes(include=[np.number]).columns.tolist()
    nuclear_share_df = nuclear_share_df.rename(columns={cand[0]: 'nuclear_share'})

# clean_share列名
if 'clean_share' not in energy_share_df.columns:
    cand = [c for c in energy_share_df.columns if 'clean' in c.lower() or 'share'
    if len(cand) == 0:
        cand = energy_share_df.select_dtypes(include=[np.number]).columns.tolist()
    energy_share_df = energy_share_df.rename(columns={cand[0]: 'clean_share'})

# 月度聚合到季度：发电量 sum，占比 mean


renewable_gen_quarterly = aggregate_to_quarter(renewable_gen_df, value_col='Gwh').re
fossil_gen_quarterly = aggregate_to_quarter(fossil_gen_df, value_col='Gwh').rename(c
nuclear_share_quarterly = aggregate_to_quarter(nuclear_share_df, value_col='nuclear_
energy_share_quarterly = aggregate_to_quarter(energy_share_df, value_col='clean_shar
# ---- 3.2 GDP（季度）
gdp_df = gdp_df.copy()
gdp_df['quarter'] = pd.PeriodIndex(gdp_df['quarter'].astype(str), freq='Q')

# ---- 3.3 电价（半年 -> 季度）
electricity_price_df = preprocess_time_halfyear(electricity_price_df, date_col='time

# ---- 3.4 年度 -> 季度（依赖度 / 天气 / 政策 / 基建）
oil_dep_expanded = preprocess_time_annual(oil_dep_df, date_col='year')
gas_dep_expanded = preprocess_time_annual(gas_dep_df, date_col='year')

# 兼容依赖度百分比列名：percentage
if 'percentage' not in oil_dep_expanded.columns:
    cand = [c for c in oil_dep_expanded.columns if c.lower() in ['percentage', 'per
    if len(cand) == 0:
        cand = oil_dep_expanded.select_dtypes(include=[np.number]).columns.tolist()
    oil_dep_expanded = oil_dep_expanded.rename(columns={cand[0]: 'percentage'})

if 'percentage' not in gas_dep_expanded.columns:
    cand = [c for c in gas_dep_expanded.columns if c.lower() in ['percentage', 'per
    if len(cand) == 0:
        cand = gas_dep_expanded.select_dtypes(include=[np.number]).columns.tolist()
    gas_dep_expanded = gas_dep_expanded.rename(columns={cand[0]: 'percentage'})

dependency_df = pd.merge(
    oil_dep_expanded[['country', 'quarter', 'percentage']].rename(columns={'percenta
    gas_dep_expanded[['country', 'quarter', 'percentage']].rename(columns={'percenta
    on=['country', 'quarter'], how='inner'
)

# 天气
ghi_expanded = preprocess_time_annual(ghi_df, date_col='year')
if 'GHI' not in ghi_expanded.columns:
    cand = [c for c in ghi_expanded.columns if 'ghi' in c.lower()]
    if len(cand) == 0:
        cand = ghi_expanded.select_dtypes(include=[np.number]).columns.tolist()
    ghi_expanded = ghi_expanded.rename(columns={cand[0]: 'GHI'})
ghi_expanded = ghi_expanded.rename(columns={'GHI': 'solar_irradiance'})

windspeed_expanded = preprocess_time_annual(windspeed_df, date_col='year')
if 'WindSpeed' in windspeed_expanded.columns and 'wind_speed' not in windspeed_expa
    windspeed_expanded = windspeed_expanded.rename(columns={'WindSpeed': 'wind_speed
if 'wind_speed' not in windspeed_expanded.columns:
    cand = [c for c in windspeed_expanded.columns if 'wind' in c.lower()]
    if len(cand) == 0:
        cand = windspeed_expanded.select_dtypes(include=[np.number]).columns.tolist
    windspeed_expanded = windspeed_expanded.rename(columns={cand[0]: 'wind_speed'})

precip_expanded = precip_df.copy()
precip_expanded['quarter'] = pd.PeriodIndex(precip_expanded['quarter'].astype(str),
if 'Precipitation' in precip_expanded.columns and 'precipitation' not in precip_exp
    precip_expanded = precip_expanded.rename(columns={'Precipitation': 'precipitatio
if 'precipitation' not in precip_expanded.columns:
    cand = [c for c in precip_expanded.columns if 'precip' in c.lower()]
    if len(cand) == 0:
        cand = precip_expanded.select_dtypes(include=[np.number]).columns.tolist()
    precip_expanded = precip_expanded.rename(columns={cand[0]: 'precipitation'})

weather_df = pd.merge(
    ghi_expanded[['country', 'quarter', 'solar_irradiance']],
    precip_expanded[['country', 'quarter', 'precipitation']],
    on=['country', 'quarter'], how='inner'
)
weather_df = pd.merge(
    weather_df,
    windspeed_expanded[['country', 'quarter', 'wind_speed']],
    on=['country', 'quarter'], how='inner'
)

repowerEU_expanded = preprocess_time_annual(repowerEU_df, date_col='year')
# 兼容 repowereu 列名
if 'repowereu' not in repowerEU_expanded.columns:
    cand = [c for c in repowerEU_expanded.columns if 'repower' in c.lower() or 'in
    if len(cand) == 0:
        cand = repowerEU_expanded.select_dtypes(include=[np.number]).columns.tolist
    repowerEU_expanded = repowerEU_expanded.rename(columns={cand[0]: 'repowereu'})

infra_expanded = preprocess_time_annual(infra_df, date_col='year')
if 'renewable_infra' not in infra_expanded.columns:
    if 'value' in infra_expanded.columns:
        infra_expanded = infra_expanded.rename(columns={'value': 'renewable_infra'})
    else:
        cand = [c for c in infra_expanded.columns if 'infra' in c.lower() or 'cap'
        if len(cand) == 0:
            cand = infra_expanded.select_dtypes(include=[np.number]).columns.tolist
        infra_expanded = infra_expanded.rename(columns={cand[0]: 'renewable_infra'})

# ================================
# 4. 合并所有数据集（核心）
# ================================
merged_df = pd.merge(energy_share_quarterly, gdp_df[['country', 'quarter', 'gdp']],
merged_df = pd.merge(merged_df, electricity_price_df, on=['country', 'quarter'], how
merged_df = pd.merge(merged_df, nuclear_share_quarterly, on=['country', 'quarter'],
merged_df = pd.merge(merged_df, renewable_gen_quarterly, on=['country', 'quarter'],
merged_df = pd.merge(merged_df, fossil_gen_quarterly, on=['country', 'quarter'], how
merged_df = pd.merge(merged_df, dependency_df, on=['country', 'quarter'], how='left'
merged_df = pd.merge(merged_df, weather_df, on=['country', 'quarter'], how='left')
merged_df = pd.merge(merged_df, repowerEU_expanded[['country', 'quarter', 'repowereu
merged_df = pd.merge(merged_df, infra_expanded[['country', 'quarter', 'renewable_inf

# 仅保留13国
merged_df = merged_df[merged_df['country'].isin(all_countries)].copy()

# 缺失处理：前向填充 + 线性插值（保留你原逻辑）
merged_df = merged_df.sort_values(['country', 'quarter'])
for col in ['gdp', 'price', 'nuclear_share', 'renewable_gen', 'fossil_gen', 'oil_dep
            'solar_irradiance', 'precipitation', 'wind_speed', 'repowereu', 'renewab
    if col in merged_df.columns:
        merged_df[col] = merged_df.groupby('country')[col].ffill().interpolate(metho

# ================================
# 5. 核心变量构建
# ================================
merged_df['treat'] = merged_df['country'].isin(treatment_countries).astype(int)
merged_df['post'] = (merged_df['quarter'].dt.year >= 2022).astype(int)
merged_df['treat_post'] = merged_df['treat'] * merged_df['post']

merged_df['russia_dep_continuous'] = (merged_df['oil_dep'] * 0.3 + merged_df['gas_de
merged_df['dep_post'] = merged_df['russia_dep_continuous'] * merged_df['post']

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

# REPowerEU 标准化
merged_df['repowereu_std'] = scaler.fit_transform(merged_df[['repowereu']])
merged_df['repowerEU_index'] = merged_df['repowereu_std']

# EGD × 基建存量
```

```
merged_df['egd'] = (merged_df['quarter'].dt.year >= 2020).astype(int)
merged_df['egd_infra'] = merged_df['egd'] * merged_df['renewable_infra']

# 天气标准化
merged_df[['wind_std', 'solar_std', 'precip_std']] = scaler.fit_transform(
    merged_df[['wind_speed', 'solar_irradiance', 'precipitation']]
)

# 效应分解变量

merged_df['total_demand'] = merged_df['renewable_gen'] + merged_df['fossil_gen']
merged_df['renewable_gen_share'] = merged_df['renewable_gen'] / merged_df['total_dem
merged_df['fossil_gen_share'] = merged_df['fossil_gen'] / merged_df['total_demand']

# 面板索引
merged_df['quarter_timestamp'] = merged_df['quarter'].dt.to_timestamp()
merged_df = merged_df.set_index(['country', 'quarter_timestamp'])
# 替代原有的 dropna()，仅删除核心因变量缺失的样本，保留控制变量缺失的样本
merged_df = merged_df.dropna(subset=['clean_share'])  # 只删因变量缺失
# 对控制变量的缺失值，用国家均值填充（而非删除）
for col in ['gdp', 'price', 'repowereu', 'egd_infra', 'wind_std']:
    if col in merged_df.columns:
        merged_df[col] = merged_df.groupby('country')[col].fillna(lambda x: x.mean(

print("最终数据维度：", merged_df.shape)
print("变量列表：", merged_df.columns.tolist())
print("处理组国家样本数：", merged_df[merged_df['treat'] == 1].index.get_level_value
print("对照组国家样本数：", merged_df[merged_df['treat'] == 0].index.get_level_value

# ================================
# 6. 描述性统计与趋势图
# ================================
desc_stats = merged_df[['clean_share', 'renewable_gen', 'fossil_gen',
                        'russia_dep_continuous', 'repowerEU_index', 'wind_std']].des
print("\n核心变量描述性统计：")
print(desc_stats)

trend_data = merged_df.reset_index().copy()
trend_data['year'] = trend_data['quarter_timestamp'].dt.year
yearly_avg = trend_data.groupby(['year', 'treat'])[['clean_share', 'renewable_gen',

fig, axes = plt.subplots(1, 3, figsize=(18, 6))

for treat in [0, 1]:
    subset = yearly_avg[yearly_avg['treat'] == treat]
    label = 'Treatment Group' if treat == 1 else 'Control Group'
    axes[0].plot(subset['year'], subset['clean_share'], label=label, marker='o')
axes[0].axvline(x=2022, color='r', linestyle='--', label='War Start')
axes[0].set_title('Clean Energy Share Trend (%)')
axes[0].set_ylabel('Share (%)')
axes[0].legend()

for treat in [0, 1]:
    subset = yearly_avg[yearly_avg['treat'] == treat]
    label = 'Treatment Group' if treat == 1 else 'Control Group'
    axes[1].plot(subset['year'], subset['renewable_gen'] / 1000, label=label, marker
axes[1].axvline(x=2022, color='r', linestyle='--')
axes[1].set_title('Renewable Generation Trend (TWh)')
axes[1].set_ylabel('Generation (TWh)')

for treat in [0, 1]:
    subset = yearly_avg[yearly_avg['treat'] == treat]
    label = 'Treatment Group' if treat == 1 else 'Control Group'
    axes[2].plot(subset['year'], subset['fossil_gen'] / 1000, label=label, marker='o
axes[2].axvline(x=2022, color='r', linestyle='--')
axes[2].set_title('Fossil (Gas) Generation Trend (TWh)')
axes[2].set_ylabel('Generation (TWh)')

plt.tight_layout()
plt.savefig(r"D:\大修\trend_analysis_13countries.png", dpi=300, bbox_inches='tight')
plt.show()

# ================================
# 7. 基线DID回归（不含核电控制变量）
# ================================
print("\n=== 基线DID回归结果（clean_share，不含核电控制）===")
baseline_model = PanelOLS.from_formula(
    """
    clean_share ~ treat_post + dep_post + repowerEU_index
    + gdp + price + wind_std + solar_std
    + EntityEffects + TimeEffects
    """,
    data=merged_df,
    drop_absorbed=True
)
baseline_results = baseline_model.fit(cov_type='clustered', cluster_entity=True, cl
print(baseline_results.summary)
```

```
最终数据维度： (576, 29)
变量列表： ['quarter', 'clean_share', 'gdp', 'price', 'nuclear_share', 'renewable_ge
n', 'fossil_gen', 'oil_dep', 'gas_dep', 'solar_irradiance', 'precipitation', 'wind_s
peed', 'repowereu', 'renewable_infra', 'treat', 'post', 'treat_post', 'russia_dep_co
ntinuous', 'dep_post', 'repowereu_std', 'repowerEU_index', 'egd', 'egd_infra', 'wind
_std', 'solar_std', 'precip_std', 'total_demand', 'renewable_gen_share', 'fossil_gen
_share']
处理组国家样本数： 8
对照组国家样本数： 8

