Good/Bad Visualization Project

Grace Shao

10/31/22

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from datetime import datetime
from datetime import date
from datetime import timedelta

import altair as alt
import matplotlib
from matplotlib import pyplot as plt
plt.rcParams['figure.dpi'] = 300
from matplotlib.widgets import Slider, Button

# from transformers import pipeline
# import seaborn as sns
# import geopandas as geo
```

For this assignment, you will create two visualizations of a dataset of your choice. (1) A visualization engineered to obscure some truth of the data; and (2) A visualization with an "honest" presentation of the same data.

This project explores reviews of scented candles on Amazon. The data was accessed October 2022, from Kate Petrova's github (https://github.com/kateptrv/Candles). The dataset includes a random sample of US Amazon candle consumers, from unscented and scented candles. The ~3.6K reviews for unscented candles span three candles, from 2014 to 2020; and the 17.7K reviews for scented spans five candles, from 2005 to 2020. The COVID-19 information was accessed October 2022, released by New York Times (https://www.kaggle.com/datasets/gniwnyc/nytimescovid19usdataset?select=us.csv). It covers US Country level COVID-19 information, from January 2020 to April 2020.

```
unscented = pd.read_excel("Unscented_all.xlsx")
# scented = pd.read_excel("Scented_all.xlsx")
```

```
scented['Date'] = pd.to_datetime(scented['Date'])
```

```
len(unscented),len(scented)
```

```
# scented.CandleID.value_counts()
# scented.sort_values('Date')
```

```
summarizer = pipeline("summarization")
zero_shot = pipeline("zero-shot-classification")
```

```
scented1 = scented.loc[:len(scented)/3]
scented2 = scented.loc[len(scented)/3:2*len(scented)/3]
scented3 = scented.loc[2*len(scented)/3:]
```

```
def score_unscent(reviews):
    res = []
    for i in reviews.index:
        zero = zero_shot(reviews[i],['scented','unscented'])
        score = dict(zip(zero['labels'],zero['scores']))['unscented']
        res.append(score)
    return res
```

```
# scented['score'] = score_unscent(scented['Review'])
```

```
# scented.to_csv('scored_scented.csv')
scented = pd.read_csv('scored_scented.csv')
```

Good visual

```
# Querying for top three candles for both scented and unscented.
top = {1,2,3}
scented = scented.query('CandleID in @top')
unscented = unscented.query('CandleID in @top')
```

```
len(scented), len(unscented)
```

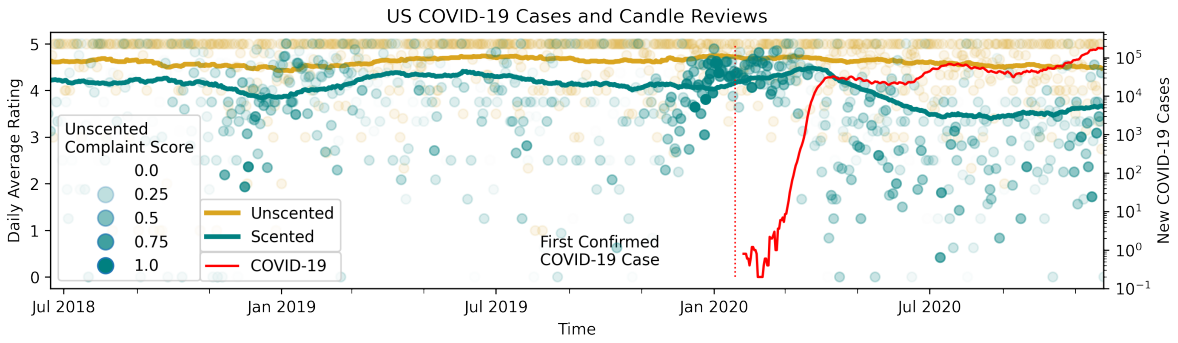
```
(8614, 3597)
```

```
# Average daily rating
bydate = scented[['Date','Rating']].groupby('Date').mean().reset_index()
bydate['Date'] = pd.to_datetime(bydate['Date'])
bydate['score'] = scented[['Date','score']].groupby('Date').sum().values
bydate['year'] = [int(str(bydate['Date'][i])[:4]) for i in bydate.index]
bydate['roll'] = bydate['Rating'].rolling(window=100).mean()

bydate_unscented = unscented[['Date','Rating']].groupby('Date').mean().reset_index()
bydate_unscented['year'] = [int(str(bydate_unscented['Date'][i])[:4]) for i in bydate_unsc
bydate_unscented['roll'] = bydate_unscented['Rating'].rolling(window=100).mean()
```

```
#Relevant covid data
first_case = date(2020,1,19) # from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7092802/

covid = pd.read_csv('covid_count.csv')
covid['date'] = pd.to_datetime(covid['date'])
covid['change'] = covid['cases'].diff().rolling(5).mean()

covid['year'] = [int(str(covid['date'][i])[:4]) for i in covid.index]
covid = covid.query('year <= 2020')
```

```
# graphing
fig,ax = plt.subplots(figsize=(12,3))
alpha = 0.1
c1,c2 = 'goldenrod','teal'

### reviews
ax.scatter(bydate_unscented['Date'],bydate_unscented['Rating'],
           alpha=alpha,c=c1)
ax_score = ax.twinx()
ax_score.scatter(bydate['Date'],bydate['Rating'],
           c=c2,alpha=bydate['score']/max(bydate['score'])) # using opacity to show compl
ax_score.tick_params(axis='y',right=False,labelcolor='white')

ax.plot(bydate_unscented['Date'],bydate_unscented['roll'],
        c=c1,label='Unscented',lw=3)
```

```
ax.plot(bydate['Date'],bydate['roll'],
         c=c2,label='Scented',lw=3)

### relevant covid information
ax2 = ax.twinx()
ax2.plot(covid['date'],covid['change'],c='red',label='COVID-19 ')
ax2.set_yscale('log')
ax2.set_ylabel('New COVID-19 Cases')

ax.plot([first_case,first_case],[0,5],c='r',linestyle='dotted',lw=1)
ax.text(first_case-timedelta(days=165),0.25,'First Confirmed \nCOVID-19 Case')

### formatting
matplotlib.dates.MonthLocator(interval=6)
ax.xaxis.set_major_formatter(matplotlib.dates.DateFormatter('%b %Y'))
ax.set_xlim([date(2018,6,20),date(2020,11,25)])
ax.xaxis.set_major_locator(matplotlib.dates.MonthLocator(interval=6))
ax.xaxis.set_minor_locator(matplotlib.dates.MonthLocator(interval=2))
ax.tick_params(which='minor', length=3)
ax.set_ylabel('Daily Average Rating')
ax.set_xlabel('Time')
ax.set_title('US COVID-19 Cases and Candle Reviews')

ax.legend(loc=(0.142,0.144))
ax2.legend(loc=(0.142,0.03))
alpha_labels = []
for a in np.arange(0,1.1,0.25):
    alpha_labels.append(matplotlib.lines.Line2D([0],[0],linestyle="none",marker="o",alpha=
ax_score.legend(alpha_labels,np.arange(0,1.1,0.25),title='Unscented \nComplaint Score',loc
fig.set_dpi(300)
plt.show()
```



Caption: The figure compares candle reviews with COVID-19 cases, depicting daily average ratings, rolling average ratings, and log new cases. Before COVID-19 the ratings for scented and unscented candles don't stray far from each other, with slight dips in scented candle ratings every winter, likely due to a higher expectation for holiday candles. After the start of COVID-19, the average ratings for scented candles falls down, coinciding with a jump in the log of new COVID-19 cases. The unscented complaint score* is also higher at the time of and after the first confirmed COVID-19 case. One of COVID-19's symptoms is the loss of smell, which is likely the reason for an increased unsatisfaction with scented candles.

**The unscented complaint score: Using a zero-shot-estimator, each review was scored on a scale of "scented" to "unscented". The unscented complaint score is the total "unscented" score for that day divided by the maximum total present. I used sums instead of averages, because this is a metric to highlight the increase in complaints regarding absence of scent.*

Bad Visual

```
bydate['pchange'] = bydate['Rating'].pct_change()
bydate['proll'] = bydate['pchange'].rolling(window=200).mean()
```

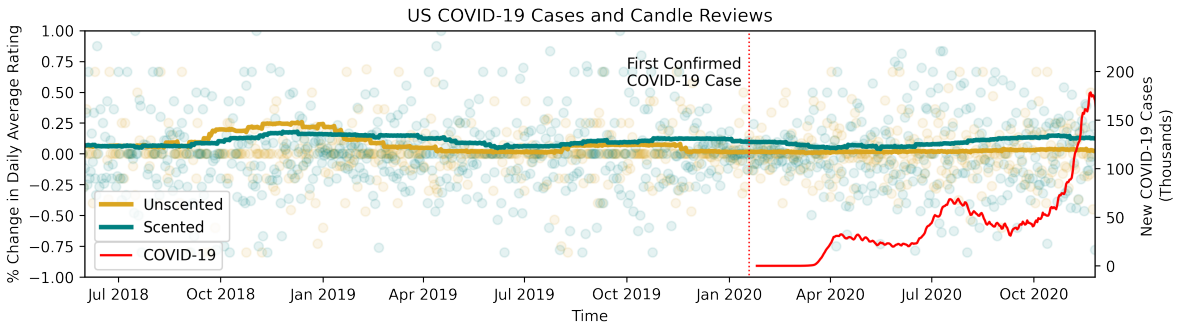
```
bydate_unscented['pchange'] = bydate_unscented['Rating'].pct_change()
bydate_unscented['proll'] = bydate_unscented['pchange'].rolling(window=100).mean()
```

```
covid['K'] = covid['change']/1000
```

```
sizes = max(bydate['score']) - bydate['score'] + 1
# inverted_alpha = (max(bydate['score'])-bydate['score']) / max(bydate['score']) / max(inv
```

```
fig,ax = plt.subplots(figsize=(12,3))
alpha = 0.1
c1,c2 = 'goldenrod','teal'

### reviews
ax.plot(bydate_unscented['Date'],bydate_unscented['proll'],
         c=c1,label='Unscented',lw=3)
ax.plot(bydate['Date'],bydate['proll'],
         c=c2,label='Scented',lw=3)

ax.scatter(bydate_unscented['Date'],bydate_unscented['pchange'],
            alpha=alpha,c=c1)
```

```
ax.scatter(bydate['Date'],bydate['pchange'],
           alpha=alpha,c=c2)
# ax_score = ax.twiny()
# ax_score.scatter(bydate['Date'],bydate['pchange'],
#                  c=c2,alpha=alpha,
#                  s=sizes) # using opacity to show complaints about unscented
# ax_score.tick_params(axis='x',top=False,labelcolor='white')

### relevant covid information
ax2 = ax.twinx()
ax2.plot(covid['date'],covid['K'],c='red',label='COVID-19')
ax2.set_ylabel('New COVID-19 Cases \n(Thousands)')

### formatting
matplotlib.dates.MonthLocator(interval=6)
ax.xaxis.set_major_formatter(matplotlib.dates.DateFormatter('%b %Y'))
ax.set_xlim([date(2018,6,1),date(2020,11,25)])
ax.set_ylim([-1,1])
ax.set_ylabel('% Change in Daily Average Rating')
ax.set_xlabel('Time')
ax.set_title('US COVID-19 Cases and Candle Reviews')

ax.legend(loc=(0.01,0.144))
ax2.legend(loc=(0.01,0.03))
ax.plot([first_case,first_case],[-5,5],c='r',linestyle='dotted',lw=1)
ax.text(first_case-timedelta(days=110),0.55,'First Confirmed \nCOVID-19 Case')
alpha_labels = []
for a in np.arange(0,1.1,0.25):
    alpha_labels.append(matplotlib.lines.Line2D([0],[0],linestyle="none",marker="o",alpha=
ax_score.legend(alpha_labels,np.arange(0,1.1,0.25),title='Unscented \nComplaint Score',loc
plt.show()
```



Caption: The figure compares candle reviews with COVID-19 cases, depicting the percent change in daily average ratings and new COVID-19 cases. There are slight peaks in the change in ratings for scented candles every winter, likely due to an increase in candles bought for the holidays. There is no relationship between the change in ratings and COVID-19 cases.

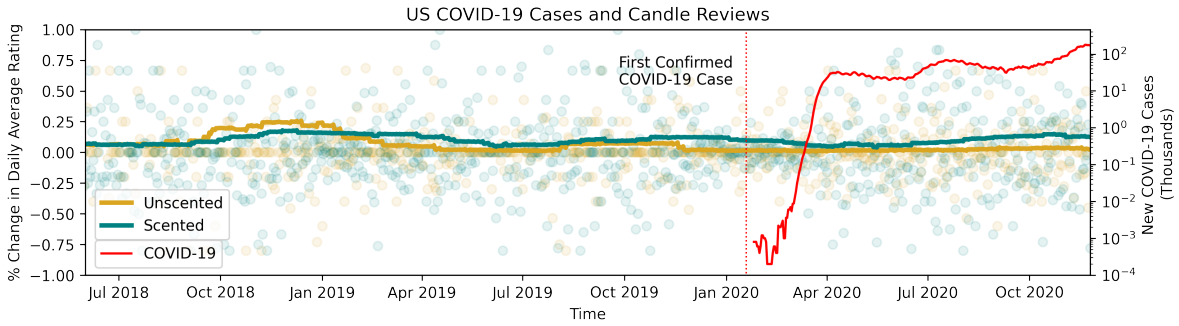
These two visualizations demonstrate the importance of what the y-axis represents. Graphing the percent change rather than average values hides how the actual ratings changing in favor of showing how the change changes. If the average ratings are consistently lower after the start of COVID-19 (with the same consistency as before COVID-19), then the percent change is stable, and the y-axis would not reflect the decrease in the ratings. Additionally, the new COVID-19 cases are represented here on a linear rather than a logarithmic scale. Since COVID-19 spreads at an exponential rate, a logarithmic scale is more appropriate to understanding its growth, while a linear scale is less intuitive. While both the logarithmic and linear scale would not have an obvious correlation to the percent change in ratings, I chose to graph COVID-19 on the linear scale to further obfuscate the data.

Additional bad graphs

```
fig,ax = plt.subplots(figsize=(12,3))
alpha = 0.1
c1,c2 = 'goldenrod','teal'

### reviews
ax.plot(bydate_unscented['Date'],bydate_unscented['proll'],
         c=c1,label='Unscented',lw=3)
ax.plot(bydate['Date'],bydate['proll'],
         c=c2,label='Scented',lw=3)

ax.scatter(bydate_unscented['Date'],bydate_unscented['pchange'],
            alpha=alpha,c=c1)
ax.scatter(bydate['Date'],bydate['pchange'],
            alpha=alpha,c=c2)
# ax_score = ax.twiny()
# ax_score.scatter(bydate['Date'],bydate['pchange'],
#                  c=c2,alpha=alpha,
#                  s=sizes) # using opacity to show complaints about unscented
# ax_score.tick_params(axis='x',top=False,labelcolor='white')

### relevant covid information
```

```
ax2 = ax.twinx()
ax2.plot(covid['date'],covid['K'],c='red',label='COVID-19')
ax2.set_yscale('log')
ax2.set_ylabel('New COVID-19 Cases \n(Thousands)')

### formatting
matplotlib.dates.MonthLocator(interval=6)
ax.xaxis.set_major_formatter(matplotlib.dates.DateFormatter('%b %Y'))
ax.set_xlim([date(2018,6,1),date(2020,11,25)])
ax.set_ylim([-1,1])
ax.set_ylabel('% Change in Daily Average Rating')
ax.set_xlabel('Time')
ax.set_title('US COVID-19 Cases and Candle Reviews')

ax.legend(loc=(0.01,0.144))
ax2.legend(loc=(0.01,0.03))
ax.plot([first_case,first_case],[-5,5],c='r',linestyle='dotted',lw=1)
ax.text(first_case-timedelta(days=115),0.55,'First Confirmed \nCOVID-19 Case')
alpha_labels = []
for a in np.arange(0,1.1,0.25):
    alpha_labels.append(matplotlib.lines.Line2D([0],[0],linestyle="none",marker="o",alpha=
ax_score.legend(alpha_labels,np.arange(0,1.1,0.25),title='Unscented \nComplaint Score',loc
#
```

```
<matplotlib.legend.Legend at 0x22612dadfd0>
```

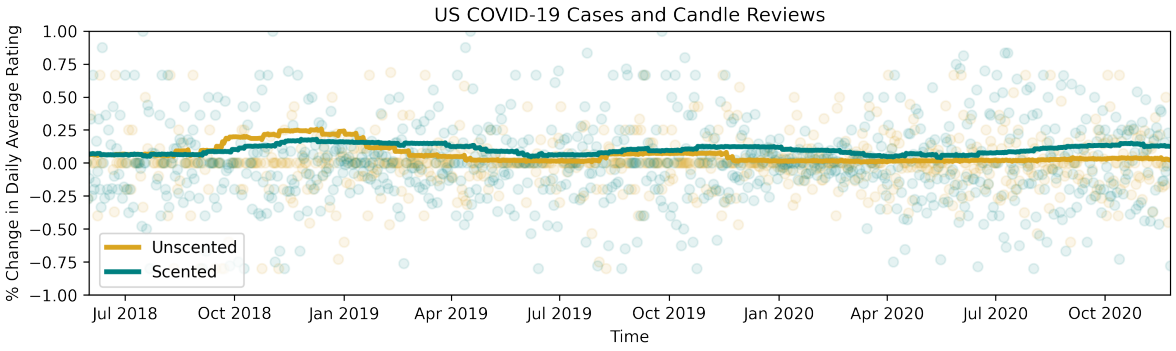


Using the logarithmic scale for new COVID-19 cases.

```
fig,ax = plt.subplots(figsize=(12,3))
alpha = 0.1
```

```
c1,c2 = 'goldenrod','teal'

### reviews
ax.plot(bydate_unscented['Date'],bydate_unscented['proll'],
         c=c1,label='Unscented',lw=3)
ax.plot(bydate['Date'],bydate['proll'],
         c=c2,label='Scented',lw=3)

ax.scatter(bydate_unscented['Date'],bydate_unscented['pchange'],
            alpha=alpha,c=c1)
ax.scatter(bydate['Date'],bydate['pchange'],
            alpha=alpha,c=c2)
# ax_score = ax.twiny()
# ax_score.scatter(bydate['Date'],bydate['pchange'],
#                  c=c2,alpha=alpha,
#                  s=sizes) # using opacity to show complaints about unscented
# ax_score.tick_params(axis='x',top=False,labelcolor='white')

### formatting
matplotlib.dates.MonthLocator(interval=6)
ax.xaxis.set_major_formatter(matplotlib.dates.DateFormatter('%b %Y'))
ax.set_xlim([date(2018,6,1),date(2020,11,25)])
ax.set_ylim([-1,1])
ax.set_ylabel('% Change in Daily Average Rating')
ax.set_xlabel('Time')
ax.set_title('US COVID-19 Cases and Candle Reviews')

ax.legend(loc=(0.01,0.03))
ax2.legend(loc=(0.01,0.03))
alpha_labels = []
for a in np.arange(0,1.1,0.25):
    alpha_labels.append(matplotlib.lines.Line2D([0],[0],linestyle="none",marker="o",alpha=
ax_score.legend(alpha_labels,np.arange(0,1.1,0.25),title='Unscented \nComplaint Score',loc
#
```

```
<matplotlib.legend.Legend at 0x291e1f57b20>
```

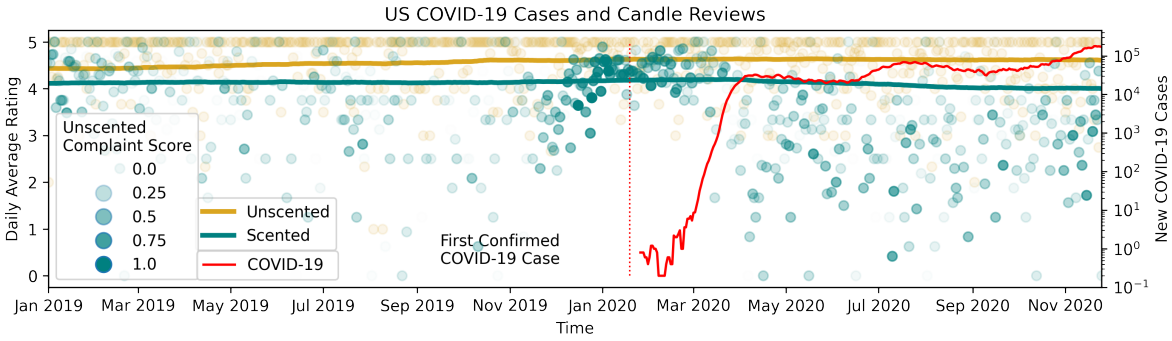


Taking away relevant information

```
fig,ax = plt.subplots(figsize=(12,3))
alpha = 0.1
c1,c2 = 'goldenrod','teal'
roll = 700

### reviews
ax.plot(bydate_unscented['Date'],bydate_unscented['Rating'].rolling(window=roll).mean(),
        c=c1,label='Unscented',lw=3)
ax.plot(bydate['Date'],bydate['Rating'].rolling(window=roll).mean(),
        c=c2,label='Scented',lw=3)

ax.scatter(bydate_unscented['Date'],bydate_unscented['Rating'],
           alpha=alpha,c=c1)
ax_score = ax.twinx()
ax_score.scatter(bydate['Date'],bydate['Rating'],
           c=c2,alpha=bydate['score']/max(bydate['score'])) # using opacity to show compl
ax_score.tick_params(axis='y',right=False,labelcolor='white')

### relevant covid information
ax2 = ax.twinx()
ax2.plot(covid['date'],covid['change'],c='red',label='COVID-19 ')
ax2.set_yscale('log')
ax2.set_ylabel('New COVID-19 Cases')

ax.plot([first_case,first_case],[0,5],c='r',linestyle='dotted',lw=1)
ax.text(first_case-timedelta(days=125),0.25,'First Confirmed \nCOVID-19 Case')

### formatting
```

```
matplotlib.dates.MonthLocator(interval=6)
ax.xaxis.set_major_formatter(matplotlib.dates.DateFormatter('%b %Y'))
ax.set_xlim([date(2019,1,1),date(2020,11,25)])
ax.set_ylabel('Daily Average Rating')
ax.set_xlabel('Time')
ax.set_title('US COVID-19 Cases and Candle Reviews')

ax.legend(loc=(0.14,0.144))
ax2.legend(loc=(0.141,0.03))
alpha_labels = []
for a in np.arange(0,1.1,0.25):
    alpha_labels.append(matplotlib.lines.Line2D([0],[0],linestyle="none",marker="o",alpha=
ax_score.legend(alpha_labels,np.arange(0,1.1,0.25),title='Unscented \nComplaint Score',loc
#
```

```
<matplotlib.legend.Legend at 0x2261d695670>
```



Overly smoothing the data.