WileyWinters_finalProject

March 9, 2024

1 MSDS 670 Final Project

Wiley Winters MSDS 670 Data Visualization 2024-MAR-10

Import required packages and libraries

[1]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
from matplotlib import rcParams
from labellines import labelLine, labelLines
import numpy as np

# plotly
from plotly.offline import init_notebook_mode, iplot, plot
import plotly as py
import plotly.express as px
init_notebook_mode(connected=True)

# Suppress Warnings
import warnings
warnings.filterwarnings('ignore')

# Set seaborn autoconfig to True
rcParams.update({'figure.autolayout': True})
```

Read datasets into Pandas DataFrames. The main dataset is the one from the NCHS with the 10 leading causes of death in the United States. The `states1999` and `states2011` contain population data for each state plus the total for the United States. In order to have the option to create a choropleth map of the United States, an abbreviation dataset was created.

[2]:

```
causes_df = pd.read_csv('../data/NCHS_-_Leading_Causes_of_Death__United_States.
  ↪csv')
states2000_df = pd.read_csv('../data/states2000-2010.csv')
states2010_df = pd.read_csv('../data/states2011-2019.csv')
abbrev_df = pd.read_csv('../data/stateAbb.csv')
```

```
print(causes_df.head())
print(states2000_df.head())
print(states2010_df.head())
print(abbrev_df.head())
```

```
   Year                                    113 Cause Name  \
0  2017  Accidents (unintentional injuries) (V01-X59,Y8…
1  2017  Accidents (unintentional injuries) (V01-X59,Y8…
2  2017  Accidents (unintentional injuries) (V01-X59,Y8…
3  2017  Accidents (unintentional injuries) (V01-X59,Y8…
4  2017  Accidents (unintentional injuries) (V01-X59,Y8…

               Cause Name          State  Deaths  Age-adjusted Death Rate
0  Unintentional injuries  United States  169936                     49.4
1  Unintentional injuries        Alabama    2703                     53.8
2  Unintentional injuries         Alaska     436                     63.7
3  Unintentional injuries        Arizona    4184                     56.2
4  Unintentional injuries       Arkansas    1625                     51.8
   year          state  population
0  2000  United States   282162411
1  2000        Alabama     4452173
2  2000         Alaska      627963
3  2000        Arizona     5160586
4  2000       Arkansas     2678588
   year          state  population
0  2011  United States   311583481
1  2011        Alabama     4799069
2  2011         Alaska      722128
3  2011        Arizona     6472643
4  2011       Arkansas     2940667
        state abbreviation
0     Alabama           AL
1      Alaska           AK
2     Arizona           AZ
3    Arkansas           AR
4  California           CA
```

The column names in the causes_df DataFrame are not in a user friendly format. I will rename them.

```
[3]: causes_df.rename({'Year':'year', '113 Cause Name': '113_cause_name',
                      'Cause Name':'cause_name', 'State': 'state', 'Deaths':'deaths',
                      'Age-adjusted Death Rate':'age_adjusted'}, axis=1,␣
      ↪inplace=True)
     causes_df.columns
```

```
[3]: Index(['year', '113_cause_name', 'cause_name', 'state', 'deaths',
            'age_adjusted'],
      dtype='object')
```

Combine all DataFrames into one.

[4]:

```
# Concatenate states1999_df and states2011_df
states = pd.concat([states2000_df, states2010_df], ignore_index=True)
# Merge DataFrames and add abbreviations
all_df = pd.merge(causes_df, states, on=['year', 'state'], how='inner')
all_df = all_df.merge(abbrev_df, on='state', how='left')
```

Ensure all_df is sane enough to use

[5]:

```
print(all_df.info())
print('\\nNaN Values:\\n', all_df.isna().sum())
print('\\nDuplicates: ', all_df.duplicated().sum())
print('\\nSize: ', all_df.size)
print('\\nDistribution:\\n', all_df.describe().T)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10296 entries, 0 to 10295
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   year            10296 non-null  int64
 1   113_cause_name  10296 non-null  object
 2   cause_name      10296 non-null  object
 3   state           10296 non-null  object
 4   deaths          10296 non-null  int64
 5   age_adjusted    10296 non-null  float64
 6   population      10296 non-null  int64
 7   abbreviation    10098 non-null  object
dtypes: float64(1), int64(3), object(4)
memory usage: 643.6+ KB
None
\nNaN Values:\n year                0
113_cause_name      0
cause_name          0
state               0
deaths              0
age_adjusted        0
population          0
abbreviation      198
dtype: int64
\nDuplicates:  0
\nSize:  82368
\nDistribution:\n                  count          mean           std       min
25%  \
year          10296.0  2.008500e+03  5.188379e+00    2000.0     2004.00
deaths        10296.0  1.548406e+04  1.131075e+05      21.0      617.75
```

```
age_adjusted  10296.0  1.266776e+02  2.222074e+02       2.6       19.20
population    10296.0  1.171272e+07  4.157954e+07  494300.0  1716741.25

                    50%         75%          max
year             2008.5     2013.00       2017.0
deaths           1727.5     5756.50    2813503.0
age_adjusted       35.9      151.10       1061.2
population    4336593.5  7220489.75  325122128.0
```

[6]:
```
all_df.head()
```

[6]:
```
   year                                     113_cause_name  \
0  2017  Accidents (unintentional injuries) (V01-X59,Y8…
1  2017  Accidents (unintentional injuries) (V01-X59,Y8…
2  2017  Accidents (unintentional injuries) (V01-X59,Y8…
3  2017  Accidents (unintentional injuries) (V01-X59,Y8…
4  2017  Accidents (unintentional injuries) (V01-X59,Y8…

               cause_name          state  deaths  age_adjusted  population  \
0  Unintentional injuries  United States  169936          49.4   325122128
1  Unintentional injuries        Alabama    2703          53.8     4874486
2  Unintentional injuries         Alaska     436          63.7      739700
3  Unintentional injuries        Arizona    4184          56.2     7044008
4  Unintentional injuries       Arkansas    1625          51.8     3001345

  abbreviation
0          NaN
1           AL
2           AK
3           AZ
4           AR
```

The `113_cause_name` column appears to be a more complicated version of `cause_name`. I will drop the `113_cause_name` column. It is not required for this analysis.

[7]:
```
all_df.drop('113_cause_name', axis=1, inplace=True)
```

When researching this project, I discovered that the *crude death rate* is often used by researchers as the death rate measure of choice. The formula for it is $crudedeathrate = (numberofdeaths/totalpopulation) * 100,000$. This will give the *crude death rate* per *100,000* people. This will make it easier to compare states with large and small populations without having scaling issues.

[8]:
```
# Create the crude_deaths column by calculating the crude death rate per 100,000
all_df['crude_deaths'] = round((all_df['deaths'] / all_df['population']) *␣
 ↪100000)
```

```
# Take a look at the results
all_df.head()
```

```
[8]:    year              cause_name          state  deaths  age_adjusted  \
     0  2017  Unintentional injuries  United States  169936          49.4
     1  2017  Unintentional injuries        Alabama    2703          53.8
     2  2017  Unintentional injuries         Alaska     436          63.7
     3  2017  Unintentional injuries        Arizona    4184          56.2
     4  2017  Unintentional injuries       Arkansas    1625          51.8

        population abbreviation  crude_deaths
     0   325122128          NaN          52.0
     1     4874486           AL          55.0
     2      739700           AK          59.0
     3     7044008           AZ          59.0
     4     3001345           AR          54.0
```

Explore the dataset to see what I have to work with

```
[9]: print('start--> ', all_df.year.min())
     print('end-->   ', all_df.year.max())
```

```
start-->  2000
end-->    2017
```

```
[10]: print('age min: ', all_df.age_adjusted.min())
      print('age max: ', all_df.age_adjusted.max())
      print('crude_deaths min: ', all_df.crude_deaths.min())
      print('crude_deaths max: ', all_df.crude_deaths.max())
      print('deaths min: ', all_df.deaths.min())
      print('deaths max: ', all_df.deaths.max())
```

```
age min:  2.6
age max:  1061.2
crude_deaths min:  3.0
crude_deaths max:  1281.0
deaths min:  21
deaths max:  2813503
```

```
[11]: all_df.value_counts('year')
```

```
[11]: year
      2000    572
      2001    572
      2016    572
      2015    572
      2014    572
```

```
2013    572
2012    572
2011    572
2010    572
2009    572
2008    572
2007    572
2006    572
2005    572
2004    572
2003    572
2002    572
2017    572
Name: count, dtype: int64
```

[12]:

```python
all_df['cause_name'].value_counts()
```

[12]:

```
cause_name
Unintentional injuries     936
All causes                 936
Alzheimer's disease        936
Stroke                     936
CLRD                       936
Diabetes                   936
Heart disease              936
Influenza and pneumonia    936
Suicide                    936
Cancer                     936
Kidney disease             936
Name: count, dtype: int64
```

The `Unintentional injuries` cause name is just another name for accidental death. I will change the value to be `Accidents`. It is easier to read.

[13]:

```python
all_df.cause_name = all_df.cause_name.apply(lambda x: 'Accidents'
                                            if x == 'Unintentional injuries'
                                            else x)
```

Another cause name that many people may not be familiar with is *CLRD*. It stands for *Chronic Lower Respiratory Disease*. It is a group of disorders affecting the lungs and airways and is one of the leading causes of death in the United States. I will rename this cause to `Respiratory disease`. It is easier to understand.

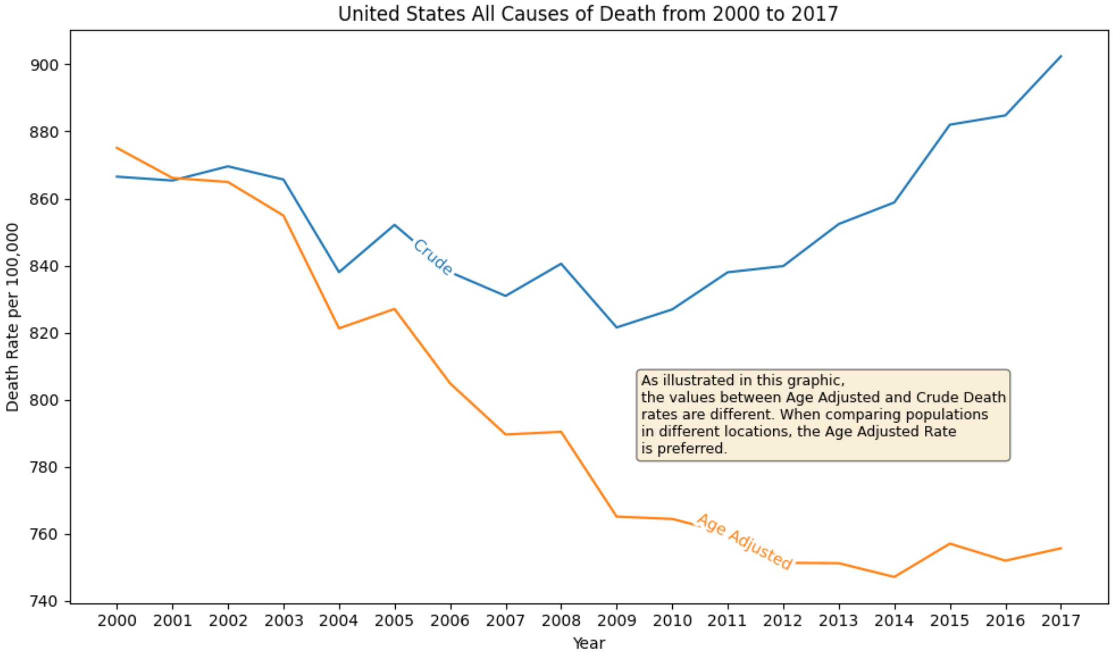
[14]:

```python
all_df.cause_name = all_df.cause_name.apply(lambda x: 'Respiratory disease'
                                            if x == 'CLRD' else x)
```

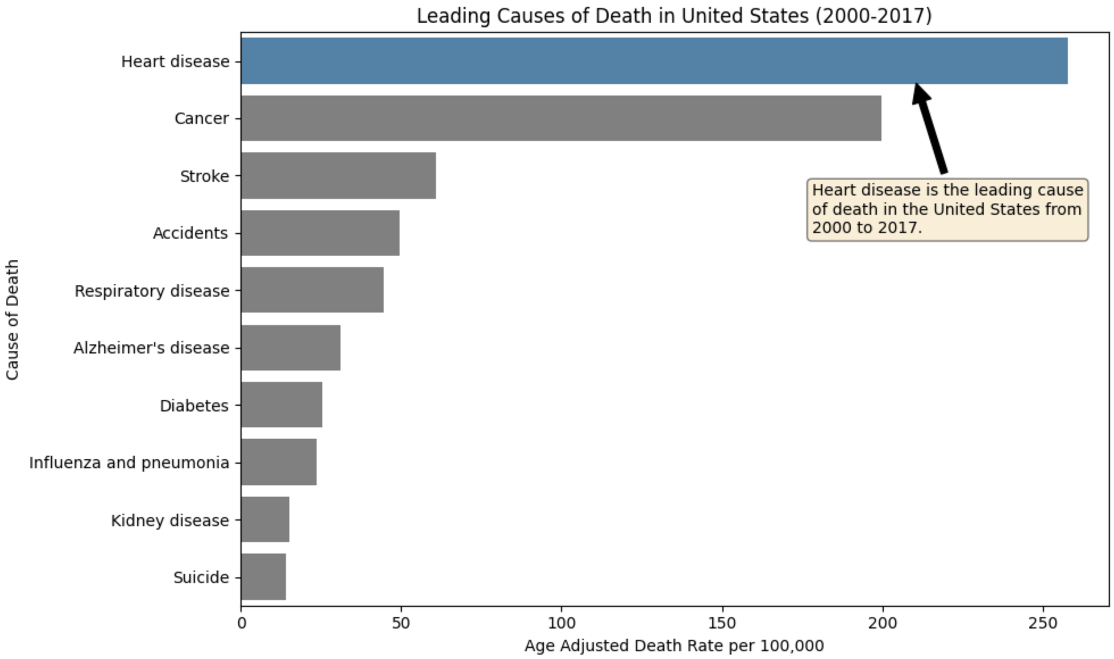
Look at some groupings

[15]:

```
# filter on all causes and break into crude death and age adjusted rates
all = all_df[(all_df.cause_name == 'All causes') & \
             (all_df.state != 'United States')]
crude = all.groupby(['year', 'state', 'cause_name']).agg({'crude_deaths':␣
 ↪'max'}). \
            reset_index()
age = all.groupby(['year', 'state', 'cause_name']).agg({'age_adjusted': 'max'}).
 ↪ \
          reset_index()

# plot crude and age adjusted death rates
fig, ax = plt.subplots(figsize=(10, 6))
ax.set(xlabel='Year', ylabel='Death Rate per 100,000',
       title='United States All Causes of Death from 2000 to 2017')
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
text1 = '\n'.join(('As illustrated in this graphic,',
                   'the values between Age Adjusted and Crude Death',
                   'rates are different. When comparing populations',
                   'in different locations, the Age Adjusted Rate',
                   'is preferred.'))
ax.xaxis.set_ticks(np.arange(2000, 2018, 1))
p = sns.lineplot(data=crude, x='year', y='crude_deaths', ci=None, label='Crude')
p = sns.lineplot(data=age, x='year', y='age_adjusted', ci=None, label='Age␣
 ↪Adjusted')
ax.text(0.55, 0.4, text1, transform=ax.transAxes, fontsize=9,
        verticalalignment='top', bbox=props)
ax.get_legend().remove()
labelLines(plt.gca().get_lines())
plt.show()
fig.savefig('../images/compareCDR-AADR.png', bbox_inches='tight', dpi=300)
```



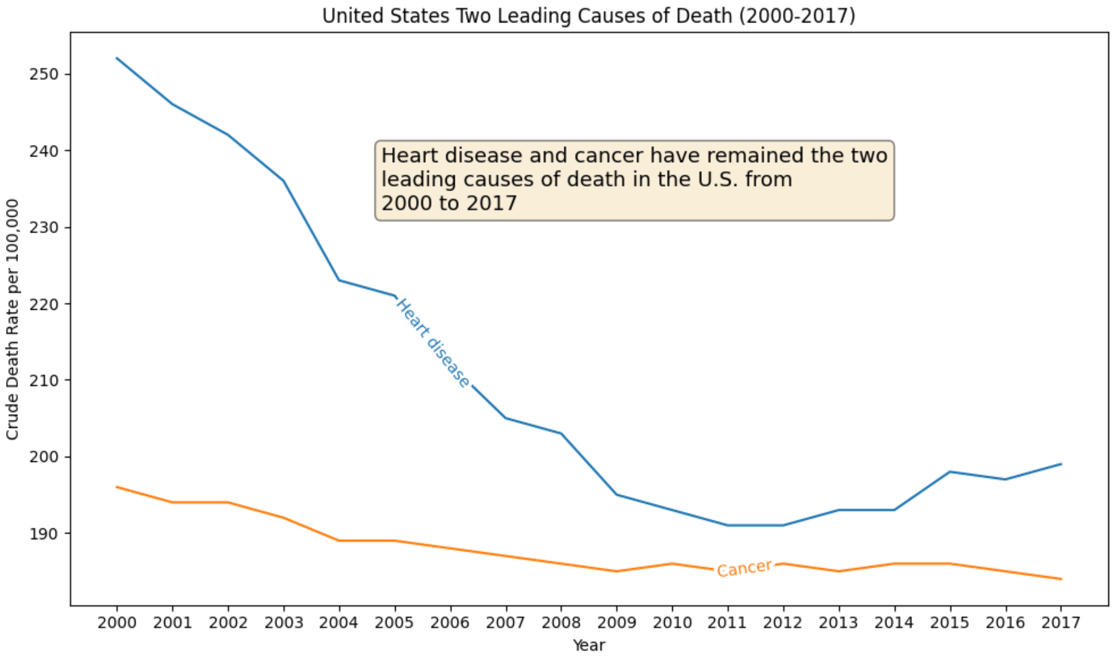
[16]:
```
# filter data
not_all = all_df[(all_df.cause_name != 'All causes') & \
                 (all_df.state == 'United States')]
p_usa = not_all.groupby('cause_name').agg({'age_adjusted':'max'}). \
                sort_values('age_adjusted', ascending=False).reset_index()

# Create bar plot
fig, ax = plt.subplots(figsize=(10,6))
cols = ['grey' if (x < max(p_usa.age_adjusted)) else 'steelblue' \
               for x in p_usa.age_adjusted]
sns.barplot(data=p_usa, y='cause_name', x='age_adjusted', ci=None, palette=cols)
ax.set(xlabel='Age Adjusted Death Rate per 100,000', ylabel='Cause of Death',
       title='Leading Causes of Death in United States (2000-2017)')
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
text1 = '\n'.join(('Heart disease is the leading cause',
                   'of death in the United States from',
                   '2000 to 2017.'))
ax.annotate(text1, xy=(210,0.3), xytext=(178,3), bbox=props,
            fontsize=10, arrowprops=dict(facecolor='black', shrink=0.05))
plt.show()
fig.savefig('../images/allLeadingCauses.png', bbox_inches='tight', dpi=300)
```



Has the leading causes of death remained constant through the time period of this analysis?

```
[17]: # Create filters
usa = all_df[all_df.state == 'United States']
causes = ['Stroke', 'Accidents', 'Respiratory disease', 'Alzheimer\'s disease',
          'Diabetes', 'Influenza and pneumonia', 'Kidney disease', 'Suicide']
t_causes = ['Heart disease', 'Cancer']

# Configure two plotting environments
fig, ax = plt.subplots(figsize=(10,6))
ax.set(xlabel='Year', ylabel='Crude Death Rate per 100,000',
       title='United States Two Leading Causes of Death (2000-2017)')
ax.xaxis.set_ticks(np.arange(2000, 2018, 1))
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
text1 = '\n'.join(('Heart disease and cancer have remained the two',
                   'leading causes of death in the U.S. from',
                   '2000 to 2017'))

# Plot top two causes
for t_cause in t_causes:
    t_name = usa[usa.cause_name == t_cause].groupby('year'). \
                 agg({'crude_deaths': 'mean'}).reset_index()
    p = sns.lineplot(data=t_name, x='year', y='crude_deaths', ci=None,
                     label=t_cause)
ax.text(0.3, 0.8, text1, transform=ax.transAxes, fontsize=13,
        verticalalignment='top', bbox=props)
```

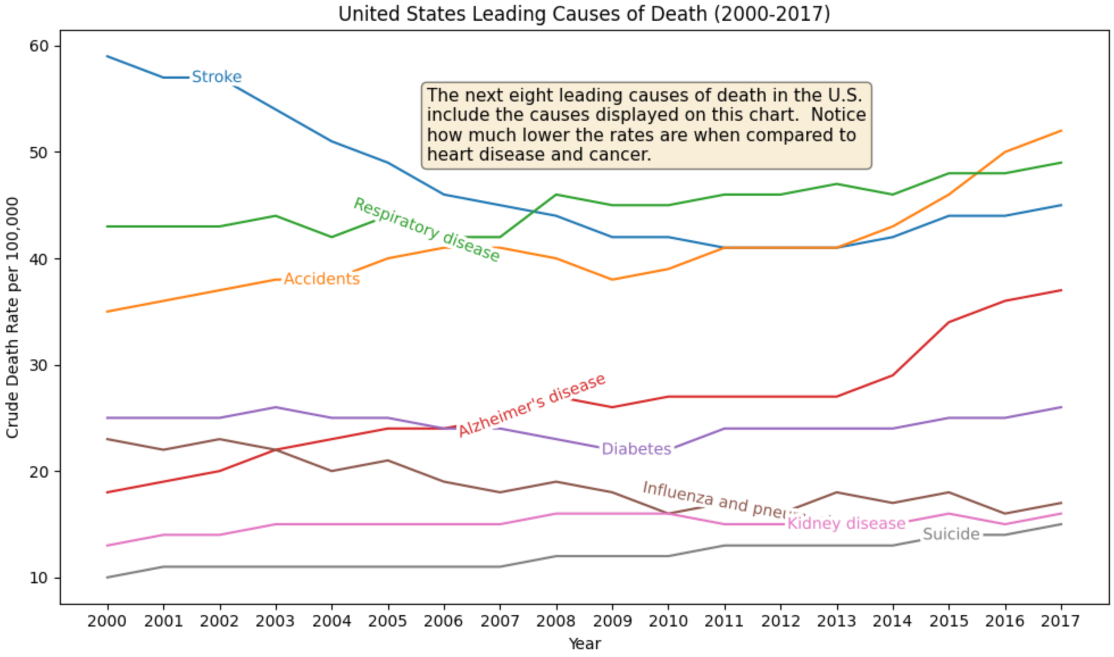
```
ax.get_legend().remove()
labelLines(ax.get_lines())
plt.show()
fig.savefig('../images/usTwoLeadingCausesLine.png', bbox_inches='tight',␣
 ↪dpi=300)
```



Heart disease and cancer have remained the two
leading causes of death in the U.S. from
2000 to 2017

[18]:
```
# Plot next eight causes
fig, ax = plt.subplots(figsize=(10,6))
ax.set(xlabel='Year', ylabel='Crude Death Rate per 100,000',
       title='United States Leading Causes of Death (2000-2017)')
ax.xaxis.set_ticks(np.arange(2000, 2018, 1))
text1 = '\n'.join(('The next eight leading causes of death in the U.S.',
                   'include the causes displayed on this chart.  Notice',
                   'how much lower the rates are when compared to',
                   'heart disease and cancer.'))
for cause in causes:
    name = usa[usa.cause_name == cause].groupby('year'). \
               agg({'crude_deaths': 'mean'}).reset_index()
    p = sns.lineplot(data=name, x='year', y='crude_deaths', ci=None,
                     label=cause)
ax.text(0.35, 0.9, text1, transform=ax.transAxes, fontsize=11,
        verticalalignment='top', bbox=props)
ax.get_legend().remove()
labelLines(ax.get_lines())
plt.show()
```

```
fig.savefig('../images/usNextEightCauseLine.png', bbox_inches='tight', dpi=300)
```



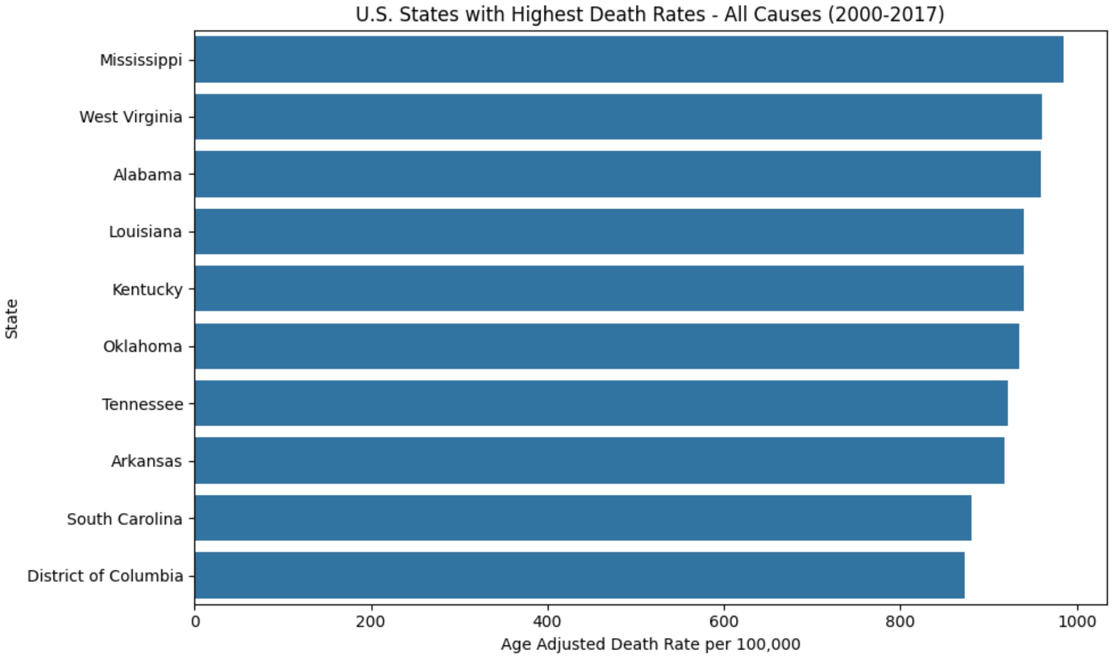
United States Leading Causes of Death (2000-2017)

The next eight leading causes of death in the U.S. include the causes displayed on this chart. Notice how much lower the rates are when compared to heart disease and cancer.

Take a closer look at deaths caused by heart disease

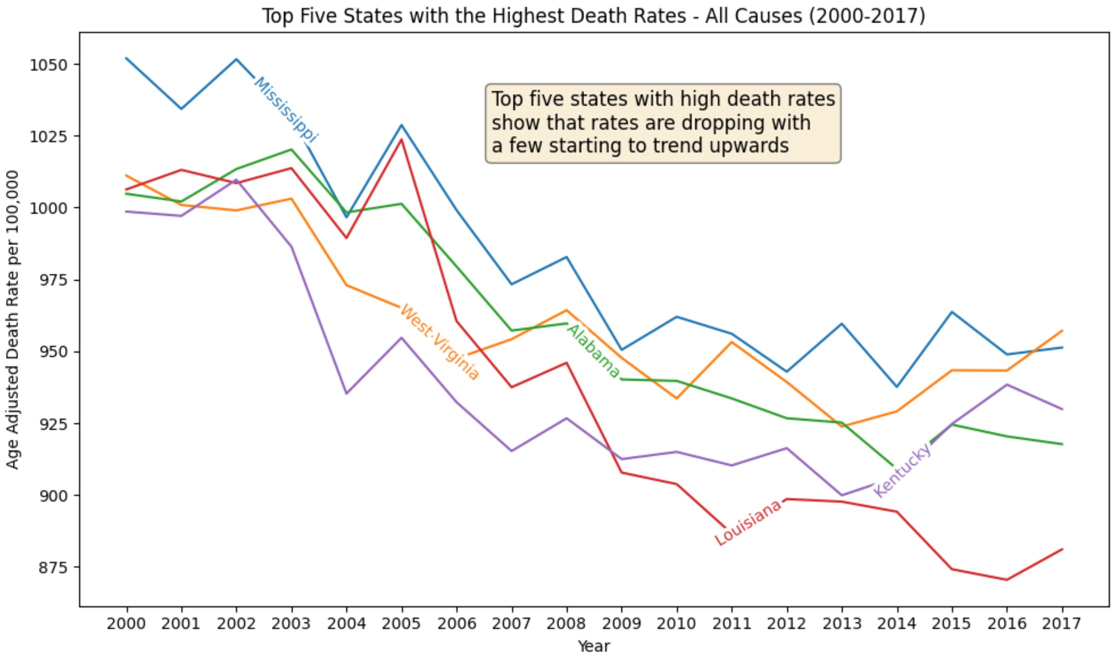
[19]:

```
# filter information to heart disease and top 10 states
states = all_df[(all_df.cause_name == 'All causes') & \
                 (all_df.state != 'United States')]
plot = states.groupby('state').agg({'age_adjusted':'mean'}). \
              sort_values('age_adjusted', ascending=False).head(10).
 ↪reset_index()

# Plot results
fig, ax = plt.subplots(figsize=(10,6))
ax.set(xlabel='Age Adjusted Death Rate per 100,000', ylabel='State',
       title='U.S. States with Highest Death Rates - All Causes (2000-2017)')
sns.barplot(data=plot, y='state', x='age_adjusted', ci=None)
plt.show()
fig.savefig('../images/statesMaxDeath.png', bbox_inches='tight', dpi=300)
```



[20]:
```
top10States = states.groupby('state').agg({'age_adjusted':'mean'}). \
              sort_values('age_adjusted', ascending=False).head(10).
↪reset_index()
p_states = top10States.state.head(10).tolist()
top5 = p_states[:5]
bottom5 = p_states[-5:]
```

[21]:
```
# Plot top 5 states
all_causes = all_df[all_df.cause_name == 'All causes']
fig, ax = plt.subplots(figsize=(10,6))
ax.set(xlabel='Year', ylabel='Age Adjusted Death Rate per 100,000',
       title='Top Five States with the Highest Death Rates - '\
              'All Causes (2000-2017)')
ax.xaxis.set_ticks(np.arange(2000, 2018, 1))
text1 = '\n'.join(('Top five states with high death rates',
                   'show that rates are dropping with',
                   'a few starting to trend upwards'))

# Loop through top 5 states
for state in top5:
    name = all_causes[all_causes.state == state].groupby('year'). \
           agg({'age_adjusted': 'mean'}).reset_index()
    p = sns.lineplot(data=name, x='year', y='age_adjusted', ci=None,␣
↪label=state)
ax.text(0.4, 0.9, text1, transform=ax.transAxes, fontsize=12,
        verticalalignment='top', bbox=props)
ax.get_legend().remove()
labelLines(ax.get_lines())
plt.show()
fig.savefig('../images/top5DeathRateLine.png', bbox_inches='tight', dpi=300)
```



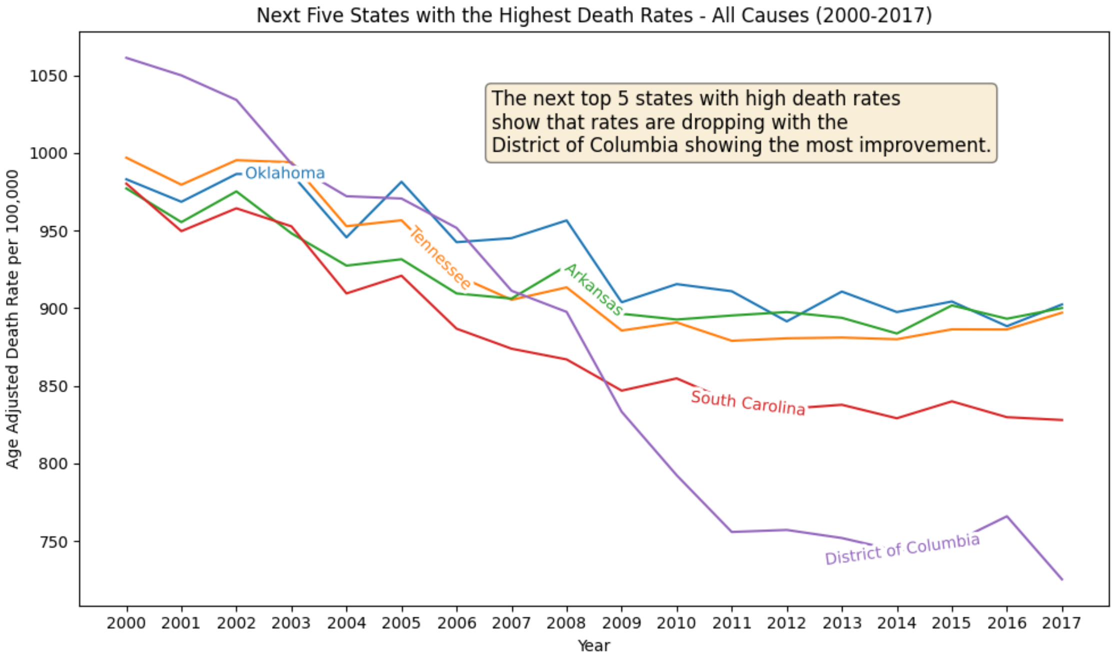
Top Five States with the Highest Death Rates - All Causes (2000-2017)

Top five states with high death rates
show that rates are dropping with
a few starting to trend upwards

[22]:

```
# Plot next 5 states
all_causes = all_df[all_df.cause_name == 'All causes']
fig, ax = plt.subplots(figsize=(10,6))
ax.set(xlabel='Year', ylabel='Age Adjusted Death Rate per 100,000',
       title='Next Five States with the Highest Death Rates - '\
              'All Causes (2000-2017)')
ax.xaxis.set_ticks(np.arange(2000, 2018, 1))
text1 = '\n'.join(('The next top 5 states with high death rates',
                   'show that rates are dropping with the',
                   'District of Columbia showing the most improvement.'))

# Loop through top 5 states
for state in bottom5:
    name = all_causes[all_causes.state == state].groupby('year'). \
           agg({'age_adjusted': 'mean'}).reset_index()
    p = sns.lineplot(data=name, x='year', y='age_adjusted', ci=None,␣
 ↪label=state)
ax.text(0.4, 0.9, text1, transform=ax.transAxes, fontsize=12,
        verticalalignment='top', bbox=props)
```

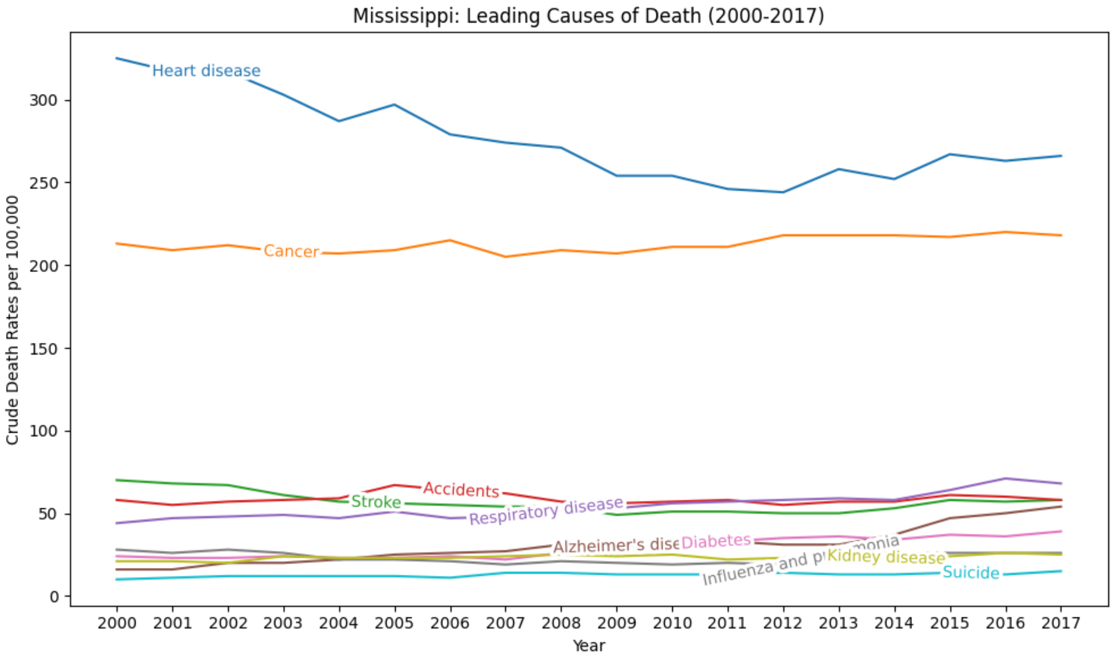
```
ax.get_legend().remove()
labelLines(ax.get_lines())
plt.show()
fig.savefig('../images/next5DeathRateLine.png', bbox_inches='tight', dpi=300)
```

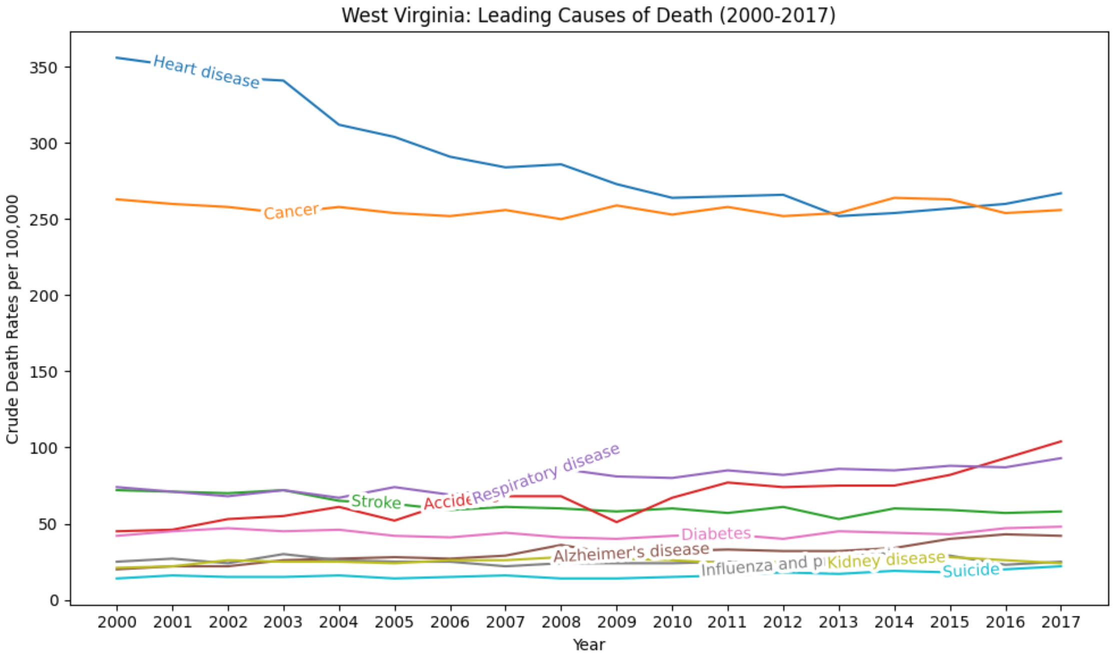

Next Five States with the Highest Death Rates - All Causes (2000-2017)

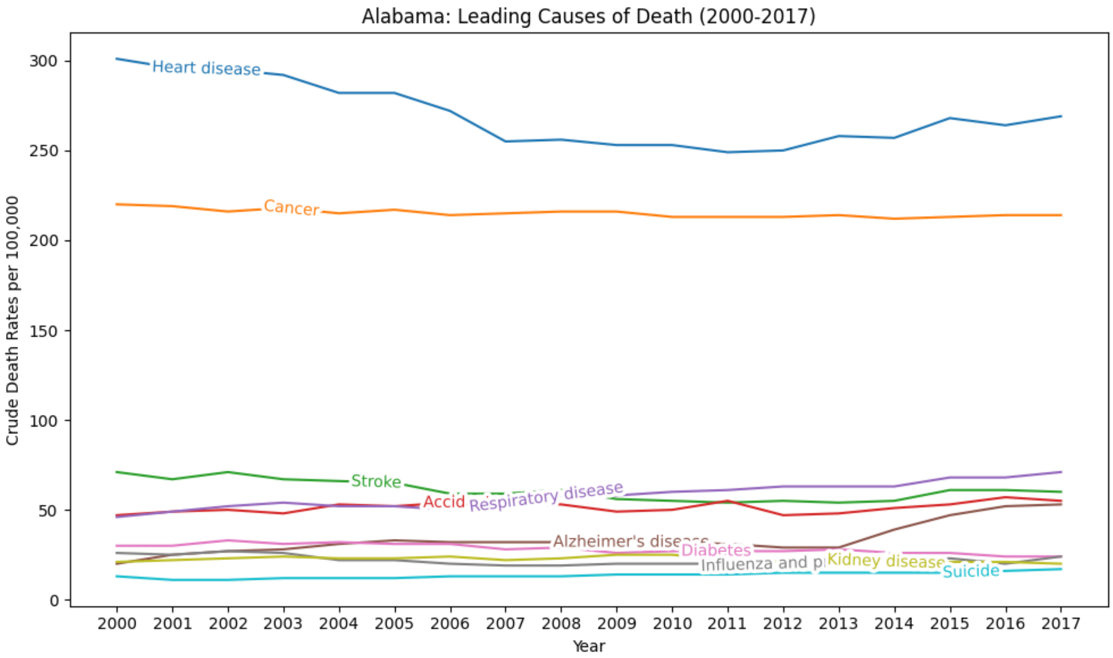
The next top 5 states with high death rates
show that rates are dropping with the
District of Columbia showing the most improvement.

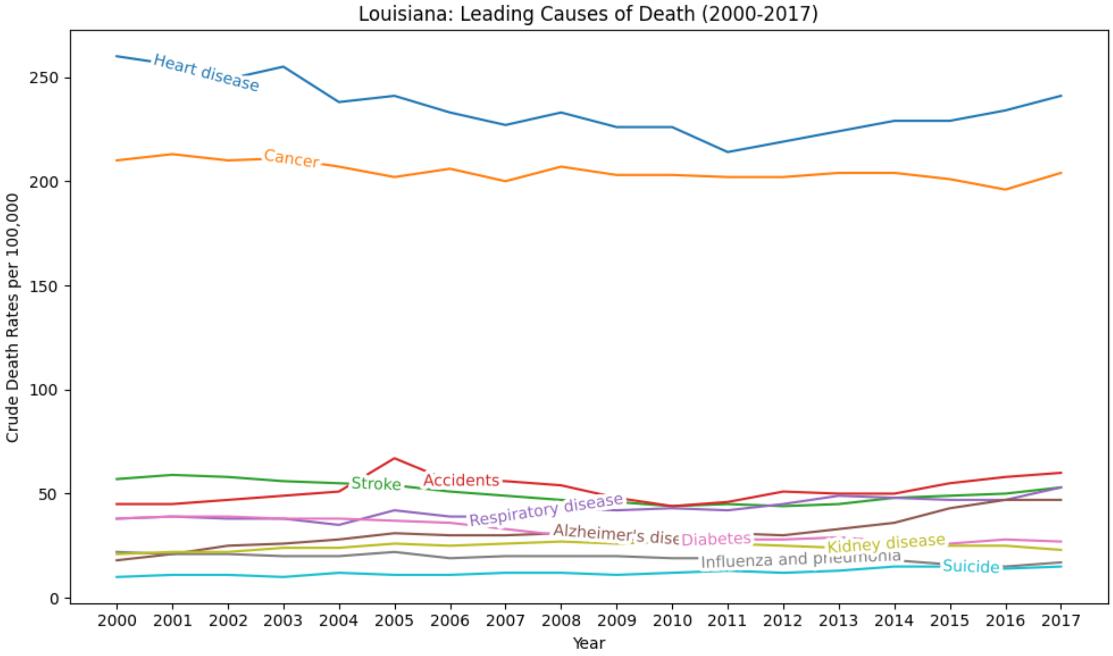
```
[23]: causes = ['Heart disease', 'Cancer', 'Stroke', 'Accidents', 'Respiratory␣
       ↪disease',
                 'Alzheimer\'s disease', 'Diabetes', 'Influenza and pneumonia',
                 'Kidney disease', 'Suicide']
      states = ['Mississippi', 'Oklahoma', 'District of Columbia', 'West Virginia',
                 'Kentucky', 'Alabama', 'New York', 'Tennessee', 'Louisiana',␣
       ↪'Missouri']

      for state in p_states:
          fig, ax = plt.subplots(figsize=(10, 6))
          for cause in causes:
              name = all_df[(all_df.state == state) & (all_df.cause_name ==cause)]. \
                     groupby('year').agg({'crude_deaths': 'mean'}).reset_index()
              p = sns.lineplot(data=name, x='year', y='crude_deaths', ci=None,
                                label=cause)
              ax.xaxis.set_ticks(np.arange(2000, 2018, 1))
              ax.set(xlabel='Year', ylabel='Crude Death Rates per 100,000',
                     title=state+': Leading Causes of Death (2000-2017)')
              ax.get_legend().remove()
          labelLines(ax.get_lines())
```

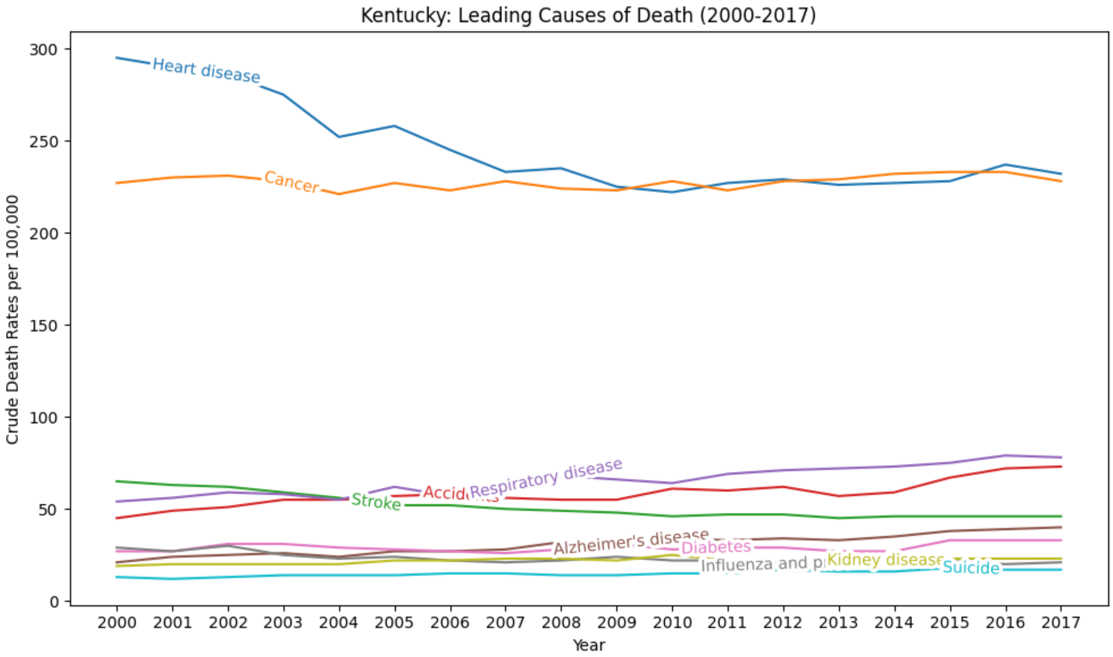
```
    plt.show()
    fig.savefig('../images/'+state+'CausesLine.png', bbox_inches='tight', dpi=300)
```

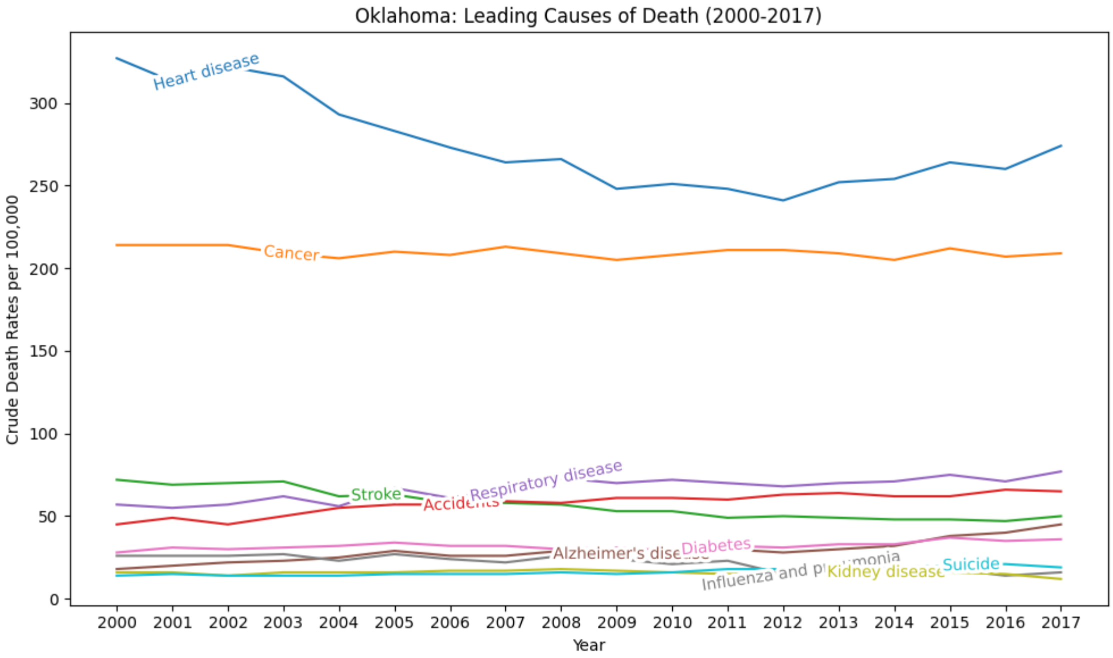


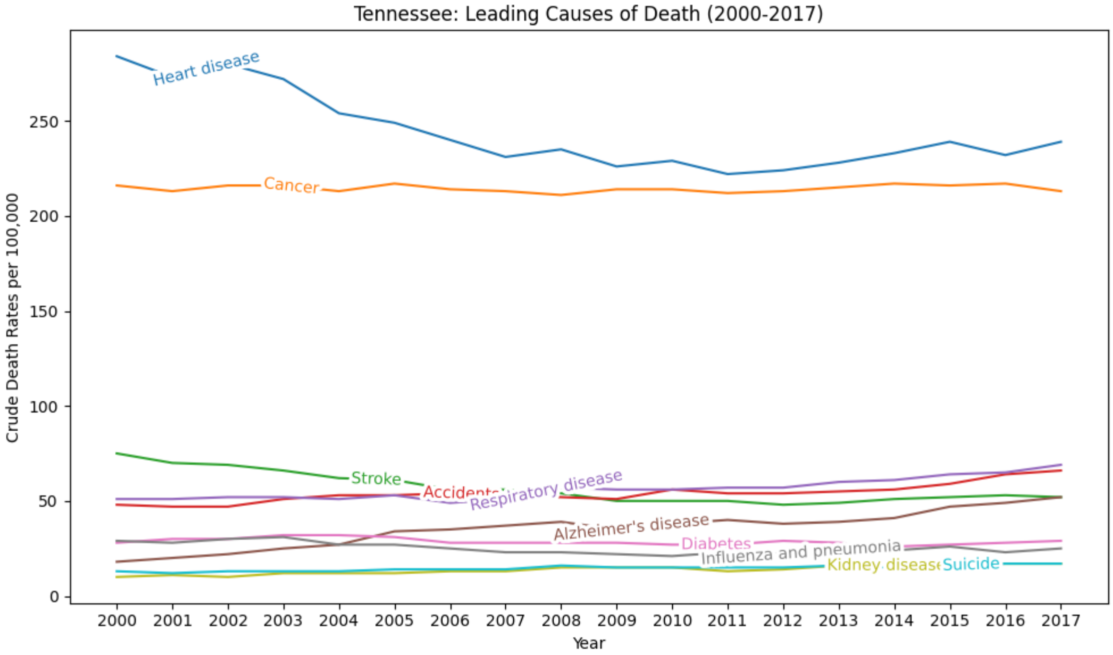


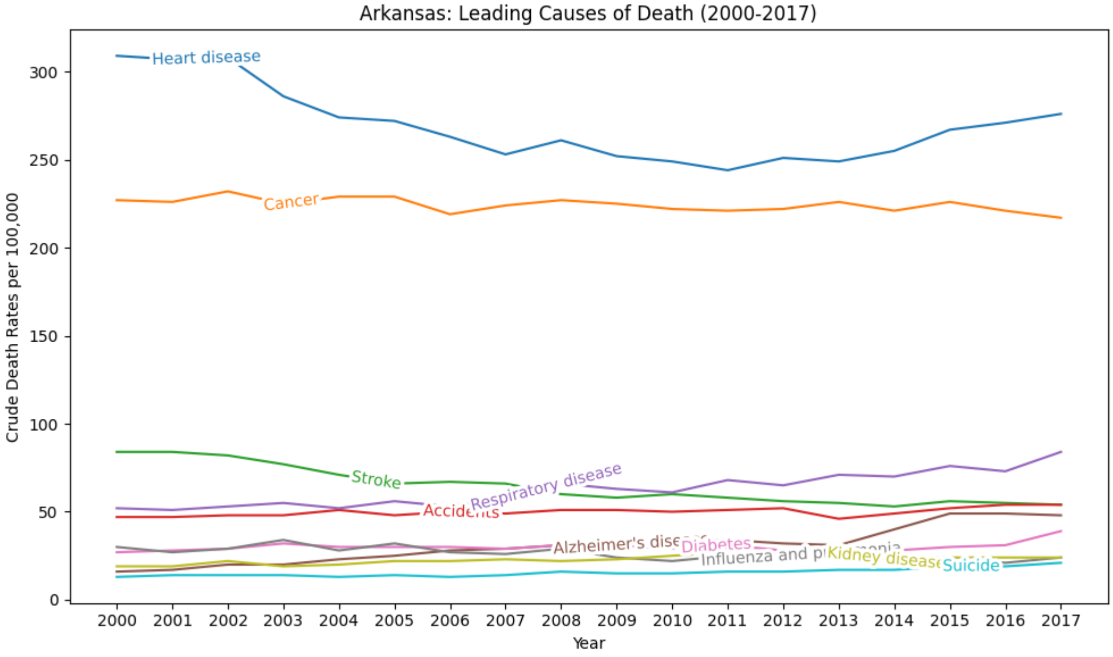

Alabama: Leading Causes of Death (2000-2017)
Crude Death Rates per 100,000
0
50
100
150
200
250
300
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
Year
Heart disease
Cancer
Stroke
Accid
Respiratory disease
Alzheimer's dise
Diabetes
Influenza and p
Kidney disease
Suicide

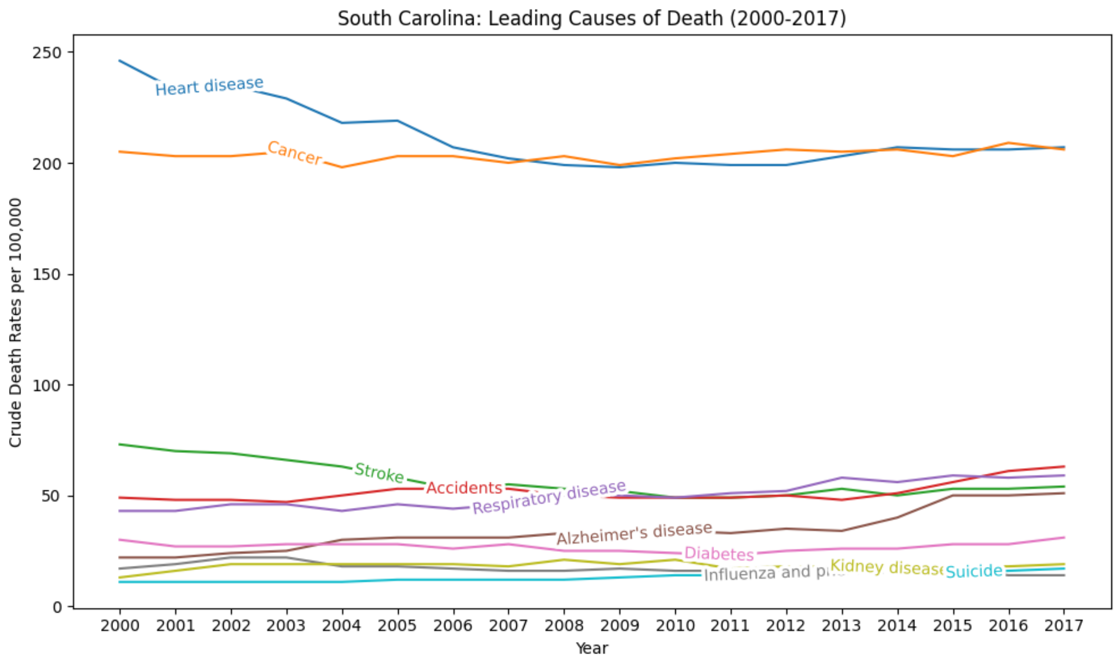

Louisiana: Leading Causes of Death (2000-2017)
Crude Death Rates per 100,000
0
50
100
150
200
250
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
Year
Heart disease
Cancer
Accidents
Stroke
Respiratory disease
Alzheimer's dise
Diabetes
Influenza and pneumonia
Kidney disease
Suicide

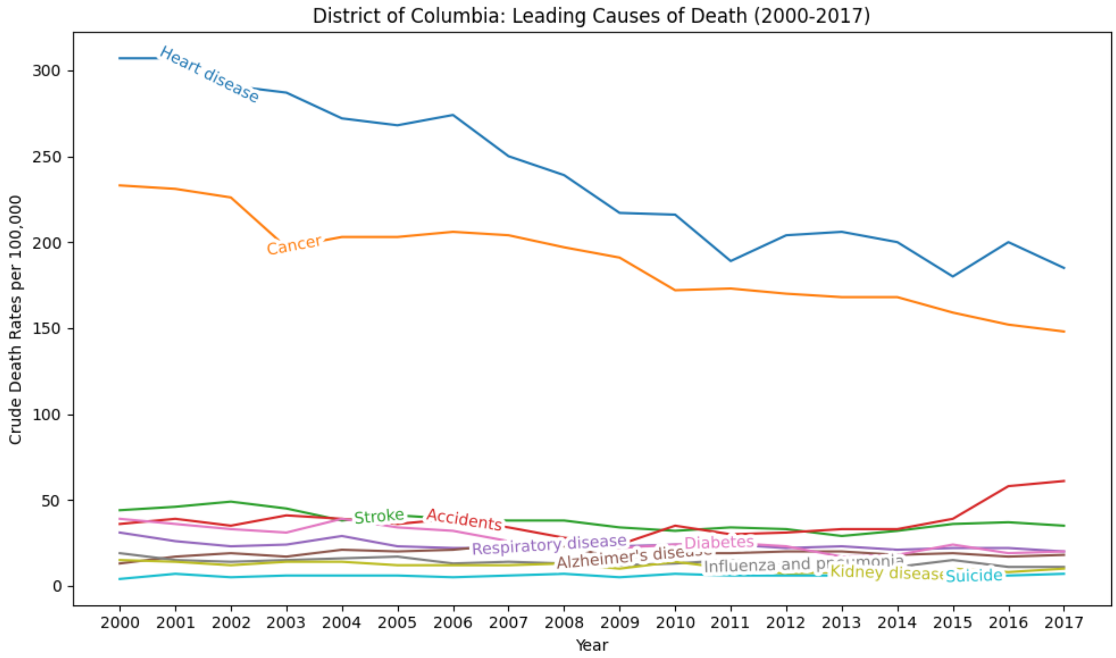

Kentucky: Leading Causes of Death (2000-2017)
Crude Death Rates per 100,000
300
250
200
150
100
50
0
Heart disease
Cancer
Respiratory disease
Accidents
Stroke
Alzheimer's disease
Diabetes
Influenza and p
Kidney disease
Suicide
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
Year


Oklahoma: Leading Causes of Death (2000-2017)
Crude Death Rates per 100,000
300
250
200
150
100
50
0
Heart disease
Cancer
Stroke
Respiratory disease
Accidents
Alzheimer's disease
Diabetes
Influenza and pneumonia
Kidney disease
Suicide
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
Year


Tennessee: Leading Causes of Death (2000-2017)
Crude Death Rates per 100,000
0
50
100
150
200
250
Year
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
Heart disease
Cancer
Stroke
Accidents
Respiratory disease
Alzheimer's disease
Diabetes
Influenza and pneumonia
Kidney disease
Suicide

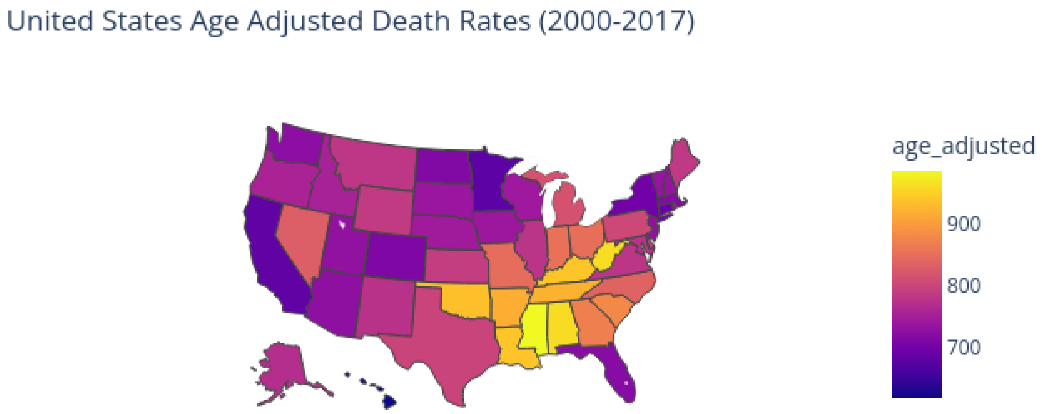

Arkansas: Leading Causes of Death (2000-2017)
Crude Death Rates per 100,000
0
50
100
150
200
250
300
Year
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
Heart disease
Cancer
Stroke
Accidents
Respiratory disease
Alzheimer's disease
Diabetes
Influenza and pneumonia
Kidney disease
Suicide





```
[24]: all_causes = all_df[(all_df.cause_name == 'All causes') & \
                          (all_df.state != 'United States')]
      plot = all_causes.groupby('abbreviation').agg({'age_adjusted':'mean'}).
       ↪reset_index()
```

```
fig = px.choropleth(plot, locations='abbreviation', locationmode='USA-states',
                    color='age_adjusted', scope='usa',
                    title='United States Age Adjusted Death Rates (2000-2017)',
                    hover_data='age_adjusted')
fig.update_layout(hoverlabel=dict(bgcolor='wheat', font_size=15))
fig.show()
fig.write_html('../images/mapUSA.html')
fig.write_image('../images/mapUSA.png')
```



[ ]: