Final Project: Group 33

Dale Jin, Katherine Tu, Yuliana Zhang

2024-12-02

```
import pandas as pd
import altair as alt
alt.renderers.enable("png")
import time
import warnings
warnings.filterwarnings('ignore')
import os
import json

import geopandas as gpd
from shapely.geometry import Point
from shapely import wkt
import matplotlib.pyplot as plt
import numpy as np
```

1. Data Cleaning

```
# Loading the Data
df_building =
 ↪  pd.read_csv("data/Chicago_Energy_Benchmarking_-_Covered_Buildings_20240715.csv")
df_energy = pd.read_csv("data/Chicago_Energy_Benchmarking_20240715.csv")

chicago_communities = pd.read_csv('data/CommAreas_20241130.csv')
chicago_communities['the_geom'] =
 ↪  chicago_communities['the_geom'].apply(wkt.loads)
```

```
# Merge by building id
df_merge = pd.merge(df_energy, df_building, how = 'inner', left_on = 'ID',
 ↪  right_on = 'Building ID')
```

```
# Filter for necessary variables
df = df_merge[[
    'Data Year',
    'Community Area',
    'Primary Property Type',
    'Gross Floor Area - Buildings (sq ft)',
    'Year Built',
    'Electricity Use (kBtu)',
    'Natural Gas Use (kBtu)',
    'Total GHG Emissions (Metric Tons CO2e)',
    'Latitude_x',
    'Longitude_x'
]]

# Rename
df = df.rename(columns = {
    'Data Year': 'year',
    'Community Area': 'community',
    'Primary Property Type': 'property_type',
    'Gross Floor Area - Buildings (sq ft)': 'area',
    'Year Built': 'built_year',
    'Electricity Use (kBtu)': 'electricity',
    'Natural Gas Use (kBtu)': 'gas',
    'Total GHG Emissions (Metric Tons CO2e)': 'ghg',
    'Latitude_x': 'latitude',
    'Longitude_x': 'longitude'
})

# Treated (prior retrofitted) communities
treat = [
    "Rogers Park","Belmont Cragin","Hermosa","Humboldt Park", "Austin","North
    ↪  Lawndale","South Lawndale","Grand Boulevard","Washington
    ↪  Park","Woodlawn","South Shore","Chatham","South
    ↪  Chicago","Roseland","East Side","McKinley Park","Gage Park","West
    ↪  Englewood","Englewood","Auburn Gresham"]
treat = [community.upper() for community in treat]

# Merge treated communities to main df
df['treated'] = df['community'].isin(treat).astype(int)
```

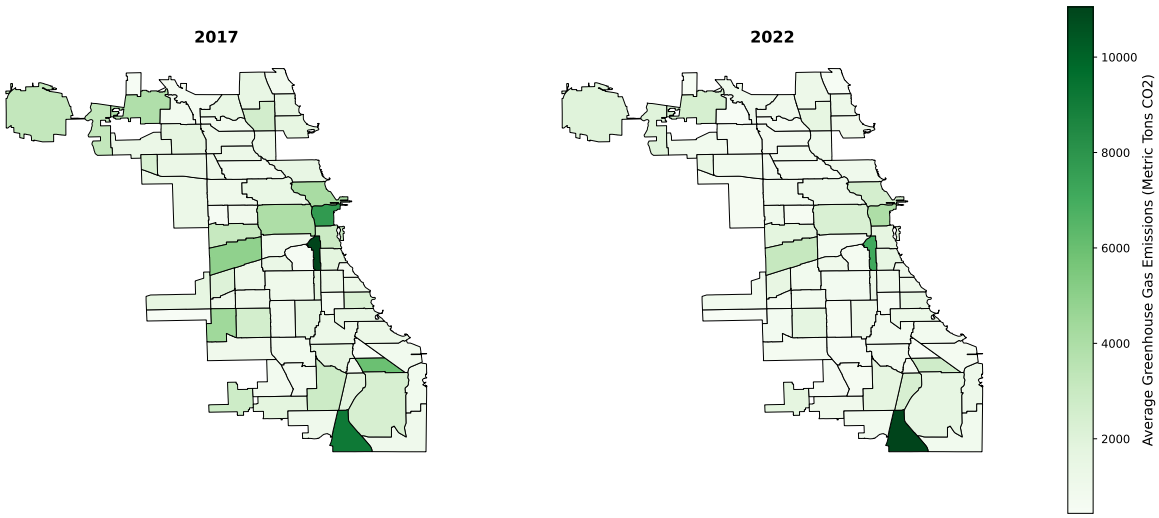
```
# Cleaning Community Names
df['community'] = df['community'].str.upper()
df['community'] = df['community'].str.replace("'", "", regex = False)
df['community'] = df['community'].replace('LAKE VIEW', 'LAKEVIEW')

# Create columns for efficiency
df['ghg_efficiency'] = (df['ghg']/df['area'])*1000
df['elec_efficiency'] = (df['electricity']/df['area'])*1000
df['gas_efficiency'] = (df['gas']/df['area']*1000)

# Create bins for longitudes and latitudes
df['lat_bin'] = ((df["latitude"]//0.01)*0.01).round(2)
df['lon_bin'] = ((df["longitude"]//0.01)*0.01).round(2)
```

```
# Create a df with no NA values
df_clean = df.dropna()
df_clean_2022 = df_clean[df_clean['year'] == 2022]
```

2. Build Geopanda Graphs by Year

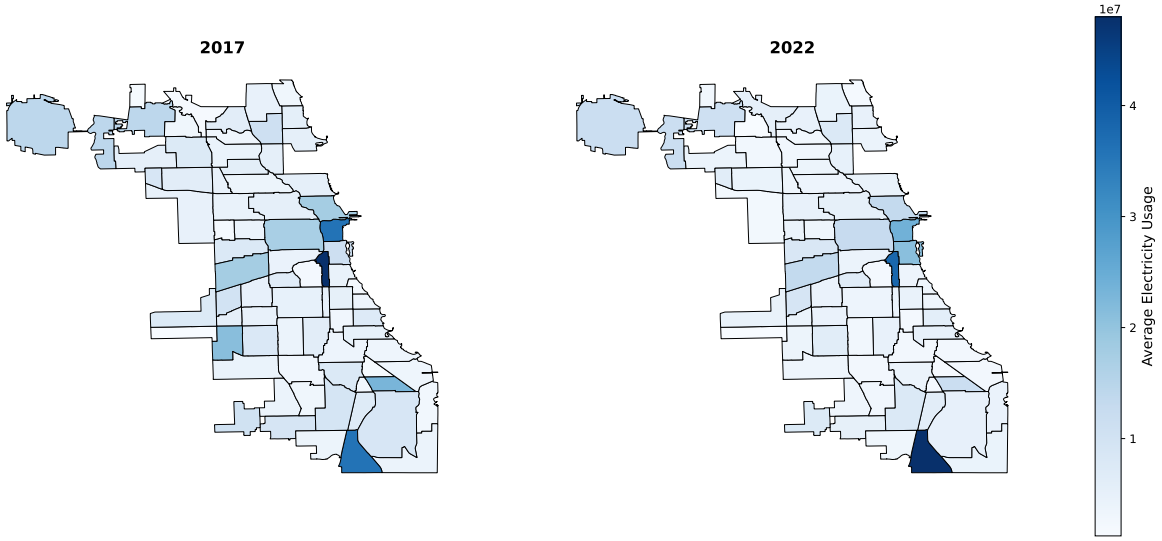
```
## 1. GHG plot
fig, axes = plt.subplots(1, 2, figsize = (20, 8), sharex = True, sharey =
 ↪  True)  # Create 1x2 grid with shared axes

# Data for 2017
df_2017 = df[df['year'] == 2017]
geoplot_2017 = df_2017.groupby('community')[['ghg',
 ↪  'treated']].mean().reset_index()
geoplot_2017 = geoplot_2017.merge(chicago_communities, left_on = 'community',
 ↪  right_on = 'COMMUNITY', how = 'left')
geoplot_2017 = gpd.GeoDataFrame(geoplot_2017, geometry = 'the_geom')

# Plot for 2017
geoplot_2017.plot(
