# analysis9-2021

July 2, 2025

## 1 Colorado Spills

### 1.1 OLS, NBR, Interrupted Time Series Analysis

### 1.2 Just top 3 Counties

#### 1.2.1 Author: David P. Adams, PhD

**Date: 2025-07-02**

---

## 2 Setup

[1]:
```
import pandas as pd
from sqlalchemy import create_engine
import geopandas as gpd
from dotenv import load_dotenv
load_dotenv()

import os

# Database connection details from zshrc environment variables
#  use .env file for local development


db_name = 'colorado_spills'
user = os.getenv('DB_USER')
password = os.getenv('DB_PASSWORD')
host = os.getenv('DB_HOST')
port = os.getenv('DB_PORT')


# Create an engine to connect to the PostgreSQL database
engine = create_engine(f'postgresql+psycopg2://{user}:{password}@{host}:{port}/
 ↪{db_name}')

# Read in the spills_with_demographics_geog as spills
spills = pd.read_sql_table('spills_with_demographics_geog', engine)
```

```
/home/dadams/miniconda3/envs/spatial_env2/lib/python3.10/site-
packages/pandas/io/sql.py:1725: SAWarning: Did not recognize type 'geometry' of
column 'geometry'
  self.meta.reflect(bind=self.con, only=[table_name], views=True)
```

[2]:
```
# use longitude and latitude to create a GeoDataFrame from spills data
spills['geometry'] = gpd.points_from_xy(spills['Longitude'], spills['Latitude'])
```

[3]:
```
spills_gdf = gpd.GeoDataFrame(spills, crs='EPSG:4326')
```

[4]:
```
# Ensure date columns are in datetime format
spills_gdf['Date of Discovery'] = pd.to_datetime(spills_gdf['Date of␣
 ↪Discovery'])
spills_gdf['Initial Report Date'] = pd.to_datetime(spills_gdf['Initial Report␣
 ↪Date'])
```

[5]:
```
# drop 2024 data for date of discovery and initial report date
spills_gdf = spills_gdf[spills_gdf['Date of Discovery'].dt.year < 2024]
spills_gdf = spills_gdf[spills_gdf['Initial Report Date'].dt.year < 2024]
```

[6]:
```
# Separate Historical vs. Recent Spills
historical_spills = spills_gdf[spills_gdf['Spill Type'] == 'Historical']
recent_spills = spills_gdf[spills_gdf['Spill Type'] == 'Recent']
```

[8]:
```
# only use the top 3 counties by number of spills
top_counties = spills_gdf['county'].value_counts().nlargest(3).index.tolist()
# filter spills_gdf to only include top counties
spills_gdf = spills_gdf[spills_gdf['county'].isin(top_counties)]
# filter historical spills to only include top counties
historical_spills = historical_spills[historical_spills['county'].
 ↪isin(top_counties)]
# filter recent spills to only include top counties
recent_spills = recent_spills[recent_spills['county'].isin(top_counties)]
```

[9]:
```
import statsmodels.formula.api as smf

# Create 'Period' column
spills_gdf['Period'] = spills_gdf['Report Year'].apply(lambda x: 'Before 2020'␣
 ↪if x < 2020 else '2020 and After')
```

```
/home/dadams/miniconda3/envs/spatial_env2/lib/python3.10/site-
packages/geopandas/geodataframe.py:1819: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  super().__setitem__(key, value)
```

[10]:
```
spills_gdf = spills_gdf.rename(columns={
    'Report Delay (Days)': 'report_delay',
    'Spill Type': 'spill_type'
})
```

---

## 3 OLS Regression

[11]:
```
import statsmodels.formula.api as smf

model = smf.ols("report_delay ~ C(spill_type) * C(Period)", data=spills_gdf).
  ↪fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           report_delay   R-squared:                       0.011
Model:                            OLS   Adj. R-squared:                  0.011
Method:                 Least Squares   F-statistic:                     43.65
Date:                Wed, 02 Jul 2025   Prob (F-statistic):           4.78e-28
Time:                        12:42:27   Log-Likelihood:                -73126.
No. Observations:               12042   AIC:                         1.463e+05
Df Residuals:                   12038   BIC:                         1.463e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
==================================
                                                    coef    std err
t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
----------------------------------
Intercept                                          3.6818      2.207
1.668      0.095      -0.644       8.008
C(spill_type)[T.Recent]                           -1.1410      2.864
-0.398      0.690      -6.755       4.473
C(Period)[T.Before 2020]                          27.7277      3.105
8.930      0.000      21.641      33.814
C(spill_type)[T.Recent]:C(Period)[T.Before 2020]  -25.6845      3.952
-6.498      0.000     -33.432     -17.937
==============================================================================
Omnibus:                    29347.589   Durbin-Watson:                   1.919
Prob(Omnibus):                  0.000   Jarque-Bera (JB):        463915680.878
Skew:                          25.574   Prob(JB):                         0.00
Kurtosis:                     963.197   Cond. No.                         8.10
==============================================================================
```

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

---

## 4 Negative Binomial Regression

[12]:
```python
import statsmodels.formula.api as smf
import statsmodels.api as sm

# Make sure spill_type and Period are clean, and report_delay is numeric
spills_gdf = spills_gdf.rename(columns={
    'Report Delay (Days)': 'report_delay',
    'Spill Type': 'spill_type'
})

# Fit the Negative Binomial model with interaction
nb_model = smf.glm(
    formula="report_delay ~ C(spill_type) * C(Period)",
    data=spills_gdf,
    family=sm.families.NegativeBinomial()
).fit()

# View the results
print(nb_model.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:           report_delay   No. Observations:                12042
Model:                            GLM   Df Residuals:                    12038
Model Family:        NegativeBinomial   Df Model:                            3
Link Function:                    Log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -33697.
Date:                Wed, 02 Jul 2025   Deviance:                       38539.
Time:                        12:43:21   Pearson chi2:                 1.03e+06
No. Iterations:                    13   Pseudo R-squ. (CS):             0.5954
Covariance Type:            nonrobust
==============================================================================
                                                 coef    std err
z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
------------------------------------
Intercept                                     1.3034      0.024
54.986      0.000       1.257       1.350
C(spill_type)[T.Recent]                      -0.3709      0.031
-11.830      0.000      -0.432      -0.309
```

```
C(Period)[T.Before 2020]                                 2.1437      0.032
67.500       0.000        2.081        2.206
C(spill_type)[T.Recent]:C(Period)[T.Before 2020]    -1.5536      0.042
-37.432       0.000       -1.635       -1.472
==============================================================================
==================================
```

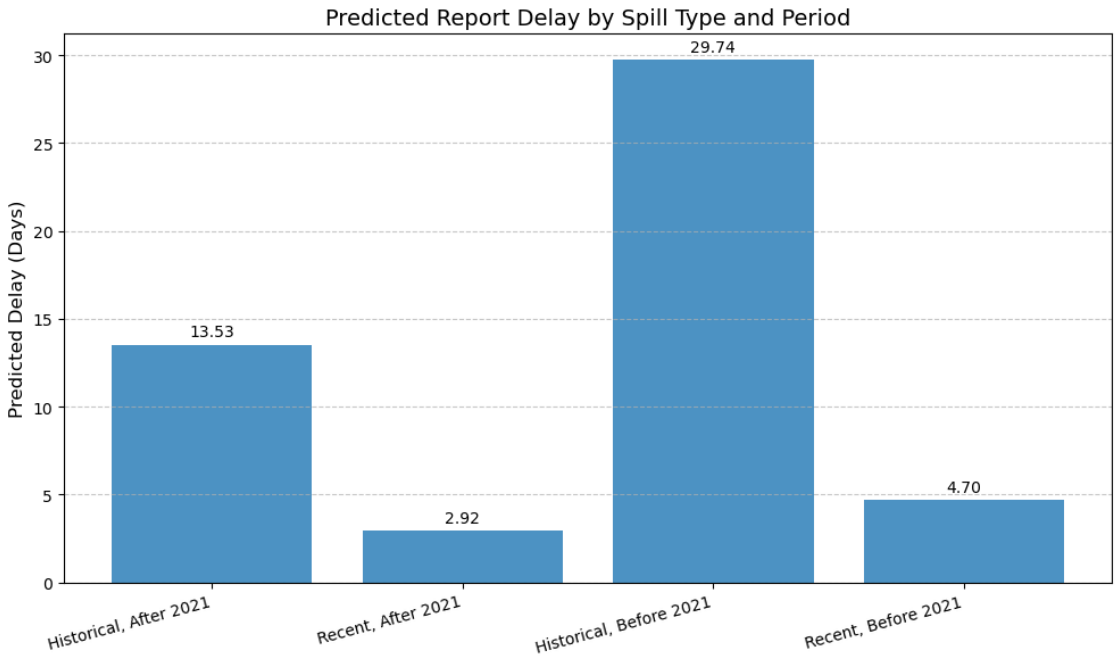
```
/home/dadams/miniconda3/envs/spatial_env2/lib/python3.10/site-
packages/statsmodels/genmod/families/family.py:1367: ValueWarning: Negative
binomial dispersion parameter alpha not set. Using default value alpha=1.0.
  warnings.warn("Negative binomial dispersion parameter alpha not "
```

[13]:
```
import numpy as np
np.exp(nb_model.params)
```

[13]:
```
Intercept                                           3.681838
C(spill_type)[T.Recent]                             0.690093
C(Period)[T.Before 2020]                            8.530935
C(spill_type)[T.Recent]:C(Period)[T.Before 2020]    0.211483
dtype: float64
```

[14]:
```
import matplotlib.pyplot as plt
import pandas as pd

# Predicted delays by group
group_labels = [
    "Historical, After 2021",
    "Recent, After 2021",
    "Historical, Before 2021",
    "Recent, Before 2021"
]

# From your exponentiated results
predicted_delays = [
    13.53,                          # Intercept only
    13.53 * 0.216,                  # Intercept * Recent
    13.53 * 2.198,                  # Intercept * Before 2021
    13.53 * 0.216 * 2.198 * 0.732  # Intercept * Recent * Before 2021 *␣
 ↪Interaction
]

# Create DataFrame for plotting
df_plot = pd.DataFrame({
    "Group": group_labels,
    "Predicted Delay (Days)": predicted_delays
})

# Plot
```

```
plt.figure(figsize=(10, 6))
bars = plt.bar(df_plot["Group"], df_plot["Predicted Delay (Days)"], alpha=0.8)
plt.title("Predicted Report Delay by Spill Type and Period", fontsize=14)
plt.ylabel("Predicted Delay (Days)", fontsize=12)
plt.xticks(rotation=15, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()

# Annotate bars
for bar in bars:
    height = bar.get_height()
    plt.annotate(f"{height:.2f}",
                 xy=(bar.get_x() + bar.get_width() / 2, height),
                 xytext=(0, 3),  # 3 points vertical offset
                 textcoords="offset points",
                 ha='center', va='bottom')

plt.show()
```



