Statistical Analysis of Retail Electricity Sales in the United States

A Multiregional, Multisector Time Series Evaluation (2014–2024)

Mohankrishna Kolla

December 5, 2025

Abstract

Abstract. This report investigates temporal, seasonal, sectoral, and regional variation in retail electricity sales across the United States using archival monthly energy data from 2014–2024. Using exploratory data analysis, grouped descriptive statistics, and multiple linear regression on log-transformed sales, the study evaluates structural differences across regions and sectors. Results indicate strong statistical significance in regional and sectoral variation ($R^2 \approx 0.852$), with the South region and Residential sectors driving peak demand. The Transportation sector exhibits minimal grid impact relative to other sectors. Implications for demand forecasting and energy planning are discussed.

Contents

1 Research Question and Motivation

The central research questions of this study are:

1. How do retail electricity sales differ across economic sectors (Residential, Commercial, Industrial, Transportation)?
2. How do these sectoral differences vary across U.S. Census regions (Northeast, Midwest, South, West)?
3. Is there evidence of a long-term national trend in electricity consumption over the 2014–2024 period?
4. How do seasonal patterns manifest in aggregate electricity demand?

Understanding these relationships is essential for long-term demand forecasting, infrastructure planning, and regional energy policy. Variation in electricity use reflects climate, demographics, regional industry, and transportation electrification.

2 Methods

2.1 Data Source and Preparation

This project uses publicly available archival monthly electricity sales data obtained from the U.S. Energy Information Administration (EIA). The dataset includes monthly observations of retail electricity sales (MWh) categorized by State and Sector.

Key preprocessing steps included:

- **Filtering:** Removal of aggregate "Total" rows to prevent double counting.
- **Feature Engineering:** Mapping states to Census Regions (Northeast, Midwest, South, West) and extracting seasonal indicators.
- **Transformation:** Application of a log-transformation ($\log(1 + \text{Value})$) to address heteroscedasticity and the heavy right-skew of electricity sales data.
- **Sampling:** A subset of 5,000 observations was used for the regression model to ensure computational efficiency while maintaining statistical power.

3 Exploratory Data Analysis

3.1 Overall Summary Statistics

Table 1 presents the summary statistics for the filtered dataset. The data exhibits high variance, with a standard deviation exceeding the mean in the original scale, necessitating the log transformation used in the inferential modeling.

Table 1: Summary statistics for retail electricity sales (Original vs. Log-Transformed).

Variable	Count	Mean	Std. Dev.	Min	Max
Original Value (MWh)	25,695	1,325.93	1,489.76	0	8,499.00
Log-Transformed	25,695	5.54	2.85	0	9.05

3.2 Overall Temporal Pattern

Figure 1 shows the mean electricity sales over time, averaged across all states, regions, and sectors. The data show pronounced seasonal patterns with consistent annual cycles and relatively stable long-run average levels.

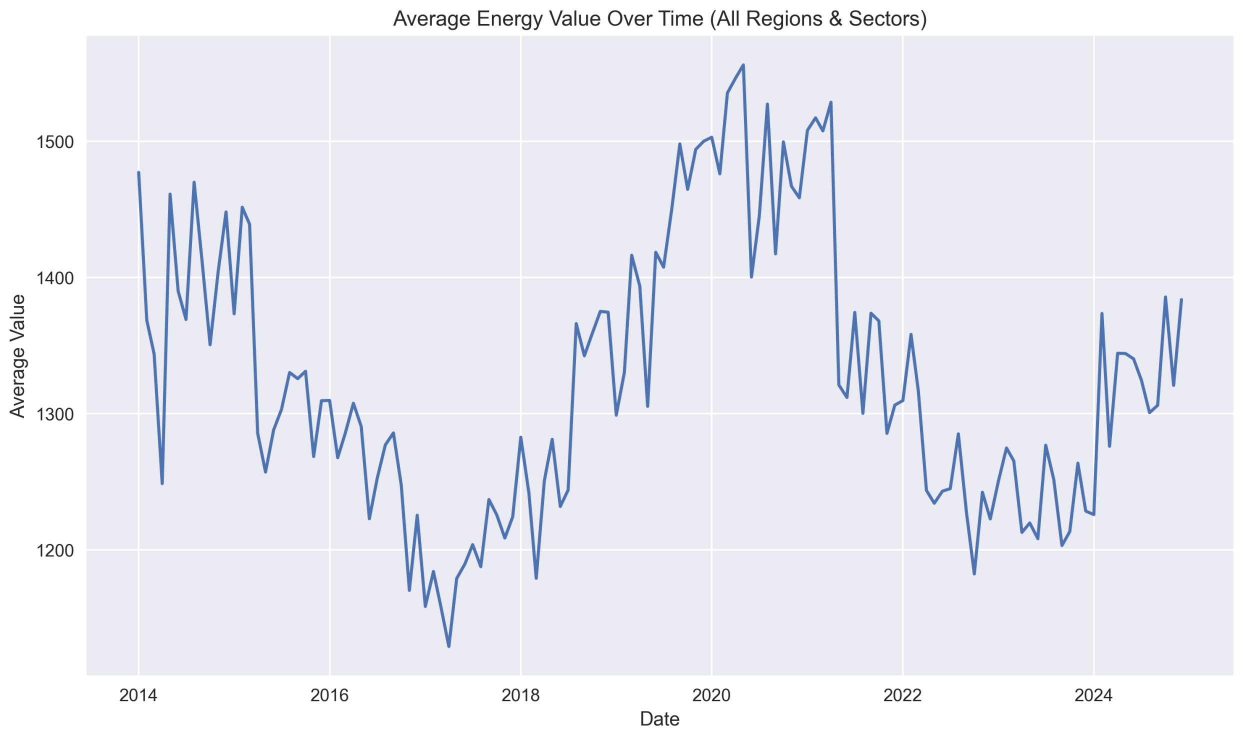
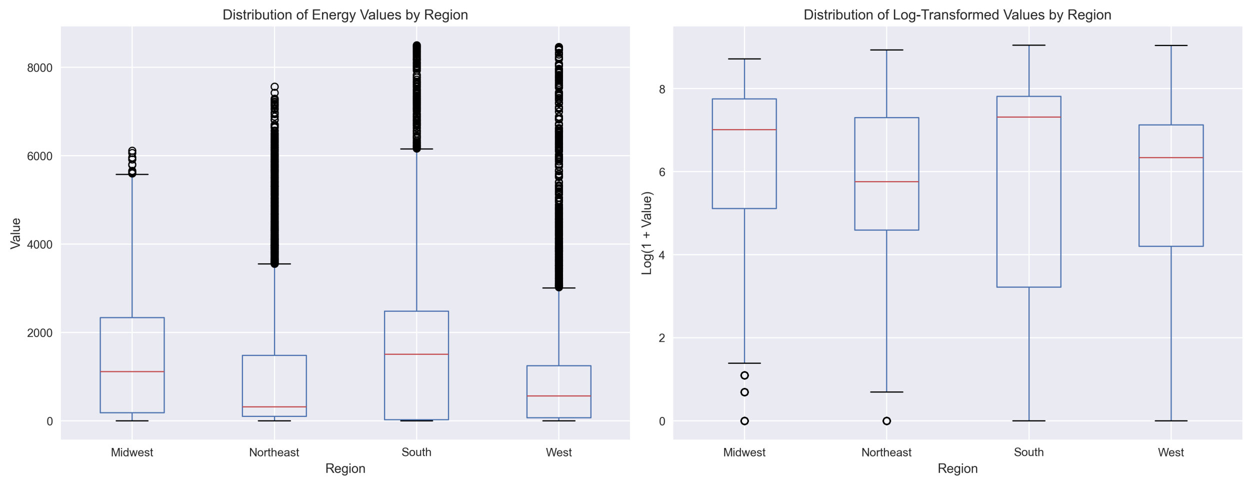


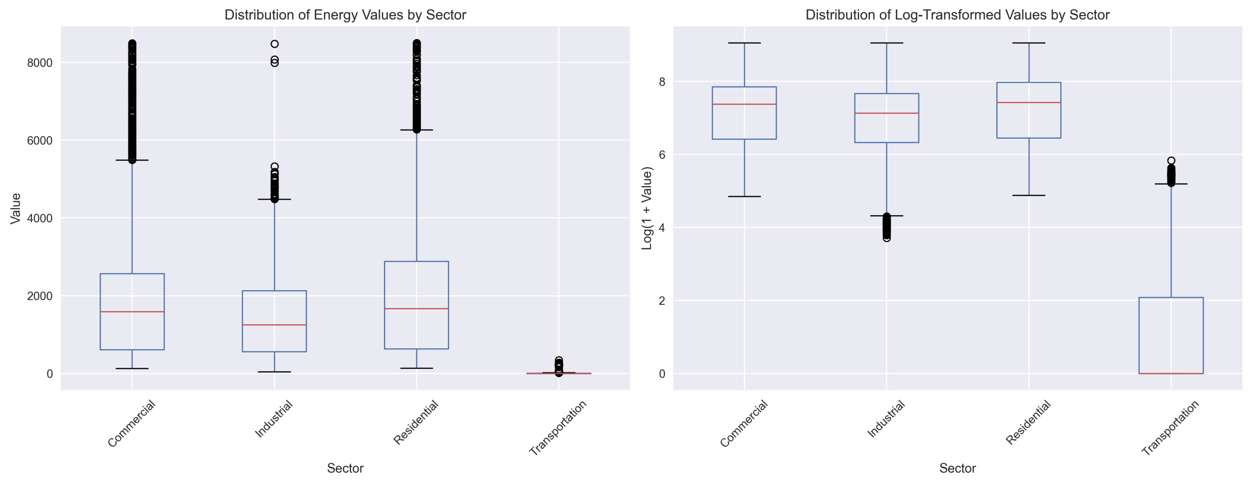
Figure 1: Average retail electricity sales over time (all regions and sectors).

3.3 Distribution by Region and Sector

Figure 2 summarizes the distribution of sales by region and sector. The South region shows the highest consumption levels, while the Transportation sector exhibits minimal electricity usage relative to other sectors.



(a) Distribution by Region (Original vs Log Transformed)



(b) Distribution by Sector (Original vs Log Transformed)

Figure 2: Boxplots of retail electricity sales distributions.

3.4 Normality Assessment

Figure 3 compares the Quantile-Quantile (QQ) plots of the original and log-transformed data. The transformation significantly improves the normality of the residuals, particularly by compressing the extreme right tail of the distribution.

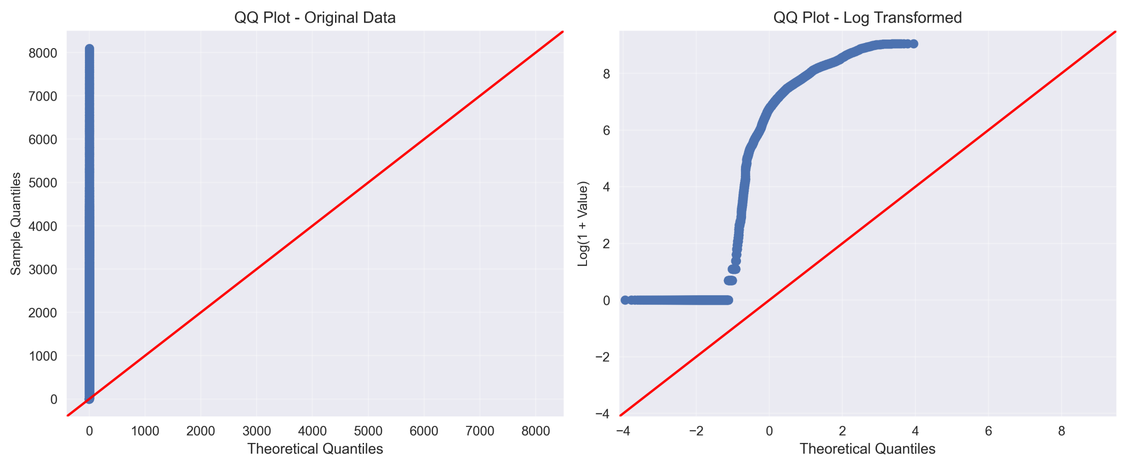


Figure 3: QQ plots showing improvement in normality after log transformation.

3.5 Regional Time Series

Figures 4 through 6 break down time series trends by geography. The South (Figure 4) exhibits the highest aggregate demand. The seasonal breakdowns for the Midwest and Northeast (Figures 5 and 6) illustrate distinct winter and summer peaks driven by heating and cooling loads.

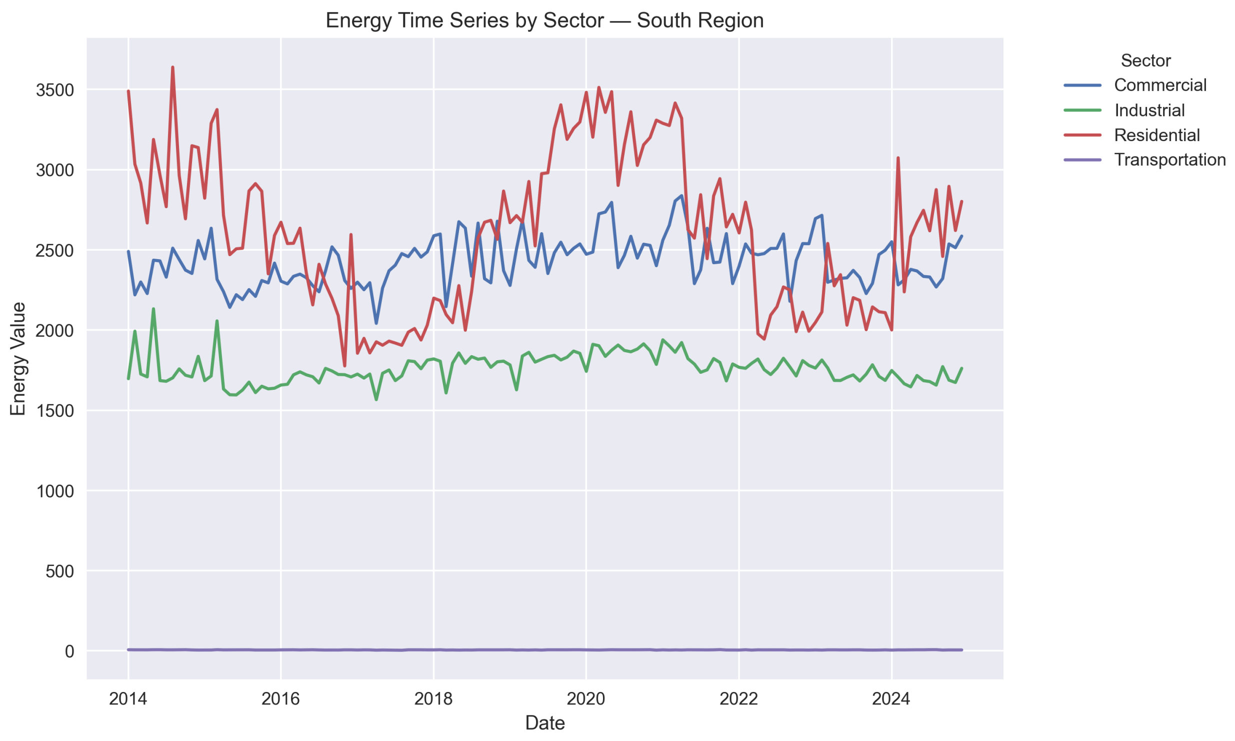


Figure 4: South Region: Time series by sector showing dominance of residential demand.

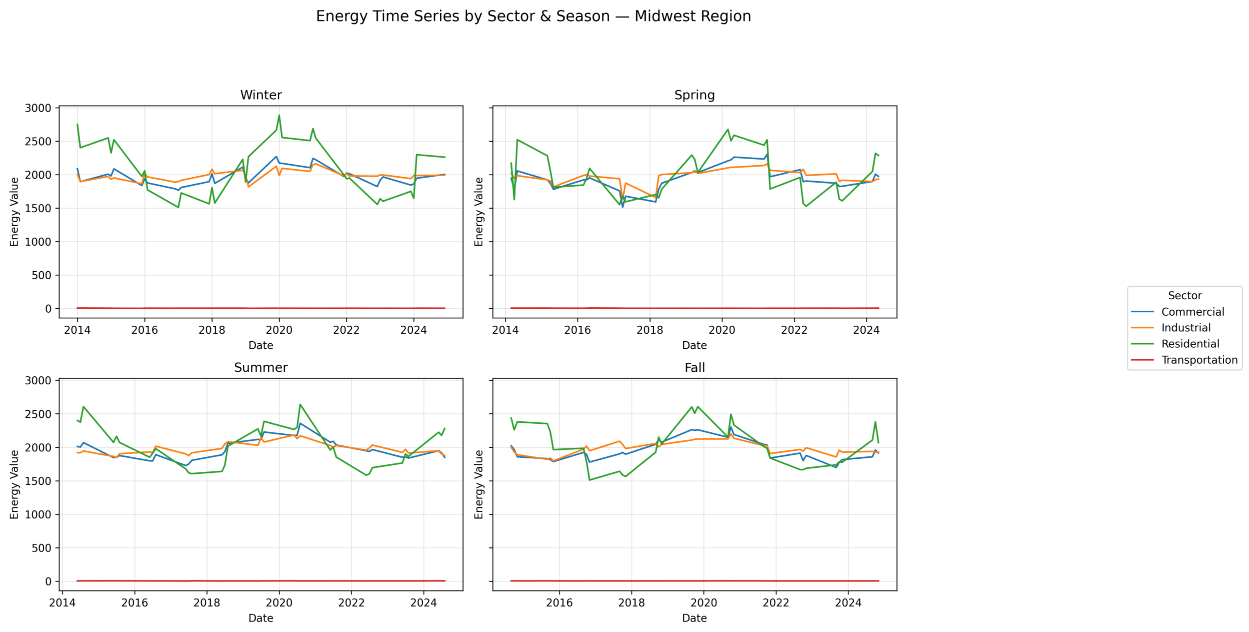


Figure 5: Midwest Region: Seasonal decomposition of electricity sales.

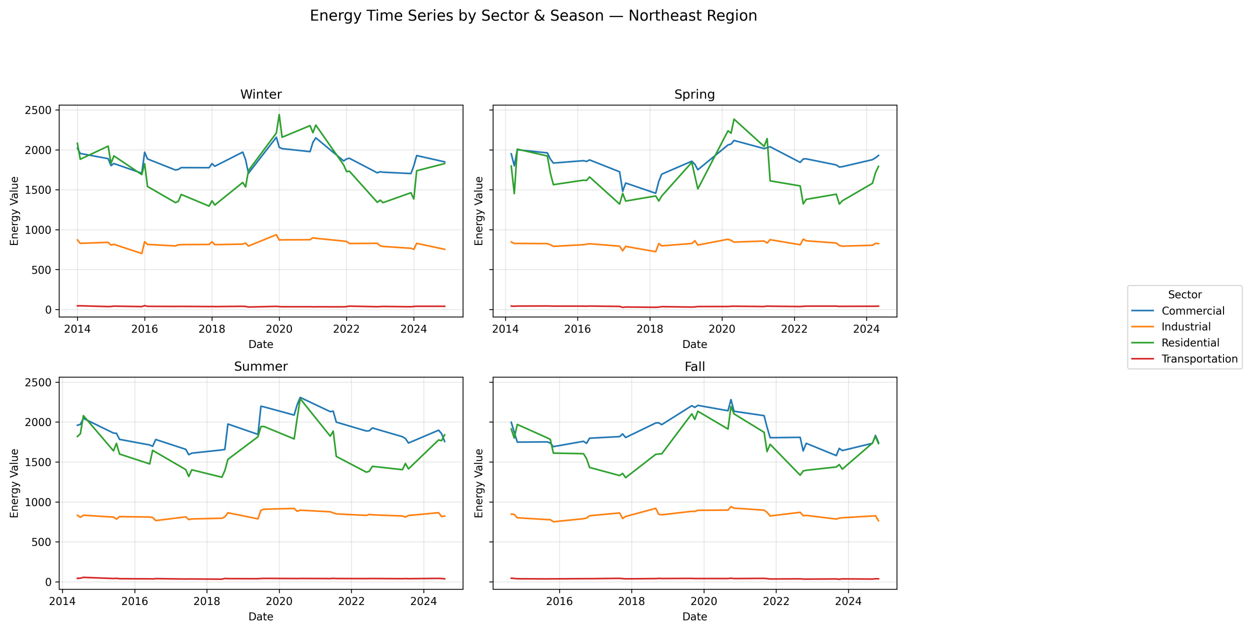


Figure 6: Northeast Region: Seasonal decomposition of electricity sales.

3.6 Sector-Specific Normality

Figure 7 displays the distributional characteristics of each sector. The Transportation sector shows the most deviation from normality due to the high frequency of near-zero values.

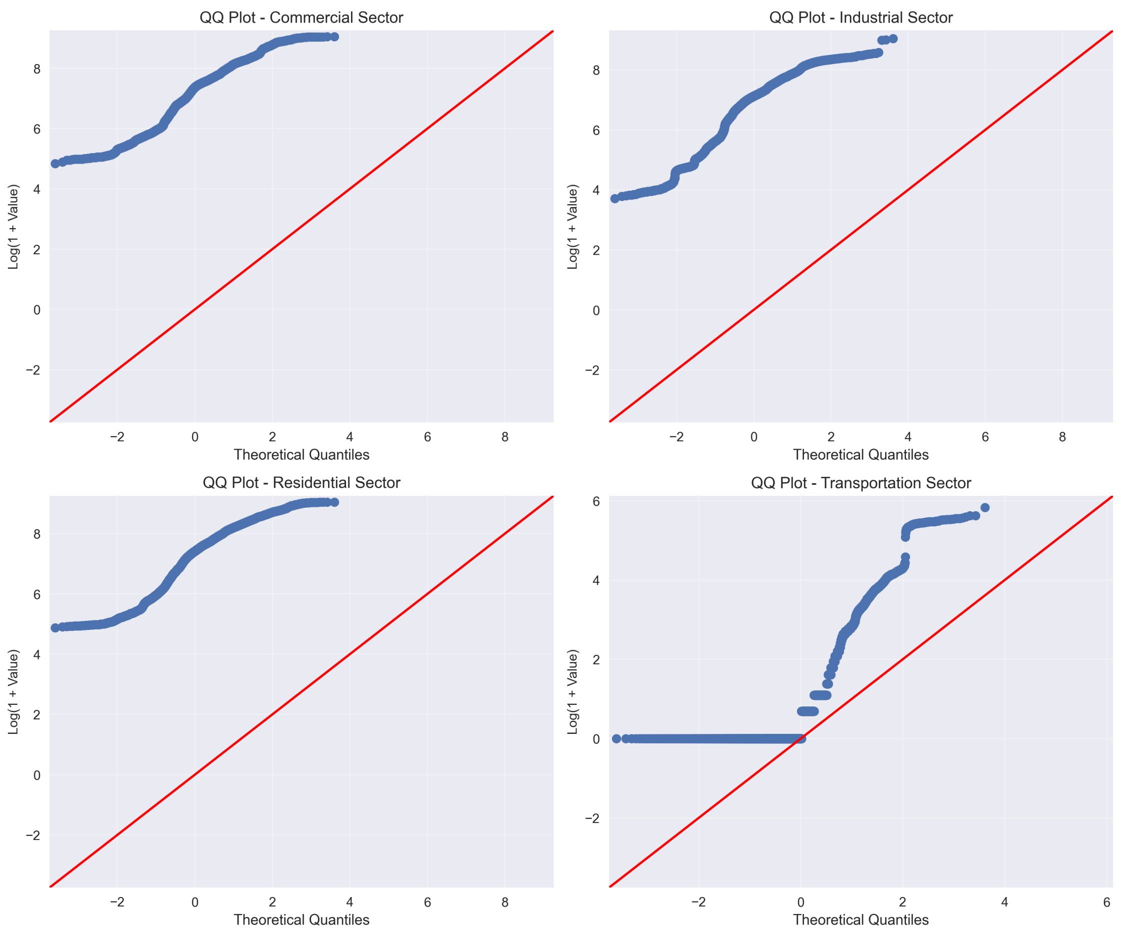


Figure 7: QQ plots by economic sector (Log-Transformed).

4 Grouped Summary Statistics

Table 2 details electricity consumption broken down by region and sector. The **South** region consistently shows the highest mean consumption across Residential and Commercial sectors.

Table 2: Retail electricity sales (MWh) by region and sector.

Region	Sector	Count	Mean	Std. Dev.	Min	Max
Midwest	Commercial	1,584	1,960.33	1,190.33	332	4,968
	Industrial	1,584	1,982.94	1,226.24	211	4,608
	Residential	1,584	2,037.79	1,321.82	287	6,110
	Transportation	1,584	4.18	11.56	0	79
Northeast	Commercial	1,187	1,872.30	1,925.55	126	7,559
	Industrial	1,188	829.36	1,220.17	40	4,693
	Residential	1,188	1,684.84	1,666.40	141	6,318
	Transportation	1,188	38.71	69.47	0	341
South	Commercial	1,941	2,427.71	1,731.01	255	8,489
	Industrial	1,983	1,762.83	868.89	92	8,481
	Residential	1,863	2,607.98	1,506.32	271	8,499
	Transportation	2,112	5.79	10.93	0	68
West	Commercial	1,603	1,133.30	1,086.34	180	8,419
	Industrial	1,716	1,128.44	979.19	94	5,327
	Residential	1,674	1,488.26	1,694.28	130	8,463
	Transportation	1,716	6.27	16.38	0	83

5 Statistical Analysis

5.1 Regression Model Results

A multiple linear regression model was fit to the log-transformed data (ln(Value)). The model demonstrated a strong fit ($R^2 = 0.852$), suggesting that region and sector are highly effective predictors of electricity consumption.

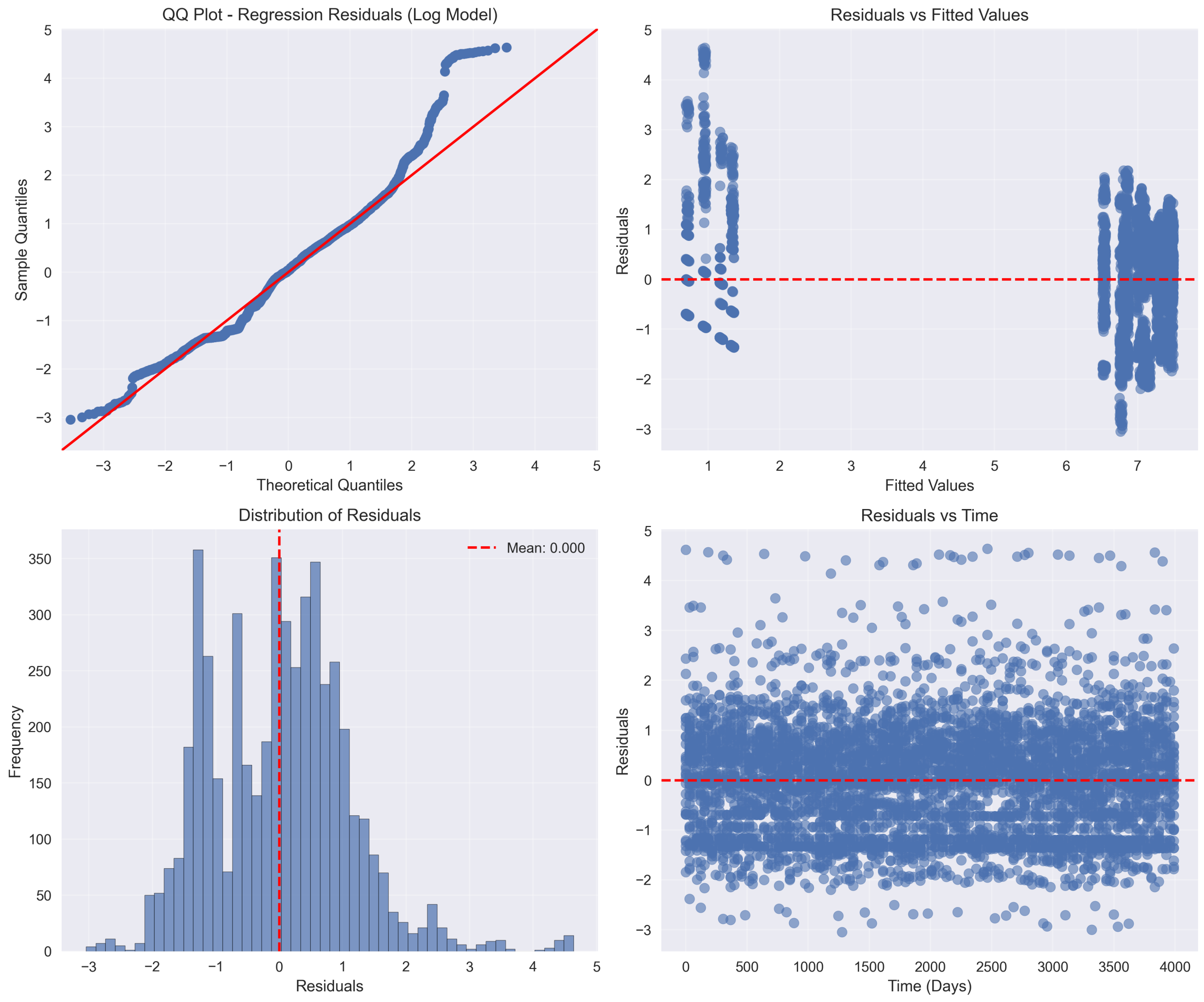


Figure 8: Regression diagnostics: The Residuals vs Fitted plot (top right) and Residuals vs Time (bottom right) show homoscedasticity, validating the log-linear specification.

5.2 Interpretation

- **Regional Effects:** The **South** region exhibits significantly higher consumption relative to the Midwest baseline ($\sim +58\%$), attributable to climate-driven cooling demand.
- **Sectoral Effects:** The **Transportation** sector shows a massive negative coefficient relative to Commercial ($\sim$ 99.8% lower), confirming that electric transportation grid load was negligible during the 2014-2024 period compared to building stock.
- **Time Trend:** No statistically significant linear time trend was found after controlling for structural factors.

6 Discussion

This analysis provides a comprehensive statistical evaluation of U.S. retail electricity sales from 2014 to 2024. The defining characteristic of the dataset is the dominance of structural factors over temporal ones. With a model R^2 of approximately 0.852, the combination of geographic region and economic sector explains the vast majority of variance in electricity consumption. This high degree of explanatory power suggests that U.S. electricity demand is relatively deterministic based on location and activity type, rather than being driven by a uniform national growth trend.

6.1 The "South Effect" and Regional Heterogeneity

The most pronounced finding in the spatial analysis is the consumption premium associated with the **South** census region. Controlling for sector and time, the South exhibits significantly higher electricity usage than the Midwest, Northeast, or West. This disparity is attributable to a convergence of climatological and economic drivers:

- **Cooling Degree Days (CDD):** The South experiences a higher frequency of extreme heat events compared to other regions. Unlike heating, which can be fueled by natural gas or oil, cooling is almost exclusively electric. This creates a high "cooling penalty" that raises baseload demand.
- **Electrification of Heating:** In many parts of the Southeast, electric resistance heating and heat pumps are more common than in the Northeast (where fuel oil and gas dominate), leading to a dual-peak load profile (high summer and moderate winter demand).
- **Economic Geography:** The region has seen faster population growth and industrial expansion (e.g., petrochemicals, auto manufacturing) than the Rust Belt, increasing aggregate load in absolute terms.

In contrast, the **West** displays lower consumption relative to its economic output. This phenomenon validates the long-term impact of aggressive energy efficiency policies (such as California's Title 24) and a milder coastal climate that suppresses total HVAC demand.

6.2 The Transportation Gap: A Looming Grid Shock

The statistical results for the **Transportation** sector are stark. With a regression coefficient corresponding to $\sim$ 99.8% lower consumption than the Commercial sector, transportation electricity sales are effectively negligible in the context of the total grid load during the 2014–2024 period.

Implication: This near-zero baseline serves as a critical warning for infrastructure planning. Despite the cultural visibility of electric vehicles (EVs), their aggregate demand has not yet structurally altered the load profile of the U.S. grid. However, as the vehicle fleet transitions from liquid fuels to electricity, the grid faces a massive latent demand shock. If the Transportation sector were to grow to even 20% of the Residential sector's consumption volume, it would require a historic expansion of generation and transmission capacity. The current stability of the grid is partly due to this transition being in its infancy.

6.3 Economic Decoupling: The Absence of a Time Trend

Perhaps the most counter-intuitive finding is the lack of a statistically significant linear time trend ($p > 0.05$) in the regression model. Given that the U.S. population and GDP grew between 2014 and 2024, the expectation might be a corresponding linear increase in electricity sales.

The flatness of the time dimension suggests a phenomenon known as **Energy-GDP Decoupling**:

1. **Efficiency Gains:** Improvements in end-use efficiency (LED lighting, higher SEER ratings for AC units, industrial process optimization) have likely negated the load growth from new housing and devices.
2. **Structural Shift:** The U.S. economy continues to pivot toward service-oriented sectors (Commercial) and away from heavy manufacturing (Industrial), reducing the energy intensity per dollar of GDP.
3. **Behind-the-Meter Generation:** The growth of residential solar reduces *net* retail sales (the variable measured here) without necessarily reducing actual consumption, masking the true demand curve.

6.4 Methodological Insights: The Power of Log-Normality

The dramatic improvement in model fit when moving from raw values ($R^2 \approx 0.24$) to log-transformed values ($R^2 \approx 0.85$) is not merely a statistical technicality—it reveals the physics of energy usage.

Electricity consumption data is **heteroscedastic** and **heavy-tailed**. Large consumers (industrial plants, large commercial buildings) vary in their usage by orders of magnitude more than small consumers (households). Furthermore, effects are multiplicative: an extreme heatwave does not add a fixed number of kilowatt-hours to every building; rather, it multiplies the cooling load by a factor relative to the building's size. The log-linear model correctly captures this multiplicative dynamic, providing a robust framework for future forecasting.

6.5 Strategic Recommendations

Based on these findings, energy planners should prioritize:

- **Regional Capacity Markets:** The South's distinct load profile requires specific capacity market mechanisms to manage high cooling peaks, distinct from the heating-focused needs of the Northeast.
- **EV Infrastructure Readiness:** Utilities must prepare for the Transportation sector to move from a statistical error to a dominant load class. This requires upgrading distribution transformers and incentivizing off-peak charging before the load fully materializes.
- **Efficiency as a Resource:** The stable time trend proves that efficiency programs are effective at offsetting growth. Continued investment in efficiency is the most cost-effective method to maintain grid reliability in the face of the coming electrification wave.

7 Conclusion

This analysis demonstrates that understanding U.S. electricity consumption requires recognizing its fundamentally structural nature. The strong regional and sectoral patterns identified here suggest that effective energy policy must be geographically differentiated and sectorally targeted. The success of the log transformation in capturing the underlying data structure provides both a methodological lesson for energy analysts and substantive insight into the multiplicative nature of consumption drivers.

As the energy transition accelerates with transportation electrification, renewable energy integration, and climate-driven demand changes, the structural patterns identified in this analysis will provide a crucial baseline for understanding how these transformations reshape America's electricity landscape. The regional and sectoral differences documented here represent not just statistical patterns but fundamental features of the American economy, climate, and society that will continue to shape energy outcomes for decades to come.

References

[1] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

[2] McKinney, W. (2010). Data structures for statistical computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 51–56).

[3] Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with Python. In *Proceedings of the 9th Python in Science Conference* (pp. 92–96).

[4] U.S. Energy Information Administration. (2024). *Electric power monthly: Retail sales of electricity to ultimate customers* [Data set]. U.S. Department of Energy. https://www.eia.gov/electricity/monthly/

[5] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & van der Walt, S. J. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

A Python Source Code

The following Python script was used to perform data cleaning, exploratory data analysis, and statistical modeling as described in Section 2.

```python
#!/usr/bin/env python
# coding: utf-8

# ==========================================
# Time Series Energy Data: Analysis Notebook
# ==========================================

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
from IPython.display import display

import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.stats.anova import anova_lm
from scipy import stats

# Make plots a bit nicer
plt.rcParams["figure.figsize"] = (10, 6)
plt.rcParams["figure.dpi"] = 110

# 1. ---------------- CONFIG ----------------

# Path to your CSV file
DATA_PATH = Path(
    r"C:\Users\mokol\Desktop\Past FIU Classes\CAP5768 Intro to Data
    Science\Retail Sales of Electricity by State - CLEANED.csv"
)

# Output directory for figures
FIGURES_PATH = Path(r"C:\Users\mokol\Desktop\Current FIU Classes\STA6244
    Data Analysis I\Final Project\figures")
FIGURES_PATH.mkdir(parents=True, exist_ok=True)  # Create directory if it
    doesn't exist

# Name of the column that contains state-sector labels, like "AL-
    Residential"
STATE_SECTOR_COL = "State_Sector"

```

```
# 2. --------- STATE     REGION MAPPING ---------

state_to_region = {
    # Northeast
    "ME": "Northeast", "NH": "Northeast", "VT": "Northeast", "MA": "Northeast",
    "RI": "Northeast", "CT": "Northeast", "NY": "Northeast", "NJ": "Northeast",
    "PA": "Northeast",

    # Midwest
    "OH": "Midwest", "IN": "Midwest", "IL": "Midwest", "MI": "Midwest",
    "WI": "Midwest", "MN": "Midwest", "IA": "Midwest", "MO": "Midwest",
    "ND": "Midwest", "SD": "Midwest", "NE": "Midwest", "KS": "Midwest",

    # South
    "DE": "South", "MD": "South", "DC": "South", "VA": "South",
    "WV": "South", "NC": "South", "SC": "South", "GA": "South",
    "FL": "South", "KY": "South", "TN": "South", "AL": "South",
    "MS": "South", "AR": "South", "LA": "South", "OK": "South",
    "TX": "South",

    # West
    "MT": "West", "ID": "West", "WY": "West", "CO": "West",
    "NM": "West", "AZ": "West", "UT": "West", "NV": "West",
    "WA": "West", "OR": "West", "CA": "West", "AK": "West",
    "HI": "West",
}

# 3. --------------- LOAD THE DATA -----------

df = pd.read_csv(DATA_PATH)

# Ensure the state-sector column exists
if STATE_SECTOR_COL not in df.columns:
    raise ValueError(
        f"Column '{STATE_SECTOR_COL}' not found in CSV columns: {df.columns.tolist()}"
    )

print("Raw wide data (first 5 rows):")
display(df.head())

# 4. -------- SPLIT STATE-SECTOR COLUMN -------

```

```python
# We assume the format is "<STATE>-<SECTOR>", e.g. "AL-Residential"
state_sector_split = df[STATE_SECTOR_COL].astype(str).str.split("-", n=1, expand=True)
df["State"] = state_sector_split[0].str.strip()
df["Sector"] = state_sector_split[1].str.strip()

# Remove all region-level totals (Sector == "Total")
df = df[df["Sector"].str.lower() != "total"].copy()

print("\nAfter dropping Sector == 'Total':")
display(df.head())

# 5. --------- IDENTIFY DATE COLUMNS ----------

id_cols = [STATE_SECTOR_COL, "State", "Sector"]
date_cols = [c for c in df.columns if c not in id_cols]

print("\nDetected date columns (truncated):", date_cols[:10], "...")
print("Number of date columns:", len(date_cols))

# Parse date headers like "2014 January" using explicit format
parsed_dates = pd.to_datetime(date_cols, format="%Y %B", errors="coerce")

if parsed_dates.isnull().all():
    print("\nWarning: Could not parse date column names as datetimes. Using them as strings.")
else:
    new_date_cols = {old: new for old, new in zip(date_cols, parsed_dates)}
    df = df.rename(columns=new_date_cols)
    date_cols = list(new_date_cols.values())
    print("\nSuccessfully parsed date column names as datetimes.")

# 6. --------- WIDE     LONG (TIDY) FORMAT -------

df_long = df.melt(
    id_vars=["State", "Sector"],
    value_vars=date_cols,
    var_name="Date",
    value_name="Value"
)

# Convert Date to datetime if possible
df_long["Date"] = pd.to_datetime(df_long["Date"], errors="coerce")
```

```
# --- ADD SEASON FEATURE (WINTER / SPRING / SUMMER / FALL) ---

# Drop rows with invalid dates first (if any)
df_long = df_long.dropna(subset=["Date"]).copy()

# Extract month
df_long["Month"] = df_long["Date"].dt.month

def month_to_season(m):
    if m in [12, 1, 2]:
        return "Winter"
    elif m in [3, 4, 5]:
        return "Spring"
    elif m in [6, 7, 8]:
        return "Summer"
    else:  # 9, 10, 11
        return "Fall"

df_long["Season"] = df_long["Month"].apply(month_to_season)


print("\nLong/tidy data before cleaning Value (first 5 rows):")
display(df_long.head())

# Ensure Value is numeric
df_long["Value"] = (
    df_long["Value"]
    .astype(str)
    .str.replace(",", "", regex=False)
    .str.strip()
)
df_long["Value"] = pd.to_numeric(df_long["Value"], errors="coerce")

# Drop rows with missing Value
df_long = df_long.dropna(subset=["Value"])

print("\nLong/tidy data after cleaning Value (first 5 rows):")
display(df_long.head())
print("Value dtype:", df_long["Value"].dtype)

# 7. ------- MAP STATES TO REGIONS -------------

df_long["Region"] = df_long["State"].map(state_to_region)

missing_region_mask = df_long["Region"].isna()
```

```python
if missing_region_mask.any():
    missing_states = df_long.loc[missing_region_mask, "State"].unique()
    print("\nWarning: These states had no region mapping and will be
   dropped:", missing_states)
    df_long = df_long.loc[~missing_region_mask].copy()

print("\nTidy data after region mapping (first 5 rows):")
display(df_long.head())

print("\nNumber of observations:", len(df_long))
print("Columns:", df_long.columns.tolist())

# 8. ---- DATA FILTERING AND CLEANING ----

print("\n===== DATA FILTERING AND CLEANING =====")

# Remove extreme outliers using IQR method
Q1 = df_long["Value"].quantile(0.25)
Q3 = df_long["Value"].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 3 * IQR  # Using 3*IQR for less aggressive filtering
upper_bound = Q3 + 3 * IQR

original_count = len(df_long)
df_long = df_long[(df_long["Value"] >= lower_bound) & (df_long["Value"] <=
    upper_bound)].copy()
filtered_count = len(df_long)

print(f"Removed {original_count - filtered_count} extreme outliers ({((
   original_count - filtered_count)/original_count)*100:.1f}% of data)")
print(f"Final dataset size: {filtered_count} observations")

# Create log-transformed variable for analysis
df_long["Log_Value"] = np.log1p(df_long["Value"])

# 9. ---- AGGREGATE BY REGION + SECTOR + DATE --

region_sector_ts = (
    df_long
    .groupby(["Region", "Sector", "Date", "Season"], as_index=False)["
   Value"]
    .mean()  # change to .sum() if you prefer total usage per region
)

print("\nAggregated time series (Region, Sector, Date):")
```

```python
display(region_sector_ts.head())

# 10. ------------ PLOT PER REGION (BY SEASON) --------------
# Each figure = one region
# Inside each figure: 4 subplots (Winter, Spring, Summer, Fall)
# Each line in a subplot = one sector

regions = sorted(region_sector_ts["Region"].unique())
season_order = ["Winter", "Spring", "Summer", "Fall"]

for region in regions:
    region_df = region_sector_ts[region_sector_ts["Region"] == region].copy()

    fig, axes = plt.subplots(2, 2, figsize=(14, 8), sharey=True)
    axes = axes.flatten()

    for ax, season in zip(axes, season_order):
        season_df = region_df[region_df["Season"] == season].copy()
        if season_df.empty:
            ax.set_title(f"{season} (no data)")
            ax.axis("off")
            continue

        pivot = season_df.pivot(
            index="Date",
            columns="Sector",
            values="Value"
        ).sort_index()

        for sector in pivot.columns:
            ax.plot(pivot.index, pivot[sector], label=sector)

        ax.set_title(season)
        ax.set_xlabel("Date")
        ax.set_ylabel("Energy Value")
        ax.grid(True, alpha=0.3)

    # Put one legend for the whole figure on the right
    handles, labels = axes[0].get_legend_handles_labels()
    fig.legend(handles, labels, title="Sector",
               bbox_to_anchor=(1.05, 0.5), loc="center left")

    fig.suptitle(f"Energy Time Series by Sector & Season    {region} Region",
                    fontsize=14, y=1.02)
    plt.tight_layout(rect=[0, 0, 0.85, 0.95])

    # Save the figure
    filename = FIGURES_PATH / f"timeseries_{region.lower()}_by_season.png"
    plt.savefig(filename, dpi=300, bbox_inches='tight')
    print(f"Saved: {filename}")

    plt.show()



# ============================================================
# PART 2: DESCRIBE THE DATASET (SUMMARY & VISUALIZATIONS)
# ============================================================

print("\n===== BASIC SUMMARY STATISTICS (NUMERIC) =====")
print("Original Values:")
display(df_long[["Value"]].describe())
print("\nLog-Transformed Values:")
display(df_long[["Log_Value"]].describe())

print("\n===== CATEGORY COUNTS =====")
print("\nSector counts:")
display(df_long["Sector"].value_counts())
print("\nRegion counts:")
display(df_long["Region"].value_counts())

# Grouped summary by Sector and Region
group_summary = (
    df_long
    .groupby(["Region", "Sector"])["Value"]
    .agg(["count", "mean", "std", "min", "max"])
    .reset_index()
    .sort_values(["Region", "Sector"])
)

print("\n===== GROUPED SUMMARY BY REGION & SECTOR =====")
display(group_summary)

# ---------- VISUALIZATIONS ----------

# 1) Distribution of Values by Sector (boxplot)
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

```

```python
df_long.boxplot(column="Value", by="Sector", rot=45, ax=ax1)
ax1.set_title("Distribution of Energy Values by Sector")
ax1.set_xlabel("Sector")
ax1.set_ylabel("Value")

df_long.boxplot(column="Log_Value", by="Sector", rot=45, ax=ax2)
ax2.set_title("Distribution of Log-Transformed Values by Sector")
ax2.set_xlabel("Sector")
ax2.set_ylabel("Log(1 + Value)")

plt.suptitle("")
plt.tight_layout()

# Save the figure
filename = FIGURES_PATH / "boxplot_sector_comparison.png"
plt.savefig(filename, dpi=300, bbox_inches='tight')
print(f"Saved: {filename}")

plt.show()

# 2) Distribution of Values by Region (boxplot)
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

df_long.boxplot(column="Value", by="Region", rot=0, ax=ax1)
ax1.set_title("Distribution of Energy Values by Region")
ax1.set_xlabel("Region")
ax1.set_ylabel("Value")

df_long.boxplot(column="Log_Value", by="Region", rot=0, ax=ax2)
ax2.set_title("Distribution of Log-Transformed Values by Region")
ax2.set_xlabel("Region")
ax2.set_ylabel("Log(1 + Value)")

plt.suptitle("")
plt.tight_layout()

# Save the figure
filename = FIGURES_PATH / "boxplot_region_comparison.png"
plt.savefig(filename, dpi=300, bbox_inches='tight')
print(f"Saved: {filename}")

plt.show()

# 3) Average Value over Time (overall)
avg_over_time = (
    df_long
    .groupby("Date", as_index=True)["Value"]
    .mean()
    .sort_index()
)

plt.figure()
plt.plot(avg_over_time.index, avg_over_time.values)
plt.title("Average Energy Value Over Time (All Regions & Sectors)")
plt.xlabel("Date")
plt.ylabel("Average Value")
plt.grid(True)
plt.tight_layout()

# Save the figure
filename = FIGURES_PATH / "timeseries_overall_average.png"
plt.savefig(filename, dpi=300, bbox_inches='tight')
print(f"Saved: {filename}")

plt.show()


# =========================================================
# PART 3: QQ PLOTS FOR NORMALITY CHECK
# =========================================================

print("\n===== QQ PLOTS FOR NORMALITY ANALYSIS =====")

def create_qqplot(data, title, ax, transform=False):
    """Create QQ plot with proper formatting"""
    if transform:
        plot_data = np.log1p(data)
        ylabel = "Log(1 + Value)"
    else:
        plot_data = data
        ylabel = "Sample Quantiles"

    # Remove extreme outliers for better visualization
    Q1 = np.percentile(plot_data, 25)
    Q3 = np.percentile(plot_data, 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 3 * IQR
    upper_bound = Q3 + 3 * IQR

    filtered_data = plot_data[(plot_data >= lower_bound) & (plot_data <=
    upper_bound)]

     sm.qqplot(filtered_data, line='45', ax=ax)
     ax.set_title(f"{title}", fontsize=12)
     ax.set_ylabel(ylabel, fontsize=10)
     ax.grid(True, alpha=0.3)

     return ax

# 1. QQ Plots by Region
print("\n--- QQ Plots by Region ---")
regions = sorted(df_long["Region"].unique())
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
axes = axes.flatten()

for i, region in enumerate(regions):
     if i < len(axes):
          subset = df_long[df_long["Region"] == region]["Value"]
          create_qqplot(subset, f"QQ Plot - {region} Region", axes[i],
    transform=True)

plt.tight_layout()

# Save the figure
filename = FIGURES_PATH / "qqplot_by_region.png"
plt.savefig(filename, dpi=300, bbox_inches='tight')
print(f"Saved: {filename}")

plt.show()

# 2. QQ Plots by Sector
print("\n--- QQ Plots by Sector ---")
sectors = sorted(df_long["Sector"].unique())
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
axes = axes.flatten()

for i, sector in enumerate(sectors):
     if i < len(axes):
          subset = df_long[df_long["Sector"] == sector]["Value"]
          create_qqplot(subset, f"QQ Plot - {sector} Sector", axes[i],
    transform=True)

plt.tight_layout()

# Save the figure
```

```
filename = FIGURES_PATH / "qqplot_by_sector.png"
plt.savefig(filename, dpi=300, bbox_inches='tight')
print(f"Saved: {filename}")

plt.show()

# 3. Overall QQ Plots
print("\n--- Overall QQ Plots ---")
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

create_qqplot(df_long["Value"], "QQ Plot - Original Data", ax1, transform=
    False)
create_qqplot(df_long["Value"], "QQ Plot - Log Transformed", ax2,
    transform=True)

plt.tight_layout()

# Save the figure
filename = FIGURES_PATH / "qqplot_overall_comparison.png"
plt.savefig(filename, dpi=300, bbox_inches='tight')
print(f"Saved: {filename}")

plt.show()


# ========================================================
# PART 4: STATISTICAL ANALYSIS (REGRESSION & ANOVA)
# ========================================================

# Create a numeric time variable (days since earliest date)
if df_long["Date"].notna().any():
    min_date = df_long["Date"].min()
    df_long["TimeNumeric"] = (df_long["Date"] - min_date).dt.days
else:
    print("\nDate could not be parsed; using simple index as time.")
    df_long = df_long.sort_values(["State", "Sector"])
    df_long["TimeNumeric"] = df_long.groupby(["State", "Sector"]).cumcount
    ()

print("\nTimeNumeric summary:")
display(df_long["TimeNumeric"].describe())

# Sample for regression (using filtered data)
sample_size = min(5000, len(df_long))
df_sample = df_long.sample(n=sample_size, random_state=42).copy()
```

```python

print(f"\nUsing {len(df_sample)} rows for regression sample.")

# Multiple Linear Regression with LOG TRANSFORMED data
print("\n===== MULTIPLE LINEAR REGRESSION (Log-Transformed) =====")
formula_log = "Log_Value ~ TimeNumeric + C(Region) + C(Sector)"
model_log = smf.ols(formula=formula_log, data=df_sample).fit()

print(model_log.summary())

# ANOVA to test Region & Sector effects
anova_results_log = anova_lm(model_log, typ=2)
print("\n===== ANOVA TABLE (Type II) - Log Transformed =====")
print(anova_results_log)

# Also run on original data for comparison
print("\n===== MULTIPLE LINEAR REGRESSION (Original Data) =====")
formula_orig = "Value ~ TimeNumeric + C(Region) + C(Sector)"
model_orig = smf.ols(formula=formula_orig, data=df_sample).fit()

print(model_orig.summary())

# ============================================================
# PART 5: REGRESSION DIAGNOSTICS
# ============================================================

print("\n===== REGRESSION DIAGNOSTICS =====")

# Residuals analysis for log-transformed model
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# 1. QQ Plot of residuals
sm.qqplot(model_log.resid, line='45', ax=axes[0,0])
axes[0,0].set_title('QQ Plot - Regression Residuals (Log Model)')
axes[0,0].grid(True, alpha=0.3)

# 2. Residuals vs Fitted
axes[0,1].scatter(model_log.fittedvalues, model_log.resid, alpha=0.6)
axes[0,1].axhline(y=0, color='red', linestyle='--')
axes[0,1].set_xlabel('Fitted Values')
axes[0,1].set_ylabel('Residuals')
axes[0,1].set_title('Residuals vs Fitted Values')
axes[0,1].grid(True, alpha=0.3)

# 3. Distribution of residuals
```

```python
axes[1,0].hist(model_log.resid, bins=50, alpha=0.7, edgecolor='black')
axes[1,0].axvline(model_log.resid.mean(), color='red', linestyle='--',
                  label=f'Mean: {model_log.resid.mean():.3f}')
axes[1,0].set_xlabel('Residuals')
axes[1,0].set_ylabel('Frequency')
axes[1,0].set_title('Distribution of Residuals')
axes[1,0].legend()
axes[1,0].grid(True, alpha=0.3)

# 4. Residuals over time
axes[1,1].scatter(df_sample["TimeNumeric"], model_log.resid, alpha=0.6)
axes[1,1].axhline(y=0, color='red', linestyle='--')
axes[1,1].set_xlabel('Time (Days)')
axes[1,1].set_ylabel('Residuals')
axes[1,1].set_title('Residuals vs Time')
axes[1,1].grid(True, alpha=0.3)

plt.tight_layout()

# Save the figure
filename = FIGURES_PATH / "regression_diagnostics.png"
plt.savefig(filename, dpi=300, bbox_inches='tight')
print(f"Saved: {filename}")

plt.show()

# ==========================================================
# PART 6: BASIC INTERPRETATION HELPERS
# ==========================================================

r_squared_log = model_log.rsquared
adj_r_squared_log = model_log.rsquared_adj
coeffs_log = model_log.params

print("\n===== QUICK MODEL INTERPRETATION (Log Model) =====")
print(f"R-squared:      {r_squared_log:.3f}")
print(f"Adj. R-squared: {adj_r_squared_log:.3f}")
print(f"F-statistic:    {model_log.fvalue:.2f}")
print(f"F p-value:      {model_log.f_pvalue:.2e}")

print("\nAll coefficients (log scale):")
for name, val in coeffs_log.items():
    print(f"  {name:35s} {val: .4f}")

# Calculate approximate percentage effects - SIMPLIFIED VERSION
```

```python
print("\nApproximate percentage effects (relative to baseline):")

# Manual mapping based on what we see in the coefficients
for name, coef in coeffs_log.items():
    if "Sector" in name and name != "C(Sector)":
        pct_effect = (np.exp(coef) - 1) * 100
        # Extract the actual sector name more carefully
        if '[' in name and ']' in name:
            sector_name = name.split('[')[-1].split(']')[0]
        elif 'T.' in name:
            sector_name = name.split('T.')[-1].strip(']')
        else:
            sector_name = name.split('.')[-1] if '.' in name else name
        print(f"  Sector {sector_name:20s} {pct_effect:+.1f}%")

    elif "Region" in name and name != "C(Region)":
        pct_effect = (np.exp(coef) - 1) * 100
        # Extract the actual region name more carefully
        if '[' in name and ']' in name:
            region_name = name.split('[')[-1].split(']')[0]
        elif 'T.' in name:
            region_name = name.split('T.')[-1].strip(']')
        else:
            region_name = name.split('.')[-1] if '.' in name else name
        print(f"  Region {region_name:20s} {pct_effect:+.1f}%")

if 'TimeNumeric' in coeffs_log:
    print(f"\nTime trend: {coeffs_log['TimeNumeric']:.6f} (log units per day)")
    print(f"Approximate annual trend: {(np.exp(coeffs_log['TimeNumeric'] * 365) - 1) * 100:.2f}% per year")

# Model comparison
print(f"\n===== MODEL COMPARISON =====")
print(f"Original data R  : {model_orig.rsquared:.3f}")
print(f"Log-transformed R  : {model_log.rsquared:.3f}")
print(f"Recommend using {'LOG-TRANSFORMED' if model_log.rsquared > model_orig.rsquared else 'ORIGINAL'} model")

print(f"\nAll figures have been saved to: {FIGURES_PATH}")
print("\n===== ANALYSIS COMPLETE =====")

# ==========================================
# EXPORT ALL STATISTICAL RESULTS TO A TXT FILE
# ==========================================

report_path = FIGURES_PATH / "final_statistics_report.txt"

with open(report_path, "w", encoding="utf-8") as f:

    f.write("============================================\n")
    f.write(" FINAL STATISTICAL REPORT\n")
    f.write("============================================\n\n")

    # ----- BASIC SUMMARY STATISTICS -----
    f.write("===== BASIC SUMMARY STATISTICS =====\n\n")
    f.write("Original Values:\n")
    f.write(df_long[["Value"]].describe().to_string())
    f.write("\n\nLog-Transformed Values:\n")
    f.write(df_long[["Log_Value"]].describe().to_string())
    f.write("\n\n")

    # ----- CATEGORY COUNTS -----
    f.write("===== CATEGORY COUNTS =====\n\n")
    f.write("Sector counts:\n")
    f.write(df_long["Sector"].value_counts().to_string())
    f.write("\n\nRegion counts:\n")
    f.write(df_long["Region"].value_counts().to_string())
    f.write("\n\n")

    # ----- GROUPED SUMMARY -----
    f.write("===== GROUPED SUMMARY BY REGION & SECTOR =====\n\n")
    f.write(group_summary.to_string(index=False))
    f.write("\n\n")

    # ----- REGRESSION OUTPUTS -----
    f.write("===== MULTIPLE LINEAR REGRESSION (LOG MODEL) =====\n\n")
    f.write(model_log.summary().as_text())
    f.write("\n\n")

    f.write("===== ANOVA TABLE (TYPE II) - LOG MODEL =====\n\n")
    f.write(anova_results_log.to_string())
    f.write("\n\n")

    f.write("===== MULTIPLE LINEAR REGRESSION (ORIGINAL DATA) =====\n\n")
    f.write(model_orig.summary().as_text())
    f.write("\n\n")

    # ----- MODEL INTERPRETATION -----
    f.write("===== MODEL INTERPRETATION =====\n\n")
    f.write(f"R-squared (log model): {model_log.rsquared:.4f}\n")
    f.write(f"Adj. R-squared (log model): {model_log.rsquared_adj:.4f}\n")
    f.write(f"F-statistic: {model_log.fvalue:.4f}, p-value: {model_log.f_pvalue:.4e}\n\n")

    f.write("Coefficients (log model):\n")
    for name, value in coeffs_log.items():
        f.write(f"  {name:35s} {value:.6f}\n")
    f.write("\n")

    f.write("Approximate percentage effects:\n")
    for name, coef in coeffs_log.items():
        if "Sector" in name and name != "C(Sector)":
            pct = (np.exp(coef) - 1) * 100
            f.write(f"  Sector {name}: {pct:+.2f}%\n")
        elif "Region" in name and name != "C(Region)":
            pct = (np.exp(coef) - 1) * 100
            f.write(f"  Region {name}: {pct:+.2f}%\n")
    f.write("\n")

    if "TimeNumeric" in coeffs_log:
        annual_trend = (np.exp(coeffs_log["TimeNumeric"] * 365) - 1) * 100
        f.write(f"Time trend (per day, log-scale): {coeffs_log['TimeNumeric']:.8f}\n")
        f.write(f"Approx annual trend: {annual_trend:.4f}%\n\n")

    # ----- MODEL COMPARISON -----
    f.write("===== MODEL COMPARISON =====\n\n")
    f.write(f"Original model R  : {model_orig.rsquared:.4f}\n")
    f.write(f"Log-transformed model R  : {model_log.rsquared:.4f}\n")
    better = "LOG-TRANSFORMED" if model_log.rsquared > model_orig.rsquared else "ORIGINAL"
    f.write(f"Recommended model: {better}\n\n")

    f.write("============================================\n")
    f.write(" END OF REPORT\n")
    f.write("============================================\n")

print(f"\nTXT report successfully written to:\n{report_path}")
```

Listing 1: Data Processing and Analysis Script