    column = "ghg",
    cmap = "Greens",
    legend = False,
    edgecolor = "black",
    linewidth = 0.5,
    ax = axes[0]
)
```

```
axes[0].set_title("2017", fontsize = 14, fontweight = 'bold')
axes[0].set_axis_off()

# Data for 2022
df_2022 = df[df['year'] == 2022]
geoplot_2022 = df_2022.groupby('community')[['ghg',
 ↪  'treated']].mean().reset_index()
geoplot_2022 = geoplot_2022.merge(chicago_communities, left_on = 'community',
 ↪  right_on = 'COMMUNITY', how = 'left')
geoplot_2022 = gpd.GeoDataFrame(geoplot_2022, geometry = 'the_geom')

# Plot for 2022
geoplot_2022.plot(
    column = "ghg",
    cmap = "Greens",
    legend = False,
    edgecolor = "black",
    linewidth = 0.5,
    ax = axes[1]
)
axes[1].set_title("2022", fontsize = 14, fontweight = 'bold')
axes[1].set_axis_off()

# Add title
fig.suptitle("Greenhouse Gas Emissions by Community (2017 vs 2022)",
             fontsize = 18, fontweight='bold')

# Add legend
legend = plt.cm.ScalarMappable(cmap = "Greens", norm = plt.Normalize(vmin =
 ↪  min(geoplot_2017['ghg'].min(), geoplot_2022['ghg'].min()), vmax =
 ↪  max(geoplot_2017['ghg'].max(), geoplot_2022['ghg'].max())))

# Adjust legend position
cbar = fig.colorbar(legend, ax = axes).set_label("Average Greenhouse Gas
 ↪  Emissions (Metric Tons CO2)", fontsize = 12)
```