## 5 Interrupted Time Series

[15]:
```
import pandas as pd
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt

# 1. Convert to monthly time series
spills_gdf['report_month'] = spills_gdf['Initial Report Date'].dt.to_period('M')
monthly = spills_gdf.groupby('report_month')['report_delay'].mean().
 ↪reset_index()
monthly['report_month'] = monthly['report_month'].dt.to_timestamp()

# 2. Create ITS variables
monthly['time'] = (monthly['report_month'] - monthly['report_month'].min()).dt.
 ↪days // 30
monthly['post_2021'] = (monthly['report_month'] >= pd.Timestamp('2021-01-01')).
 ↪astype(int)
monthly['time_post'] = monthly['time'] * monthly['post_2021']

# 3. Run the ITS model
its_model = smf.ols("report_delay ~ time + post_2021 + time_post",␣
 ↪data=monthly).fit()
print(its_model.summary())

# 4. Predict and plot
monthly['predicted'] = its_model.predict(monthly)

plt.figure(figsize=(12, 6))
plt.plot(monthly['report_month'], monthly['report_delay'], marker='o',␣
 ↪label='Observed', alpha=0.6)
plt.plot(monthly['report_month'], monthly['predicted'], linestyle='--',␣
 ↪color='red', label='Predicted (ITS)')
plt.axvline(x=pd.Timestamp('2021-01-01'), color='black', linestyle='--',␣
 ↪label='Policy Change (2021)')
plt.title('Interrupted Time Series: Monthly Report Delay')
plt.xlabel('Month')
plt.ylabel('Mean Report Delay (Days)')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

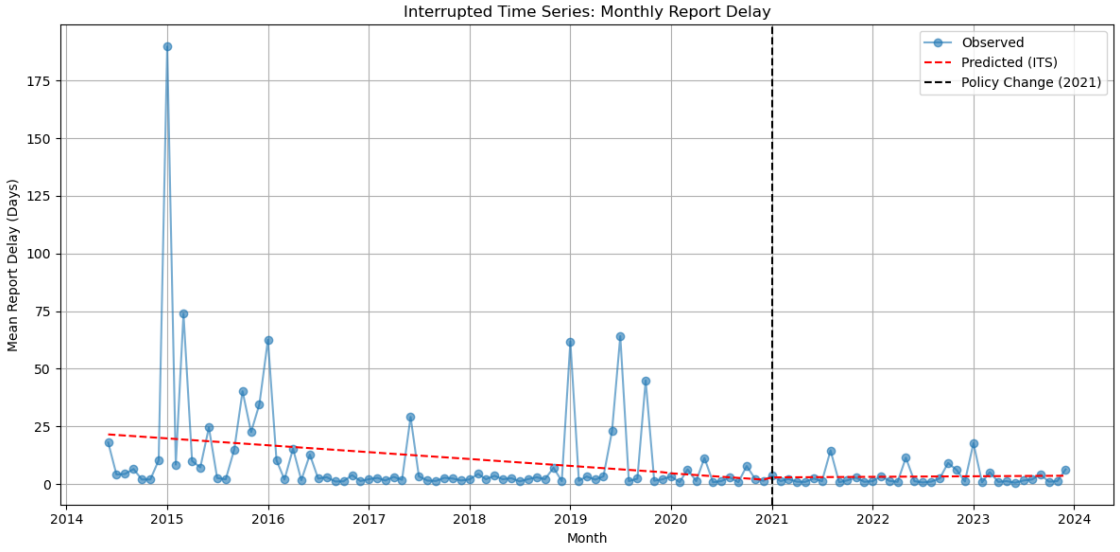
```
                            OLS Regression Results
==============================================================================
Dep. Variable:           report_delay   R-squared:                       0.082
Model:                            OLS   Adj. R-squared:                  0.057
Method:                 Least Squares   F-statistic:                     3.285
```

| | | | |
|---|---|---|---|
| Date: | Wed, 02 Jul 2025 | Prob (F-statistic): | 0.0235 |
| Time: | 12:44:49 | Log-Likelihood: | -512.04 |
| No. Observations: | 115 | AIC: | 1032. |
| Df Residuals: | 111 | BIC: | 1043. |
| Df Model: | 3 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| Intercept | 21.5177 | 4.692 | 4.586 | 0.000 | 12.220 | 30.815 |
| time | -0.2484 | 0.103 | -2.404 | 0.018 | -0.453 | -0.044 |
| post_2021 | -20.3023 | 33.589 | -0.604 | 0.547 | -86.862 | 46.257 |
| time_post | 0.2689 | 0.355 | 0.758 | 0.450 | -0.434 | 0.972 |

| | | | |
|---|---|---|---|
| Omnibus: | 164.881 | Durbin-Watson: | 2.021 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 7376.455 |
| Skew: | 5.376 | Prob(JB): | 0.00 |
| Kurtosis: | 40.734 | Cond. No. | 1.41e+03 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.41e+03. This might indicate that there are strong multicollinearity or other numerical problems.



### 5.1 ITS by Spill Type

[16]:
```
import pandas as pd
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt

# Group by month and spill type
spills_gdf['report_month'] = spills_gdf['Initial Report Date'].dt.to_period('M')
monthly_by_type = spills_gdf.groupby(['report_month',␣
 ↪'spill_type'])['report_delay'].mean().reset_index()
monthly_by_type['report_month'] = monthly_by_type['report_month'].dt.
 ↪to_timestamp()