核心变量描述性统计：
       clean_share  renewable_gen   fossil_gen  russia_dep_continuous  \
count   576.000000     576.000000   576.000000             576.000000   
mean    127.839424   11989.618052  4001.343872               0.435354   
std      78.038530   16091.307511  6695.347326               0.322402   
min       3.000000       3.000000     3.681000               0.000000   
25%      60.093750     780.149750   274.808250               0.101970   
50%     118.261500    3752.000000   683.078000               0.439645   
75%     184.932500   17199.378500  3229.248500               0.724110   
max     298.353000   68879.740000 32938.908000               0.912390   

       repowerEU_index      wind_std  
count     5.560000e+02  5.760000e+02  
mean      5.111818e-17  1.973730e-16  
std       1.000900e+00  1.000869e+00  
min      -7.667103e-01 -1.820763e+00  
25%      -6.998639e-01 -6.652324e-01  
50%      -2.920816e-01  1.436388e-01  
75%       3.407890e-01  5.769628e-01  
max       4.377003e+00  1.876934e+00  
```



```
=== 基线DID回归结果（clean_share，不含核电控制）===
                          PanelOLS Estimation Summary
================================================================================
Dep. Variable:            clean_share   R-squared:                        0.1278
Estimator:                   PanelOLS   R-squared (Between):             -1.9608
No. Observations:                 556   R-squared (Within):               0.3251
Date:                Sun, Feb 22 2026   R-squared (Overall):             -1.9821
Time:                        22:59:52   Log-likelihood                   -2491.2
Cov. Estimator:             Clustered
                                        F-statistic:                      10.425
Entities:                          16   P-value                           0.0000
Avg Obs:                       34.750   Distribution:                   F(7,498)
Min Obs:                       16.000
Max Obs:                       36.000   F-statistic (robust):             1.8765
                                        P-value                           0.0714
Time periods:                      36   Distribution:                   F(7,498)
Avg Obs:                       15.444
Min Obs:                       15.000
Max Obs:                       16.000

                                Parameter Estimates
==================================================================================
                Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
----------------------------------------------------------------------------------
treat_post        -40.551     23.605    -1.7179     0.0864     -86.929      5.8263
dep_post           103.52     43.176     2.3976     0.0169      18.690      188.35
repowerEU_index    8.2218     4.9790     1.6513     0.0993     -1.5606      18.004
gdp            -9.022e-05     0.0001    -0.8459     0.3980     -0.0003      0.0001
price              37.122     47.476     0.7819     0.4346     -56.156      130.40
wind_std          -30.608     15.682    -1.9519     0.0515     -61.418      0.2020
solar_std          150.73     195.64     0.7705     0.4414     -233.65      535.12
==================================================================================

F-test for Poolability: 61.016
P-value: 0.0000
Distribution: F(50,498)

Included effects: Entity, Time
```

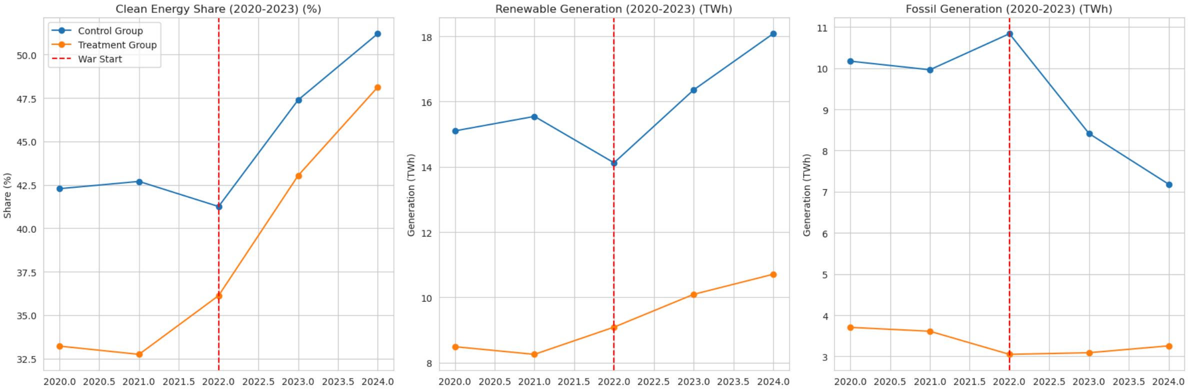
In [ ]:

```
# =================================
# 8.8 稳健性检验：限制时间窗口（2020-2023）
# =================================
#  筛选2020-2023年的观测值
# 先重置索引，方便筛选年份（也可直接通过索引层级筛选）
merged_df_reset = merged_df.reset_index()
# 筛选条件：年份在2020到2023之间（包含边界）
merged_df_2020_2023 = merged_df_reset[
    (merged_df_reset['quarter_timestamp'].dt.year >= 2020) &
    (merged_df_reset['quarter_timestamp'].dt.year <= 2024)
].copy()

# 重新设置面板数据索引（PanelOLS要求）
```

```
merged_df_2020_2023 = merged_df_2020_2023.set_index(['country', 'quarter_timestamp']
# 删除筛选后可能产生的缺失值（保持与基线回归一致的处理）
merged_df_2020_2023 = merged_df_2020_2023.dropna()

#  检查筛选后的数据基本信息（验证数据有效性）
print("\n=== 稳健性检验：2020-2023时间窗口数据信息 ===")
print(f"筛选后数据维度：{merged_df_2020_2023.shape}")
print(f"处理组国家样本数：{merged_df_2020_2023[merged_df_2020_2023['treat']==1].inde
print(f"对照组国家样本数：{merged_df_2020_2023[merged_df_2020_2023['treat']==0].inde
print(f"时间范围：{merged_df_2020_2023.index.get_level_values('quarter_timestamp').m

#  运行时间窗口限制后的基线DID回归（与原公式完全一致）
print("\n=== 稳健性检验：2020-2023时间窗口DID回归结果 ===")
robust_model_2020_2023 = PanelOLS.from_formula(
    '''clean_share ~ treat_post + dep_post + repowerEU_index  +
    gdp + price + wind_std + solar_std  +
    EntityEffects + TimeEffects''',
    data=merged_df_2020_2023
)
robust_results_2020_2023 = robust_model_2020_2023.fit(
    cov_type='clustered',
    cluster_entity=True,
    cluster_time=True
)
print(robust_results_2020_2023.summary)

#  核心系数对比（方便快速判断稳健性）
print("\n=== 核心系数对比（全样本 vs 2020-2023窗口）===")
# 提取全样本的核心系数
baseline_coef = {
    'treat_post': baseline_results.params['treat_post'],
    'dep_post': baseline_results.params['dep_post'],
    'treat_post_pval': baseline_results.pvalues['treat_post'],
    'dep_post_pval': baseline_results.pvalues['dep_post']
}
# 提取2020-2023窗口的核心系数
robust_coef = {
    'treat_post': robust_results_2020_2023.params['treat_post'],
    'dep_post': robust_results_2020_2023.params['dep_post'],
    'treat_post_pval': robust_results_2020_2023.pvalues['treat_post'],
    'dep_post_pval': robust_results_2020_2023.pvalues['dep_post']
}
# 输出对比表
comparison_df = pd.DataFrame({
    '全样本系数': [baseline_coef['treat_post'], baseline_coef['dep_post']],
    '全样本P值': [baseline_coef['treat_post_pval'], baseline_coef['dep_post_pval']],
    '2020-2023系数': [robust_coef['treat_post'], robust_coef['dep_post']],
    '2020-2023 P值': [robust_coef['treat_post_pval'], robust_coef['dep_post_pval']]
}, index=['treat_post (DID核心项)', 'dep_post (依赖度交互项)'])
print(comparison_df.round(4))

#  可选：可视化2020-2023窗口内的趋势（与全样本趋势对比）
trend_data_2020_2023 = merged_df_2020_2023.reset_index()
trend_data_2020_2023['year'] = trend_data_2020_2023['quarter_timestamp'].dt.year
yearly_avg_2020_2023 = trend_data_2020_2023.groupby(['year', 'treat'])[['clean_share

fig, axes = plt.subplots(1, 3, figsize=(18, 6))
# 清洁能源占比趋势（2020-2023）
for treat in [0, 1]:
    subset = yearly_avg_2020_2023[yearly_avg_2020_2023['treat'] == treat]
    label = 'Treatment Group' if treat == 1 else 'Control Group'
    axes[0].plot(subset['year'], subset['clean_share'], label=label, marker='o')
axes[0].axvline(x=2022, color='r', linestyle='--', label='War Start')
axes[0].set_title('Clean Energy Share (2020-2023) (%)')
```