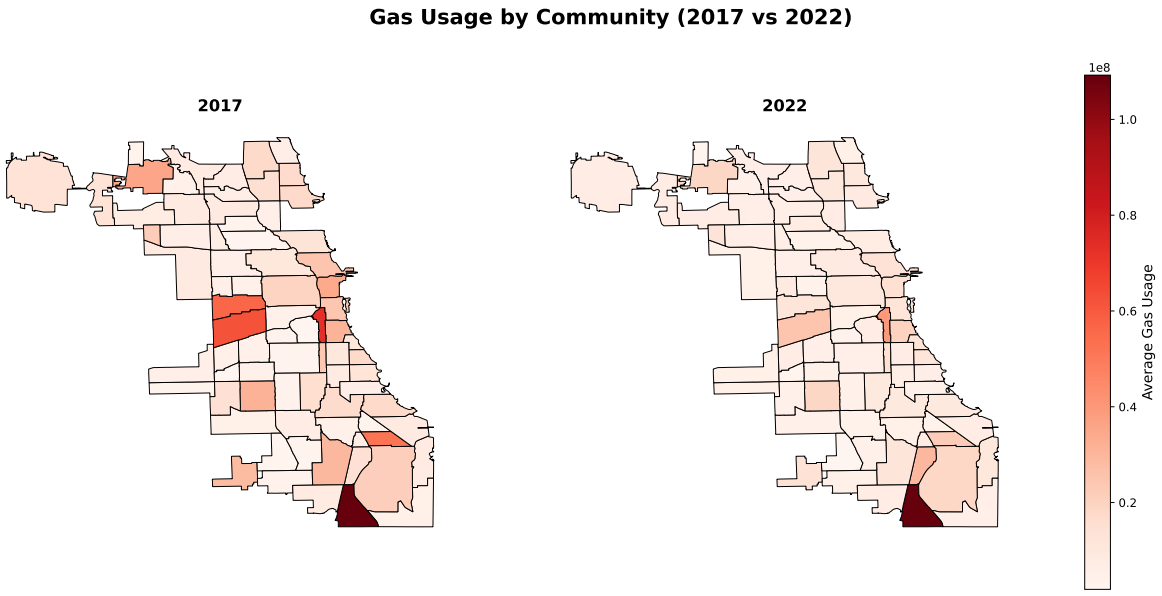

```
## 2. Electricity plot
fig, axes = plt.subplots(1, 2, figsize = (20, 8), sharex = True, sharey =
 ↪  True)  # Create 1x2 grid with shared axes

# Data for 2017
df_2017 = df[df['year'] == 2017]
geoplot_2017 = df_2017.groupby('community')[['electricity',
 ↪  'treated']].mean().reset_index()
geoplot_2017 = geoplot_2017.merge(chicago_communities, left_on = 'community',
 ↪  right_on = 'COMMUNITY', how = 'left')
geoplot_2017 = gpd.GeoDataFrame(geoplot_2017, geometry = 'the_geom')

# Plot for 2017
geoplot_2017.plot(
    column = "electricity",
    cmap = "Blues",
    legend = False,
    edgecolor = "black",
    linewidth = 0.5,
    ax = axes[0]
)
axes[0].set_title("2017", fontsize = 14, fontweight = 'bold')
axes[0].set_axis_off()

# Data for 2022
```

```
df_2022 = df[df['year'] == 2022]
geoplot_2022 = df_2022.groupby('community')[['electricity',
 ↪  'treated']].mean().reset_index()
geoplot_2022 = geoplot_2022.merge(chicago_communities, left_on = 'community',
 ↪  right_on = 'COMMUNITY', how = 'left')
geoplot_2022 = gpd.GeoDataFrame(geoplot_2022, geometry = 'the_geom')

# Plot for 2022
geoplot_2022.plot(
    column = "electricity",
    cmap = "Blues",
    legend = False,
    edgecolor = "black",
    linewidth = 0.5,
    ax = axes[1]
)
axes[1].set_title("2022", fontsize = 14, fontweight = 'bold')
axes[1].set_axis_off()

# Add title
fig.suptitle("Electricity Usage by Community (2017 vs 2022)",
             fontsize = 18, fontweight = 'bold')

# Add legend
legend = plt.cm.ScalarMappable(cmap = "Blues", norm = plt.Normalize(vmin =
 ↪  min(geoplot_2017['electricity'].min(),
 ↪  geoplot_2022['electricity'].min()), vmax =
 ↪  max(geoplot_2017['electricity'].max(),
 ↪  geoplot_2022['electricity'].max())))

# Adjust legend position
cbar = fig.colorbar(legend, ax = axes).set_label("Average Electricity Usage",
 ↪  fontsize=12)
```




```
## 3. Gas plot
fig, axes = plt.subplots(1, 2, figsize = (20, 8), sharex = True, sharey =
 ↪  True)  # Create 1x2 grid with shared axes

# Data for 2017
df_2017 = df[df['year'] == 2017]
geoplot_2017 = df_2017.groupby('community')[['gas',
 ↪  'treated']].mean().reset_index()
geoplot_2017 = geoplot_2017.merge(chicago_communities, left_on = 'community',
 ↪  right_on = 'COMMUNITY', how = 'left')
geoplot_2017 = gpd.GeoDataFrame(geoplot_2017, geometry = 'the_geom')

# Plot for 2017
geoplot_2017.plot(
    column = "gas",
    cmap = "Reds",
    legend = False,
    edgecolor = "black",
    linewidth = 0.5,
    ax = axes[0]
)
axes[0].set_title("2017", fontsize = 14, fontweight = 'bold')
axes[0].set_axis_off()

# Data for 2022
```

```
df_2022 = df[df['year'] == 2022]
geoplot_2022 = df_2022.groupby('community')[['gas',
 ↪  'treated']].mean().reset_index()
geoplot_2022 = geoplot_2022.merge(chicago_communities, left_on = 'community',
 ↪  right_on = 'COMMUNITY', how = 'left')
geoplot_2022 = gpd.GeoDataFrame(geoplot_2022, geometry = 'the_geom')

# Plot for 2022
geoplot_2022.plot(
    column = "gas",
    cmap = "Reds",
    legend = False,
    edgecolor = "black",
    linewidth = 0.5,
    ax = axes[1]
)
axes[1].set_title("2022", fontsize = 14, fontweight = 'bold')
axes[1].set_axis_off()

# Add title
fig.suptitle("Gas Usage by Community (2017 vs 2022)", fontsize = 18,
 ↪  fontweight = 'bold')

# Add legend
legend = plt.cm.ScalarMappable(cmap = "Reds", norm=plt.Normalize(vmin =
 ↪  min(geoplot_2017['gas'].min(), geoplot_2022['gas'].min()), vmax =
 ↪  max(geoplot_2017['gas'].max(), geoplot_2022['gas'].max())))

# Adjust legend position
cbar = fig.colorbar(legend, ax = axes).set_label("Average Gas Usage",
 ↪  fontsize = 12)
```



3. Shiny dynamic plots 1 – community trends

```
## Create dynamic plot for the average trend by year by community
average_use = df_clean.groupby(["year","community"], as_index =
 ↪  False)[["elec_efficiency","gas_efficiency","ghg_efficiency"]].mean()

average_use = average_use.rename(columns = {
    'elec_efficiency':'Electricity',
    'gas_efficiency':'Gas',
    'ghg_efficiency':'Greenhouse Gas'
})

average_use.to_csv("./shiny-app/community_trends/community_use.csv")
```

4. Shiny dynamic plots 2 – property trends

```
## Create dynamic plot for the trend of energy usage by building type
property_type_trend = df_clean.groupby(["year","property_type"], as_index =
 ↪  False)[["elec_efficiency","gas_efficiency","ghg_efficiency"]].mean()

property_type_trend = property_type_trend.rename(columns= {
    'elec_efficiency':'Electricity',
    'gas_efficiency':'Gas',
    'ghg_efficiency':'Greenhouse Gas'
})
```

```
# Save it as a csv
property_type_trend.to_csv("./shiny-app/property_trends/property_type_trend.csv")
```

4. Dynamic altair maps

```
## Top 500 buildings with the lowest energy efficiencies by each property
 ↪  type in 2022

#sort value by electricity, gas, and ghg efficiencies
top_ghg = df_clean_2022.sort_values(by = 'ghg_efficiency', ascending =
 ↪  False).head(500)
top_elec = df_clean_2022.sort_values(by = 'elec_efficiency', ascending =
 ↪  False).head(500)
top_gas = df_clean_2022.sort_values(by = 'gas_efficiency', ascending =
 ↪  False).head(500)

#Label their datasets
top_ghg['type of efficiency'] = 'Greenhouse Gas'
top_elec['type of efficiency'] = 'Electricity'
top_gas['type of efficiency'] = 'Gas'

# Concatenate the three datasets
least_efficient = pd.concat([top_ghg, top_elec, top_gas])

# Save as a new CSV
least_efficient.to_csv("./shiny-app/interactive_map/least_efficient.csv")
```

```
## Create a map to show the distribution of these buildings

#Import Chicago Map
import requests

url =
 ↪  "http://data.cityofchicago.org/api/geospatial/bbvz-uum9?method=export&format=GeoJSON"

response = requests.get(url)

if response.status_code == 200:
    with open("./shiny-app/interactive_map/chicago-boundaries.geojson","wb")
     ↪  as file:
```

```
        file.write(response.content)
    print("GeoJSON file downloaded successfully!")
else:
    print(f"Failed to download file. Status code: {response.status_code}")
```

```
GeoJSON file downloaded successfully!
```