# ITS variables
monthly_by_type['time'] = (monthly_by_type['report_month'] -␣
 ↪monthly_by_type['report_month'].min()).dt.days // 30
monthly_by_type['post_2021'] = (monthly_by_type['report_month'] >= pd.
 ↪Timestamp('2021-01-01')).astype(int)
monthly_by_type['time_post'] = monthly_by_type['time'] *␣
 ↪monthly_by_type['post_2021']

# Run ITS models and print summaries
its_results_by_type = {}
for stype in monthly_by_type['spill_type'].unique():
    df = monthly_by_type[monthly_by_type['spill_type'] == stype].copy()
    model = smf.ols("report_delay ~ time + post_2021 + time_post", data=df).
 ↪fit()
    df['predicted'] = model.predict(df)
    its_results_by_type[stype] = (model, df)

    print(f"\n=== ITS Model Summary for {stype} Spills ===\n")
    print(model.summary())
```

```
=== ITS Model Summary for Historical Spills ===

                            OLS Regression Results
==============================================================================
Dep. Variable:           report_delay   R-squared:                       0.065
Model:                            OLS   Adj. R-squared:                  0.040
Method:                 Least Squares   F-statistic:                     2.575
Date:                Wed, 02 Jul 2025   Prob (F-statistic):             0.0575
Time:                        12:45:09   Log-Likelihood:                -632.13
No. Observations:                 115   AIC:                             1272.
Df Residuals:                     111   BIC:                             1283.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
```

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| Intercept | 49.9824 | 13.331 | 3.749 | 0.000 | 23.566 | 76.398 |
| time | -0.6047 | 0.293 | -2.060 | 0.042 | -1.186 | -0.023 |
| post_2021 | -49.3080 | 95.435 | -0.517 | 0.606 | -238.418 | 139.802 |
| time_post | 0.6365 | 1.007 | 0.632 | 0.529 | -1.360 | 2.633 |

| | | | |
|---|---|---|---|
| Omnibus: | 176.312 | Durbin-Watson: | 2.082 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 9469.685 |
| Skew: | 5.973 | Prob(JB): | 0.00 |
| Kurtosis: | 45.820 | Cond. No. | 1.41e+03 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.41e+03. This might indicate that there are strong multicollinearity or other numerical problems.

=== ITS Model Summary for Recent Spills ===

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | report_delay | R-squared: | 0.009 |
| Model: | OLS | Adj. R-squared: | -0.018 |
| Method: | Least Squares | F-statistic: | 0.3223 |
| Date: | Wed, 02 Jul 2025 | Prob (F-statistic): | 0.809 |
| Time: | 12:45:09 | Log-Likelihood: | -400.04 |
| No. Observations: | 115 | AIC: | 808.1 |
| Df Residuals: | 111 | BIC: | 819.1 |
| Df Model: | 3 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| Intercept | 4.5634 | 1.772 | 2.576 | 0.011 | 1.053 | 8.074 |
| time | -0.0067 | 0.039 | -0.173 | 0.863 | -0.084 | 0.071 |
| post_2021 | -3.1197 | 12.683 | -0.246 | 0.806 | -28.252 | 22.013 |
| time_post | 0.0202 | 0.134 | 0.151 | 0.880 | -0.245 | 0.285 |

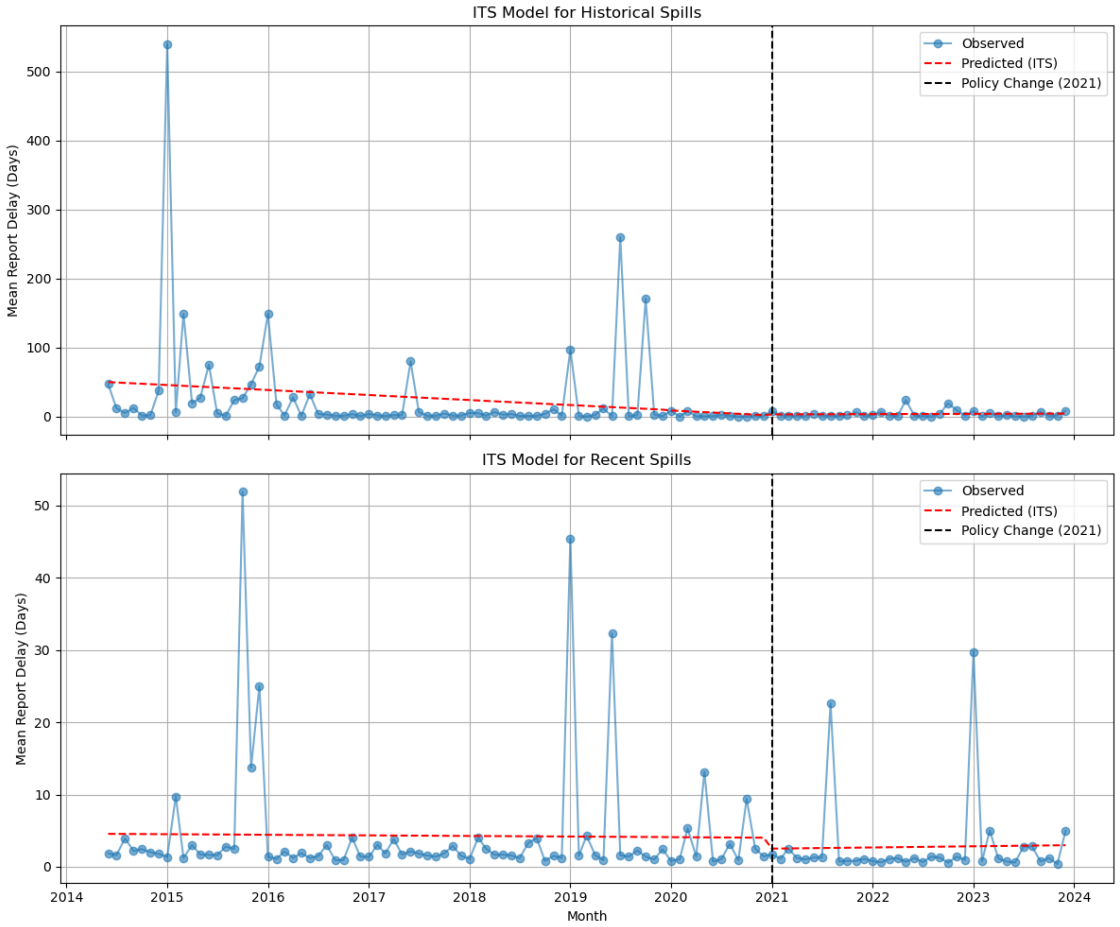
| | | | |
|---|---|---|---|
| Omnibus: | 132.583 | Durbin-Watson: | 1.949 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 2032.755 |
| Skew: | 4.250 | Prob(JB): | 0.00 |
| Kurtosis: | 21.761 | Cond. No. | 1.41e+03 |

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