```
axes[0].set_ylabel('Share (%)')
axes[0].legend()

# 可再生绝对发电量趋势（2020-2023）
for treat in [0, 1]:
    subset = yearly_avg_2020_2023[yearly_avg_2020_2023['treat'] == treat]
    label = 'Treatment Group' if treat == 1 else 'Control Group'
    axes[1].plot(subset['year'], subset['renewable_gen']/1000, label=label, marker='
axes[1].axvline(x=2022, color='r', linestyle='--')
axes[1].set_title('Renewable Generation (2020-2023) (TWh)')
axes[1].set_ylabel('Generation (TWh)')

# 化石绝对发电量趋势（2020-2023）
for treat in [0, 1]:
    subset = yearly_avg_2020_2023[yearly_avg_2020_2023['treat'] == treat]
    label = 'Treatment Group' if treat == 1 else 'Control Group'
    axes[2].plot(subset['year'], subset['fossil_gen']/1000, label=label, marker='o')
axes[2].axvline(x=2022, color='r', linestyle='--')
axes[2].set_title('Fossil Generation (2020-2023) (TWh)')
axes[2].set_ylabel('Generation (TWh)')

plt.tight_layout()
plt.savefig(r"D:\Dissertation 1\trend_analysis_2020_2023.png", dpi=300, bbox_inches=
plt.show()
```

```
=== 稳健性检验：2020-2023时间窗口数据信息 ===
筛选后数据维度：(312, 29)
处理组国家样本数：8
对照组国家样本数：8
时间范围：2020-01-01 00:00:00 至 2024-10-01 00:00:00

=== 稳健性检验：2020-2023时间窗口DID回归结果 ===
                          PanelOLS Estimation Summary
================================================================================
Dep. Variable:            clean_share   R-squared:                        0.0526
Estimator:                   PanelOLS   R-squared (Between):             -0.0322
No. Observations:                 312   R-squared (Within):              -0.0116
Date:                Sun, Feb 22 2026   R-squared (Overall):             -0.0268
Time:                        23:13:50   Log-likelihood                   -998.88
Cov. Estimator:             Clustered
                                        F-statistic:                      2.1409
Entities:                          16   P-value                           0.0398
Avg Obs:                       19.500   Distribution:                   F(7,270)
Min Obs:                       12.000
Max Obs:                       20.000   F-statistic (robust):            -19.835
                                        P-value                           1.0000
Time periods:                      20   Distribution:                   F(7,270)
Avg Obs:                       15.600
Min Obs:                       15.000
Max Obs:                       16.000

                                Parameter Estimates
===================================================================================
                 Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
-----------------------------------------------------------------------------------
treat_post          8.5038     4.1336     2.0572     0.0406      0.3657      16.642
dep_post           -8.4904     7.3481    -1.1555     0.2489     -22.957      5.9764
repowerEU_index    -2.7232     1.0463    -2.6027     0.0098     -4.7832     -0.6633
gdp             -2.248e-07  1.995e-05    -0.0113     0.9910   -3.95e-05   3.905e-05
price              -3.7224     15.881    -0.2344     0.8149     -34.989      27.545
wind_std           -0.1839     4.0056    -0.0459     0.9634     -8.0700      7.7022
solar_std           5.3265     4.4354     1.2009     0.2308     -3.4058      14.059
===================================================================================

F-test for Poolability: 58.661
P-value: 0.0000
Distribution: F(34,270)

Included effects: Entity, Time

=== 核心系数对比（全样本 vs 2020-2023窗口）===
                        全样本系数   全样本P值   2020-2023系数   2020-2023 P值
treat_post（DID核心项）  -40.5513   0.0864        8.5038          0.0406
dep_post（依赖度交互项）    103.5199   0.0169       -8.4904          0.2489
```


Clean Energy Share (2020-2023) (%)
Control Group
Treatment Group
War Start
Share (%)
Renewable Generation (2020-2023) (TWh)
Generation (TWh)
Fossil Generation (2020-2023) (TWh)
Generation (TWh)