DemographicAnalysis

December 17, 2025

[1]:
```
# Imports
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import os
import sys
os.makedirs('../figures', exist_ok = True)
# Define the relative path to the 'src' folder
module_path = os.path.abspath(os.path.join('..', 'src'))
# Add the path to sys.path
if module_path not in sys.path:
    sys.path.append(module_path)
import analysis_utils as utils
```

[2]:
```
df_od_age_group = pd.read_csv("../data/overdose_age_data_clean.csv")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[2], line 1
----> 1 df_od_age_group = pd.read_csv(                                      )

File /srv/conda/envs/notebook/lib/python3.12/site-packages/pandas/io/parsers/
 ↪readers.py:1026, in read_csv(filepath_or_buffer, sep, delimiter, header,␣
 ↪names, index_col, usecols, dtype, engine, converters, true_values,␣
 ↪false_values, skipinitialspace, skiprows, skipfooter, nrows, na_values,␣
 ↪keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates,␣
 ↪infer_datetime_format, keep_date_col, date_parser, date_format, dayfirst,␣
 ↪cache_dates, iterator, chunksize, compression, thousands, decimal,␣
 ↪lineterminator, quotechar, quoting, doublequote, escapechar, comment,␣
 ↪encoding, encoding_errors, dialect, on_bad_lines, delim_whitespace,␣
 ↪low_memory, memory_map, float_precision, storage_options, dtype_backend)
   1013 kwds_defaults = _refine_defaults_read(
   1014     dialect,
   1015     delimiter,
   (…)   1022     dtype_backend=dtype_backend,
   1023 )
   1024 kwds.update(kwds_defaults)
-> 1026 return _read(filepath_or_buffer, kwds)
```

```
File /srv/conda/envs/notebook/lib/python3.12/site-packages/pandas/io/parsers/
 ↪readers.py:620, in _read(filepath_or_buffer, kwds)
    617 _validate_names(kwds.get("names", None))
    619 # Create the parser.
--> 620 parser = TextFileReader(filepath_or_buffer, **kwds)
    622 if chunksize or iterator:
    623     return parser

File /srv/conda/envs/notebook/lib/python3.12/site-packages/pandas/io/parsers/
 ↪readers.py:1620, in TextFileReader.__init__(self, f, engine, **kwds)
   1617     self.options["has_index_names"] = kwds["has_index_names"]
   1619 self.handles: IOHandles | None = None
-> 1620 self._engine = self._make_engine(f, self.engine)

File /srv/conda/envs/notebook/lib/python3.12/site-packages/pandas/io/parsers/
 ↪readers.py:1880, in TextFileReader._make_engine(self, f, engine)
   1878     if "b" not in mode:
   1879         mode += "b"
-> 1880 self.handles = get_handle(
   1881     f,
   1882     mode,
   1883     encoding=self.options.get(          , None),
   1884     compression=self.options.get(             , None),
   1885     memory_map=self.options.get(            , False),
   1886     is_text=is_text,
   1887     errors=self.options.get(                 ,         ),
   1888     storage_options=self.options.get(                , None),
   1889 )
   1890 assert self.handles is not None
   1891 f = self.handles.handle

File /srv/conda/envs/notebook/lib/python3.12/site-packages/pandas/io/common.py:
 ↪873, in get_handle(path_or_buf, mode, encoding, compression, memory_map,␣
 ↪is_text, errors, storage_options)
    868 elif isinstance(handle, str):
    869     # Check whether the filename is to be opened in binary mode.
    870     # Binary mode does not support 'encoding' and 'newline'.
    871     if ioargs.encoding and "b" not in ioargs.mode:
    872         # Encoding
--> 873         handle = open(
    874             handle,
    875             ioargs.mode,
    876             encoding=ioargs.encoding,
    877             errors=errors,
    878             newline=  ,
    879         )
    880     else:
    881         # Binary mode
```

```
    882         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory: '../data/
 ↪overdose_age_data_clean.csv'
```

