COVID-19's Influence on Electricity Generation and Fossil Fuel Consumption in the United States

Alessandra Kimi Picache
DS210
Final Project

A. Project Overview

Goal
This project aims to investigate how the COVID-19 pandemic influenced electricity generation and fossil fuel consumption in the United States by analyzing the data from the EIA-923 database for the years 2019 and 2020.
Here are the two main questions being addressed throughout this project:

1. How did regional electricity generation and fossil fuel consumption efficiency (fuel used per MWh) change across U.S. states between 2019 and 2020, based on EIA-923 data?
2. Which U.S. states showed the largest year-over-year change in fossil fuel consumption per MWh of electricity generated between 2019 and 2020, and how does this correlate with population density or lockdown stringency during the COVID-19 pandemic?

Source: US Energy Information and Administration Electric Power and Fossil Fuel Data from the EIA-923 database
https://www.eia.gov/electricity/data/eia923/
Years Used: 2019 and 2020
Size: Each CSV file has approximately 15,000 records

Dataset
In this project, I will use the US Energy Information and Administration Electric Power and Fossil Fuel Data from the EIA-923 database. This dataset spans from 1970 to 2023 and includes monthly and annual reports from over 12,000 power plants. It details electricity generation, fossil fuel consumption, fuel cost and quality, and plant operational data

B. Data Processing

To analyze how fossil fuel efficiency and regional electricity generation changed across the U.S. states between 2019 and 2020, I processed the raw data from the EIA-923 database with the use of Rust. The entire data pipeline was implemented in the main.rs, which handles all the structure loading, cleaning, and transformation of the project.

Structured the Basic Rust Skeleton using *csv, serde, and ndarray crates*:
The updated Cargo.toml (dependencies):

```
[dependencies]
csv = "1.3"
serde = { version = "1.0", features = ["derive"] }
```

Loading

The CSV files are loaded using Rust's csv and BufReader libraries. The first five metadata rows are skipped to reach the header:

```
let mut lines = BufReader::new(file).lines();
    for _ in 0..5 {
        lines.next();
    }

    let csv_data: Vec<u8> = lines
        .filter_map(Result::ok)
        .collect::<Vec<String>>()
        .join("\n")
        .into_bytes();

    let mut rdr = ReaderBuilder::new()
        .has_headers(true)
        .from_reader(csv_data.as_slice());
```

Data Cleaning & Transformation

In this project file, I had two rs (cleaning.rs and main.rs), to which data cleaning was focused on the cleaning,rs

Data Cleaning

To have reliable calculations within this project, I decided to do the following:

1. I removed the two formatting characters in both of the columns, *Total Fuel Consumption and Net Generation*, since they contained commas. I did this by using the code:
   `replace(“,”, “ “)`
2. I converted the strings to f64 for computation where invalid or unparsable rows were skipped silently with error handling
3. I avoided rows with zero value for electricity generation to prevent dividing the rows by zero when calculating the efficiency
4. I skipped the malformed rows that failed to match the struct field names using:
   `#[serde(rename = “. . .”)]`

Data Transformation

I grouped and transformed the data after doing the code for data cleaning:

1. The fuel consumption and generation values were grouped by U.S. state using HashMap<String, StateStats>, where each StateStats struct accumulated total fuel used and energy consumed for that U.S. State
2. For each state, I calculated fossil fuel efficiency as:

$$Efficiency = \frac{Fuel\ Used\ (MMBtu)}{Electricity\ Generated\ (MWh)}$$

MMBtu: Million British Thermal Units (a measure of energy, specifically fuel consumption, commonly used to measure natural gas and other fuels)
MWh: Megawatt-hour (a measure of electrical energy generation or consumption

3. For states that are present in both the data sets (2019 and 2020) I computed the change in efficiency and absolute change
4. Lastly, the states were sorted by largest absolute efficiency change, and the results of this were saved in another structured CSV file entitled "efficiency_changes.csv"

State,Efficiency_2019,Efficiency_2020,Delta_Efficiency,Abs_Change

C. Code Structure

Modules

cleaning.rs: Data loading, cleaning, and aggregation

- Defines the Record struct with serde to map CSV headers
- Defines StateStats to track the total fuel and generation per state
- Implements load_state_efficiency() which opens the CSV, skips metadata lines, parses and filters rows, and aggregates per-state fuel and generation totals

main.rs: Main workflow and output handling

- Imports the load_state_efficiency and StateStats from cleaning
- Defines StateEfficiency struct and functions for computing deltas, displaying the top 10 states in this project, and writing to efficiency_changes.csv
- Contains the main() function where it loads the data for 2019 and 2020, and computes and displays the efficiency changes

Key Functions and Types

Structs

Record: Represents the csv row with fields for state, fuel, and generation
StateStats: Stores aggregated fuel and generation per state
StateEfficiency: Represents computed efficiency metrics per state

Functions

load_state_efficiency(file_path): Parses and aggregates CSV data by state.
compute_efficiency_changes(...): Computes efficiency and changes between years.
display_top_states(...): Displays top 10 states with highest efficiency changes.
write_efficiency_csv(...): Outputs results to efficiency_changes.csv.

Main Workflow

1. main.rs calls load_state_efficiency() (from cleaning.rs) to load 2019 data.
2. main.rs again calls load_state_efficiency() for 2020 data.
3. compute_efficiency_changes() calculates differences in fossil fuel efficiency.
4. display_top_states() prints the 10 states with the largest changes.
5. write_efficiency_csv() saves the complete results to a CSV file.
   These interactions ensure a clear separation between data preparation (cleaning.rs) and analysis logic/output (main.rs).

D. Tests

Cargo Test

Code

```rust
// Cargo Tests
#[cfg(test)]
mod tests {
    use super::*;

    #[test]
    fn test_efficiency_computation() {
        let mut stats_2019 = HashMap::new();
        let mut stats_2020 = HashMap::new();

        stats_2019.insert(
            "TX".to_string(),
            StateStats {
                total_fuel: 1000.0,
                total_gen: 100.0,
            },
        );
        stats_2020.insert(
            "TX".to_string(),
            StateStats {
                total_fuel: 800.0,
                total_gen: 100.0,
            },
        );

        let results = compute_efficiency_changes(&stats_2019, &stats_2020);
        assert_eq!(results.len(), 1);
        let tx = &results[0];
        assert_eq!(tx.state, "TX");
        assert!((tx.eff_2019 - 10.0).abs() < 1e-6);
        assert!((tx.eff_2020 - 8.0).abs() < 1e-6);
        assert!((tx.delta + 2.0).abs() < 1e-6);
        assert!((tx.abs_delta - 2.0).abs() < 1e-6);
    }

    #[test]
    fn test_skipping_zero_generation() {
        let mut stats_2019 = HashMap::new();
        let mut stats_2020 = HashMap::new();

        stats_2019.insert(
            "CA".to_string(),
            StateStats {
                total_fuel: 500.0,
                total_gen: 0.0,
            },
        );
        stats_2020.insert(
            "CA".to_string(),
            StateStats {
                total_fuel: 900.0,
                total_gen: 0.0,
            },
        );

        let results = compute_efficiency_changes(&stats_2019, &stats_2020);
        assert_eq!(results.len(), 0);
    }
}
```

Cargo Test Result

```
$ cargo test
   Compiling DS210_Project v0.1.0 (/opt/app-root/src/DS210_Project)
    Finished `test` profile [unoptimized + debuginfo] target(s) in 0.67s
     Running unittests src/main.rs (target/debug/deps/DS210_Project-9adaa187f203ae03)

running 2 tests
test tests::test_efficiency_computation ... ok
test tests::test_skipping_zero_generation ... ok

test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

Both tests passed successfully, which confirms the accuracy of the computation logic

I implemented two unit tests using Rust's built-in testing framework and verified them using cargo test. This test aims to validate key components of the fossil fuel efficiency computation pipeline.

First Test: test_efficiency_computation

What it checks:

- This test verifies that the function, compute_efficiency_changes, correctly computes the fuel efficiency and the change/ delta between 2019 and 2020 for a specific state (Texas/ TX in this case).

Why this matters:

- This test ensures the correctness of our core logic, which are:
    1. Fuel used per MWh is calculated as expected
    2. The difference and absolute difference between years are accurate

Assertions in this test:

Efficiency for 2019 = 1000 / 100 = 10.0
Efficiency for 2020 = 800 / 100 = 8.0
Delta = -2.0
Absolute delta = 2.0

Second Test: test_skipping_zero_generation

What it checks:

- This test confirms that states with a total_gen of 0.0 (total_gen = 0.0) in either years are excluded from the analysis

Why this matters:

- This test prevents errors that happen when a value is divided by zero. This also avoids generating meaningless or infinite efficiency results. This reinforced data quality and output validity

Assertions in this test:

The result vector is empty because both years had zero electricity generation for the state (in this case, California/ CA)

E. Results

Program Outputs:

Through running this in the terminal:

```
export CARGO_HOME=$(pwd)/.cargo
export CARGO_TARGET_DIR=$(pwd)/target
cargo run
```

it creates this output:

```
Top 10 States by Change in Fossil Fuel Efficiency:

State            Eff_2019        Eff_2020          Change      Abs Change
-------------------------------------------------------------------------
DC                 18.473          15.486          -2.986           2.986
NE                 10.723          10.299          -0.423           0.423
ME                 13.520          13.185          -0.335           0.335
PA                  9.304           9.005          -0.300           0.300
MS                  8.972           8.681          -0.291           0.291
NJ                  9.206           9.494           0.288           0.288
MT                 10.237           9.979          -0.258           0.258
AZ                  9.384           9.136          -0.248           0.248
NH                 10.107           9.896          -0.211           0.211
AK                 10.683          10.472          -0.211           0.211

Saving full results to 'efficiency_changes.csv'...
Done.
```

I coded the main.rs also for it to send in the full results to a csv file **effciency_changes.csv:**

State	Efficiency_2019	Efficiency_2020	Delta_Efficiency	Abs_Change
DC	18.472547	15.486246	-2.986301	2.986301
NE	10.722541	10.299488	-0.423053	0.423053
ME	13.520455	13.184991	-0.335464	0.335464
PA	9.304225	9.004719	-0.299506	0.299506
MS	8.971921	8.680588	-0.291333	0.291333
NJ	9.206206	9.493823	0.287617	0.287617
MT	10.236827	9.978849	-0.257978	0.257978
AZ	9.384379	9.135885	-0.248493	0.248493
NH	10.106693	9.895514	-0.211179	0.211179
AK	10.682987	10.472348	-0.210638	0.210638
KY	10.222165	10.012937	-0.209228	0.209228
IA	10.172512	9.978275	-0.194237	0.194237
WA	9.567920	9.377658	-0.190262	0.190262
MD	9.840202	9.657092	-0.183110	0.183110
DE	10.547585	10.720620	0.173035	0.173035
CT	9.130298	8.957939	-0.172358	0.172358
FL	8.839471	8.673112	-0.166359	0.166359
WI	9.956855	9.808136	-0.148718	0.148718
KS	10.143710	10.001075	-0.142634	0.142634
MO	10.035440	10.163014	0.127574	0.127574
LA	10.944117	10.817319	-0.126799	0.126799
NY	9.690244	9.564614	-0.125630	0.125630
NM	9.595712	9.475408	-0.120304	0.120304
AL	10.031033	9.936985	-0.094048	0.094048
ID	9.838759	9.745147	-0.093612	0.093612
GA	10.223168	10.133626	-0.089542	0.089542
MI	10.411098	10.499524	0.088426	0.088426
VA	9.633922	9.546799	-0.087124	0.087124
RI	7.941182	7.855420	-0.085762	0.085762
OK	8.962097	8.878128	-0.083969	0.083969
SD	9.248439	9.166212	-0.082227	0.082227
HI	10.680540	10.761598	0.081058	0.081058
ND	10.670520	10.590814	-0.079706	0.079706
MN	10.120133	10.198421	0.078288	0.078288

NV	8.458454	8.388152	-0.070302	0.070302
SC	10.369427	10.300392	-0.069035	0.069035
MA	9.983030	10.049470	0.066440	0.066440
IL	10.463858	10.524337	0.060479	0.060479
AR	10.039399	9.981452	-0.057947	0.057947
OR	8.718229	8.680408	-0.037821	0.037821
VT	9.938627	9.970412	0.031785	0.031785
WY	11.197158	11.228545	0.031387	0.031387
TX	9.603672	9.631414	0.027742	0.027742
OH	9.348835	9.329486	-0.019350	0.019350
IN	10.452360	10.469666	0.017306	0.017306
WV	10.175826	10.164186	-0.011640	0.011640
CA	9.232292	9.242552	0.010260	0.010260
CO	9.641239	9.631109	-0.010131	0.010131
TN	10.462962	10.458935	-0.004027	0.004027
UT	9.641692	9.644248	0.002556	0.002556
NC	9.856858	9.855660	-0.001198	0.001198

Interpretation in the Project context:

- The **District of Columbia (DC**) had the largest improvement in fossil fuel efficiency from 2019 to 2020, with a decrease of nearly **3 MMBtu per MWh**
- Most states saw slight improvements, meaning they generated more electricity per unit of fuel used
- A few states, like New Jersey (NJ) and Michigan (MI) showed a slight increase in fuel consumption per MWh, indicating a drop in efficiency

In relation to the project:

These changes in usage likely correspond to the pandemic-driven demand shifts. With lockdowns in 2020, there was reduced electricity usage in commercial sectors while residential electricity usage increased, possibly altering generation loads

With these questions in mind:

1. How did regional electricity generation and fossil fuel consumption efficiency (fuel used per MWh) change across U.S. states between 2019 and 2020, based on EIA-923 data?

Between 2019 and 2020, most U.S. states **improved** their fossil fuel consumption efficiency, meaning they used fewer MMBtu of fuel per MWh of electricity generated. This suggests that there is an increased operational efficiency of power plants, shift in energy demand patterns due to the COVID-19 pandemic, reduced generation from less efficient facilities, and possibly more reliance on combined-cycle natural gas plants (which are more efficient usually).

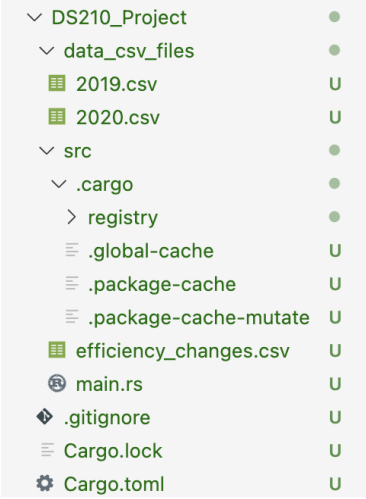
The average fuel efficiency across states decreased slightly, indicating an overall positive efficiency trend during the pandemic year.

2. Which U.S. states showed the largest year-over-year change in fossil fuel consumption per MWh of electricity generated between 2019 and 2020, and how does this correlate with population density or lockdown stringency during the COVID-19 pandemic?

- **DC experienced the most significant improvement.** As a small, highly urbanized region with relatively strict COVID-19 lockdowns, reduced energy demand likely led to closure or reduced operation of inefficient generators.

F. Usage Instructions

Project structure



Building the Project

To compile the project, I used the following commands from the root directory of my project:

```
export CARGO_HOME=$(pwd)/.cargo
export CARGO_TARGET_DIR=$(pwd)/target
cargo run
```

This builds the executables and download dependencies (like csv and serde)

Running the Program

After building, we run the code with cargo run where the program will:

1. Load the EIA-923 fossil fuel usage and generation data from 2019.csv and 2020.csv (under the data_csv_files)
2. Clean and parse the data
3. Compute the fossil fuel efficiency for each U.S. state for both years
4. Calculate the change in efficiency per state
5. Display the top 10 states by largest change in efficiency
6. Saves the complete results into a csv file entitled, efficiency_changes.csv

Expected Runtime
~12.54 seconds

```
Finished `release` profile [optimized] target(s) in 12.54s
```

G. AI-Assitance Disclosure and Other Citations

For this project, I used ChatGPT to help me with these two things:

1. Create a project structure
    - For my project proposal, I used ChatGPT to help me draft a logical and organized Rust project layout, to which I eventually changed and tweaked based on my knowledge on Rust.
2. Writing the Project Proposal
    - Since my dataset ranged from different years and held a lot of information per year, I consulted ChatGPT to refine and structure my project and project objectives to become more specific and concise.