EIA Explorer: Data Transformation and Visualization

Daniel Moscoe

Introduction

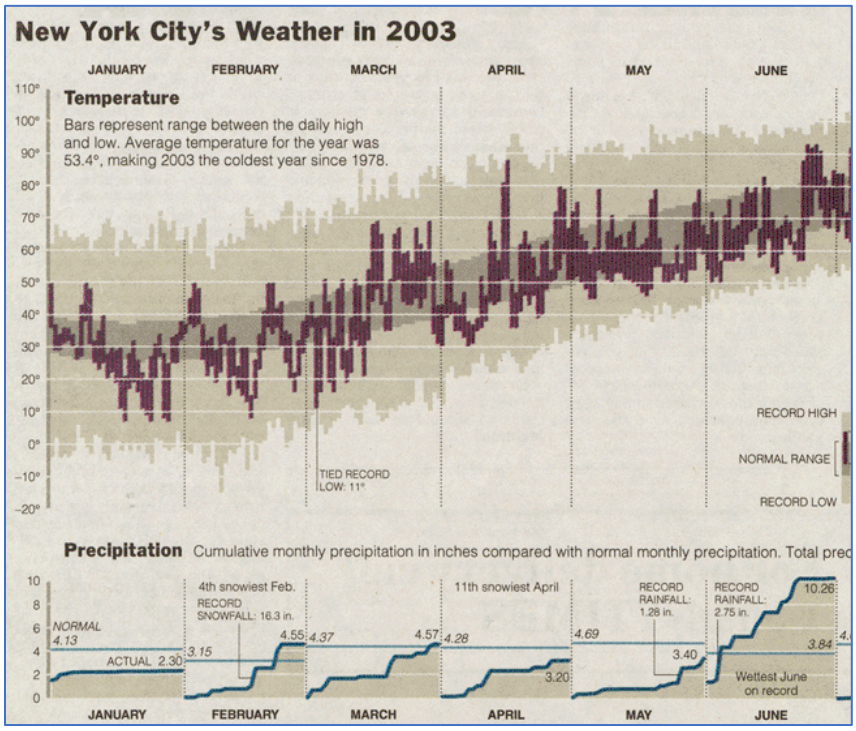
The U.S. Energy Information Administration (EIA) publishes data and analysis on energy production, consumption, and markets. As part of a course on data visualization, I implemented a website containing interactive visualizations of data pulled from the EIA's API. The visualizations below represent one portion of the site and answer the questions:

- How does a state's level of electricity consumption change over time?
- How does a state's level of electricity consumption relate to its production fuel portfolio?

In this brief report I focus on some of the design choices leading to the visualizations, how I transformed data from the EIA, and what conclusions can be drawn from the data. Other aspects of the website, such as layout, working with Plotly, collecting user input, and optimizing for speed, are not discussed.