[ ]:
```
df_od_age_group.head(5)
```

[ ]:
```
# filter for all drug overdose deaths
all_od_deaths = df_od_age_group[df_od_age_group["PANEL_NUM"] == 0]
# remove "All from age_group and sex since we want to look at distinct␣
 ↪age_groups and distinct sex
all_od_deaths = all_od_deaths[(all_od_deaths["age_group"] != "All") &␣
 ↪(all_od_deaths["sex"] != "All")]
all_od_deaths.head(5)
```

[ ]:
```
# get female and make deaths by year
male_female_ave = all_od_deaths[(all_od_deaths["sex"] == "Female") |␣
 ↪(all_od_deaths["sex"] == "Male")].groupby(["YEAR", "sex"]).mean("ESTIMATE").
 ↪reset_index()

male_female_ave.head(5)
```

[ ]:
```
# plot year vs estimated deaths for each sex

fig, ax = plt.subplots(figsize=(10, 5))
sns.scatterplot(data = male_female_ave, x='YEAR', y='ESTIMATE', hue='sex')
sns.lineplot(data = male_female_ave, x='YEAR', y='ESTIMATE', hue='sex')
# Add a legend and show the plot
ax.set_ylabel('Death Rate (per 100,000)')

# Get the current handles and labels
handles, labels = ax.get_legend_handles_labels()

# Create a dictionary to store unique labels and handles, which automatically␣
 ↪removes duplicates
# Using dict.fromkeys preserves the insertion order in Python 3.7+
unique_labels_handles = dict(zip(labels, handles))

# Extract the unique handles and labels
unique_handles = unique_labels_handles.values()
unique_labels = unique_labels_handles.keys()

# Apply the unique handles and labels to the legend
ax.legend(unique_handles, unique_labels, title='Sex')
```

```
ax.set_title('Drug overdose death rates by sex')

fig.savefig('../figures/drug_overdose_death_rates_by_sex.png', dpi=300,␣
 ↪bbox_inches='tight')
print("  Saved: figures/drug_overdose_death_rates_by_sex.png")
```

[ ]:
```
# filter data by age-group
#Trend by Age group

plt.figure(figsize=(14, 8))

df_od_age_group_distinct = df_od_age_group[df_od_age_group["age_group"] !=␣
 ↪"All"]
age_groups = df_od_age_group_distinct['age_group'].unique()

for age_grp in age_groups:
    age_trend = df_od_age_group_distinct[
        (df_od_age_group_distinct['age_group'] == age_grp)].
 ↪groupby('YEAR')['ESTIMATE'].mean().reset_index()

    if len(age_trend) > 0:
        plt.plot(age_trend['YEAR'], age_trend['ESTIMATE'],
                 marker='o', linewidth=2, label=age_grp, markersize=4)

plt.title('Drug overdose death rates by age group',
          fontsize=16, fontweight='bold', pad=20)
plt.xlabel('Year', fontsize=12, fontweight='bold')
plt.ylabel('Death Rate (per 100,000)', fontsize=12, fontweight='bold')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', fontsize=9)
plt.grid(True, alpha=0.3)
plt.tight_layout()

plt.savefig('../figures/drug_overdose_death_rates_by_age_group.png', dpi=300,␣
 ↪bbox_inches='tight')
print("  Saved: figures/drug_overdose_death_rates_by_age_group.png")
```

[ ]:
```
# heatmap of rates by age x year
# create heatmap_data for distinct age_groups, not "All", and for sex not "All"
heatmap_data = all_od_deaths[(all_od_deaths["age_group"] != "All") &␣
 ↪(all_od_deaths["sex"] != "All")]

heatmap_df = heatmap_data[['YEAR', 'sex', 'age_group', 'ESTIMATE']]
heatmap_grouped = heatmap_df.groupby(['YEAR', 'sex', 'age_group']).
 ↪mean("ESTIMATE").reset_index()
heatmap_grouped
```

[ ]:
```
# Pivot the data into a matrix format (months as rows, years as columns)
heatmap_matrix = heatmap_grouped.pivot_table(index="YEAR", columns="age_group",␣
 ↪values="ESTIMATE")
ax = sns.heatmap(heatmap_matrix, annot=True, cmap="YlGnBu")

# Get the Figure object and save it
fig = ax.get_figure()

fig.savefig('../figures/heatmap_by_age_group.png', dpi=300, bbox_inches='tight')
print("  Saved: figures/heatmap_by_age_group.png")
```

[ ]:
```
# Pivot the data into a matrix format (months as rows, years as columns)
heatmap_matrix = heatmap_grouped.pivot_table(index="YEAR", columns="sex",␣
 ↪values="ESTIMATE")
ax = sns.heatmap(heatmap_matrix, annot=True, cmap="YlGnBu")

# Get the Figure object and save it
fig = ax.get_figure()

fig.savefig('../figures/heatmap_by_sex.png', dpi=300, bbox_inches='tight')
print("  Saved: figures/heatmap_by_sex.png")
```

[ ]:
```
# compute rate of change
heatmap_data_m = heatmap_data[heatmap_data["sex"] == "Male"]
df_age_rate_of_change_m = utils.compute_rate_change(heatmap_data_m, 1999, 2018,␣
 ↪group_col="age_group")
df_age_rate_of_change_m
```

[ ]:
```
# compute rate of change
heatmap_data_f = heatmap_data[heatmap_data["sex"] == "Female"]
df_age_rate_of_change_f = utils.compute_rate_change(heatmap_data_f, 1999, 2018,␣
 ↪group_col="age_group")
df_age_rate_of_change_f
```

[ ]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

attr_name = 'percent_change'
attr_label = 'Percent change'
title_label = 'Drug OD Deaths Percent Rate of Change for '

dataset = df_age_rate_of_change_m
sex_label = "Male"

if len(dataset) > 0:
    n_colors = len(dataset)
    colors = sns.color_palette("YlGnBu_r", n_colors)
    bars = ax1.bar(dataset['age_group'], dataset[attr_name],
```

```
                    color=colors, edgecolor='black', linewidth=1.5)
    ax1.set_xticks(range(len(dataset)), dataset['age_group'],
                rotation=45, ha='right', fontsize=10)

    ax1.set_title(title_label+sex_label,
                fontsize=12, fontweight='bold', pad=20)
    ax1.set_ylabel(attr_label,
                fontsize=10, fontweight='bold')
    for i, bar in enumerate(bars):
        height = bar.get_height()
        ax1.text(bar.get_x() + bar.get_width()/2., height,
                 f'{height:.1f}', ha='center', va='bottom', fontsize=10,
                 fontweight='bold')

dataset = df_age_rate_of_change_f
sex_label = "Female"

if len(dataset) > 0:

    n_colors = len(dataset)
    colors = sns.color_palette("YlGnBu_r", n_colors)
    bars = ax2.bar(dataset['age_group'], dataset[attr_name],
                    color=colors, edgecolor='black', linewidth=1.5)
    ax2.set_xticks(range(len(dataset)), dataset['age_group'],
                rotation=45, ha='right', fontsize=10)

    ax2.set_title(title_label+sex_label,
                fontsize=12, fontweight='bold', pad=20)
    ax2.set_ylabel(attr_label,
                fontsize=10, fontweight='bold')
    for i, bar in enumerate(bars):
        height = bar.get_height()
        ax2.text(bar.get_x() + bar.get_width()/2., height,
                 f'{height:.1f}', ha='center', va='bottom', fontsize=10,
                 fontweight='bold')

plt.tight_layout()
# Display the plots

fig.savefig('../figures/drug_overdose_deaths_percent_change_by_sex.png',␣
 ↪dpi=300, bbox_inches='tight')
print("  Saved: figures/drug_overdose_deaths_percent_change_by_sex.png")
```

[ ]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

attr_name = 'absolute_change'
attr_label = 'Absolute change'
```

```
title_label = 'Drug OD Deaths Absolute Rate of Change for '

dataset = df_age_rate_of_change_m
sex_label = "Male"

if len(dataset) > 0:
    n_colors = len(dataset)
    colors = sns.color_palette("YlGnBu_r", n_colors)
    bars = ax1.bar(dataset['age_group'], dataset[attr_name],
                   color=colors, edgecolor='black', linewidth=1.5)
    ax1.set_xticks(range(len(dataset)), dataset['age_group'],
               rotation=45, ha='right', fontsize=10)

    ax1.set_title(title_label+sex_label,
              fontsize=12, fontweight='bold', pad=20)
    ax1.set_ylabel(attr_label,
               fontsize=10, fontweight='bold')
    for i, bar in enumerate(bars):
        height = bar.get_height()
        ax1.text(bar.get_x() + bar.get_width()/2., height,
                 f'{height:.1f}', ha='center', va='bottom', fontsize=10,
                 fontweight='bold')

dataset = df_age_rate_of_change_f
sex_label = "Female"

if len(dataset) > 0:

    n_colors = len(dataset)
    colors = sns.color_palette("YlGnBu_r", n_colors)
    bars = ax2.bar(dataset['age_group'], dataset[attr_name],
                   color=colors, edgecolor='black', linewidth=1.5)
    ax2.set_xticks(range(len(dataset)), dataset['age_group'],
               rotation=45, ha='right', fontsize=10)

    ax2.set_title(title_label+sex_label,
              fontsize=12, fontweight='bold', pad=20)
    ax2.set_ylabel(attr_label,
               fontsize=10, fontweight='bold')
    for i, bar in enumerate(bars):
        height = bar.get_height()
        ax2.text(bar.get_x() + bar.get_width()/2., height,
                 f'{height:.1f}', ha='center', va='bottom', fontsize=10,
                 fontweight='bold')

plt.tight_layout()
```

```
fig.savefig('../figures/drug_overdose_deaths_absolute_change_by_sex.png',␣
 ↪dpi=300, bbox_inches='tight')
print("  Saved: figures/drug_overdose_deaths_absolute_change_by_sex.png")
```

[ ]:
```
# Male groups with largest percent change
attr_name = 'percent_change'
dataset = df_age_rate_of_change_m
threshold = 200

dataset_sorted = dataset.sort_values(by = attr_name, ascending=False)
dataset_sorted_subset = dataset_sorted[dataset_sorted[attr_name] > threshold]
dataset_sorted_subset = dataset_sorted_subset[["age_group", attr_name]]
print(f"Top age groups for males experiencing the greater than {threshold}␣
 ↪percent change:")
dataset_sorted_subset.head(10)
```

[ ]:
```
# Male groups with largest change
attr_name = 'absolute_change'
dataset = df_age_rate_of_change_m
threshold = 10

dataset_sorted = dataset.sort_values(by = attr_name, ascending=False)
dataset_sorted_subset = dataset_sorted[dataset_sorted[attr_name] > threshold]
dataset_sorted_subset = dataset_sorted_subset[["age_group", attr_name]]
print("Top age groups for males experiencing the largest absolute change:")
dataset_sorted_subset.head(10)
```

[ ]:
```
# Female groups with largest change
attr_name = 'percent_change'
dataset = df_age_rate_of_change_f
threshold = 200

dataset_sorted = dataset.sort_values(by = attr_name, ascending=False)
dataset_sorted_subset = dataset_sorted[dataset_sorted[attr_name] > threshold]
dataset_sorted_subset = dataset_sorted_subset[["age_group", attr_name]]
print(f"Top age groups for females experiencing the greater than {threshold}␣
 ↪percent change:")
dataset_sorted_subset.head(10)
```

[ ]:
```
# Female groups with largest change
attr_name = 'absolute_change'
dataset = df_age_rate_of_change_f
threshold = 10

dataset_sorted = dataset.sort_values(by = attr_name, ascending=False)
dataset_sorted_subset = dataset_sorted[dataset_sorted[attr_name] > threshold]
dataset_sorted_subset = dataset_sorted_subset[["age_group", attr_name]]
```

```
print("Top age groups for females experiencing the largest absolute change:")
dataset_sorted_subset.head(10)
```

[ ]: