lab5

February 22, 2017

1 Data: Past, Present, Future | Lab 5 | 2/23/2017

2 Principal Component Analysis (PCA) & the pleasures (and hubris) of classification

2.0.1 DISCUSSION SUMMARYűű

2.0.2 (running time: ~50 min)

PART 1: A BLISTERINGLY SHORT INTRO TO Principal Component Analysis (PCA) [20 minutes total]

```
1. What it does? (7 min)
    - Historical & Material Context
    - Math explanation

2. PCA & measuring intelligence (13 min, student work)
```

PART 2: PCA and classification [30 minutes total]

```
1. PCA to classify texts, language, and people (7 min)
    - Words as Vectors in unique dimensions
    - Texts as summation of Word Vectors
    - 76 Novels example

2. Example: 4 Texts (13 min, student work, supplied data)
    - chunking and new insights...

3. Classifying People (10 min, student work, their data)
```

PART 3: Things to Try... (if there's time)

```
1. Epigraphs from 19th C novels
2. US State of the Union Addresses
```

2.0.3 Part 1: History of PCA and of Intelligence

1. Pearson, Galton, and Intelligence

Drawings of PCA in two-dims

NEED DATA SET FOR IQs. Preferably the original historical data set. PLOT PCA OF INTELLIGENCE

2.0.4 Part 2: Classifying texts, classifying people

1. drawings of PCA word vectors in 2 dims

76 novels example

In [1]:
```
# FUNCTIONS AND LIBRARIES
# YOU MUST EXECUTE THIS BLOCK TO USE THIS NOTEBOOK

%matplotlib inline
import text_analysis as ta #artisanally crafted, use with care
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
```

In [2]:
```
### to LOAD 1000 novel dataset generated in the "Prepare, Tokenize, and Count Words of C
wordcounts, total_word_counts, corpus_word_count, word_frequencies, wordlist, textnames
chunk_corpus_flag = 0
number_of_MFWs_used = 0
```

In [3]:
```
## Inspect most frequent words (MFWs) ------------------
text_index_to_compare_MFWs = 0 # column identifer for a particular text; full list in "t
MFW = ta.obtain_MFW(word_frequencies, text_index_to_compare_MFWs, textnames)
MFW.head(25) #list first 25 words for all texts
```

```
MFW list relative to Thackeray_Esmond_1852
Type 'MFW.head(X)' to list the first X most frequent words.
```

Out[3]:

	0	1	2	3	4	5	\
,	0.001192	0.002155	0.000353	0.000768	0.000603	2.928994e-04	
the	0.000742	0.001091	0.000226	0.000720	0.000382	2.646084e-04	
and	0.000600	0.000927	0.000134	0.000303	0.000238	1.206014e-04	
.	0.000419	0.001159	0.000347	0.000587	0.000285	2.198386e-04	
of	0.000401	0.000621	0.000097	0.000461	0.000153	1.257783e-04	
to	0.000354	0.000735	0.000121	0.000368	0.000126	7.685238e-05	
a	0.000305	0.000557	0.000091	0.000311	0.000138	1.269450e-04	
his	0.000242	0.000298	0.000031	0.000152	0.000075	5.906112e-05	
in	0.000215	0.000457	0.000059	0.000153	0.000090	6.715469e-05	
was	0.000210	0.000244	0.000073	0.000141	0.000065	5.322793e-05	
he	0.000198	0.000340	0.000074	0.000181	0.000090	8.202934e-05	
that	0.000177	0.000368	0.000069	0.000093	0.000046	6.343602e-05	
;	0.000161	0.000101	0.000029	0.000123	0.000092	2.442651e-05	

``	0.000150	0.000558	0.000149	0.000007	0.000146	8.822711e-05
her	0.000147	0.000195	0.000064	0.000195	0.000008	6.562347e-07
''	0.000147	0.000573	0.000151	0.000011	0.000146	1.052892e-04
's	0.000138	0.000167	0.000029	0.000113	0.000000	1.553089e-05
as	0.000134	0.000221	0.000030	0.000075	0.000032	3.623874e-05
had	0.000132	0.000154	0.000045	0.000090	0.000046	3.128052e-05
i	0.000132	0.000687	0.000100	0.000080	0.000086	6.416517e-05
with	0.000128	0.000247	0.000033	0.000070	0.000048	4.090530e-05
my	0.000126	0.000268	0.000012	0.000041	0.000034	1.137474e-05
for	0.000109	0.000177	0.000034	0.000119	0.000048	2.311405e-05
him	0.000107	0.000152	0.000020	0.000087	0.000033	2.034328e-05
at	0.000105	0.000182	0.000033	0.000081	0.000031	2.508275e-05

	6	7	8	9	...	66	67	\
,	0.000295	0.000714	0.000797	0.000906	...	0.001244	0.000309	
the	0.000194	0.000517	0.000542	0.000528	...	0.000594	0.000225	
and	0.000185	0.000329	0.000360	0.000427	...	0.000516	0.000116	
.	0.000201	0.000492	0.000581	0.000339	...	0.000557	0.000231	
of	0.000077	0.000255	0.000301	0.000373	...	0.000323	0.000072	
to	0.000115	0.000314	0.000252	0.000287	...	0.000370	0.000082	
a	0.000095	0.000233	0.000244	0.000275	...	0.000295	0.000102	
his	0.000093	0.000109	0.000111	0.000075	...	0.000135	0.000041	
in	0.000057	0.000175	0.000172	0.000193	...	0.000221	0.000051	
was	0.000077	0.000142	0.000145	0.000109	...	0.000207	0.000043	
he	0.000143	0.000189	0.000149	0.000103	...	0.000164	0.000064	
that	0.000053	0.000139	0.000118	0.000127	...	0.000224	0.000040	
;	0.000032	0.000062	0.000054	0.000043	...	0.000092	0.000027	
``	0.000092	0.000227	0.000208	0.000195	...	0.000277	0.000110	
her	0.000030	0.000128	0.000212	0.000088	...	0.000085	0.000017	
''	0.000096	0.000233	0.000259	0.000198	...	0.000293	0.000112	
's	0.000036	0.000064	0.000077	0.000044	...	0.000080	0.000055	
as	0.000031	0.000104	0.000078	0.000122	...	0.000129	0.000025	
had	0.000055	0.000112	0.000115	0.000064	...	0.000153	0.000017	
i	0.000059	0.000227	0.000134	0.000405	...	0.000483	0.000050	
with	0.000032	0.000072	0.000104	0.000115	...	0.000128	0.000033	
my	0.000012	0.000041	0.000023	0.000142	...	0.000151	0.000009	
for	0.000022	0.000076	0.000063	0.000097	...	0.000101	0.000023	
him	0.000044	0.000069	0.000071	0.000036	...	0.000084	0.000020	
at	0.000031	0.000076	0.000072	0.000073	...	0.000119	0.000022	

	68	69	70	71	72	73	74	\
,	0.000531	0.000527	0.001020	0.000905	0.000367	0.001240	0.000851	
the	0.000359	0.000406	0.000788	0.000597	0.000280	0.001089	0.000561	
and	0.000207	0.000167	0.000404	0.000395	0.000130	0.000520	0.000720	
.	0.000255	0.000517	0.000544	0.000703	0.000311	0.000822	0.000261	
of	0.000182	0.000244	0.000418	0.000312	0.000136	0.000527	0.000268	
to	0.000145	0.000334	0.000303	0.000384	0.000129	0.000607	0.000250	
a	0.000138	0.000157	0.000318	0.000273	0.000124	0.000273	0.000142	

his	0.000044	0.000063	0.000138	0.000207	0.000040	0.000140	0.000039
in	0.000090	0.000151	0.000211	0.000161	0.000072	0.000361	0.000110
was	0.000070	0.000120	0.000122	0.000164	0.000067	0.000236	0.000130
he	0.000052	0.000102	0.000120	0.000255	0.000078	0.000158	0.000077
that	0.000079	0.000121	0.000157	0.000149	0.000074	0.000226	0.000166
;	0.000039	0.000062	0.000018	0.000062	0.000019	0.000029	0.000179
``	0.000094	0.000162	0.000007	0.000252	0.000001	0.000228	0.000000
her	0.000011	0.000113	0.000023	0.000154	0.000025	0.000240	0.000238
''	0.000092	0.000167	0.000013	0.000261	0.000017	0.000229	0.000000
's	0.000027	0.000054	0.000055	0.000129	0.000026	0.000123	0.000021
as	0.000049	0.000058	0.000119	0.000081	0.000030	0.000150	0.000098
had	0.000040	0.000094	0.000093	0.000096	0.000044	0.000175	0.000067
i	0.000115	0.000349	0.000218	0.000193	0.000082	0.000537	0.000154
with	0.000049	0.000057	0.000128	0.000079	0.000037	0.000138	0.000070
my	0.000038	0.000143	0.000109	0.000056	0.000019	0.000248	0.000049
for	0.000050	0.000059	0.000106	0.000100	0.000030	0.000127	0.000107
him	0.000022	0.000047	0.000054	0.000125	0.000021	0.000081	0.000043
at	0.000038	0.000056	0.000084	0.000072	0.000035	0.000154	0.000043

	75
,	0.000156
the	0.000122
and	0.000054
.	0.000127
of	0.000078
to	0.000082
a	0.000060
his	0.000042
in	0.000040
was	0.000038
he	0.000066
that	0.000060
;	0.000004
``	0.000055
her	0.000029
''	0.000061
's	0.000030
as	0.000020
had	0.000035
i	0.000027
with	0.000020
my	0.000010
for	0.000023
him	0.000015
at	0.000016

[25 rows x 76 columns]

In [4]:
```
## Perform PCA on Word Frequencies

    if chunk_corpus_flag == 0:
        ## Perform PCA on Word Frequencies
        text_index_to_compare_MFWs = 0 # column identifer for a particular text; full list i
        number_of_components = 2 # how many dimensions for PCA
        pca_coordinates, pca_results = ta.PCAnalysis(word_frequencies, number_of_MFWs_used,
```

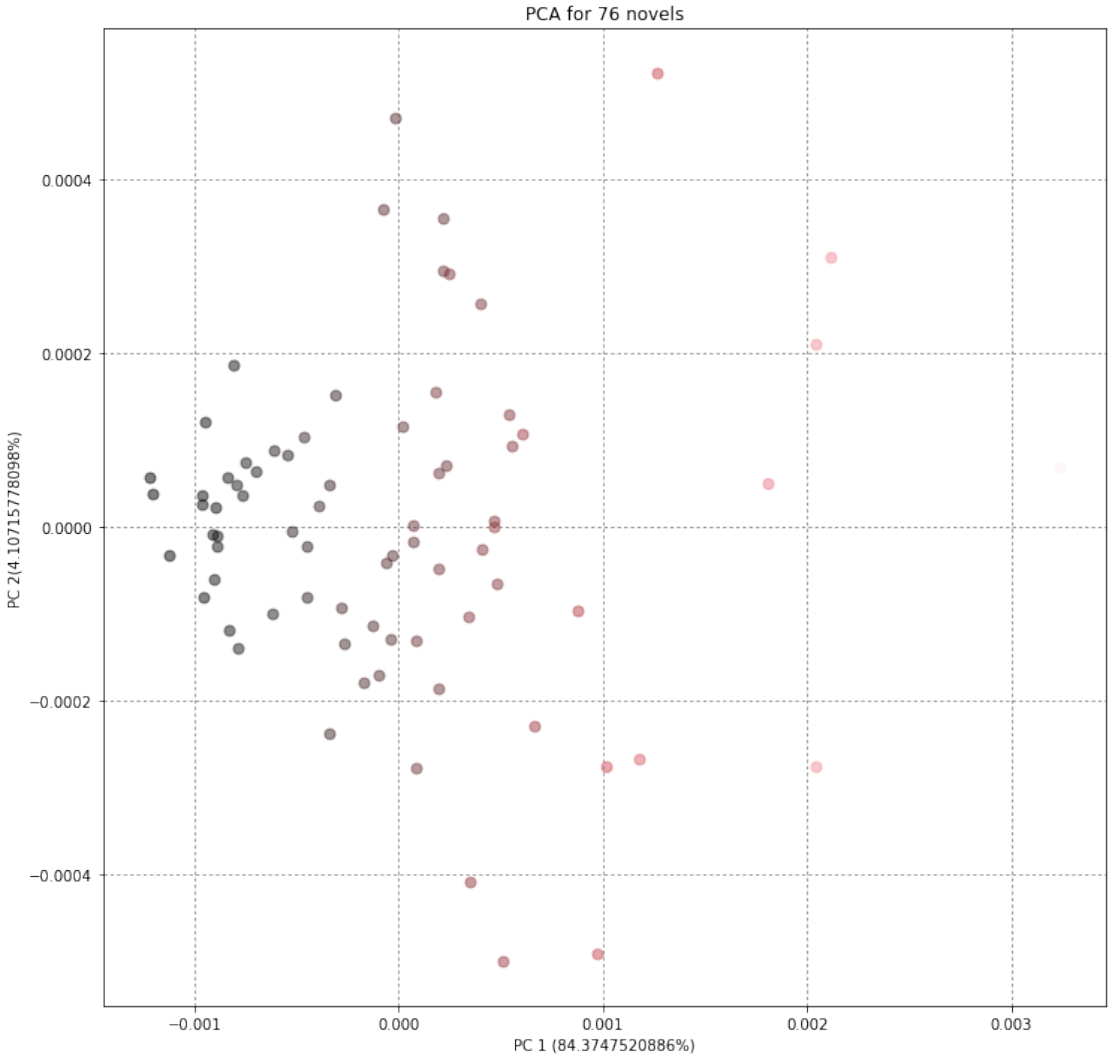
```
Corpus Word Count:13714605
Using corpus word count (13714605 words) for PCA in 2-dimensions...
Time to execute PCA: 8.221241
```

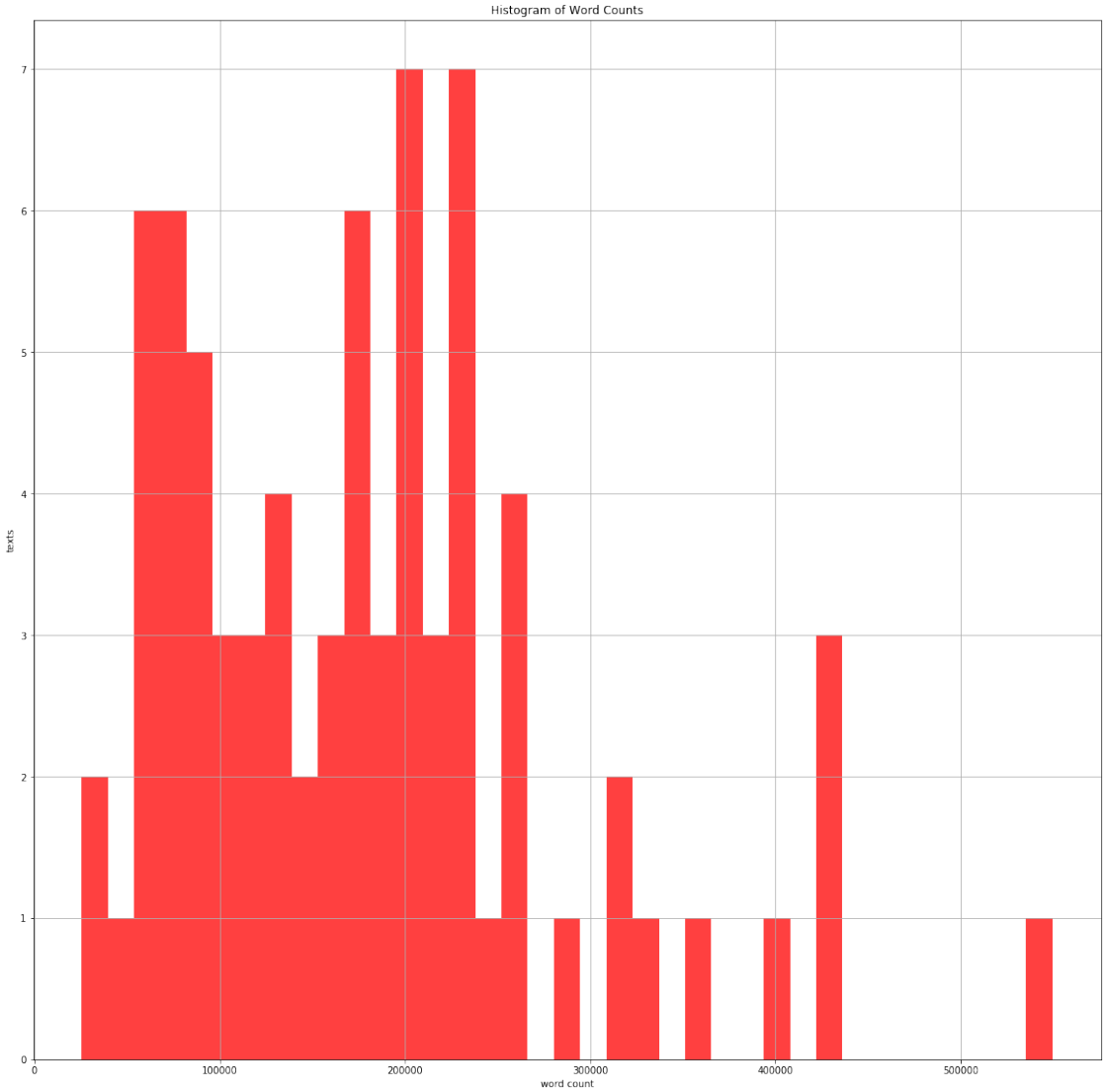
In [6]:
```
if chunk_corpus_flag == 0:

    ## generate color spectrum based on word count of texts ----------
        total_word_counts_nparray = np.array(total_word_counts) # convert word count list in
        colors_for_texts = ta.assign_text_colors_via_word_counts(total_word_counts_nparray,

    ##plot PCA without legend or tables -------
        size_of_plot = 20
        name_of_file = "pca-"+ str(number_of_MFWs_used) +"-Full-wc_color_coded"
        ta.plot_PCA(pca_coordinates, pca_results, colors_for_texts, textnames, size_of_plot,

    #### For reference, also generate historgram
        ta.histogram_of_word_counts(total_word_counts_nparray, 0, "histo-"+str(number_of_MFW
```


PCA for 76 novels
0.0004
0.0002
0.0000
−0.0002
−0.0004
PC 2(4.107157778098%)
−0.001
0.000
0.001
0.002
0.003
PC 1 (84.3747520886%)



2. Classifying Language and Imagining Relationships (text chunking) 4 novels example

```
In [7]: ## PCA Data Parameters --------------------------
        data_location = "./dat/four_novels/"
        chunk_corpus_flag = 1      # 0 = don't chunk texts; 1 = do chunk texts
        number_of_MFWs_used = 0  #number of MFWs used to plot PCA graphs; 0 = use all words

        ## Chunking Parameters ----------------------
        chunk_size_used = 5000     # number of words per chunk (but will give >chunk_size final u

In [8]: ## Prepare, Tokenize, and Count Words of Corpus ----------------------

        if (chunk_corpus_flag == 1):
```

```
        ## read & tokenize all txts in data_location directory; translate txt files into lis
        wordlist, textnames = ta.tokenized_texts_and_textname_list(ta.filenames_of_txts_in_d

        ## CHUNK TEXTS
        ## chunk_wordlist = same as wordlist, but each element is now a "text chunk" instead
        ## chunk_index = provides start and end elements for each text in chunk_wordlist
        chunk_wordlist, chunk_index = ta.chunk_all_texts(wordlist, textnames, chunk_size_use

        ## GET WORD COUNTS FOR ALL CHUNKS;
        ## WHERE chunk_word_counts IS A LIST OF INDIVIDUAL WORD COUNTS PER CHUNK
        ## AND chunk_total_word_counts IS A LIST of TOTAL WORD COUNTS PER CHUNK
        chunk_word_counts, chunk_total_word_counts = ta.type_counts_and_total_token_counts(c
        corpus_word_count_for_chunks = ta.total_number_of_words_in_corpus(chunk_total_word_c

        ## CALCULATE CHUNK WORD FREQUENCIES (RELATIVE TO ENTIRE CORPUS)
        chunk_word_frequencies = ta.word_freq(chunk_word_counts, corpus_word_count_for_chunk

        print("corpus word count: " + str(corpus_word_count_for_chunks))
```

```
Examining 4 texts...
Time to tokenize texts: 4.938937000000003
Time to chunk texts: 0.00778800000000146
Time to count words: 27.164153
Time to compute frequencies: 0.03459399999999846
corpus word count: 831919
```

In [9]:
```
## save or export results for chunked texts

## to export, uncomment line below
## WARNING: THIS WILL DELETE EXISTING SAVED RESULTS
#ta.export_chunked_text_analysis('four_novels', chunk_word_counts, chunk_total_word_coun

## to import, uncomment line below
chunk_corpus_flag = 1
number_of_texts = 4  #change to correct value if necessary
chunk_size_used = 5000  #change to correct value if necessary
chunk_word_counts, chunk_total_word_counts, corpus_word_count_for_chunks, chunk_word_fre
```

In [10]:
```
if (chunk_corpus_flag == 1):
    number_of_PCs = 2
    text_index_compare_MFWs = 0
    number_of_MFWs_used = 0  #PCAnalysis will use full corpus of words when examining c
    pca_coordinates_for_chunks, pca_results_for_chunks = ta.PCAnalysis(chunk_word_frequ
```

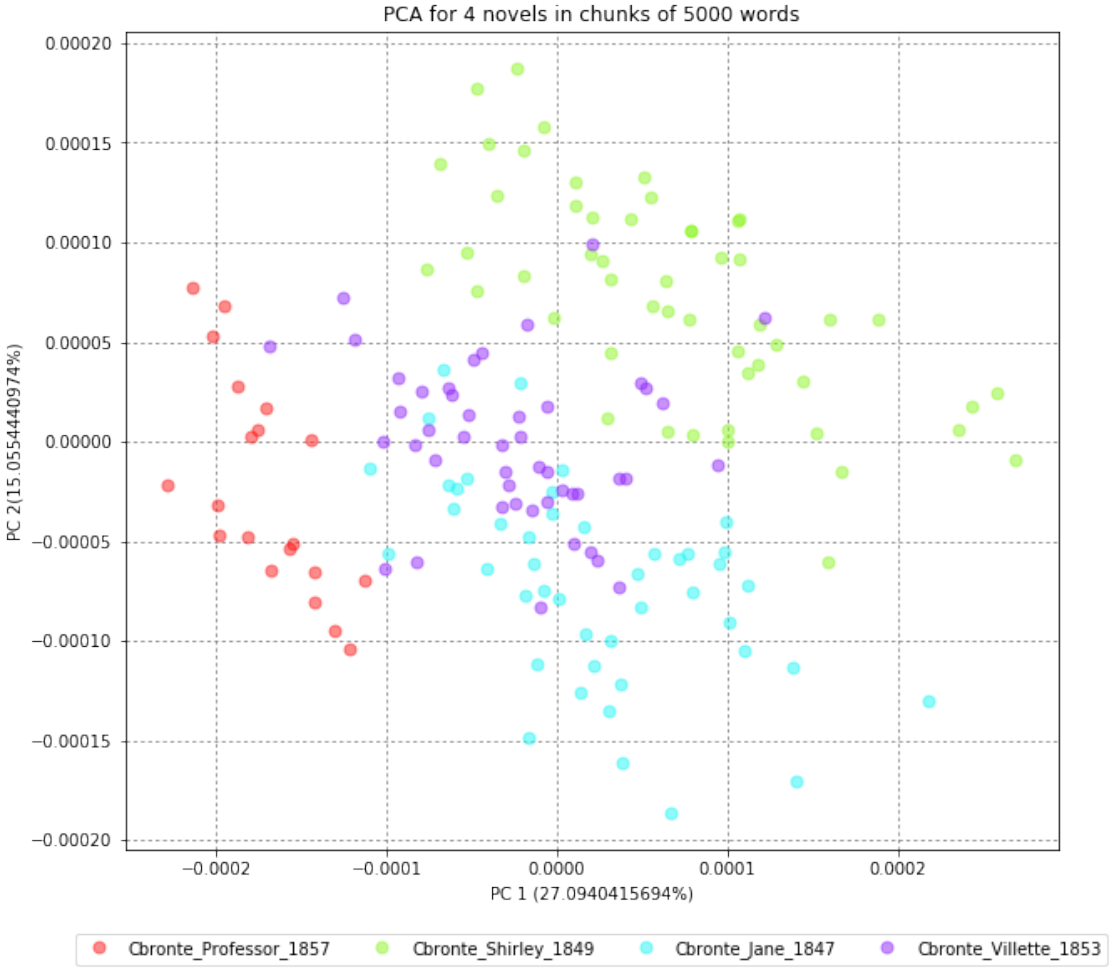
```
Corpus Word Count:831919
Using corpus word count (831919 words) for PCA in 2-dimensions...
Time to execute PCA: 3.3021499999999975
```

In [11]:
```
## color scheme for chunk graph
## this code will randomly select colors for each "text"
number_of_texts = 4 #change based on how many documents you have
colors_for_texts = ta.generate_colors(number_of_texts)
colors_for_texts
```

Out[11]:
```
[(0.9896353773773667, 0.09366810913369128, 0.09366810913369128),
 (0.55291067081843703, 0.9655912853414365, 0.1402300562954374),
 (0.12808983657476358, 0.96573217056726846, 0.9657321705672686),
 (0.56582334911085175, 0.14813889358152754, 0.9835078046401766)]
```

In [13]:
```
# You may need to run this code twice to get proper display of graph

output_flag = 1
plot_name = 0 #reverts to default name; to supply alternate name, provide a string here
plot_size = 10.0
ta.plot_PCA_chunked_with_legend(pca_coordinates_for_chunks, pca_results_for_chunks, chu
```



EXPERIMENTS WITH STUDENT DATA!

In [ ]: