CMSC25025 HW1 P5

April 9, 2018

Question 5: Presidential Logorrhea

```
In [239]: import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt
          import sklearn
          import pickle
          import pyspark
          import nltk
          nltk.download('punkt')
          import itertools
```

```
[nltk_data] Downloading package punkt to /Users/ontheroad/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
In [47]: f=open('speeches.pkl','rb')
         speeches = pickle.load(f)
```

```
In [132]: speeches = pd.DataFrame(speeches,
                                  columns = ['president','words','year'])
          speeches['year'] = pd.to_numeric(speeches['year'])
          #speeches.iloc[1]['words']
          #speeches['year'].value_counts()
```

(a) parse sentences

```
In [133]: ### clean the data first
          speeches['words_cleaned'] = speeches['words'].apply(lambda x: x.replace('\r\n\r\n',
          ### parse sentences using nltk.sent_tokenize
          #speeches
          speeches['words_split'] = speeches['words_cleaned'].apply(lambda x: nltk.sent_tokeni
          #speeches.iloc[1]['words_split']
```

(b) sentence statistics

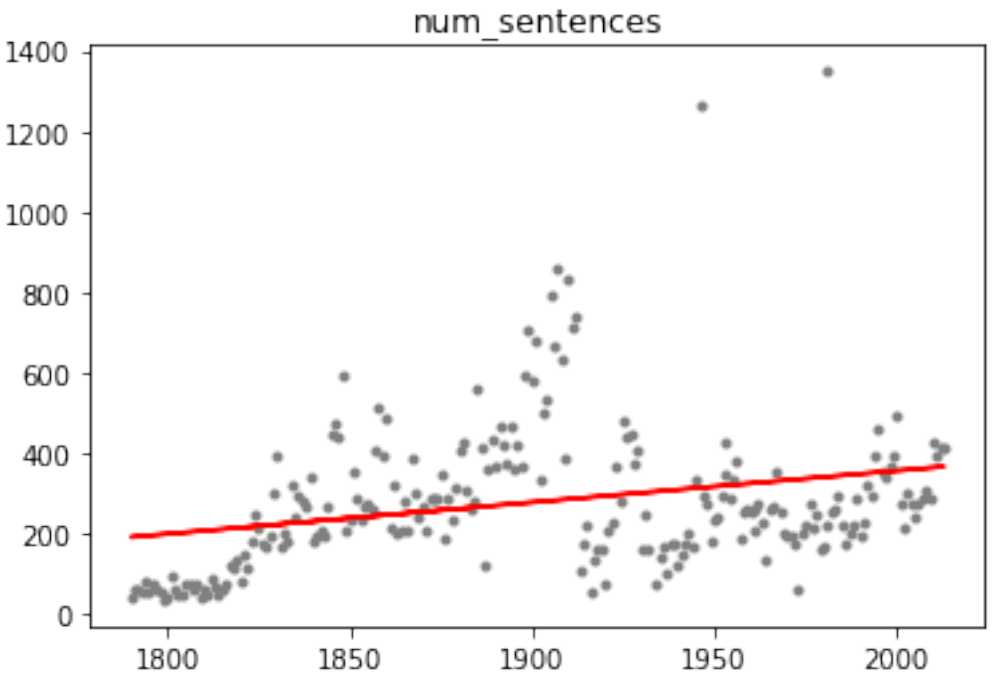
```
In [134]: ### computing num_sentences and mean sentence length in words
          speeches['num_sentences'] = speeches['words_split'].apply(lambda x: len(x))
          speeches['mean'] = speeches['words_split'].apply(lambda x: sum([len(i.split(' ')) fo
          speeches['mean'] = speeches['mean']/speeches['num_sentences']
          #speeches.iloc[1:5]
```

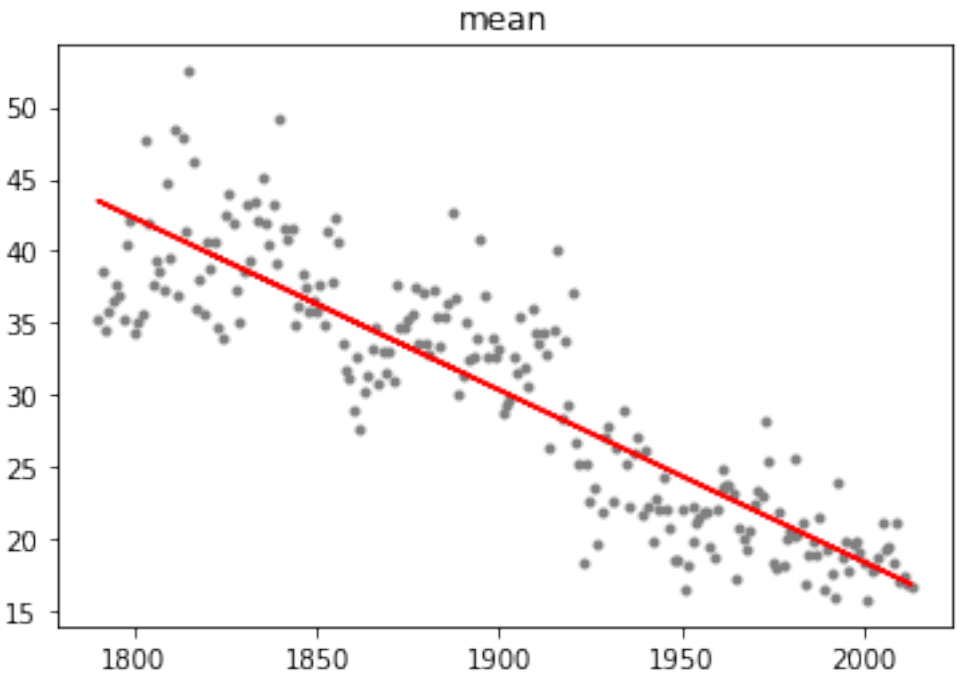
In [131]:
```
## plot and regression
## referene: https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.linal

def regression_plot(x,y,name):
    A = np.vstack([x,np.ones(len(x))]).T
    m, c = np.linalg.lstsq(A,y)[0]
    plt.figure()
    plt.plot(x,y,'.', c = '0.5')
    plt.plot(x,m*x+c,'r',label = name)
    plt.title(name)
    plt.show()

#speeches['year'] = pd.to_numeric(speeches['year'])
x = np.array(speeches['year'])
#x = speeches['year']
y1= np.array(speeches['num_sentences'])
y2= np.array(speeches['mean'])
regression_plot(x,y1,"num_sentences")
regression_plot(x,y2,"mean")
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:6: FutureWarning: `rcond` paramete
To use the future default and silence this warning we advise to pass `rcond=None`, to keep usi
```



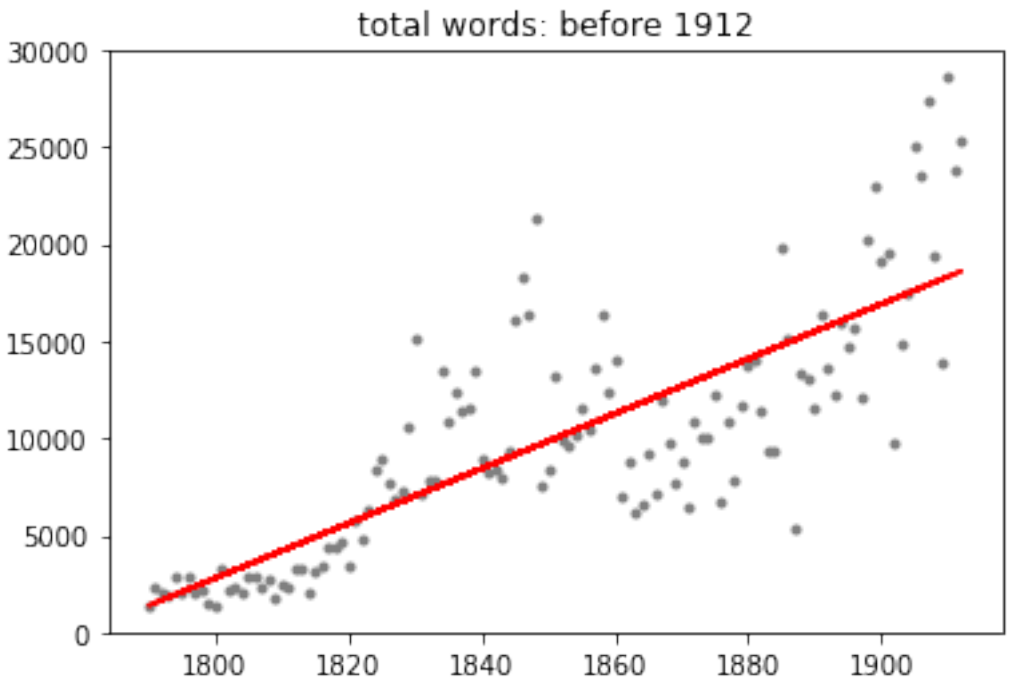


Trend: the number of sentences increase with year; the mean sentence length in words decrease with year.

(c) Compare 1790-1912 and 1913 to present

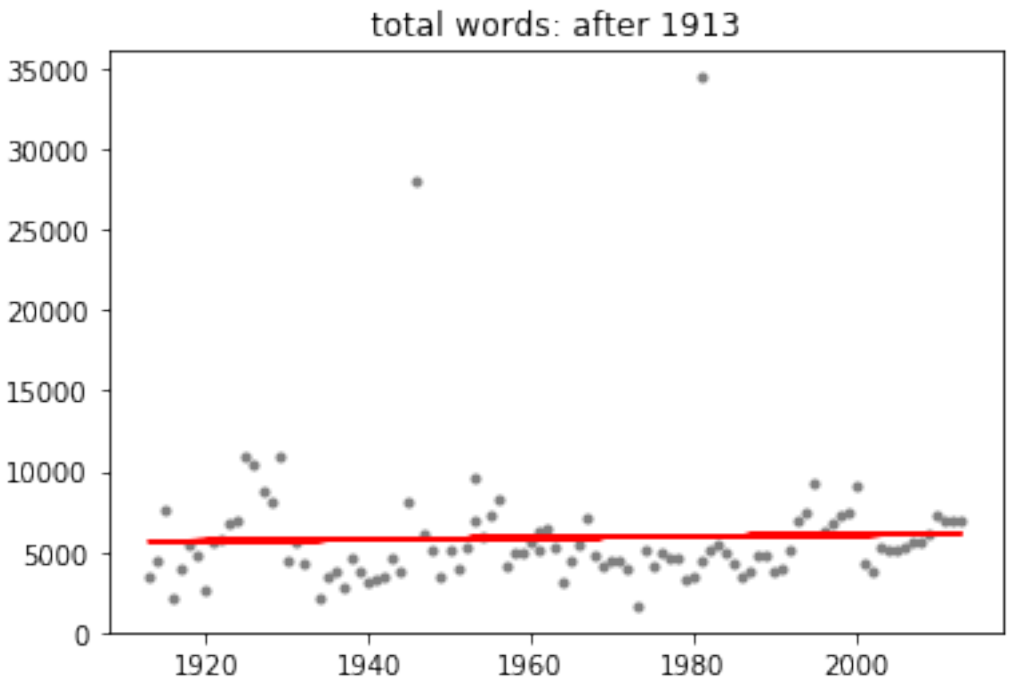
```
In [155]: #data_before1912 = speeches.filter(speeches['year'] < 1912,axis = 0)
          data_before1912 = speeches.query('year <= 1912')
          x_before1912 = np.array(data_before1912['year'])
          y_before1912_totalwords = np.array(data_before1912['num_sentences']*data_before1912[
          regression_plot(x_before1912,y_before1912_totalwords,"total words: before 1912")

/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:6: FutureWarning: `rcond` paramet
To use the future default and silence this warning we advise to pass `rcond=None`, to keep usi
```




```
In [156]: data_after1913 = speeches.query('year > 1912')
          x_after1913 = np.array(data_after1913['year'])
          y_after1913_totalwords = np.array(data_after1913['num_sentences']*data_after1913['mea
          regression_plot(x_after1913,y_after1913_totalwords,"total words: after 1913")
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:6: FutureWarning: `rcond` paramete
To use the future default and silence this warning we advise to pass `rcond=None`, to keep usi
```



Before 1912 there is an increasing trend of the number of words and the total words are well beyond 5000 later on; after 1913 however, the total number of words remain almostf the same, arounf 5000.

(d) longest and shortest sentences

```
In [215]: df = speeches.copy()
```

```
#df['mean'].quantile([0.25,0.75])
df = df.groupby(['president']).mean()
#num_pre = len(df['mean'])
num_pre = df.shape[0]
#df.head()
df = df.sort_values(by = 'mean')
df['index'] = range(len(df['mean']))

#df[['mean']].head()
#df[0].iloc['president']
#print('champion in bullshitting is ', df.loc[df['index'] == 0))
print("the president that speaks the least is:")

df.loc[df['index'] == 0]
```

```
the president that speaks the least is:
```

Out[215]:
```
                year  num_sentences       mean  index
president
George Bush  1990.5         258.25  17.355839      0
```

In [216]:
```
print("the president that speaks the most is:")
df.loc[df['index'] == num_pre-1]
```

```
the president that speaks the most is:
```

Out[216]:
```
                  year  num_sentences       mean  index
president
James Madison  1812.5          61.25  44.674439     40
```

In [219]:
```
print("the 25%, median,75% quantile are:")
df['mean'].quantile([0.25,0.5,0.75])
```

```
the 25%, median,75% quantile are:
```

Out[219]:
```
0.25    21.893192
0.50    32.780434
0.75    36.937670
Name: mean, dtype: float64
```

In [242]:
```
### find the longest and shortest sentence
def find_extreme(x,longest=True): ## x is a list whose elements are "sentences"
    len_list = [len(i.split(' ')) for i in x]
    if longest == True:
        return x[len_list.index(max(len_list))]
    else:
        return x[len_list.index(min(len_list))]
find_extreme(speeches.iloc[0]['words_split'])
#speeches['smallest-sentence'] = speeches['words_split'].apply(lambda x: find_smalle

all_sentences = list(itertools.chain.from_iterable(speeches['words_split']))
print("the longest sentence is:")
find_extreme(all_sentences)
```

```
the longest sentence is:
```

Out[242]:
```
'It shows that the ordinary revenues from all sources for the fiscal year ended June
```

In [244]:
```
print("the shortest sentence is:")
find_extreme(all_sentences,longest=False)
print("which, however is not quite meaningful due to inaccuracies in parsing")
```

```
the shortest sentence is:
which, however is not quite meaningful due to inaccuracies in parsing
```