3.2PythonCharts_AHarvey

January 24, 2021

[2]:
```
# Load libraries. Pandas for loading data and plots. Matplotlib for plots.␣
 ↪Plotly and Squarify for treemaps.
# Helpful resource at https://towardsdatascience.com/
 ↪treemap-basics-with-python-777e5ed173d0 regarding treemaps.
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import plotly.express as px
import squarify
```

[3]:
```
# Import Excel files into dataframe
unemp = pd.read_csv('unemployement-rate-1948-2010.csv')
```

[4]:
```
unemp.head()
```

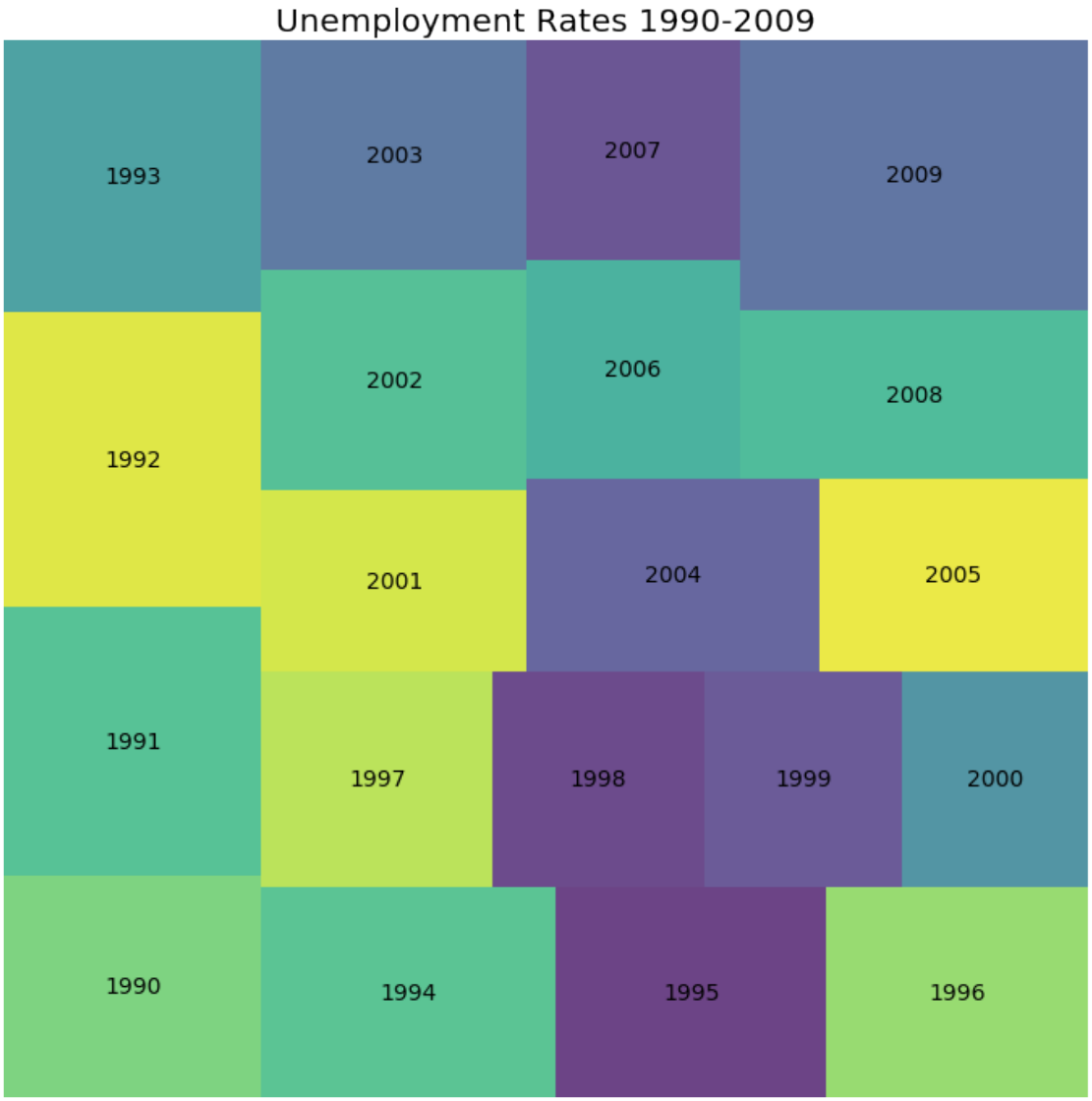
[4]:

	Series id	Year	Period	Value
0	LNS14000000	1948	M01	3.4
1	LNS14000000	1948	M02	3.8
2	LNS14000000	1948	M03	4.0
3	LNS14000000	1948	M04	3.9
4	LNS14000000	1948	M05	3.5

