MCIS6273 Data Mining (Prof. Maull) / Fall 2017 / HW0 SOLUTION

This assignment is worth up to 20 POINTS to your grade total if you complete it on time.

Points Possible	Due Date	Time Commitment (estimated)
20	Wednesday, Sep 06 @ Midnight	4 hours or less

- **GRADING:** Grading will be aligned with the completeness of the objectives.
- **INDEPENDENT WORK:** Copying, cheating, plagiarism and academic dishonesty *are not tolerated* by Univerisity or course policy. Please see the syllabus for the full departmental and University statement on the academic code of honor.

OBJECTIVES

- Familiarize yourself with Github
- Work with exploring unstructured data with Python and text
- Work with a large dataset and understand data munging in Python Pandas

WHAT TO TURN IN

You are being encouraged to turn the assignment in using the provided Jupyter Notebook. To do so, clone the course repository and modify the `hw0.ipynb` file in the `homework/` directory. If you do not know how to do this, please ask, or visit one of the many tutorials out there on the basics of using Github and cloning repositories.

Turn in a copy of a `.ipynb` file, a PDF or Word Document to Blackboard with the answers to the questions labeled with the § sign.

ASSIGNMENT TASKS

(0%) Familiarize yourself with Github

Github is the *de facto* home of open source software on the web. While there are other platforms that provide similar (and in some cases better) services, it is still *the place* for some of the largest open source software projects in the world. You will be creating an account on Github for several reasons:

1. we will be using Github to store, transmit and share homework and lecture notes,
2. we will use Github for at all assignments,
3. at some point in your future, you will *very likely* be using Github, if for private work for a company or public work on an open source project.

§ **Find ONE Github repository of interest and explore it**. There is **nothing to turn in** for this task – just begin to explore Github.

(25%) Work with exploring unstructured data with Python and text

In this part, you will explore some text data and get a little familiar with Python's parsing and text capabilities. You will grab data from the free books provided online from Project Gutenberg and use the provided code to compare these documents. Turn in the answers to the given tasks after studying the code provided in `gutenberg_get_words()` which takes a url for a book on Project Gutenberg and returns a list of the words in the book. Notice the `stopwords=` parameter is used to eliminate words that relay low or no information. This is a common technique use in text processing.

These five books will be used for the tasks for this question:

Book	URL
The Prince, Machiavelli	http://www.gutenberg.org/cache/epub/1232/pg1232.txt
Frankenstein; Or, The Modern Prometheus by Mary Wollstonecraft Shelley	http://www.gutenberg.org/cache/epub/84/pg84.txt
Siddhartha by Hermann Hesse	http://www.gutenberg.org/cache/epub/2500/pg2500.txt
The Republic by Plato	http://www.gutenberg.org/cache/epub/1497/pg1497.txt
The Federalist Papers by Alexander Hamilton, John Jay, and James Madison	http://www.gutenberg.org/cache/epub/1404/pg1404.txt

§ submit the Python code that does the following:

- using the code and the 5 books provided above, explore and apply the very nice Python library called `collections`. Use the `Counter` class to load the word frequencies of each book into a Python dictionary.
- **NOTE**: you will need to be online with an internet connection for this to work, since it loads the data directly from the URLs of the books.

```
US_STOPWORDS = ["a", "about", "above", "above", "across", "after", "afterwards", "again", "against", "a

def gutenberg_get_words(url="http://www.gutenberg.org/cache/epub/1232/pg1232.txt", range=slice(0,None),
    import requests
    import re

    r = requests.get(url)
    data = re.sub(r"[^\w\s]", "", str(r.text)).lower()

    return \
        [w for w in data.split() if w not in stopwords]
```

```
booklist = \
[
    ('The Prince, Machiavelli', 'http://www.gutenberg.org/cache/epub/1232/pg1232.txt'),
    ('Frankenstein; Or, The Modern Prometheus by Mary Wollstonecraft Shelley', 'http://www.gutenberg.or
    ('Siddhartha by Hermann Hesse', 'http://www.gutenberg.org/cache/epub/2500/pg2500.txt'),
    ('The Republic by Plato', 'http://www.gutenberg.org/cache/epub/1497/pg1497.txt'),
    ('The Federalist Papers by Alexander Hamilton, John Jay, and James Madison', 'http://www.gutenberg.
]
```

```
from collections import Counter

word_freq_top30 = {}

for book, url in booklist:
    words = gutenberg_get_words(url, stopwords=US_STOPWORDS)
    word_freq_top30[book] = Counter(words).most_common(30)
```

```
word_freq_top30
```

```
{'Frankenstein; Or, The Modern Prometheus by Mary Wollstonecraft Shelley': [('i',
   2840),
  ('man', 131),
  ('did', 119),
  ('life', 114),
  ('father', 113),
  ('shall', 107),
  ('eyes', 104),
  ('said', 102),
  ('time', 98),
  ('saw', 94),
  ('project', 87),
  ('night', 87),
  ('elizabeth', 86),
  ('mind', 85),
  ('day', 80),
  ('felt', 79),
  ('work', 78),
  ('death', 76),
  ('heart', 76),
  ('feelings', 76),
  ('thought', 74),
  ('dear', 72),
  ('soon', 71),
  ('friend', 70),
  ('passed', 67),
  ('miserable', 65),
  ('heard', 62),
  ('like', 61),
  ('place', 60),
  ('love', 59)],
 'Siddhartha by Hermann Hesse': [('i', 395),
  ('siddhartha', 379),
  ('govinda', 139),
  ('time', 139),
  ('like', 138),
  ('river', 108),
  ('saw', 99),
  ('long', 97),
  ('said', 95),
  ('did', 88),
  ('life', 88),
  ('project', 87),
  ('man', 84),
  ('thought', 82),
  ('love', 80),
  ('learned', 80),
  ('teachings', 77),
  ('just', 74),
  ('felt', 74),
  ('people', 74),
  ('kamala', 72),
  ('heart', 71),
  ('world', 68),
  ('face', 67),
  ('eyes', 66),
  ('oh', 66),
  ('friend', 65),
  ('know', 63),
  ('samana', 62),
  ('spoke', 61)],
 'The Federalist Papers by Alexander Hamilton, John Jay, and James Madison': [('states',
   863),
  ('government', 827),
  ('state', 792),
  ('power', 619),
  ('people', 614),
  ('constitution', 466),
  ('i', 382),
  ('union', 375),
  ('new', 370),
  ('national', 342),
  ('federal', 326),
  ('authority', 295),
  ('public', 294),
  ('great', 294),
  ('ought', 275),
  ('general', 270),
  ('powers', 266),
  ('shall', 257),
  ('executive', 257),
  ('united', 248),
  ('time', 248),
  ('men', 236),
  ('members', 230),
  ('number', 229),
  ('body', 225),
  ('particular', 221),
  ('different', 214),
  ('subject', 213),
  ('laws', 210),
  ('legislative', 210)],
 'The Prince, Machiavelli': [('prince', 222),
  ('i', 216),
  ('men', 161),
  ('castruccio', 136),
  ('people', 115),
  ('having', 109),
  ('ought', 94),
  ('time', 93),
  ('great', 89),
  ('duke', 88),
  ('project', 87),
  ('state', 85),
  ('did', 81),
  ('man', 79),
  ('new', 75),
  ('make', 73),
  ('good', 71),
  ('king', 69),
  ('arms', 63),
  ('fortune', 62),
  ('work', 62),
  ('way', 61),
  ('florentines', 61),
  ('necessary', 60),
  ('princes', 59),
  ('italy', 58),
  ('fear', 57),
  ('things', 57),
  ('gutenbergtm', 56),
  ('said', 55)],
 'The Republic by Plato': [('i', 1782),
  ('said', 1117),
  ('state', 684),
  ('good', 657),
  ('man', 599),
  ('true', 585),
  ('like', 477),
  ('yes', 466),
  ('men', 426),
  ('say', 420),
  ('life', 390),
  ('justice', 358),
  ('soul', 338),
  ('nature', 336),
  ('plato', 325),
  ('just', 309),
  ('knowledge', 292),
  ('replied', 271),
  ('let', 264),
  ('shall', 264),
  ('truth', 263),
  ('certainly', 260),
  ('great', 260),
  ('things', 243),
  ('world', 241),
  ('way', 235),
  ('mind', 230),
  ('evil', 226),
  ('make', 224),
  ('time', 214)]}
```

```python
for k, v in word_freq_top30.items():
    v = [word for word, count in v]
    for j, v_ in word_freq_top30.items():
        v_ = [word for word, count in v_]
        if j != k:
            common_words = set(v).intersection(set(v_))
            print("{} <=> {}\n\t{} common words: {}".format(k, j, len(common_words), common_words))
    print("\n\n")
```

```
The Prince, Machiavelli <=> Frankenstein; Or, The Modern Prometheus by Mary Wollstonecraft Shelley
```

```
    7 common words: {'time', 'work', 'project', 'did', 'said', 'man', 'i'}
The Prince, Machiavelli <=> Siddhartha by Hermann Hesse
    7 common words: {'project', 'people', 'did', 'i', 'said', 'man', 'time'}
The Prince, Machiavelli <=> The Republic by Plato
    11 common words: {'things', 'great', 'men', 'good', 'make', 'i', 'said', 'state', 'way', 'man', 'ti
The Prince, Machiavelli <=> The Federalist Papers by Alexander Hamilton, John Jay, and James Madison
    8 common words: {'time', 'great', 'men', 'people', 'new', 'state', 'i', 'ought'}



Frankenstein; Or, The Modern Prometheus by Mary Wollstonecraft Shelley <=> The Prince, Machiavelli
    7 common words: {'time', 'work', 'project', 'did', 'said', 'man', 'i'}
Frankenstein; Or, The Modern Prometheus by Mary Wollstonecraft Shelley <=> Siddhartha by Hermann Hesse
    15 common words: {'friend', 'like', 'saw', 'project', 'felt', 'did', 'i', 'said', 'thought', 'love'
Frankenstein; Or, The Modern Prometheus by Mary Wollstonecraft Shelley <=> The Republic by Plato
    8 common words: {'life', 'like', 'i', 'said', 'man', 'shall', 'time', 'mind'}
Frankenstein; Or, The Modern Prometheus by Mary Wollstonecraft Shelley <=> The Federalist Papers by Ale
    3 common words: {'time', 'shall', 'i'}



Siddhartha by Hermann Hesse <=> The Prince, Machiavelli
    7 common words: {'time', 'project', 'people', 'did', 'said', 'man', 'i'}
Siddhartha by Hermann Hesse <=> Frankenstein; Or, The Modern Prometheus by Mary Wollstonecraft Shelley
    15 common words: {'friend', 'time', 'like', 'saw', 'project', 'felt', 'did', 'said', 'thought', 'lo
Siddhartha by Hermann Hesse <=> The Republic by Plato
    8 common words: {'like', 'just', 'i', 'said', 'man', 'time', 'world', 'life'}
Siddhartha by Hermann Hesse <=> The Federalist Papers by Alexander Hamilton, John Jay, and James Madiso
    3 common words: {'time', 'people', 'i'}



The Republic by Plato <=> The Prince, Machiavelli
    11 common words: {'things', 'time', 'great', 'men', 'good', 'make', 'said', 'state', 'way', 'man',
The Republic by Plato <=> Frankenstein; Or, The Modern Prometheus by Mary Wollstonecraft Shelley
    8 common words: {'time', 'like', 'said', 'mind', 'man', 'shall', 'i', 'life'}
The Republic by Plato <=> Siddhartha by Hermann Hesse
    8 common words: {'like', 'just', 'i', 'said', 'man', 'time', 'world', 'life'}
The Republic by Plato <=> The Federalist Papers by Alexander Hamilton, John Jay, and James Madison
    6 common words: {'time', 'great', 'men', 'state', 'shall', 'i'}



The Federalist Papers by Alexander Hamilton, John Jay, and James Madison <=> The Prince, Machiavelli
    8 common words: {'time', 'great', 'men', 'people', 'new', 'state', 'i', 'ought'}
The Federalist Papers by Alexander Hamilton, John Jay, and James Madison <=> Frankenstein; Or, The Mode
    3 common words: {'time', 'shall', 'i'}
The Federalist Papers by Alexander Hamilton, John Jay, and James Madison <=> Siddhartha by Hermann Hess
    3 common words: {'people', 'time', 'i'}
The Federalist Papers by Alexander Hamilton, John Jay, and James Madison <=> The Republic by Plato
    6 common words: {'great', 'men', 'i', 'state', 'shall', 'time'}
```

§ turn in *at least* 2 sentences and any code if you used code to answering the following:

- there are similarities and differences in the top 30 words of the five provided documents – be specific about describing what they are? How similar or different are each of the top 30 words list? You can compare them by hand (look at them) or you are encouraged to write Python code to compare them automatically.

(75%) Work with a large dataset and understand data munging in Python Pandas

As we have learned in class, a great deal of time doing data mining involves understanding the data in a dataset and preprocessing it in preparation for working with it in a real analysis of some sort. We will develop an understanding of:

- importing CSV data into Pandas DataFrames,
- exporting JSON data from a Pandas DataFrame,
- understand how to compute data from DataFrame elements

You might find the Pandas IO library helpful in completing this task.

We will be using an interesting data source: **US Baseball Statistics Archive** by Sean Lahman (CCBY-SA 3.0), which can be found in these two locations. You can download the entire archive to your local machine.

- http://seanlahman.com/baseball-archive/statistics/
- https://github.com/chadwickbureau/baseballdatabank

§ **submit the code to load the batting table** from the Baseball Archive directly from the Github repository into a Pandas DataFrame.

- You can use the CSV file at this url. You will most certainly be using the `read_csv()` method.

```
import pandas as pd
df = pd.read_csv('./data/Batting.csv') # reading locally
```

§ **submit the code to write the JSON file that contains all the batting records for the Cincinnati Reds (CIN) in 1981**.

- This will require a simple query into the dataset (loaded as a DataFrame) – please read the documentation on boolean indexing.

- You will need to explore the `to_json()` method to then convert the DataFrame that results from the query above into a JSON object.

```
df.head()
```

```
<tr style="text-align: right;">
  <th></th>
  <th>playerID</th>
  <th>yearID</th>
  <th>stint</th>
  <th>teamID</th>
  <th>lgID</th>
  <th>G</th>
  <th>AB</th>
  <th>R</th>
  <th>H</th>
  <th>2B</th>
  <th>...</th>
  <th>RBI</th>
  <th>SB</th>
  <th>CS</th>
  <th>BB</th>
  <th>SO</th>
```

```
    <th>IBB</th>
    <th>HBP</th>
    <th>SH</th>
    <th>SF</th>
    <th>GIDP</th>
  </tr>
  <tr>
    <th>0</th>
    <td>abercda01</td>
    <td>1871</td>
    <td>1</td>
    <td>TRO</td>
    <td>NaN</td>
    <td>1</td>
    <td>4</td>
    <td>0</td>
    <td>0</td>
    <td>0</td>
    <td>...</td>
    <td>0.0</td>
    <td>0.0</td>
    <td>0.0</td>
    <td>0</td>
    <td>0.0</td>
    <td>NaN</td>
    <td>NaN</td>
    <td>NaN</td>
    <td>NaN</td>
    <td>NaN</td>
  </tr>
  <tr>
    <th>1</th>
    <td>addybo01</td>
    <td>1871</td>
    <td>1</td>
    <td>RC1</td>
    <td>NaN</td>
    <td>25</td>
    <td>118</td>
    <td>30</td>
    <td>32</td>
    <td>6</td>
    <td>...</td>
    <td>13.0</td>
    <td>8.0</td>
    <td>1.0</td>
    <td>4</td>
    <td>0.0</td>
    <td>NaN</td>
    <td>NaN</td>
    <td>NaN</td>
    <td>NaN</td>
    <td>NaN</td>
```

```
</tr>
<tr>
  <th>2</th>
  <td>allisar01</td>
  <td>1871</td>
  <td>1</td>
  <td>CL1</td>
  <td>NaN</td>
  <td>29</td>
  <td>137</td>
  <td>28</td>
  <td>40</td>
  <td>4</td>
  <td>...</td>
  <td>19.0</td>
  <td>3.0</td>
  <td>1.0</td>
  <td>2</td>
  <td>5.0</td>
  <td>NaN</td>
  <td>NaN</td>
  <td>NaN</td>
  <td>NaN</td>
  <td>NaN</td>
</tr>
<tr>
  <th>3</th>
  <td>allisdo01</td>
  <td>1871</td>
  <td>1</td>
  <td>WS3</td>
  <td>NaN</td>
  <td>27</td>
  <td>133</td>
  <td>28</td>
  <td>44</td>
  <td>10</td>
  <td>...</td>
  <td>27.0</td>
  <td>1.0</td>
  <td>1.0</td>
  <td>0</td>
  <td>2.0</td>
  <td>NaN</td>
  <td>NaN</td>
  <td>NaN</td>
  <td>NaN</td>
  <td>NaN</td>
</tr>
<tr>
  <th>4</th>
  <td>ansonca01</td>
  <td>1871</td>
  <td>1</td>
  <td>RC1</td>
  <td>NaN</td>
  <td>25</td>
  <td>120</td>
  <td>29</td>
  <td>39</td>
  <td>11</td>
  <td>...</td>
  <td>16.0</td>
  <td>6.0</td>
  <td>2.0</td>
  <td>2</td>
  <td>1.0</td>
  <td>NaN</td>
  <td>NaN</td>
  <td>NaN</td>
  <td>NaN</td>
  <td>NaN</td>
</tr>
```

5 rows × 22 columns

```
df_cinc81 = df[(df.teamID=='CIN') & (df.yearID==1981)]
df_cinc81.to_json('cinc81_batting.json')
```

§ submit the code that calculates the following batting rank score for the 1981 Reds (CIN)

$$\varphi = G\frac{H}{AB} + \frac{RBI}{\sqrt{R}}$$

where G, H, AB, RBI, R are *games*, *hits*, *at bats*, *runs batted in* and *runs* respectively.

- Your answer should ONLY include the batting for the **1981 Cincinnati Reds batting roster**.

```
# while not a requirement for the assignment, we should take out problematic 0 values (in a denomenator
df_cinc81 = df_cinc81[(df_cinc81.AB>0) & (df_cinc81.R>0)]

df_cinc81['score_varphi'] = df_cinc81.G * ( df_cinc81.H /  df_cinc81.AB) +  df_cinc81.RBI / (df_cinc81.

df_cinc81.head()
```

```
<tr style="text-align: right;">
  <th></th>
  <th>playerID</th>
  <th>yearID</th>
  <th>stint</th>
  <th>teamID</th>
  <th>lgID</th>
  <th>G</th>
  <th>AB</th>
  <th>R</th>
  <th>H</th>
  <th>2B</th>
  <th>...</th>
  <th>SB</th>
  <th>CS</th>
  <th>BB</th>
  <th>SO</th>
  <th>IBB</th>
```

```
    <th>HBP</th>
    <th>SH</th>
    <th>SF</th>
    <th>GIDP</th>
    <th>score_varphi</th>
  </tr>

  <tr>
    <th>58351</th>
    <td>bairdo01</td>
    <td>1981</td>
    <td>1</td>
    <td>CIN</td>
    <td>NL</td>
    <td>24</td>
    <td>3</td>
    <td>1</td>
    <td>1</td>
    <td>0</td>
    <td>...</td>
    <td>0.0</td>
    <td>0.0</td>
    <td>0</td>
    <td>1.0</td>
    <td>0.0</td>
    <td>0.0</td>
    <td>0.0</td>
    <td>0.0</td>
    <td>0.0</td>
    <td>11.000000</td>
  </tr>
  <tr>
    <th>58363</th>
    <td>barrage01</td>
    <td>1981</td>
    <td>1</td>
    <td>CIN</td>
    <td>NL</td>
    <td>9</td>
    <td>6</td>
    <td>2</td>
    <td>2</td>
    <td>0</td>
    <td>...</td>
    <td>0.0</td>
    <td>0.0</td>
    <td>0</td>
    <td>0.0</td>
    <td>0.0</td>
    <td>0.0</td>
    <td>0.0</td>
    <td>0.0</td>
    <td>0.0</td>
    <td>3.707107</td>
```

```
</tr>
<tr>
  <th>58375</th>
  <td>benchjo01</td>
  <td>1981</td>
  <td>1</td>
  <td>CIN</td>
  <td>NL</td>
  <td>52</td>
  <td>178</td>
  <td>14</td>
  <td>55</td>
  <td>8</td>
  <td>...</td>
  <td>0.0</td>
  <td>2.0</td>
  <td>17</td>
  <td>21.0</td>
  <td>3.0</td>
  <td>0.0</td>
  <td>1.0</td>
  <td>0.0</td>
  <td>4.0</td>
  <td>22.748947</td>
</tr>
<tr>
  <th>58378</th>
  <td>berenbr01</td>
  <td>1981</td>
  <td>1</td>
  <td>CIN</td>
  <td>NL</td>
  <td>21</td>
  <td>42</td>
  <td>4</td>
  <td>8</td>
  <td>3</td>
  <td>...</td>
  <td>0.0</td>
  <td>0.0</td>
  <td>2</td>
  <td>18.0</td>
  <td>0.0</td>
  <td>0.0</td>
  <td>3.0</td>
  <td>1.0</td>
  <td>1.0</td>
  <td>4.500000</td>
</tr>
<tr>
  <th>58388</th>
  <td>biittla01</td>
  <td>1981</td>
  <td>1</td>
  <td>CIN</td>
  <td>NL</td>
  <td>42</td>
  <td>61</td>
  <td>1</td>
  <td>13</td>
  <td>4</td>
  <td>...</td>
  <td>0.0</td>
  <td>0.0</td>
  <td>4</td>
  <td>4.0</td>
  <td>1.0</td>
  <td>0.0</td>
  <td>0.0</td>
  <td>1.0</td>
  <td>3.0</td>
  <td>16.950820</td>
</tr>
```

5 rows × 23 columns

§ for the 1981 data submit the top 4 batter IDs by their batting rank score, φ, previously calculated.

```
df_cinc81[['playerID', 'score_varphi']].sort_values(by='score_varphi', ascending=False).head(4)
```

```
<tr style="text-align: right;">
  <th></th>
  <th>playerID</th>
  <th>score_varphi</th>
</tr>

<tr>
  <th>58577</th>
  <td>fostege01</td>
  <td>43.076087</td>
</tr>
<tr>
  <th>58477</th>
  <td>conceda01</td>
  <td>41.354177</td>
</tr>
<tr>
  <th>58618</th>
  <td>griffke01</td>
  <td>35.588393</td>
</tr>
<tr>
  <th>58931</th>
  <td>oestero01</td>
  <td>34.735567</td>
</tr>
```