```
specified.
[2] The condition number is large, 1.41e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[17]:
```python
import pandas as pd
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt

# 1. Group by month and spill type
spills_gdf['report_month'] = spills_gdf['Initial Report Date'].dt.to_period('M')
monthly_by_type = spills_gdf.groupby(['report_month',␣
 ↪'spill_type'])['report_delay'].mean().reset_index()
monthly_by_type['report_month'] = monthly_by_type['report_month'].dt.
 ↪to_timestamp()

# 2. Create ITS variables
monthly_by_type['time'] = (monthly_by_type['report_month'] -␣
 ↪monthly_by_type['report_month'].min()).dt.days // 30
monthly_by_type['post_2021'] = (monthly_by_type['report_month'] >= pd.
 ↪Timestamp('2021-01-01')).astype(int)
monthly_by_type['time_post'] = monthly_by_type['time'] *␣
 ↪monthly_by_type['post_2021']

# 3. Run ITS models separately by spill type
its_results_by_type = {}
for stype in monthly_by_type['spill_type'].unique():
    df = monthly_by_type[monthly_by_type['spill_type'] == stype].copy()
    model = smf.ols("report_delay ~ time + post_2021 + time_post", data=df).
 ↪fit()
    df['predicted'] = model.predict(df)
    its_results_by_type[stype] = (model, df)

# 4. Plot side-by-side
fig, axs = plt.subplots(2, 1, figsize=(12, 10), sharex=True)

for ax, (stype, (model, df)) in zip(axs, its_results_by_type.items()):
    ax.plot(df['report_month'], df['report_delay'], marker='o',␣
 ↪label='Observed', alpha=0.6)
    ax.plot(df['report_month'], df['predicted'], linestyle='--', color='red',␣
 ↪label='Predicted (ITS)')
    ax.axvline(x=pd.Timestamp('2021-01-01'), color='black', linestyle='--',␣
 ↪label='Policy Change (2021)')
    ax.set_title(f"ITS Model for {stype} Spills")
    ax.set_ylabel("Mean Report Delay (Days)")
    ax.legend()
    ax.grid(True)
```

```
axs[1].set_xlabel("Month")
plt.tight_layout()
plt.show()
```




---

## 6 Conclusion: Comparative Summary

[18]:
```
def extract_regression_results(model, model_name):
    coefs = model.params
    ses = model.bse
    pvals = model.pvalues

    df = pd.DataFrame({
        'Coefficient': coefs,
        'Std. Error': ses,
        'P-Value': pvals
```

```
    })
    df['Model'] = model_name
    return df.reset_index().rename(columns={'index': 'Term'})
```

[19]:
```
# Collect regression summaries
results_frames = []

# Add your models - use your actual variable names
results_frames.append(extract_regression_results(model, 'OLS (Interaction)'))
results_frames.append(extract_regression_results(nb_model, 'Negative Binomial'))
results_frames.append(extract_regression_results(its_model, 'ITS (Monthly, All␣
 ↪Spills)'))


# ITS by spill type (e.g., 'Historical', 'Recent')
for spill_type, (model, df) in its_results_by_type.items():
    label = f"ITS (Monthly, {spill_type})"
    results_frames.append(extract_regression_results(model, label))

# Combine all
regression_summary_table = pd.concat(results_frames, ignore_index=True)

# Pivot just the coefficients for display
regression_coef_table = regression_summary_table.pivot_table(
    index='Term',
    columns='Model',
    values='Coefficient',
    aggfunc='first'
).round(3)

regression_coef_table
```

[19]:
```
Model                                              ITS (Monthly, All Spills)  \
Term
C(Period)[T.Before 2020]                                                 NaN
C(spill_type)[T.Recent]                                                  NaN
C(spill_type)[T.Recent]:C(Period)[T.Before 2020]                         NaN
Intercept                                                             21.518
post_2021                                                            -20.302
time                                                                  -0.248
time_post                                                              0.269

Model                                              ITS (Monthly, Historical)  \
Term
C(Period)[T.Before 2020]                                                 NaN
C(spill_type)[T.Recent]                                                  NaN
C(spill_type)[T.Recent]:C(Period)[T.Before 2020]                         NaN
```

| | |
|---|---|
| Intercept | 49.982 |
| post_2021 | -49.308 |
| time | -0.605 |
| time_post | 0.636 |

| Model | ITS (Monthly, Recent) \ |
|---|---|
| Term | |
| C(Period)[T.Before 2020] | NaN |
| C(spill_type)[T.Recent] | NaN |
| C(spill_type)[T.Recent]:C(Period)[T.Before 2020] | NaN |
| Intercept | 4.563 |
| post_2021 | -3.120 |
| time | -0.007 |
| time_post | 0.020 |

| Model | Negative Binomial \ |
|---|---|
| Term | |
| C(Period)[T.Before 2020] | 2.144 |
| C(spill_type)[T.Recent] | -0.371 |
| C(spill_type)[T.Recent]:C(Period)[T.Before 2020] | -1.554 |
| Intercept | 1.303 |
| post_2021 | NaN |
| time | NaN |
| time_post | NaN |

| Model | OLS (Interaction) |
|---|---|
| Term | |
| C(Period)[T.Before 2020] | NaN |
| C(spill_type)[T.Recent] | NaN |
| C(spill_type)[T.Recent]:C(Period)[T.Before 2020] | NaN |
| Intercept | 4.563 |
| post_2021 | -3.120 |
| time | -0.007 |
| time_post | 0.020 |