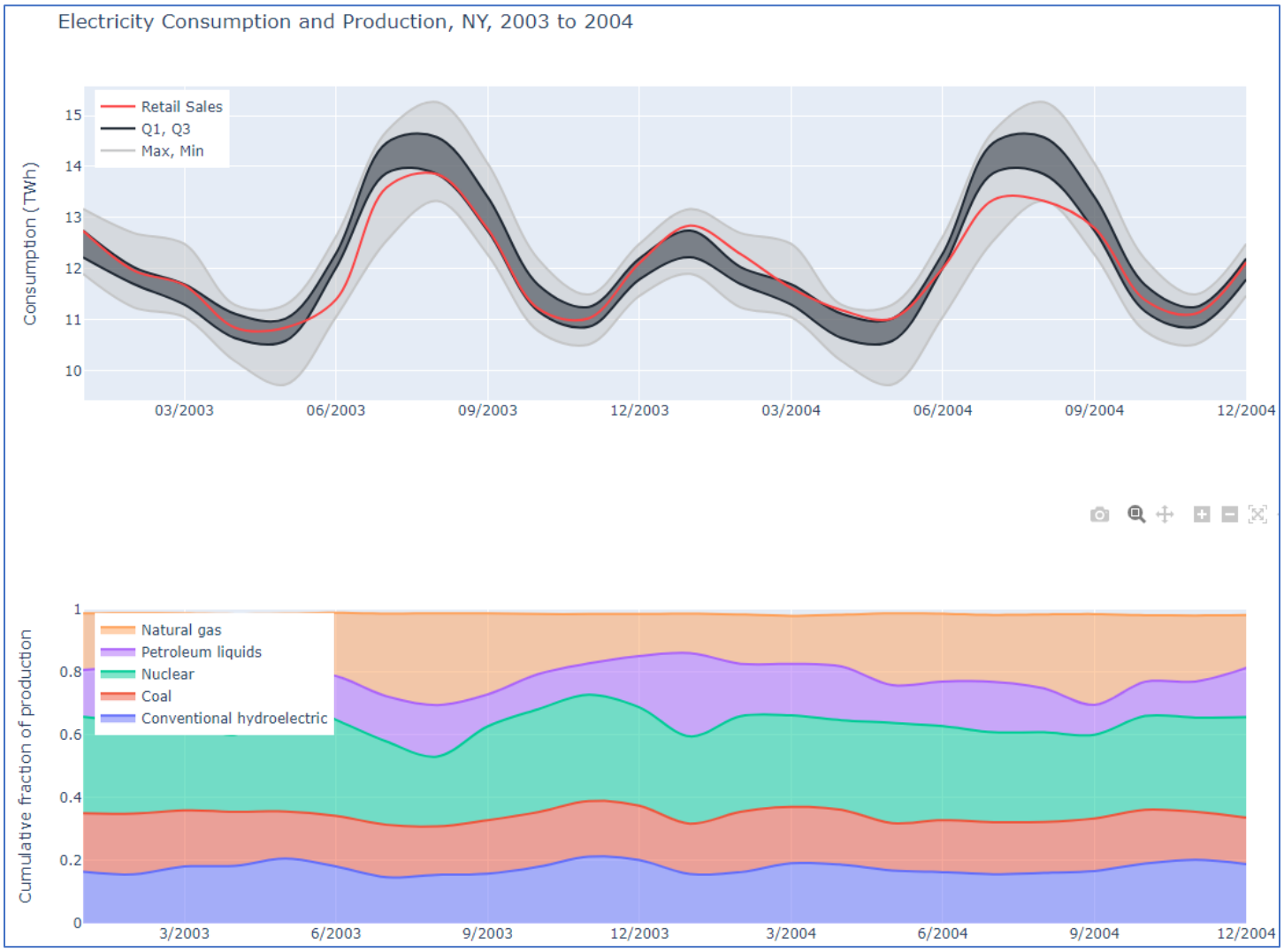
Visualization

In this plot I wanted to show how a state's level of electricity consumption changes over time, and how the fuels used to produce electricity within a state vary with consumption level. Since electricity consumption is seasonal, I wanted to contextualize values within a range appropriate to the state and time of year. I was inspired by a plot from the *New York Times*, a region of which is shown below. The plot shows temperature and precipitation in New York City for 2003.



"New York City's Weather in 2003." *New York Times*, 4 Jan 2004, p. A15.

One instance of the 2-panel plot I designed is shown below.



As in the *NYT* plot, in the top panel I used color to show how actual values compare to typical values as well as maximum and minimum values since 2001. For example, for June 2003, electricity consumption in New York was below the first quartile for all Junes since 2001, but greater than the minimum since 2001. In August 2004, electricity consumption in New York was very close to the minimum for Augusts since 2001.

The two panels above share a horizontal time axis. In the bottom panel I show how New York's electricity production fuel portfolio varies over time. In the summer months, when electricity consumption peaks in New York, the fraction of electricity generated by natural gas increases. This is shown by the thickened orange region at the top of the bottom panel, directly below the summer peaks in the top panel. The share generated by conventional hydroelectric and coal remains relatively constant by comparison. In the bottom panel, fuels are stacked according to the variance in electricity generated. Fuels with the greatest variability in electricity generation are at the top of the stack, and fuels with the least variability are at the bottom, forming the "base" of the state's production capacity.

Data Transformation

Electricity Consumption

To produce the dataframe for the Electricity Consumption plot, I implemented the function `get_retail_sales`, which takes a state and returns a dataframe with total energy consumption by month, along with min, Q1, Q3, and max summary statistics grouped by month. Below is a cell with some setup code, followed by `get_retail_sales`.

For purposes of explanation, I divided `get_retail_sales` into sections and set it to return key output from each section. The actual function supporting the site returns only `res`, the dataframe to be passed to the plotting function.

```
#Setup

import pandas as pd
import json
import urllib.request
from functools import reduce

states = ["AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE", "DC", "FL",
          "GA", "HI", "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME",
          "MD", "MA", "MI", "MN", "MS", "MO", "MT", "NE", "NV", "NH",
          "NJ", "NM", "NY", "NC", "ND", "OH", "OK", "OR", "PA", "RI",
          "SC", "SD", "TN", "TX", "UT", "VT", "VA", "WA", "WV", "WI", "WY"]

fuel_types = {"COW": "Coal",
"PEL": "Petroleum liquids",
"PC": "Petroleum coke",
"NG": "Natural gas",
"OOG": "Other gases",
"NUC": "Nuclear",
"HYC": "Conventional hydroelectric",
"AOR": "Other renewables",
"WND": "Wind",
"SUN": "All utility-scale solar",
"SPV": "Utility-scale photovoltaic",
"STH": "Utility-scale thermal",
"GEO": "Geothermal",
"WWW": "Wood and wood-derived fuels",
"WAS": "Other biomass",
"HPS": "Hydro-electric pumped storage",
"OTH": "Other",
"TSN": "All solar",
"DPV": "Small-scale solar photovoltaic",
"Other": "Other"}

years = list(range(2001, 2022))

pulled_data = {}
for state in states:
    pulled_data[state] = {}
```

```
def get_retail_sales(state):

    #Section A
    if "retail_sales" in pulled_data[state].keys():
        return pulled_data[state]["retail_sales"]
    else:
        api_call =
            "http://api.eia.gov/series/?api_key=c0b197bcf4610007c7e977fccc486830&series_id=ELEC.SALES."
            + state + "-ALL.M"
        with urllib.request.urlopen(api_call) as url:
            data = json.loads(url.read().decode())

        df = pd.DataFrame(
            data['series'][0]['data'],
            columns = ['Date', "energy"])

        section_A = df.copy()

        #Section B
        df['Year'] = df.Date.str.slice(0,4).astype(int)
        df['Month'] = df.Date.str.slice(4,6).astype(int)
        df['TWh'] = df.energy / 1000
        df['xaxis_labels'] = df.Date.str.slice(4,6) + "/" + df.Date.str.slice(0,4)
        df = df.loc[:, ['Year', 'Month', 'TWh', 'xaxis_labels']]

        section_B = df.copy()

        #Section C
        df_Min = pd.DataFrame(df.groupby(['Month'])['TWh'].min()).rename(columns = {'TWh':'Min'})
        df_Q1 = pd.DataFrame(df.groupby(['Month'])['TWh'].quantile([0.25])).rename(columns = {'TWh':'Q1'})
        df_Q3 = pd.DataFrame(df.groupby(['Month'])['TWh'].quantile([0.75])).rename(columns = {'TWh':'Q3'})
        df_Max = pd.DataFrame(df.groupby(['Month'])['TWh'].max()).rename(columns = {'TWh':'Max'})

        df_list = [df, df_Min, df_Q1, df_Q3, df_Max]

        section_C = df_list

        #Section D
        res = reduce(lambda left, right: pd.merge(left, right, on = ['Month'], how = 'outer'), df_list)

        section_D = res

        pulled_data[state]["retail_sales"] = res

        #return res
        return section_A, section_B, section_C, section_D
```

Section A checks whether the state's retail sales were previously pulled from the API. If so, the data resides in the dictionary `pulled_data`. If not, the function retrieves the data from the API. Below is an example of the raw data converted to a dataframe from its original json format.

```
state = "NY"
section_objects = get_retail_sales(state)
section_objects[0].head()
```

	Date	energy
0	202110	10966.42696
1	202109	12944.00179
2	202108	13950.11779
3	202107	13885.71184
4	202106	12040.16858

In Section B, the year and month are extracted from the strings in the Date column. The values in energy, given in gigawatt hours, are transformed to terawatt hours, reducing the numeric values on the vertical axis of the plot. Month and year are concatenated to produce labels for the horizontal axis. The result of section B is shown below.

```
section_objects[1].head()
```

	Year	Month	TWh	xaxis_labels
0	2021	10	10.966427	10/2021
1	2021	9	12.944002	09/2021
2	2021	8	13.950118	08/2021
3	2021	7	13.885712	07/2021
4	2021	6	12.040169	06/2021

In Section C, the function creates 4 new dataframes, each containing a summary statistic for the data grouped by month. These 12-row dataframes are gathered in a list, which will facilitate a succinct joining process in step D. The dataframe containing the third quartile of electricity consumption by month is shown below.

```
section_objects[2][3]
```

Month		Q3
1	0.75	12.747774
2	0.75	12.033235
3	0.75	11.696082
4	0.75	11.118763
5	0.75	11.019822
6	0.75	12.297255
7	0.75	14.462225
8	0.75	14.574946
9	0.75	13.404694
10	0.75	11.683713
11	0.75	11.250197
12	0.75	12.198354

Section D joins the DataFrames containing the raw consumption values and the summary statistics. The resulting dataframe is ready to be passed to the plotting function. The dataframe is also stored in the dictionary `pulled_data` for future use if necessary. The head of the final dataframe is shown below.

```
section_objects[3].sort_values(["Year", "Month"]).head()
```

	Year	Month	TWh	xaxis_labels	Min	Q1	Q3	Max
209	2001	1	12.747774	01/2001	11.897042	12.221600	12.747774	13.169576
188	2001	2	12.017395	02/2001	11.242659	11.700000	12.033235	12.699456
167	2001	3	11.790138	03/2001	11.053953	11.296118	11.696082	12.485820
146	2001	4	11.015864	04/2001	10.190773	10.642061	11.118763	11.290966
125	2001	5	11.019822	05/2001	9.729970	10.583232	11.019822	11.294645

Electricity Production

Producing the dataframe for the Electricity Production plot was more complicated, because I needed to combine information from a user-defined number of fuel types. I first implemented the function `get_net_gen`, which takes a state and a fuel type, and returns a dataframe with the Year and Month columns modified as above.

```
def get_net_gen(state, fuel):
    if fuel in pulled_data[state].keys(): #Did we already pull this data earlier? If so, don't call API.
        return pulled_data[state][fuel]
    else:
        try:
            api_call =
                "http://api.eia.gov/series/?api_key=c0b197bcf4610007c7e977fccc486830&series_id=ELEC.GEN."
                + fuel + "-" + state + "-99.M"
            with urllib.request.urlopen(api_call) as url:
                data = json.loads(url.read().decode())
            df = pd.DataFrame(
                data['series'][0]['data'],
                columns = ['Date', fuel])

        except KeyError:
            api_call =
                "http://api.eia.gov/series/?api_key=c0b197bcf4610007c7e977fccc486830&series_id=
                 ELEC.GEN.ALL-" + state + "-99.M"
            with urllib.request.urlopen(api_call) as url:
                data = json.loads(url.read().decode())
            df = pd.DataFrame(
                data['series'][0]['data'],
                columns = ['Date', fuel])
            df[fuel] = 0

        df['Year'] = df.Date.str.slice(0,4).astype(int)
        df['Month'] = df.Date.str.slice(4,6).astype(int)
        df = df.loc[:, ['Year', 'Month', fuel]]
        pulled_data[state][fuel] = df
        return df
```

```
state = "NY"
fuel = "NUC"

get_net_gen(state, fuel).head()
```

	Year	Month	NUC
0	2021	10	2105.477
1	2021	9	2341.896
2	2021	8	2407.000
3	2021	7	2468.055
4	2021	6	2369.921

Then I implemented a function, `get_net_gens`, to combine and transform data from several calls to `get_net_gen`. Below is `get_net_gens`, followed by an explanation of each section.

```
def get_net_gens(state, fuels, start, end):

    #Section A
    df_list = []
    fuels.append("ALL")

    for fuel in fuels:
        tmp = get_net_gen(state, fuel)
        tmp = tmp[(tmp.Year >= start) & (tmp.Year <= end)]
        df_list.append(tmp)

    section_A = df_list

    #Section B
    merged_df = reduce(lambda left, right: pd.merge(left, right, on = ['Year', 'Month'], how = 'outer'),
        df_list)
    fuels_by_variance = list(merged_df.iloc[:, 2:].var().sort_values().index)
    sorted_df = merged_df[fuels_by_variance]

    section_B = sorted_df

    #Section C
    sorted_asfractions_df = sorted_df.drop(columns = "ALL").div(sorted_df.ALL, axis = 0)
    cumulative_df = pd.DataFrame({
        'Year': merged_df.Year,
        'Month': merged_df.Month})
    for col in sorted_asfractions_df.columns:
        cumulative_df[col] = sorted_asfractions_df.loc[:, :col].sum(axis = 1)

    section_C = cumulative_df.copy()

    #Section D
    cumulative_df['xaxis_labels'] = cumulative_df.Month.astype(str) + "/" + cumulative_df.Year.astype(str)
    res = cumulative_df.sort_values(['Year', 'Month']).reset_index(drop = True)

    section_D = res

    #return res
    return section_A, section_B, section_C, section_D
```

```
state = "NY"
fuels = ["COW", "PEL", "NG", "NUC", "HYC"]
start = 2003
end = 2004
```

```
section_objects = get_net_gens(state, fuels, start, end)
```

In Section A, a call is made to `get_net_gen` for each fuel to be included in the plot. The dataframe for each fuel is then filtered for the time frame given by `start` and `end`, and appended to a list. Also appended to the list is a dataframe giving total electricity production in the state (indicated by `ALL`). The head of the dataframe for natural gas is shown below.

```
section_objects[0][2].head()
```

	Year	Month	NG
202	2004	12	1992.59225
203	2004	11	2158.02650
204	2004	10	2099.28267
205	2004	9	3322.69592
206	2004	8	2958.44805

In Section B, the dataframes from the previous section are joined by date. Then the variance of each fuel column is computed. The variances are sorted to form a list of fuels by variance. The result of this section is a dataframe containing information on all the fuels, with the fuel columns arranged in order of increasing variance.

Arranging the fuels by variance in the dataframe means that the plotting function will place low-variance fuels at the bottom of the plot, and higher variance fuels at the top. One reason for this design choice is to emphasize that fuels with low variance in electricity production form the "base" of the state's production capacity. A second reason is to locate the most active parts of the plot--those representing high variance fuels--in a place that draws attention to them.

```
section_objects[1].head()
```

	HYC	COW	NUC	PEL	NG	ALL
0	2233.29436	1750.17073	3783.890	1841.83226	1992.59225	11809.40138
1	2079.68961	1569.37453	3081.901	1176.47994	2158.02650	10260.78592
2	1889.57413	1697.85755	2962.928	1081.11342	2099.28267	9911.31058
3	1918.22403	1923.65200	3060.736	1092.62279	3322.69592	11483.11433
4	2019.95839	2038.54376	3584.710	1754.34635	2958.44805	12551.75079

In section C, the value in each fuel column is divided by the corresponding value in `ALL` to give the fraction of total production due to this fuel. The `Year` and `Month` columns are added. Then the cumulative sum across each row is computed. The value in each cell now represents the cumulative sum of the fraction of production of each fuel to the left. For example, for 11/2004, HYC, COW, and NUC together account for 0.655989 of all electricity production.

```
section_objects[2].head()
```

	Year	Month	HYC	COW	NUC	PEL	NG
0	2004	12	0.189112	0.337313	0.657726	0.813690	0.982419
1	2004	11	0.202683	0.355632	0.655989	0.770647	0.980965
2	2004	10	0.190648	0.361953	0.660897	0.769976	0.981783
3	2004	9	0.167047	0.334567	0.601110	0.696260	0.985615
4	2004	8	0.160930	0.323342	0.608936	0.748705	0.984405

In Section D, `xaxis_labels` are added, rows are sorted by date, and the index is reset. The dataframe is now ready to be passed to the plotting function.

```
section_objects[3].head()
```

	Year	Month	HYC	COW	NUC	PEL	NG	xaxis_labels
0	2003	1	0.164903	0.350659	0.658137	0.807586	0.987588	1/2003
1	2003	2	0.156441	0.349634	0.644630	0.814100	0.989087	2/2003
2	2003	3	0.181411	0.360357	0.647444	0.795211	0.987728	3/2003
3	2003	4	0.183343	0.355444	0.601197	0.782947	0.985524	4/2003
4	2003	5	0.206928	0.356840	0.715674	0.807587	0.986780	5/2003

Findings

How does New York's level of electricity consumption change over time?

Based on the plot included in this report, New York's electricity consumption exhibits yearly seasonality. Each year shows a high peak around August and a smaller peak near January. One hypothesis that might explain these peaks is that the high peak is due to increased air conditioner usage, and the low peak is due to increased electric heat usage. The peak due to heating may be smaller because, while some heat is generated by electricity, much is generated by other means. Troughs occurring near May and October may occur at times of the year when air conditioning and heating are used the least. Based on this plot, there does not appear to be an overall increasing or decreasing trend in New York's electricity consumption. Most values are within or near the range typical for their month. One exception is August 2004, when consumption approached an August minimum.

How does a state's level of electricity consumption relate to its production fuel portfolio?

New York's electricity-production fuel portfolio for 2003-2004 consists almost entirely of conventional hydroelectric, coal, nuclear, petroleum liquids, and natural gas. While nuclear and natural gas took turns producing the most electricity, no single fuel was regularly responsible for more than about 25% of total production. Nuclear, petroleum liquids, and natural gas had more variable production than coal and hydroelectric. This may be due to intermittent nuclear reactor shutdowns for testing or maintenance, as well as matching supply to demand by varying fuels that are more easily adjusted, such as petroleum liquids and natural gas. There does not appear to be any trend or structural change in the state's electricity production portfolio during the time period shown.