[5]:
```
# Create new dataframe with just 1990-2009 because the full dataset is way too␣
 ↪much information for a treemap
current = unemp[unemp["Year"]>=1990]
current = current[current['Year']<2010]
```

[6]:
```
# Attemmpt #1 with Squarify. This is as far as I could get with the information␣
 ↪I could pull from the internet.
#
# My biggest stumbling block with Squarify is that I could not figure out how␣
 ↪to do multiple levels within the chart. Every
# article I found directed me to use Plotly if I wanted to do multiple levels.␣
 ↪The other struggle was the color scheme. The
# automatic color scheme was too random, in my opinion, so I tried to do a␣
 ↪colormap based on value. But that made the largest
```

```
# values indistinguishable from each other. So, I went back to the automatic␣
 ↪color.

fig, ax = plt.subplots(1, figsize = (12,12))

# My attempt at a colormap
# cmap = matplotlib.cm.viridis
# mini=min(current['Value'])
# maxi=max(current['Value'])
# norm = matplotlib.colors.Normalize(vmin=mini, vmax=maxi)
# colors = [cmap(norm(value)) for value in current['Value']]

a = current.groupby('Year')[['Value']].mean().index.get_level_values(0).tolist()

d = current.groupby('Year')[['Value']].mean().reset_index().Value.values.
 ↪tolist()

squarify.plot(sizes=d,label=a, alpha=.8, text_kwargs={'fontsize':14})
plt.title('Unemployment Rates 1990-2009', fontsize=20)
plt.axis('off')
plt.show()
```

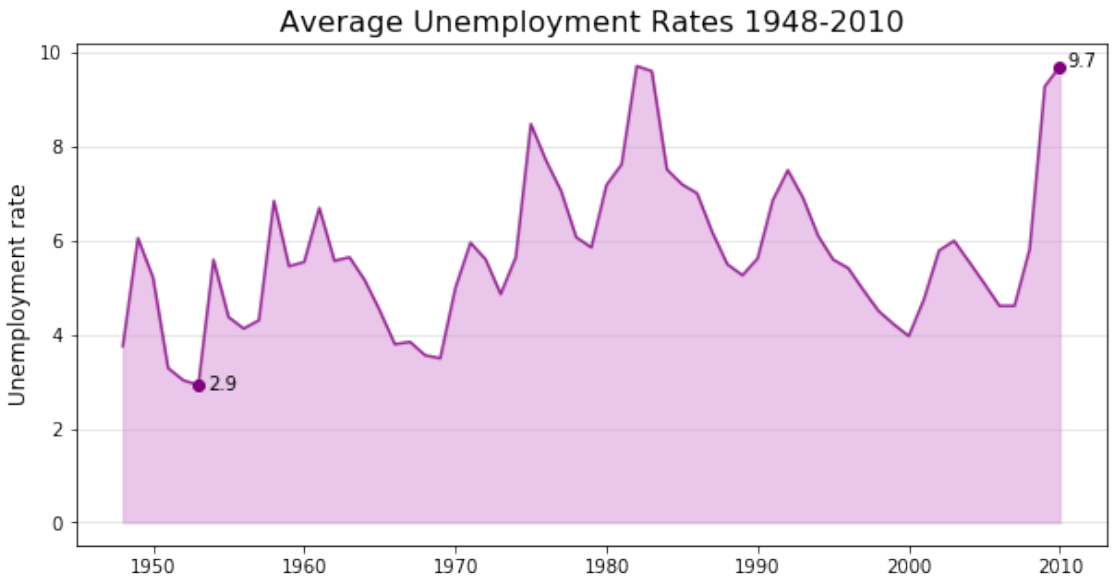


[7]:
```
# For Plotly, I was able to get multiple levels within the treemap. This source
 ↪was helpful: https://plotly.com/python/treemaps/
# I couldn't figure out how to get rid of the grey background box (which is
 ↪apparently the first node of the tree). So, I added
# a "first node" per recommendations in the articles.

data =  pd.DataFrame.from_dict(current)
data['Unemp'] = 'Unemployment'
fig = px.treemap(data, path=['Unemp','Year', 'Value'], values='Value',
                 color='Value', hover_data=['Period'],
                 color_continuous_scale='Blues',
```

```
                    color_continuous_midpoint=np.average(current['Value'],␣
 ↪weights=current['Year']),
                    title = 'Proportion of Unemployment Rates from 1990-2009',
                    width=900, height=800)
fig.update_layout(title_font_size=20, uniformtext=dict(minsize=12, mode='hide'))
fig.show()
```

[8]:
```
# Create new dataframe with just the average unemployment rate for each year
avg = unemp.groupby('Year')[['Value']].mean().reset_index()
```

