Cultural Analytics

Cultural Analytics

ENGL 64.05

Fall 2019

Prof. James E. Dobson

Any Source Can Be Quantified

- A priori, no source either requires or excludes quantification.
- Comparison is essential in order to make sense of results generally, and especially of numbers.
- In order to compare only what is comparable, one needs a well-defined corpus.
- Avoid dwelling at length on minor differences between scant corpora.
- Above all, know how to sample properly.
- Quantification makes sense only if the results obtained are commensurate with the effort required, especially in data collection.

(Lemericer and Zalc, 29)

Samples

- Do we need all the data?
  - Noise becomes less “meaningful” in larger datasets.
  - But bias can increase or be hidden.
- Random sampling can be better than “all the data.”
  - But how do we know that we’ve selected random data?
- We might combine random and stratified sampling to ensure that we have enough representation from underrepresented categories.

DTM: Document Term Matrix

- Each “Type” is a vocabulary term.
- Tokens are counted and placed in the correct columns.
- Each row represents a different text.

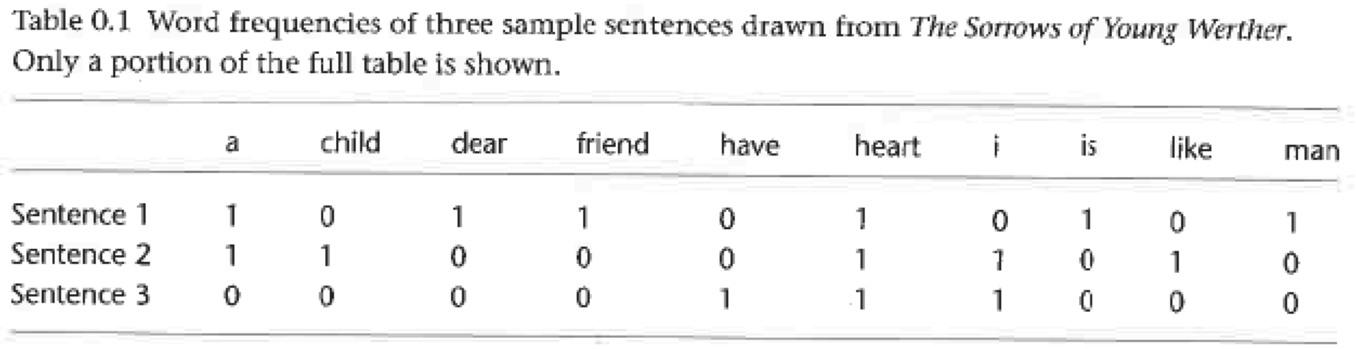
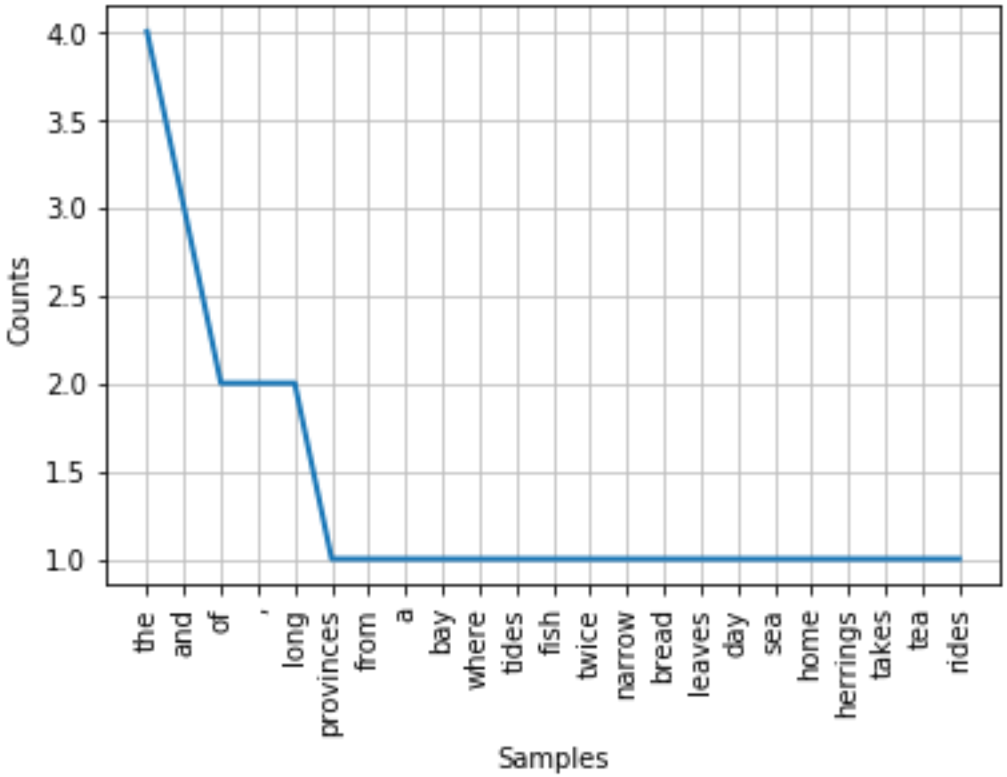


Table 0.1 Word frequencies of three sample sentences drawn from *The Sorrows of Young Werther*. Only a portion of the full table is shown.

	a	child	dear	friend	have	heart	i	is	like	man
Sentence 1	1	0	1	1	0	1	0	1	0	1
Sentence 2	1	1	0	0	0	1	1	0	1	0
Sentence 3	0	0	0	0	1	1	1	0	0	0

From narrow provinces
of fish and bread and tea,
home of the long tides
where the bay leaves the sea
twice a day and takes
the herrings long rides,

Elizabeth Bishop, from “The Moose”



Gwendolyn Brooks
“We Real Cool”

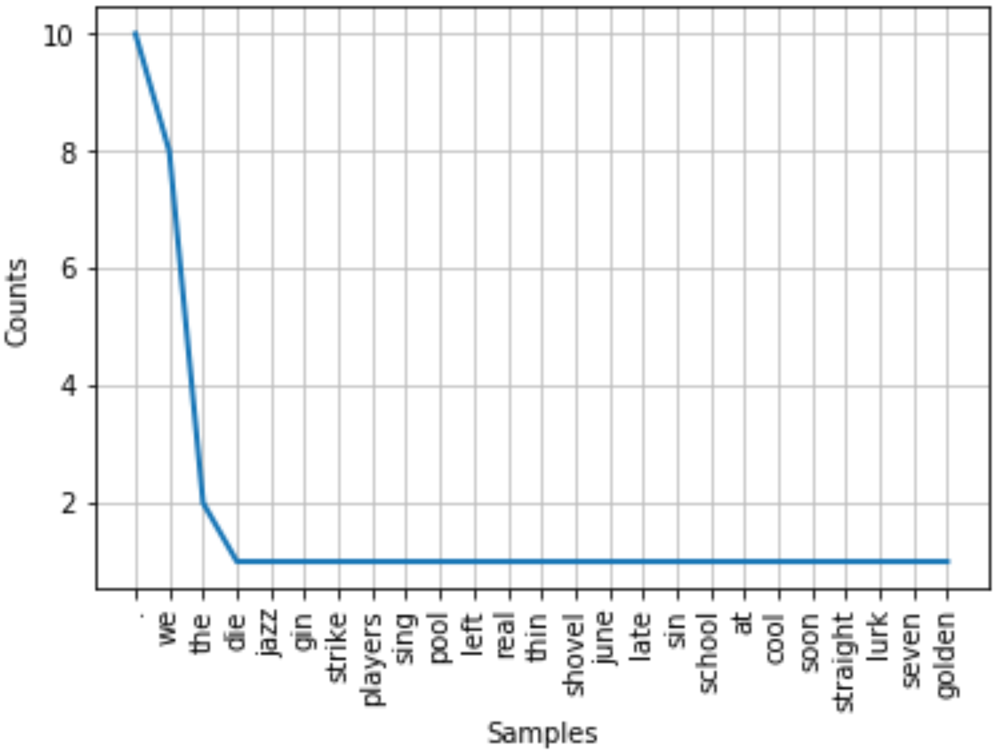
The Pool Players.
Seven at the Golden Shovel.

We real cool. We
Left school. We

Lurk late. We
Strike straight. We

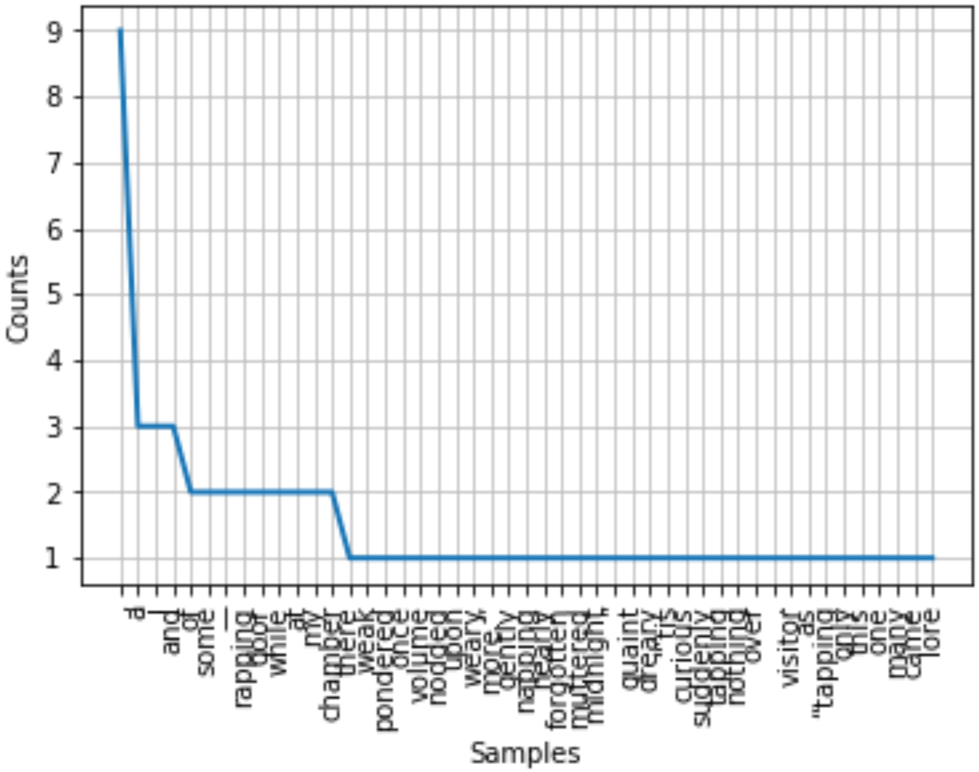
Sing sin. We
Thin gin. We

Jazz June. We
Die soon.



Once upon a midnight dreary, while I pondered, weak and weary,
Over many a quaint and curious volume of forgotten lore —
    While I nodded, nearly napping, suddenly there came a
tapping,
As of some one gently rapping, rapping at my chamber door.
"'Tis some visitor," I muttered, "tapping at my chamber door —
            Only this and nothing more."

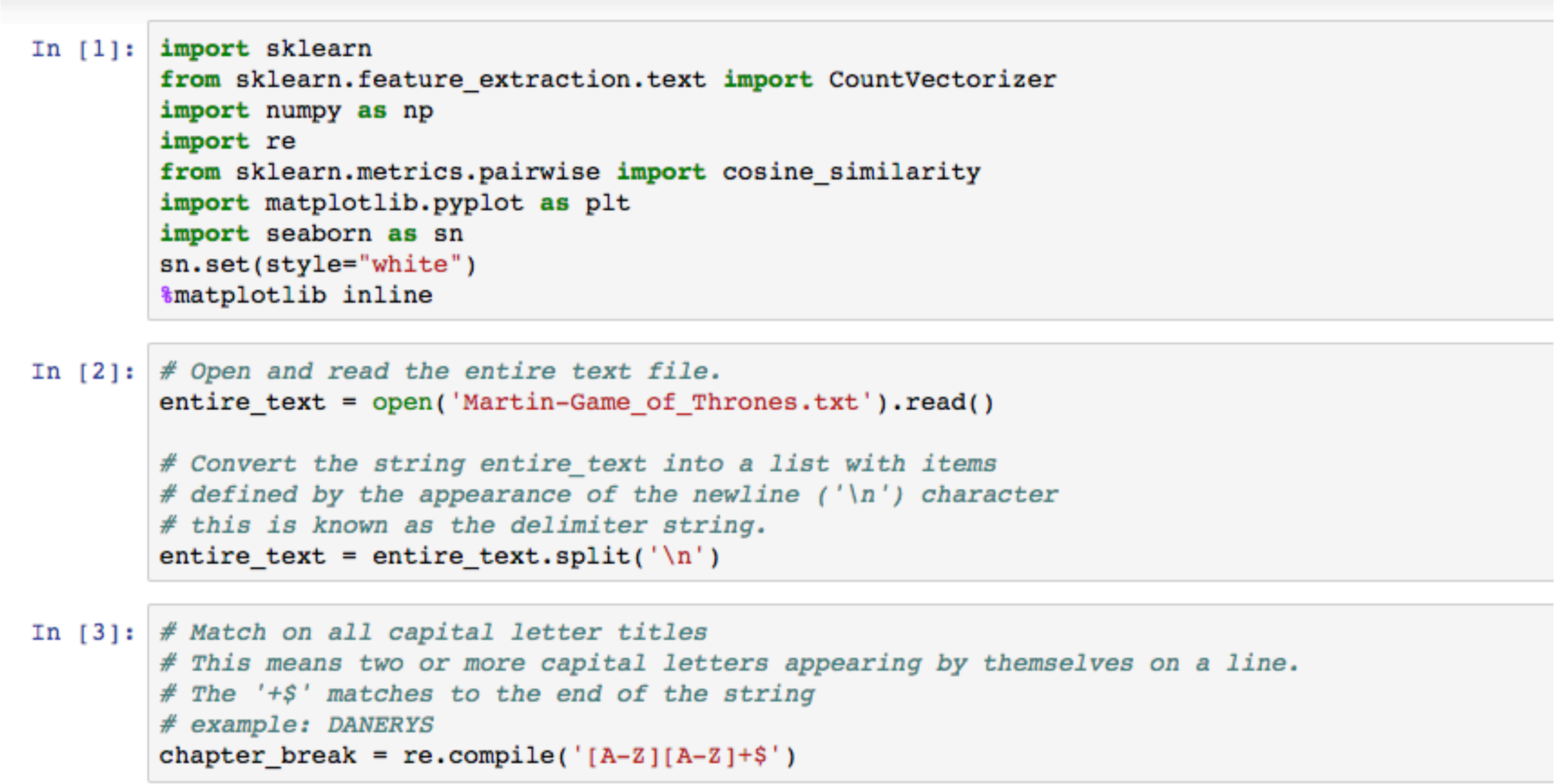
Edgar Allan Poe, from "The Raven"



Text Segmentation

- Book or document-level statistics are frequently not useful.
- Perhaps we can compare units within a single text.
- In order to compare objects of different length, we might need standardized units. Chapters? Paragraphs? Word-units?

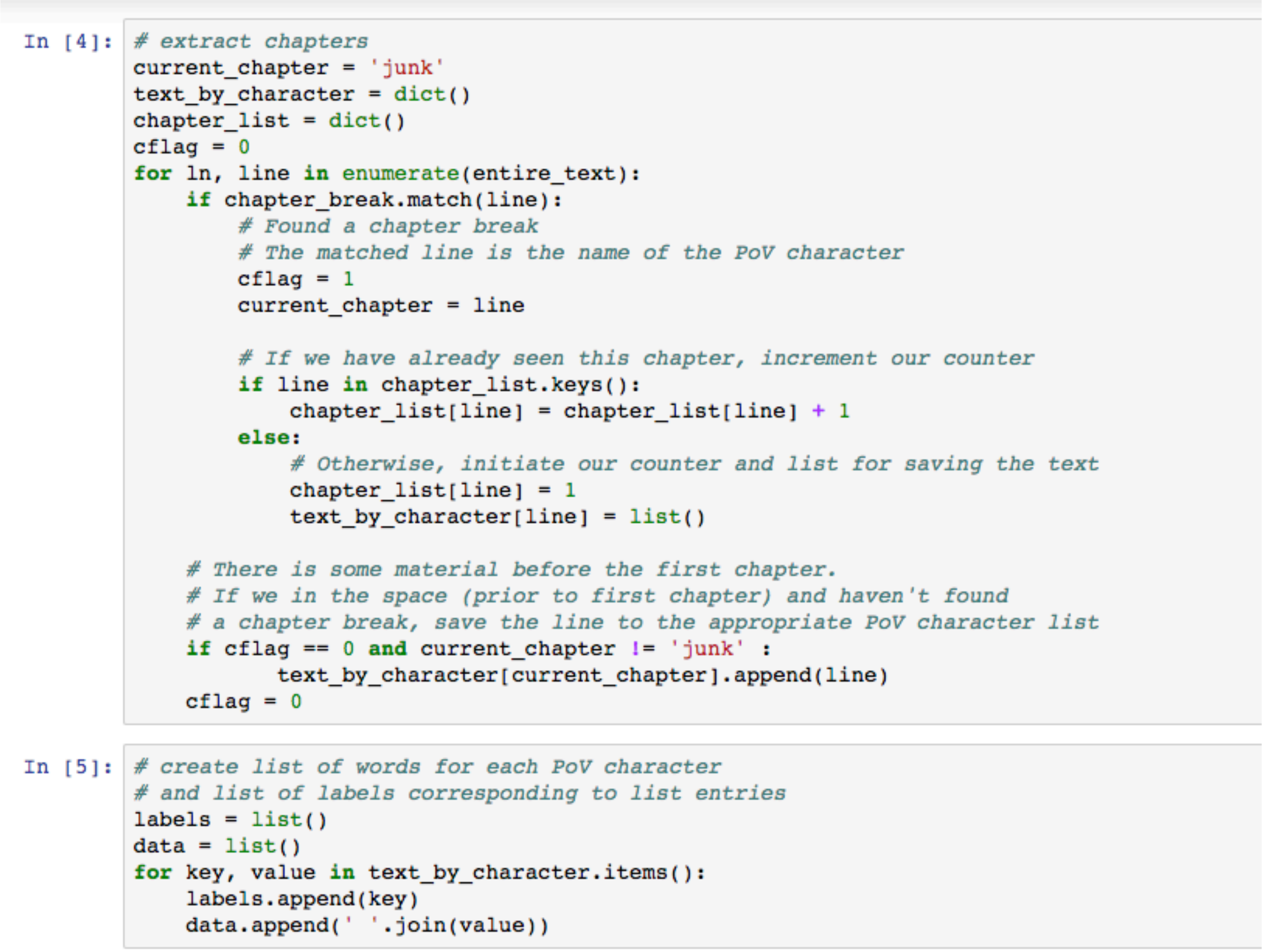
Cultural Analytics



```
In [1]: import sklearn
        from sklearn.feature_extraction.text import CountVectorizer
        import numpy as np
        import re
        from sklearn.metrics.pairwise import cosine_similarity
        import matplotlib.pyplot as plt
        import seaborn as sn
        sn.set(style="white")
        %matplotlib inline
```

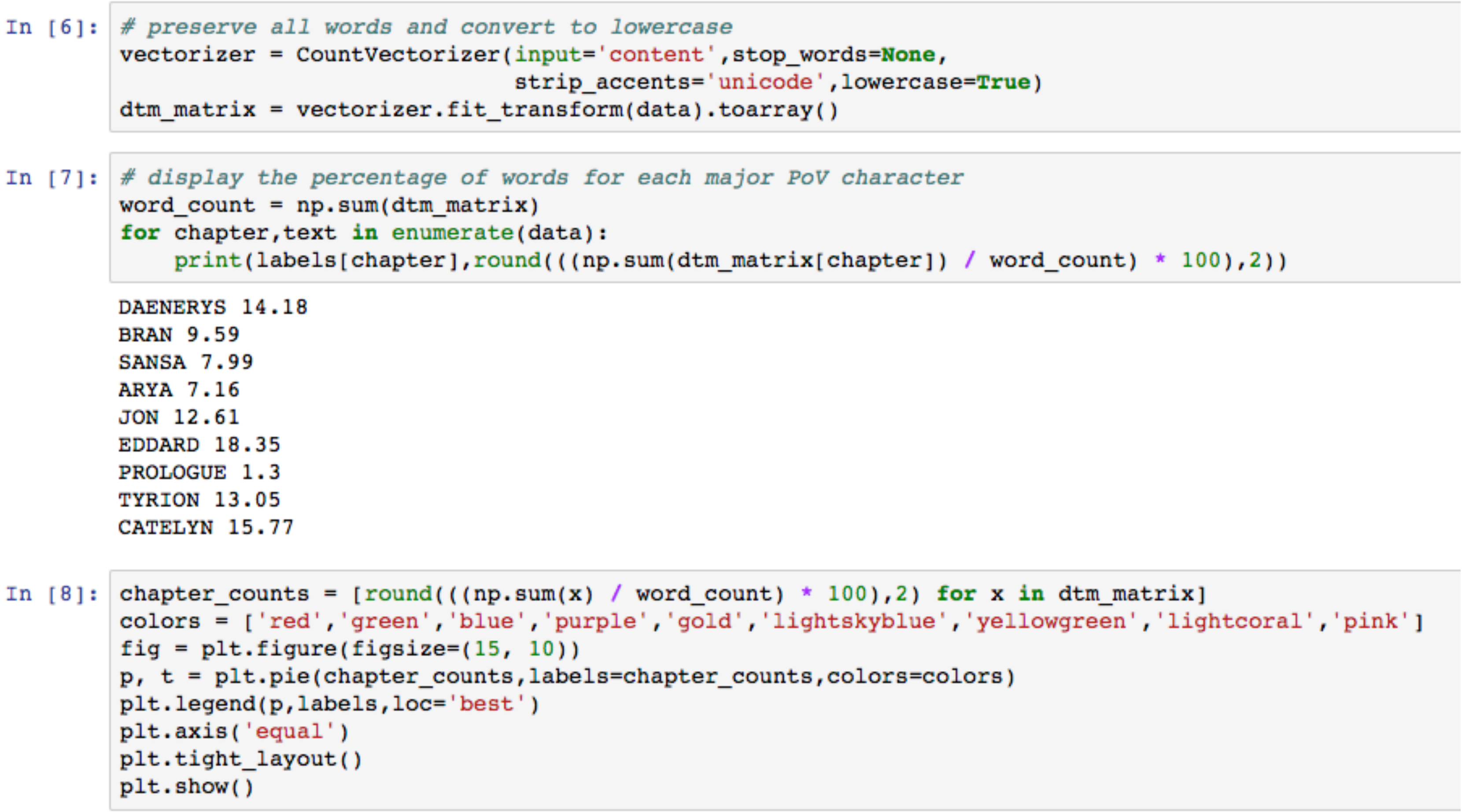
```
In [2]: # Open and read the entire text file.
        entire_text = open('Martin-Game_of_Thrones.txt').read()

        # Convert the string entire_text into a list with items
        # defined by the appearance of the newline ('\n') character
        # this is known as the delimiter string.
        entire_text = entire_text.split('\n')
```

```
In [3]: # Match on all capital letter titles
        # This means two or more capital letters appearing by themselves on a line.
        # The '+$' matches to the end of the string
        # example: DANERYS
        chapter_break = re.compile('[A-Z][A-Z]+$')
```



```
In [4]: # extract chapters
        current_chapter = 'junk'
        text_by_character = dict()
        chapter_list = dict()
        cflag = 0
        for ln, line in enumerate(entire_text):
            if chapter_break.match(line):
                # Found a chapter break
                # The matched line is the name of the PoV character
                cflag = 1
                current_chapter = line

                # If we have already seen this chapter, increment our counter
                if line in chapter_list.keys():
                    chapter_list[line] = chapter_list[line] + 1
                else:
                    # Otherwise, initiate our counter and list for saving the text
                    chapter_list[line] = 1
                    text_by_character[line] = list()

            # There is some material before the first chapter.
            # If we in the space (prior to first chapter) and haven't found
            # a chapter break, save the line to the appropriate PoV character list
            if cflag == 0 and current_chapter != 'junk' :
                   text_by_character[current_chapter].append(line)
            cflag = 0
```

```
In [5]: # create list of words for each PoV character
        # and list of labels corresponding to list entries
        labels = list()
        data = list()
        for key, value in text_by_character.items():
            labels.append(key)
            data.append(' '.join(value))
```



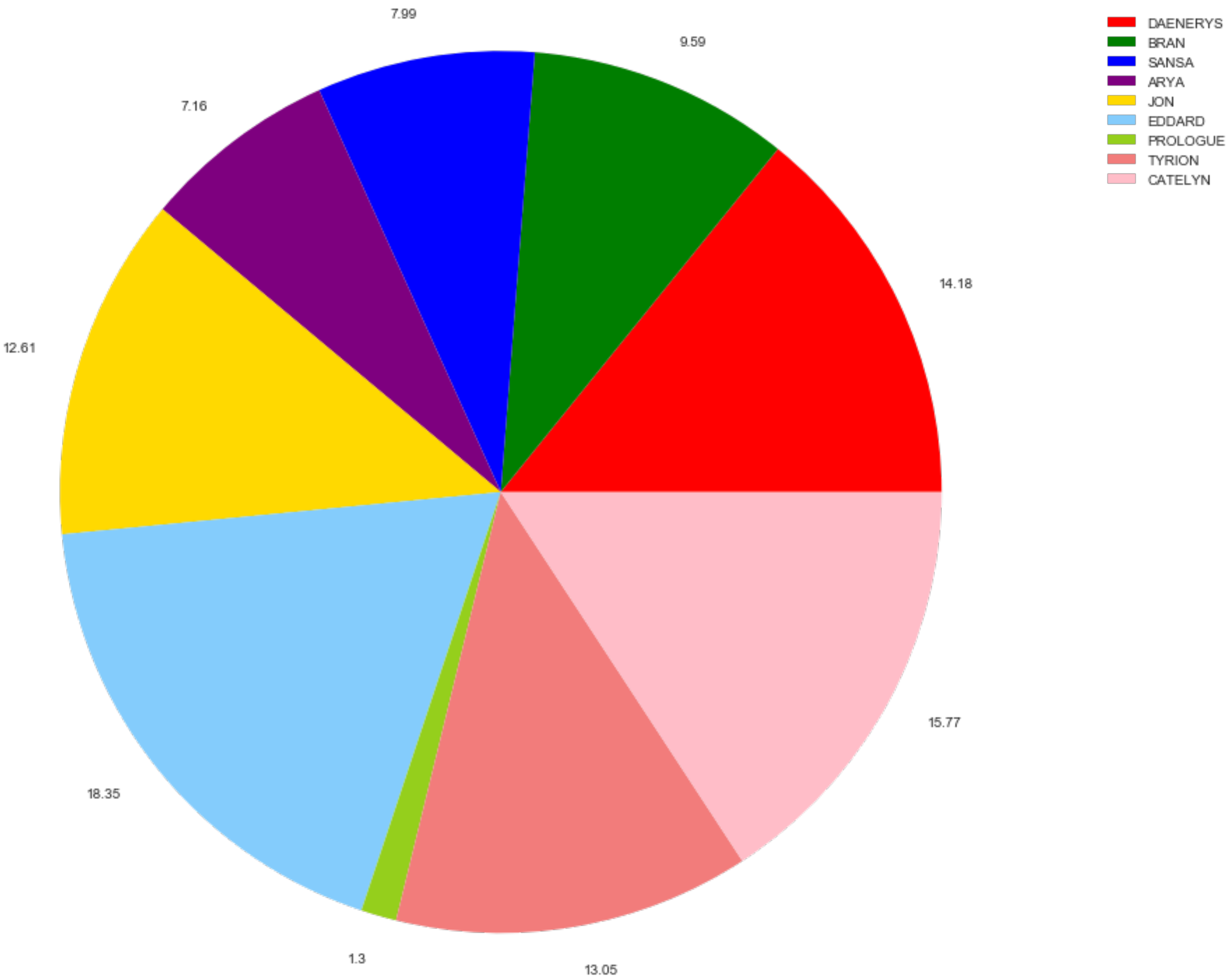
In [6]:
```
# preserve all words and convert to lowercase
vectorizer = CountVectorizer(input='content',stop_words=None,
                             strip_accents='unicode',lowercase=True)
dtm_matrix = vectorizer.fit_transform(data).toarray()
```

In [7]:
```
# display the percentage of words for each major PoV character
word_count = np.sum(dtm_matrix)
for chapter,text in enumerate(data):
    print(labels[chapter],round(((np.sum(dtm_matrix[chapter]) / word_count) * 100),2))
```

```
DAENERYS 14.18
BRAN 9.59
SANSA 7.99
ARYA 7.16
JON 12.61
EDDARD 18.35
PROLOGUE 1.3
TYRION 13.05
CATELYN 15.77
```

In [8]:
```
chapter_counts = [round(((np.sum(x) / word_count) * 100),2) for x in dtm_matrix]
colors = ['red','green','blue','purple','gold','lightskyblue','yellowgreen','lightcoral','pink']
fig = plt.figure(figsize=(15, 10))
p, t = plt.pie(chapter_counts,labels=chapter_counts,colors=colors)
plt.legend(p,labels,loc='best')
plt.axis('equal')
plt.tight_layout()
plt.show()
```

Cultural Analytics



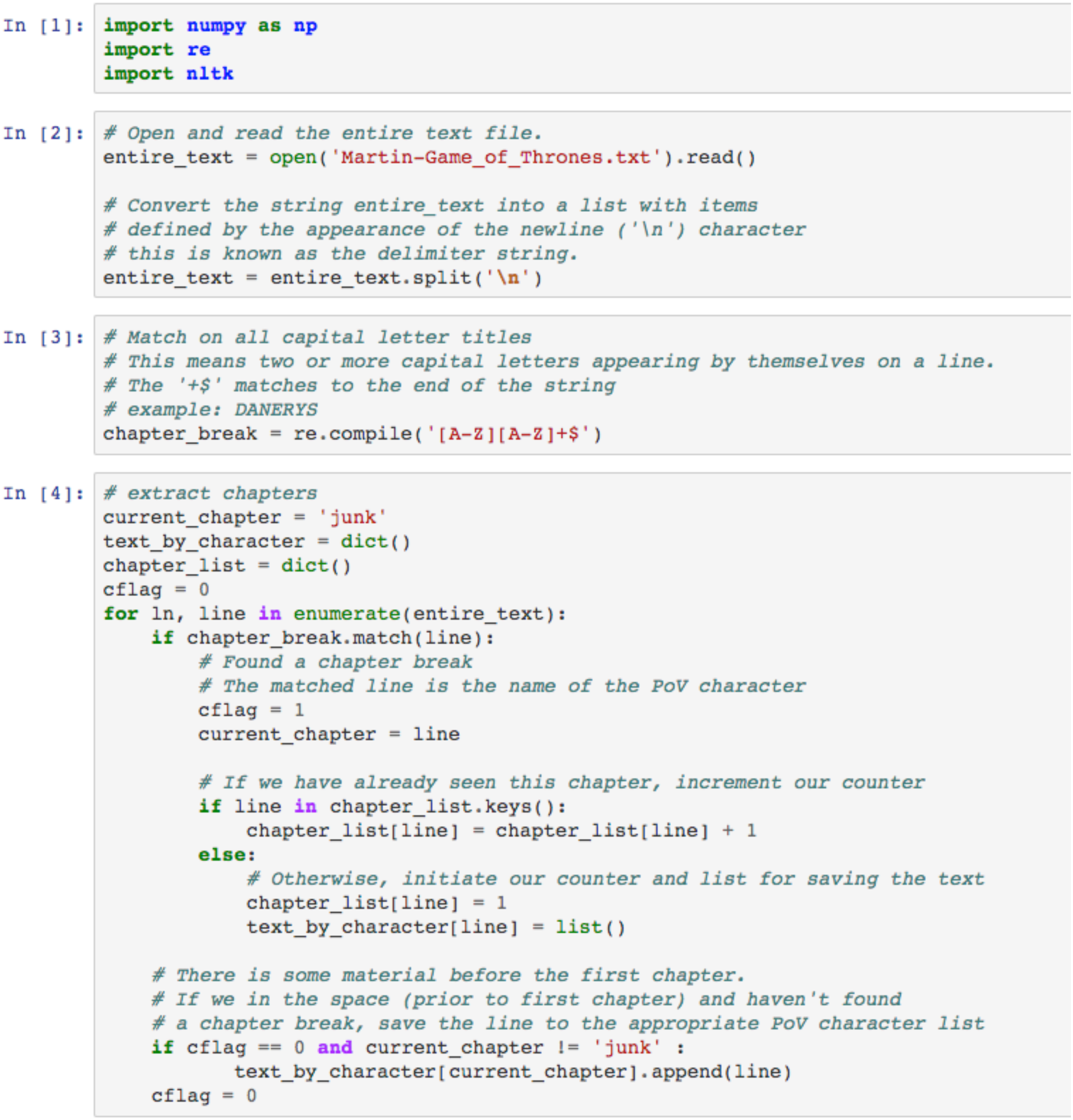
DARTMOUTH
VOX CLAMANTIS IN DESERTO
1769

Cultural Analytics

TTR: Type / Token Ratio

- One way to measure vocabulary "richness."
- Examine relationship between total number of different words and total number of words used.
- Can measure vocabulary diversity and flexibility.

Cultural Analytics



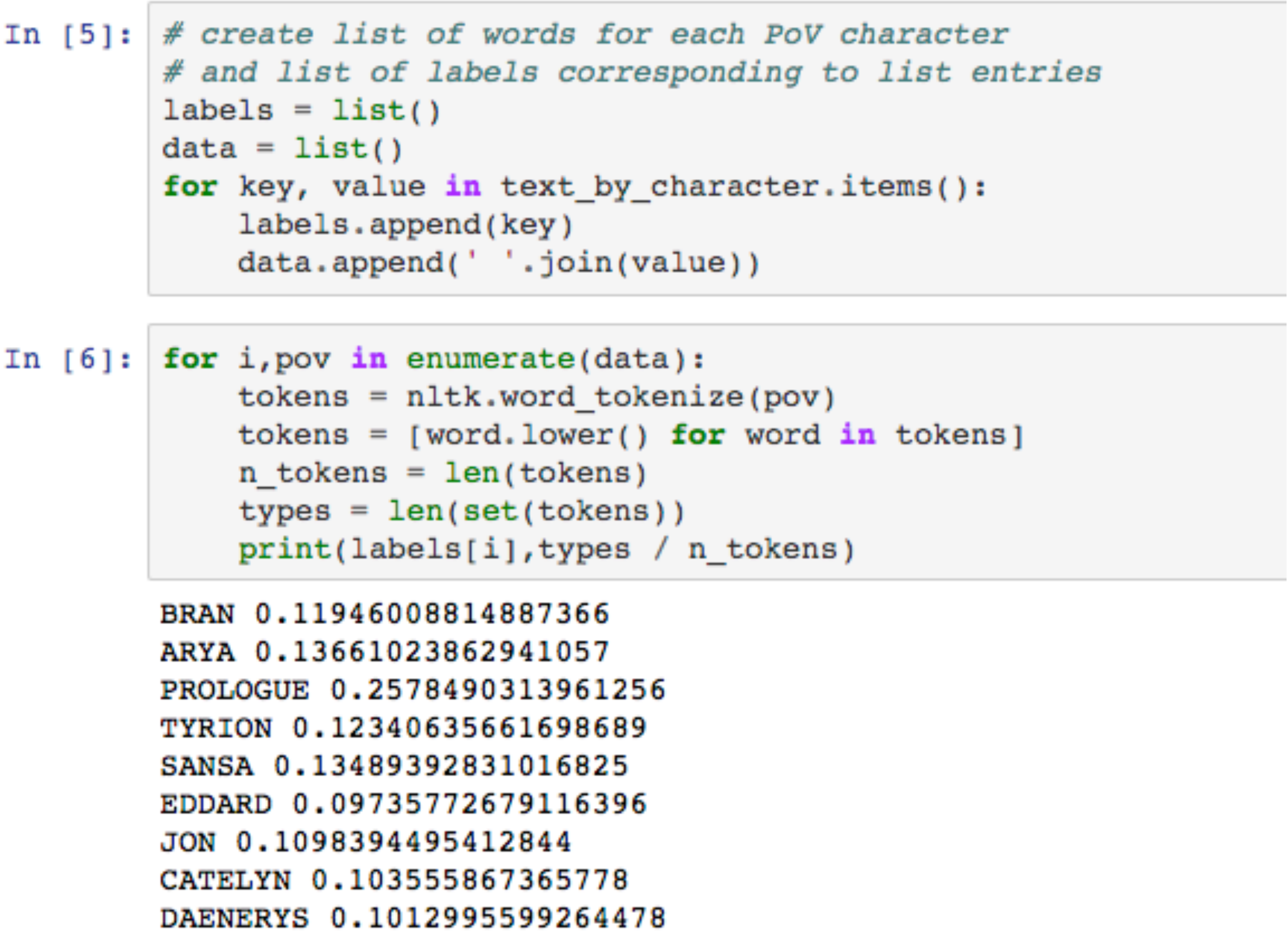
```
In [1]: import numpy as np
        import re
        import nltk
```

```
In [2]: # Open and read the entire text file.
        entire_text = open('Martin-Game_of_Thrones.txt').read()

        # Convert the string entire_text into a list with items
        # defined by the appearance of the newline ('\n') character
        # this is known as the delimiter string.
        entire_text = entire_text.split('\n')
```

```
In [3]: # Match on all capital letter titles
        # This means two or more capital letters appearing by themselves on a line.
        # The '+$' matches to the end of the string
        # example: DANERYS
        chapter_break = re.compile('[A-Z][A-Z]+$')
```

```
In [4]: # extract chapters
        current_chapter = 'junk'
        text_by_character = dict()
        chapter_list = dict()
        cflag = 0
        for ln, line in enumerate(entire_text):
            if chapter_break.match(line):
                # Found a chapter break
                # The matched line is the name of the PoV character
                cflag = 1
                current_chapter = line

                # If we have already seen this chapter, increment our counter
                if line in chapter_list.keys():
                    chapter_list[line] = chapter_list[line] + 1
                else:
                    # Otherwise, initiate our counter and list for saving the text
                    chapter_list[line] = 1
                    text_by_character[line] = list()

            # There is some material before the first chapter.
            # If we in the space (prior to first chapter) and haven't found
            # a chapter break, save the line to the appropriate PoV character list
            if cflag == 0 and current_chapter != 'junk' :
                   text_by_character[current_chapter].append(line)
            cflag = 0
```

Cultural Analytics

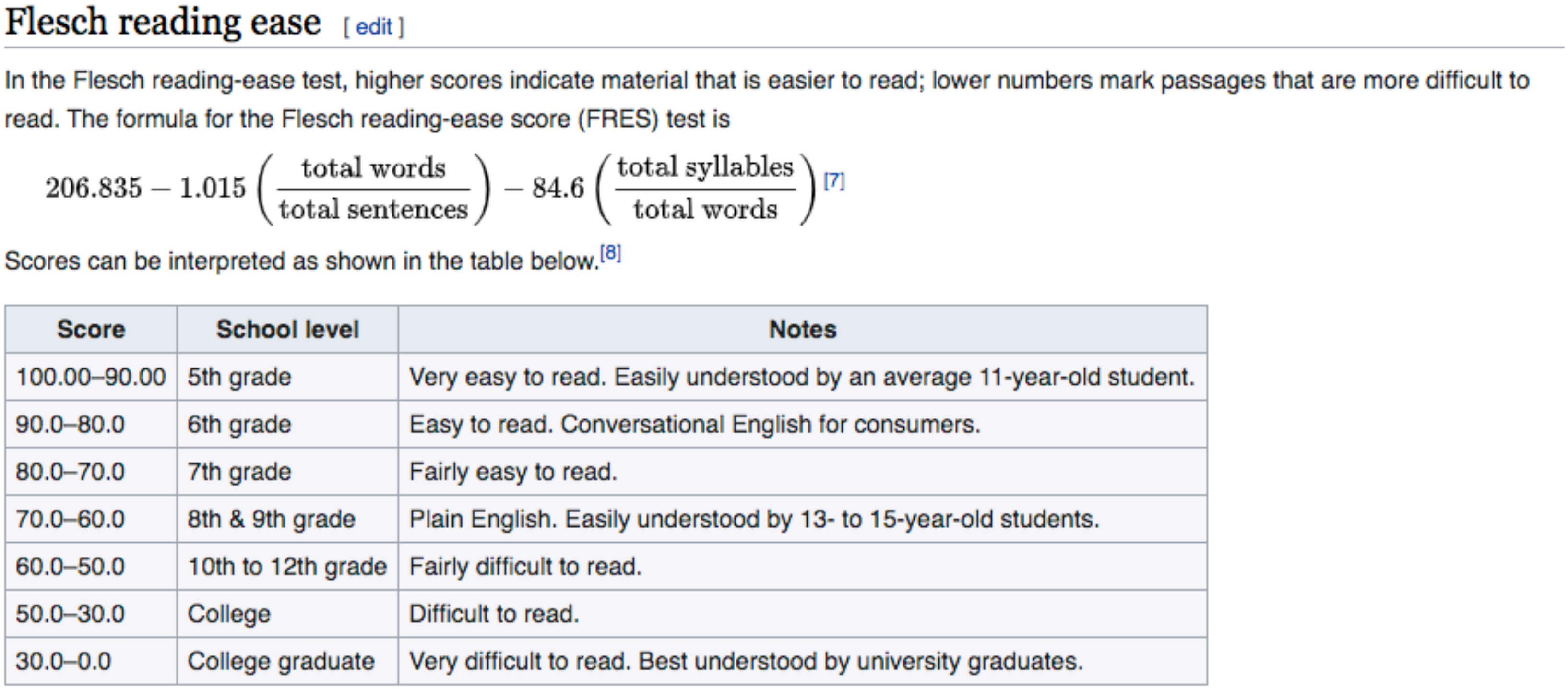


In [5]:
```
# create list of words for each PoV character
# and list of labels corresponding to list entries
labels = list()
data = list()
for key, value in text_by_character.items():
    labels.append(key)
    data.append(' '.join(value))
```

In [6]:
```
for i,pov in enumerate(data):
    tokens = nltk.word_tokenize(pov)
    tokens = [word.lower() for word in tokens]
    n_tokens = len(tokens)
    types = len(set(tokens))
    print(labels[i],types / n_tokens)
```

```
BRAN 0.11946008814887366
ARYA 0.13661023862941057
PROLOGUE 0.2578490313961256
TYRION 0.12340635661698689
SANSA 0.13489392831016825
EDDARD 0.09735772679116396
JON 0.1098394495412844
CATELYN 0.103555867365778
DAENERYS 0.1012995599264478
```

Other Measures



Flesch reading ease [edit]

In the Flesch reading-ease test, higher scores indicate material that is easier to read; lower numbers mark passages that are more difficult to read. The formula for the Flesch reading-ease score (FRES) test is

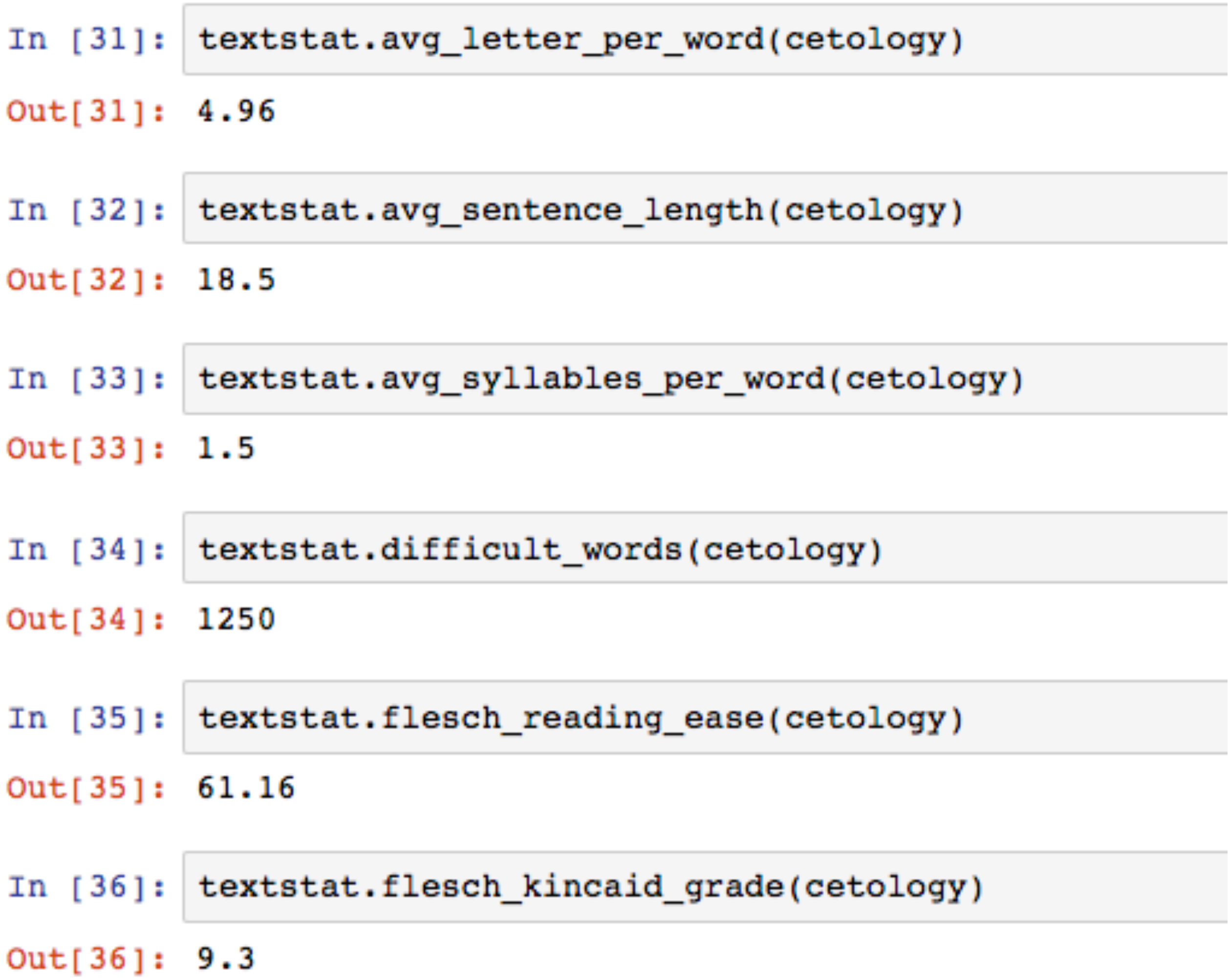
$$206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right)$$ [7]

Scores can be interpreted as shown in the table below.[8]

Score	School level	Notes
100.00–90.00	5th grade	Very easy to read. Easily understood by an average 11-year-old student.
90.0–80.0	6th grade	Easy to read. Conversational English for consumers.
80.0–70.0	7th grade	Fairly easy to read.
70.0–60.0	8th & 9th grade	Plain English. Easily understood by 13- to 15-year-old students.
60.0–50.0	10th to 12th grade	Fairly difficult to read.
50.0–30.0	College	Difficult to read.
30.0–0.0	College graduate	Very difficult to read. Best understood by university graduates.

(Thanks Wikipedia)

Other Measures: *Moby-Dick*



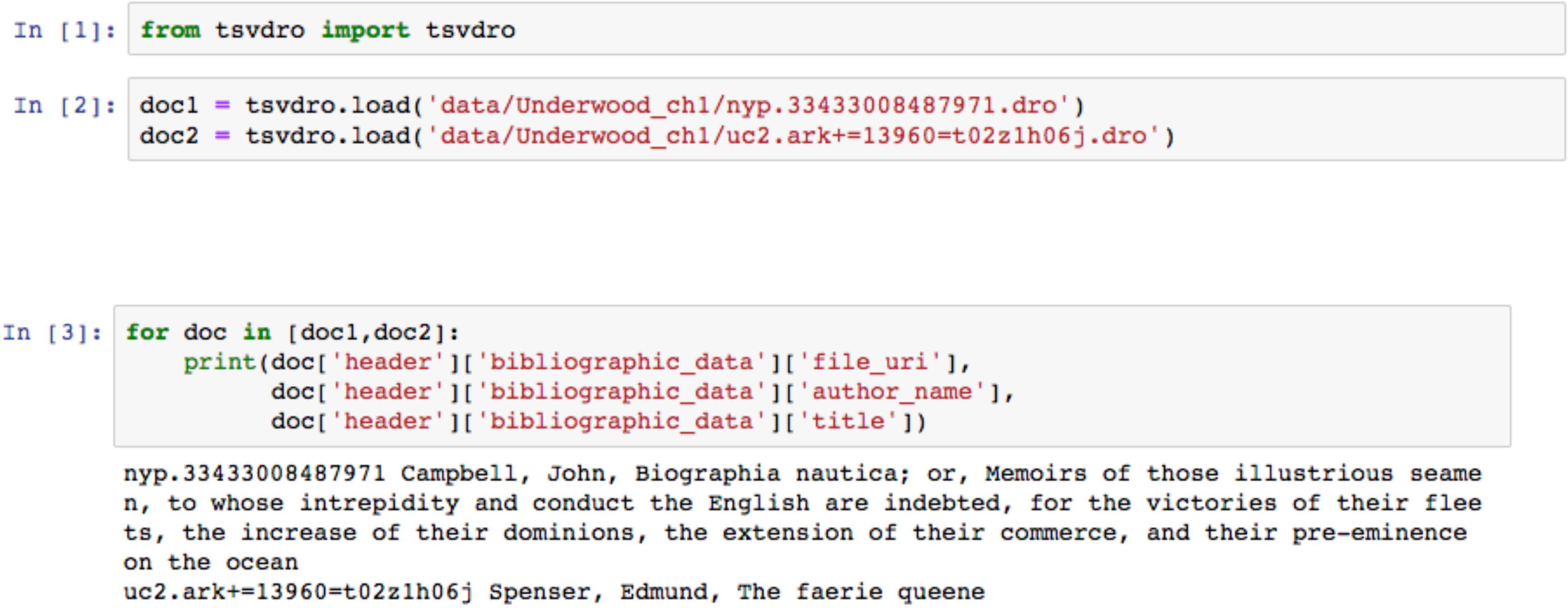
```
In [31]: textstat.avg_letter_per_word(cetology)

Out[31]: 4.96

In [32]: textstat.avg_sentence_length(cetology)

Out[32]: 18.5

In [33]: textstat.avg_syllables_per_word(cetology)

Out[33]: 1.5

In [34]: textstat.difficult_words(cetology)

Out[34]: 1250

In [35]: textstat.flesch_reading_ease(cetology)

Out[35]: 61.16

In [36]: textstat.flesch_kincaid_grade(cetology)

Out[36]: 9.3
```

Cultural Analytics

DRO: a Data Rich Object

- A simple file format for storing bibliographic information (metadata) together with type counts.
- A work-in-progress and might have some bugs.

Using DRO



```
In [1]: from tsvdro import tsvdro
```

```
In [2]: doc1 = tsvdro.load('data/Underwood_ch1/nyp.33433008487971.dro')
        doc2 = tsvdro.load('data/Underwood_ch1/uc2.ark+=13960=t02z1h06j.dro')
```

```
In [3]: for doc in [doc1,doc2]:
            print(doc['header']['bibliographic_data']['file_uri'],
                  doc['header']['bibliographic_data']['author_name'],
                  doc['header']['bibliographic_data']['title'])
```

```
nyp.33433008487971 Campbell, John, Biographia nautica; or, Memoirs of those illustrious seame
n, to whose intrepidity and conduct the English are indebted, for the victories of their flee
ts, the increase of their dominions, the extension of their commerce, and their pre-eminence
on the ocean
uc2.ark+=13960=t02z1h06j Spenser, Edmund, The faerie queene
```

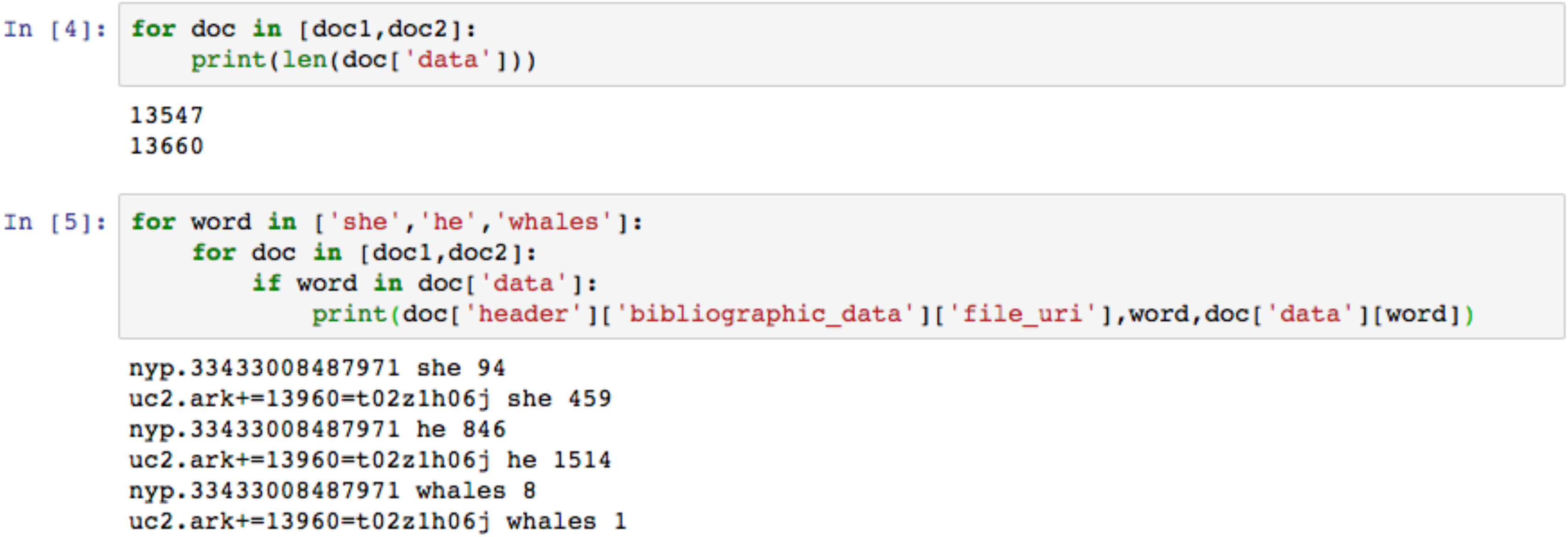
Cultural Analytics

```
In [7]: pprint.pprint(spenser['header'])
{'bibliographic_data': {'author_name': 'Spenser, Edmund,',
                        'file_uri': 'mdp.39015031001392',
                        'genre': 'poe',
                        'pages': None,
                        'publication_date': '1758',
                        'publisher': None,
                        'publisher_location': None,
                        'title': "Spenser's Faerie queene",
                        'volume': 'v.1',
                        'volumes': None},
 'tsvdro_ver': '1.0',
 'workflow': {'created_by': 'tsvdro_reference_implementation',
              'created_date': '2019-09-12 15:02',
              'created_system': 'jupyter-jed',
              'data_option': None,
              'data_type': 1,
              'last_updated': '2019-09-12 15:02',
              'token_count': None,
              'vocab_count': None}}
```

DARTMOUTH
VOX CLAMANTIS IN DESERTO
1769

Cultural Analytics

```
'soiled': '2',
'soit': '1',
'sojhall': '1',
'sojlreightly': '1',
'soke': '1',
'solace': '8',
'solaced': '1',
'solacing': '1',
'sold': '10',
'sole': '8',
'solemnise': '3',
'solemnly': '1',
'soles': '1',
'solitary': '2',
'some': '211',
'something': '1',
'sometime': '5',
'sometimes': '21',
'somewhat': '11',
'somewhere': '1',
'sommer': '1',
'somtimes': '2',
'son': '72',
'song': '19',
'songs': '1',
'sonnet': '1',
'sonnets': '1',
```



In [4]:
```
for doc in [doc1,doc2]:
    print(len(doc['data']))
```

```
13547
13660
```

In [5]:
```
for word in ['she','he','whales']:
    for doc in [doc1,doc2]:
        if word in doc['data']:
            print(doc['header']['bibliographic_data']['file_uri'],word,doc['data'][word])
```

```
nyp.33433008487971 she 94
uc2.ark+=13960=t02z1h06j she 459
nyp.33433008487971 he 846
uc2.ark+=13960=t02z1h06j he 1514
nyp.33433008487971 whales 8
uc2.ark+=13960=t02z1h06j whales 1
```

Next Class:

- Patrick Juola and Stephen Ramsay, "Probability and Statistics"