[9]:
```
# Area plot with line. Again, I find myself appreciating the simplicity of␣
 ↪matplotlib. I couldn't find a way to automate the
# labelling of only SOME of the data points. I could only find a way to do all␣
 ↪of them or one by one. So, I labeled the max/min.
plt.figure(figsize=(10, 5))

plt.fill_between(avg['Year'], avg['Value'], color='plum', alpha=0.6)
plt.plot(avg['Year'], avg['Value'], color='purple', alpha=0.8)
plt.title('Average Unemployment Rates 1948-2010', fontsize=16)
plt.ylabel('Unemployment rate', fontsize=12)
plt.grid(True, axis = 'y', linestyle='-', linewidth=0.5, alpha =0.6)

# plt.plot(avg['Year'], avg['Value'],'bo-')
# for x,y in zip(avg['Year'], avg['Value']):

#     label = "{:.2f}".format(y)

#     plt.annotate(label,
#                  (x,y),
#                  textcoords="offset points",
#                  xytext=(0,10),
#                  ha='center')
plt.plot(1953, 2.925,'bo-', color='purple')
plt.plot(2010, 9.70,'bo-', color ='purple')
plt.annotate('2.9', (1953, 2.925), xytext = (5,-3), textcoords='offset points')
plt.annotate('9.7', (2010, 9.70), xytext =(4,0), textcoords='offset points')

plt.show()
```



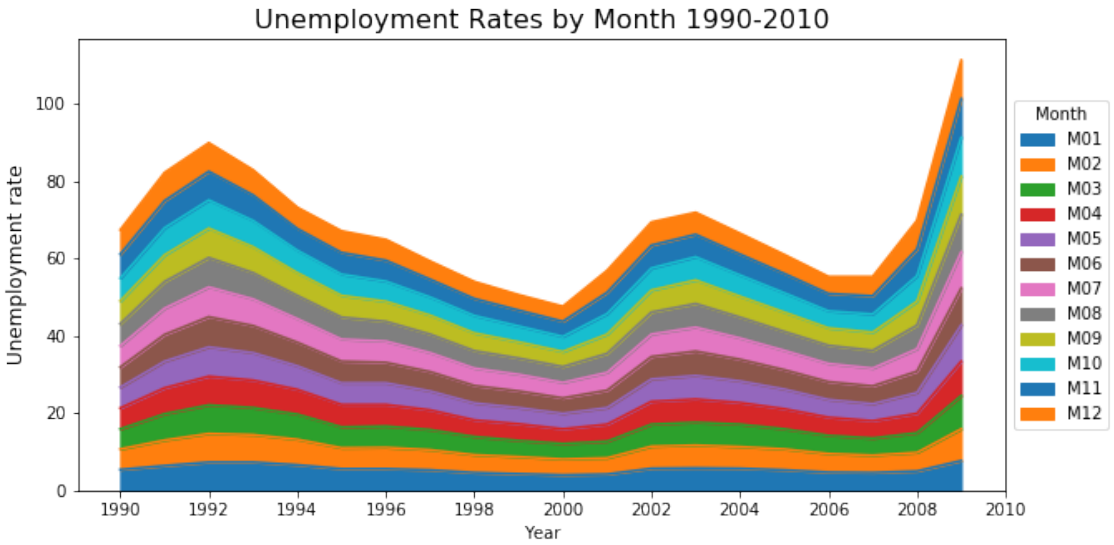
[26]:
```
# I was struggling extremely hard with the stacked line chart here until my␣
 ↪peers helped me realize I needed to pivot the data
# in order to create rows to stack.
per = current.pivot(index='Year', columns='Period', values='Value')
```

[27]:
```
per.head()
```

[27]:

Period	M01	M02	M03	M04	M05	M06	M07	M08	M09	M10	M11	M12
Year												
1990	5.4	5.3	5.2	5.4	5.4	5.2	5.5	5.7	5.9	5.9	6.2	6.3
1991	6.4	6.6	6.8	6.7	6.9	6.9	6.8	6.9	6.9	7.0	7.0	7.3
1992	7.3	7.4	7.4	7.4	7.6	7.8	7.7	7.6	7.6	7.3	7.4	7.4
1993	7.3	7.1	7.0	7.1	7.1	7.0	6.9	6.8	6.7	6.8	6.6	6.5
1994	6.6	6.6	6.5	6.4	6.1	6.1	6.1	6.0	5.9	5.8	5.6	5.5

[41]:
```
per.plot.area(figsize = (10,5), xticks = (1990, 1992, 1994, 1996, 1998, 2000,␣
 ↪2002, 2004, 2006, 2008, 2010))
plt.title('Unemployment Rates by Month 1990-2010', fontsize=16)
plt.ylabel('Unemployment rate', fontsize=12)
plt.legend(title = 'Month', loc='center left', bbox_to_anchor=(1.0, 0.5))
plt.show()
```


Unemployment Rates by Month 1990-2010
Unemployment rate
Year
0
20
40
60
80
100
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
Month
M01
M02
M03
M04
M05
M06
M07
M08
M09
M10
M11
M12

[ ]: