Text Analysis & Data Visualization

Isaiah Hull

Sveriges Riksbank

November 4, 2017

References

1. “Text Analytics with Python: A Practical Real-World Approach to Gaining Actionable Insights from Your Data”
    - Dipanjan Sarkar (2016)

Overview

Text Analysis & Data Visualization

1. Processing and understanding text
2. Text classification
3. Matplotlib
4. Bokeh
   - Sentiment analysis
   - Topic models

Processing and understanding text

Installation

```
$user pip install nltk
$user python
>>> import nltk
>>> nltk.download(`all')
```

Processing and understanding text

Introduction

- Text data is unstructured
- Must convert into linguistic components before incorporating into algorithms
  - Clean
  - Normalize
  - Pre-process

Processing and understanding text

Introduction

- Common pre-processing methods
  - Tokenization
  - Tagging
  - Chunking
  - Stemming
  - Lemmatization

Processing and understanding text

Tokenization

```
#lecture3_example1.py

$user python
>>> import nltk
>>> nltk.corpus.gutenberg.fileids()
```

Processing and understanding text

Tokenization

['austen-emma.txt',
'austen-persuasion.txt',
'austen-sense.txt',
'bible-kjv.txt',
'blake-poems.txt',
...
'milton-paradise.txt',
'shakespeare-caesar.txt',
'shakespeare-hamlet.txt',
'shakespeare-macbeth.txt',
'whitman-leaves.txt']

Tokenization

```
#lecture3_example1.py (continued)

>>> from nltk.corpus import gutenberg
>>> paradise = gutenberg.\
raw(fileids=`milton-paradise.txt')
>>> print len(paradise)
468220
>>> print paradise[:35]
[Paradise Lost by John Milton 1667]
>>> sentenceTokenizer = nltk.sent_tokenize
>>> paradiseSentences =
sentenceTokenizer(text=paradise)
>>> print paradiseSentences[500]
```

Processing and understanding text

Tokenization

```
The birds their quire apply; airs, vernal airs,
Breathing the smell of field and grove, attune
The trembling leaves, while universal Pan,
Knit with the Graces and the Hours in dance,
Led on the eternal Spring.
```

Tokenization

```
#lecture3_example1.py (continued)

>>> wordTokenizer = nltk.word_tokenize
>>> paradiseWords = wordTokenizer(text=paradise)
>>> print len(paradiseWords)
95709
>>> print paradiseWords[150:160]
```

Tokenization

```
[u'or',
u'rhyme',
u'.',
u'And',
u'chiefly',
u'thou',
u',',
u'O',
u'Spirit',
u',']
```

Tokenization

```
#lecture3_example1.py

>>> fdist = nltk.FreqDist(w.lower() for w in
paradiseWords)
>>> wlist = [`horse', `automobile']
>>> for i in wlist:
        print(i + `:', fdist[i])
(`horse:', 3)
(`automobile:', 0)
```

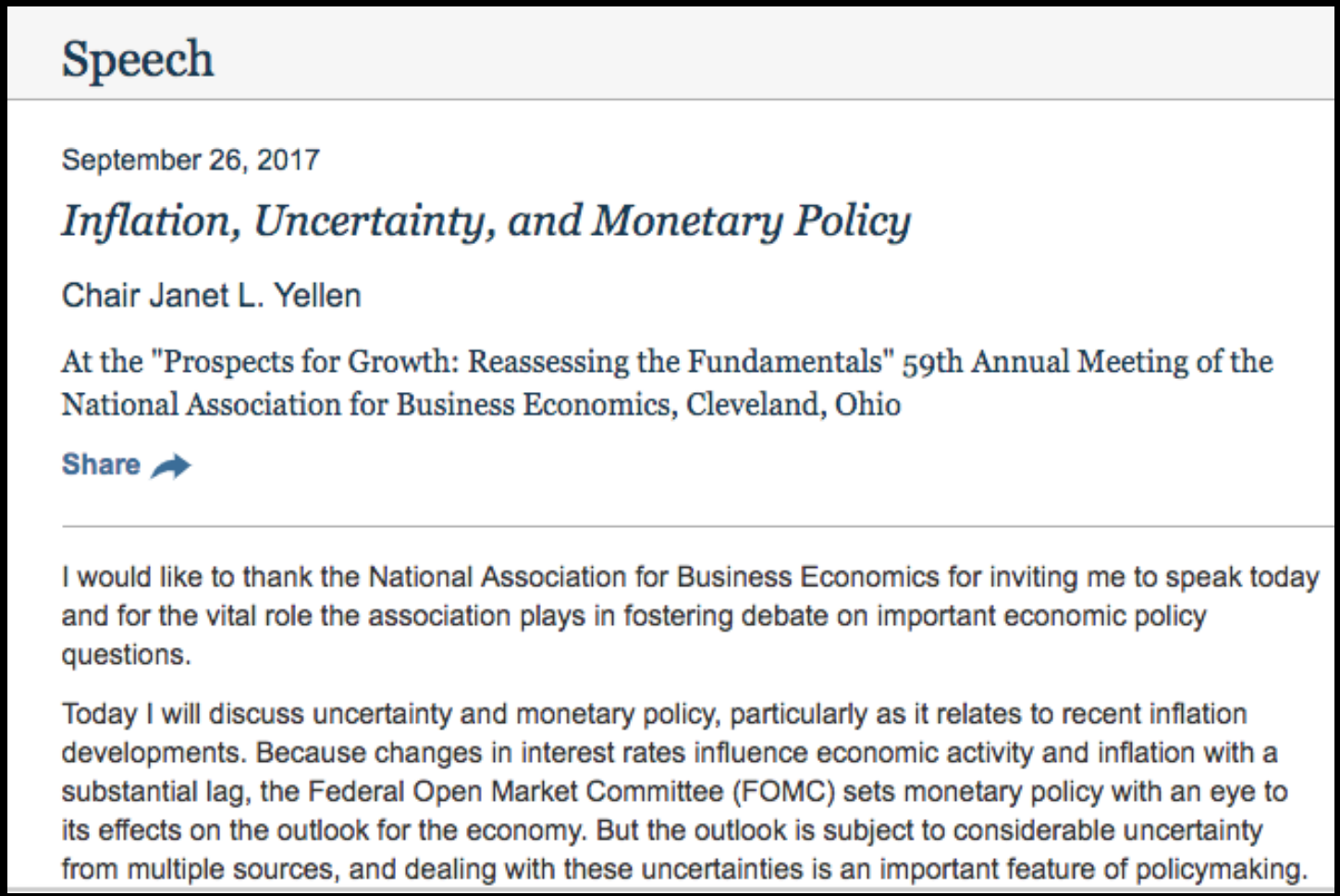
Processing and understanding text

Cleaning Raw Text

- What about unstructured text?
- Raw text is typically not suitable for use in text analysis.
- We must pre-process raw text before performing text analysis.

Processing and understanding text

Cleaning Raw Text



Speech

September 26, 2017

Inflation, Uncertainty, and Monetary Policy

Chair Janet L. Yellen

At the "Prospects for Growth: Reassessing the Fundamentals" 59th Annual Meeting of the National Association for Business Economics, Cleveland, Ohio

Share

I would like to thank the National Association for Business Economics for inviting me to speak today and for the vital role the association plays in fostering debate on important economic policy questions.

Today I will discuss uncertainty and monetary policy, particularly as it relates to recent inflation developments. Because changes in interest rates influence economic activity and inflation with a substantial lag, the Federal Open Market Committee (FOMC) sets monetary policy with an eye to its effects on the outlook for the economy. But the outlook is subject to considerable uncertainty from multiple sources, and dealing with these uncertainties is an important feature of policymaking.

Processing and understanding text

Cleaning Raw Text

Model-Based Decomposition of ECI Hourly Compensation Growth

	ECI growth	*Contributions of:* Expected inflation	Trend productivity	Slack	Other
2002-07	3.28	2.09	1.44	-0.08	-0.17
2008-09	1.80	2.16	1.18	-0.97	-0.56
2010-11	2.08	2.14	0.82	-1.09	0.22
2012-13	1.92	2.12	0.44	-0.67	0.02
2014-15	2.05	2.01	0.15	-0.20	0.09
2016-17:Q2	2.31	1.99	0.01	0.03	0.28

Note: ECI growth is reported as average percent changes at an annual rate for the periods shown; contributions are expressed in percentage points. The contribution of the model's constant term is included in the contribution for trend productivity. Contributions may not sum to total growth because of rounding.

References

Aaronson, Daniel, Luojia Hu, Arian Seifoddini, and Daniel G. Sullivan (2015). "Changing Labor Force Composition and the Natural Rate of Unemployment," Chicago Fed Letter 338. Chicago: Federal Reserve Bank of Chicago, May.

Aaronson, Stephanie, Tomaz Cajner, Bruce Fallick, Felix Galbis-Reig, Christopher Smith, and William Wascher (2014). "Labor Force Participation: Recent Developments and Future Prospects," *Brookings Papers on Economic Activity,* Fall, pp. 197-255.

Cleaning Raw Text

```
#lecture3_example2.py

$user python
>>> from urllib2 import urlopen
>>> import nltk
>>> from bs4 import BeautifulSoup
>>> url = "https://www.federalreserve.gov/
newsevents/speech/yellen20170926a.htm"
>>> html = urlopen(url)
>>> soup = BeautifulSoup(html.read())
```

Cleaning Raw Text

```
#lecture3_example2.py (continued)

>>> paragraphs = soup.findAll(`p')
>>> paragraphs = [p.text for p in paragraphs]
```

Cleaning Raw Text

```
#lecture3_example2.py (continued)

>>> len(paragraphs)
154
>>> speech = ` ´.join(paragraphs)
>>> print speech.split(`References´)[1][0:50]
Aaronson, Daniel, Luojia Hu, Arian Seifoddini, an
>>> speech = speech.split(`References´)[0]
```

Cleaning Raw Text

```
#lecture3_example2.py (continued)

>>> wordTokenizer = nltk.word_tokenize
>>> wordTokens = wordTokenizer(speech)
>>> fdist = nltk.FreqDist(wordTokens)
>>> speechLength = len(wordTokens)
>>> print fdist[`inflation']
151
>>> print fdist[`Inflation']
19
```

Cleaning Raw Text

```
#lecture3_example2.py (continued)

>>> print speechLength
10378
>>> fdist = nltk.FreqDist(w.lower() for w in
wordTokens)
>>> inflationCount = fdist[`inflation']
>>> inflationIntensity =
100.0*inflationCount/speechLength
>>> print inflationIntensity
1.63808055502
```

Processing and understanding text

Cleaning Raw Text

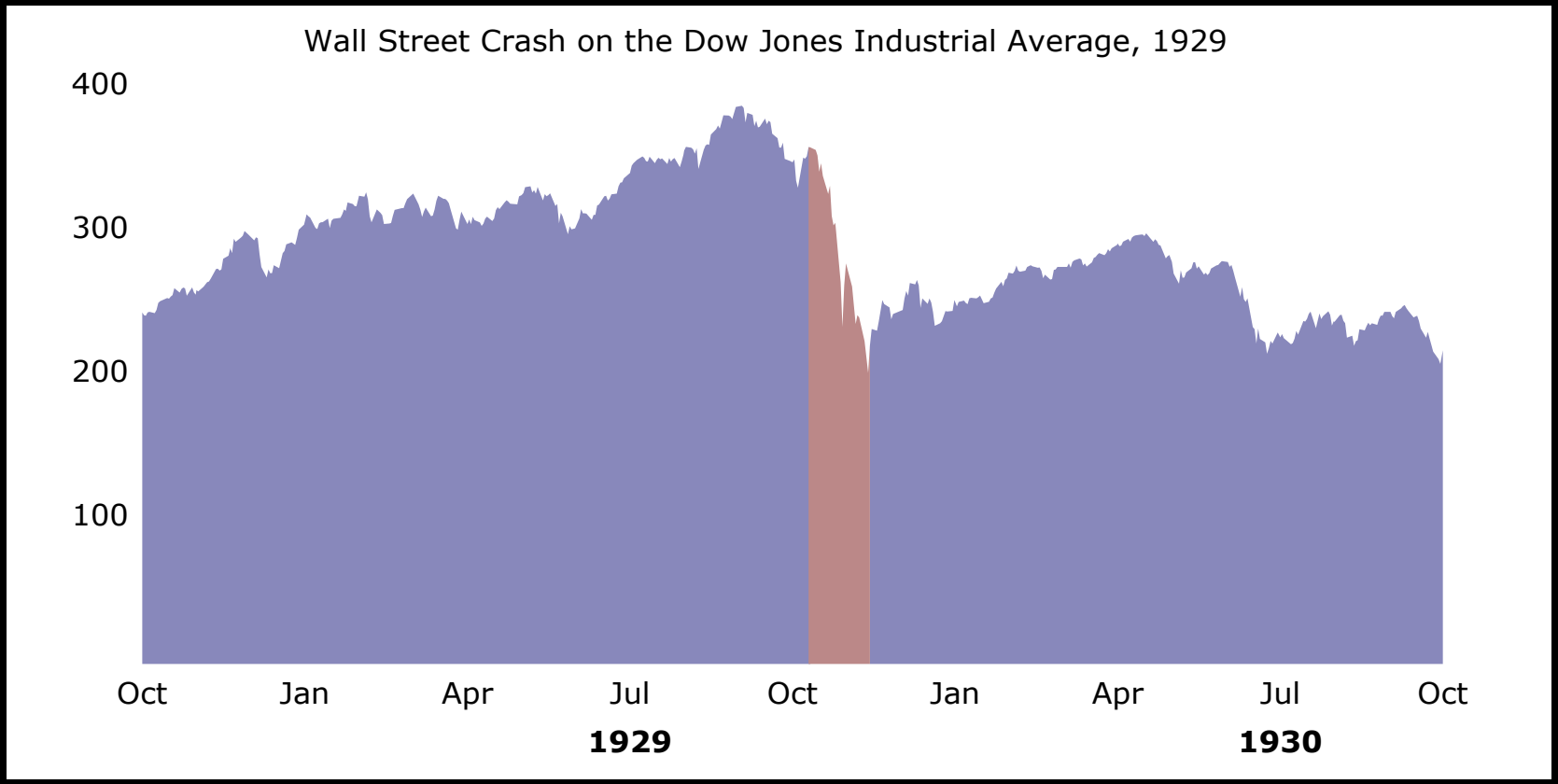
- Many problems remain:
  - Stop words → remove words without meaningful content
  - Special characters → remove symbols and punctuation
  - Stemming → retain word stem only

Cleaning Raw Text

```
#lecture3_example2.py (continued)

>>> from nltk.corpus import stopwords
>>> stops = set(stopwords.words(`english'))
>>> wordTokens = [word for word in wordTokens if
word not in stops]
>>> fdist = nltk.FreqDist(w.lower() for w in
wordTokens)
>>> print 100.0*fdist[`inflation']/len(wordTokens)
2.996254681165
```

Processing and understanding text

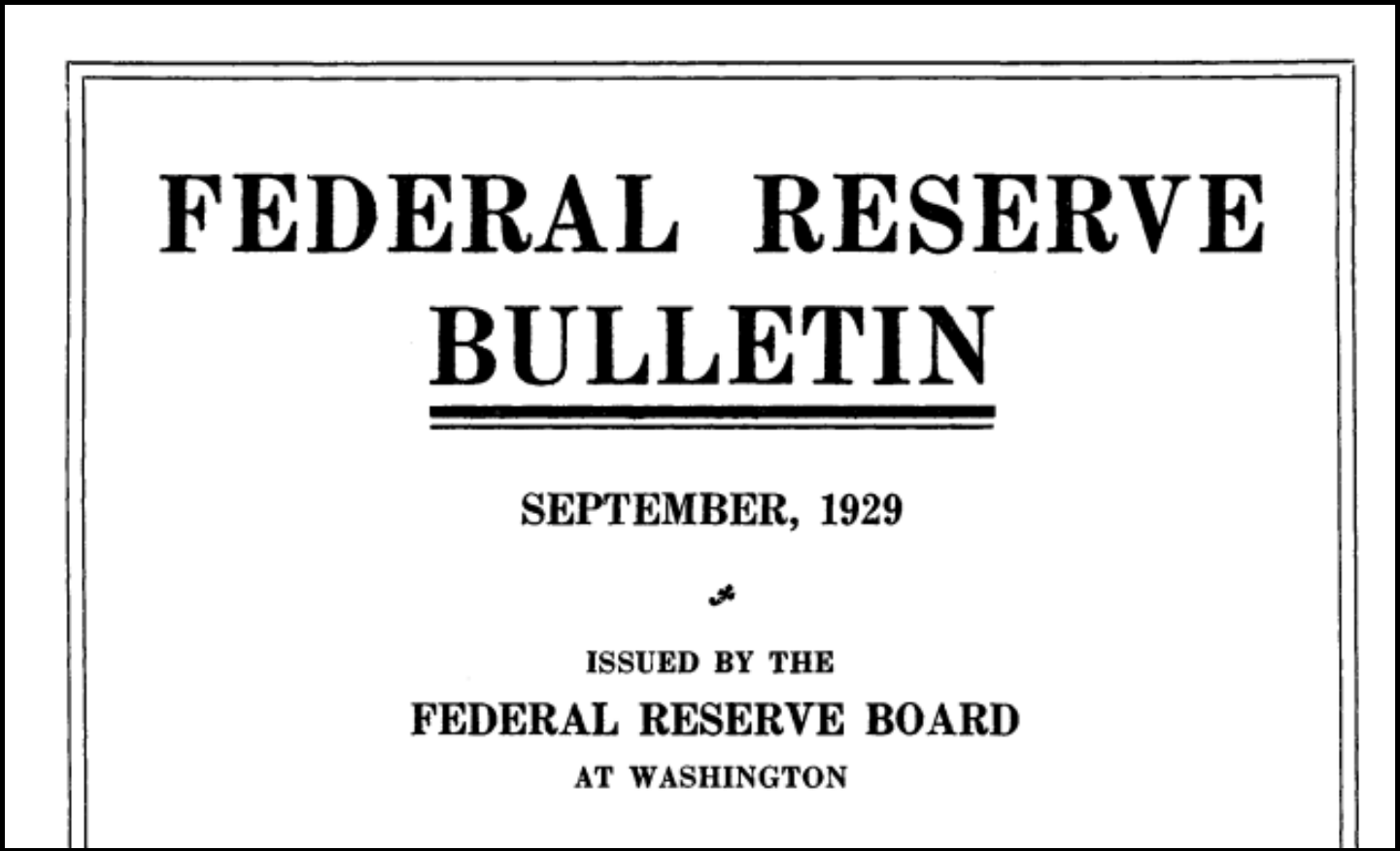
Cleaning Raw Text



https://en.wikipedia.org/wiki/Wall_Street_Crash_of_1929

Processing and understanding text

Cleaning Raw Text



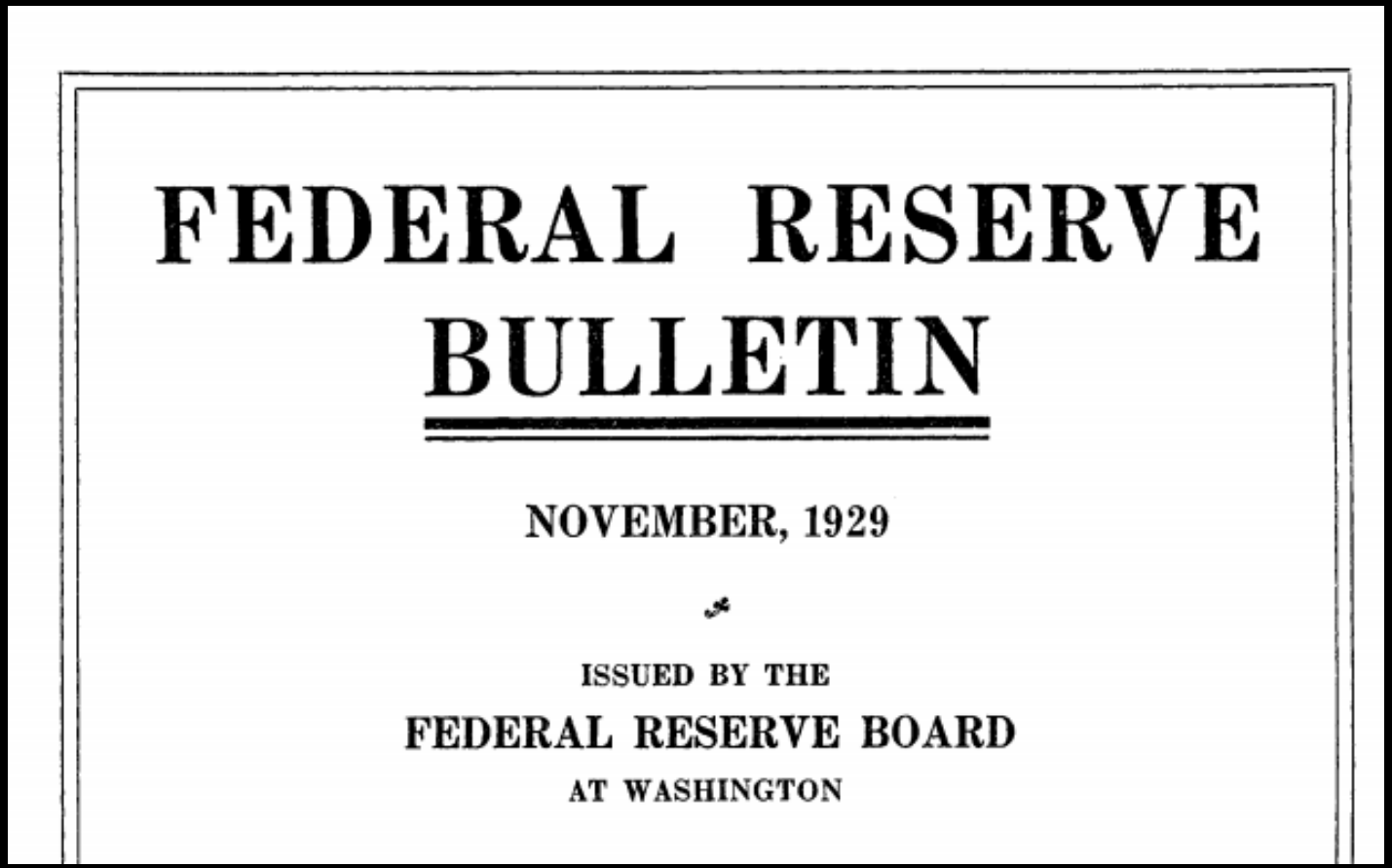
FEDERAL RESERVE
BULLETIN

SEPTEMBER, 1929

ISSUED BY THE
FEDERAL RESERVE BOARD
AT WASHINGTON

Processing and understanding text

Cleaning Raw Text



FEDERAL RESERVE
BULLETIN

NOVEMBER, 1929

ISSUED BY THE
FEDERAL RESERVE BOARD
AT WASHINGTON

Cleaning Raw Text

```
#lecture3_example3.py

$user pip install PyPDF2
$user python
>>> from urllib2 import urlopen
>>> import nltk
>>> from nltk.corpus import stopwords
>>> from nltk.stem.porter import PorterStemmer
>>> import PyPDF2
>>> import re
```

Cleaning Raw Text

```
#lecture3_example3.py (continued)

>>> save_dir = `../Downloads/'
>>> url_09_29 =
“https://fraser.stlouisfed.org/files/
docs/publications/FRB/1920s/frb_091929.pdf”
>>> url_11_29 =
“https://fraser.stlouisfed.org/files/
docs/publications/FRB/1920s/frb_111929.pdf”
>>> content_09_29 = urlopen(url_09_29).read()
>>> content_11_29 = urlopen(url_11_29).read()
```

Cleaning Raw Text

```
#lecture3_example3.py (continued)
>>> with open(save_dir+`frb_0929.pdf', `wb') as f:
        f.write(content_09_29)
>>> with open(save_dir+`frb_1129.pdf', `wb') as f:
        f.write(content_11_29)
>>> pdf_09_29 = PyPDF2.PdfFileReader(
open(save_dir+`frb_0929.pdf'), `rb')
>>> pdf_11_29 = PyPDF2.PdfFileReader(
open(save_dir+`rb_1129.pdf'), `rb')
```

Cleaning Raw Text

```
#lecture3_example3.py (continued)

>>> pdf_09_29_text = [pdf_09_29.getPage(j).
extractText() for j in range(pdf_09_29.numPages)]
>>> pdf_11_29_text = [pdf_11_29.getPage(j).
extractText() for j in range(pdf_11_29.numPages)]
>>> pdf_09_29_text = ` ´.join(pdf_09_29_text)
>>> pdf_11_29_text = ` ´.join(pdf_11_29_text)
```

Cleaning Raw Text

```
#lecture3_example3.py (continued)
>>> def tokenize(text):
        text = re.sub(`[^A-Za-z]+', ` ', text)
        wordTokens = nltk.word_tokenize(text)
        wordTokens = [token.lower() for token in
        wordTokens if len(token)>1]
        stops = set(stopwords.words("english"))
        wordTokens = [token for token in
        wordTokens if token not in stops]
        return wordTokens
>>> tokens_09_29 = tokenize(pdf_09_29_text)
>>> tokens_11_29 = tokenize(pdf_11_29_text)
```

Cleaning Raw Text

```
#lecture3_example3.py (continued)

>>> porter_stemmer = PorterStemmer()
>>> stems_09_29 = [porter_stemmer.stem(token) for
token in tokens_09_29]
>>> stems_11_29 = [porter_stemmer.stem(token) for
token in tokens_11_29]
>>> fdist_09_29 = nltk.FreqDist(stems_09_29)
>>> fdist_11_29 = nltk.FreqDist(stems_11_29)
```

Cleaning Raw Text

```
#lecture3_example3.py (continued)
>>> print fdist_09_29.most_common(5)
```

Cleaning Raw Text

1. (u'bank', 573)
2. (u'aug', 315)
3. (u'reserv', 258)
4. (u'feder', 161)
5. (u'juli', 130)]

Cleaning Raw Text

```
#lecture3_example3.py (continued)
>>> print fdist_11_29.most_common(5)
```

Cleaning Raw Text

1. (u'oct', 415)
2. (u'bank', 305)
3. (u'reserv', 184)
4. (u'feder', 149)
5. (u'index', 106)]

Cleaning Raw Text

```
#lecture3_example3.py (continued)

>>> more_stops = [`june',`juli',`aug',`sep',`sept',
`oct',`nov',`octob',`feder',`reserv',`nation',
`cent',`new', `year',`month',`per',`total']
>>> fdist_09_29 = nltk.FreqDist([stem for stem in
stems_09_29 if stem not in more_stops])
>>> fdist_11_29 = nltk.FreqDist([stem for stem in
stems_11_29 if stem not in more_stops])
```

Cleaning Raw Text

```
#lecture3_example3.py (continued)
>>> print fdist_09_29.most_common(5)
```

Cleaning Raw Text

1. (u'bank', 573)
2. (u'rate', 125)
3. (u'loan', 120)
4. (u'gold', 110)
5. (u'foreign', 107)

Cleaning Raw Text

```
#lecture3_example3.py (continued)
>>> print fdist_11_29.most_common(5)
```

Cleaning Raw Text

1. (u'bank', 305)
2. (u'index', 106)
3. (u'industri', 101)
4. (u'loan', 63)
5. (u'employ', 55)

Cleaning Raw Text

```
#lecture3_example4.py

$user python
>>> import nltk
>>> from nltk.corpus import twitter_samples
>>> from nltk.corpus import stopwords
>>> from nltk.stem.porter importer PorterStemmer
>>> from nltk import ngrams
>>> import re
>>> twitter_samples.fileids()
```

Processing Raw Text

```
[u`negative_tweets.json',
u`positive_tweets.json',
u`tweets.20150430-223406.json']
```

Processing Raw Text

```
#lecture3_example4.py (continued)

>>> def clean_text(tweet):
        tweet = tweet.strip()
        pattern = "(@[A-Za-z0-9]+)|(#[A-Za-z0-9]+)|
       ([^0-9A-Za-z \t])|(\w+:\/\/\S+)|(RT)"
        cleaned_tweet = ' '.join(re.sub(pattern,
        ' ',tweet).split())
        wordTokens = nltk.word_tokenize(cleaned_tweet)
        wordTokens = [token.lower() for token in
        wordTokens if len(token)>1]
        stops = set(stopwords.words("english"))
        wordTokens = [token for token in
        wordTokens if token not in stops]
        cleaned_tweet = ' '.join(wordTokens)
        return cleaned_tweet
```

Processing Raw Text

```
#lecture3_example4.py (continued)
>>> tweets = twitter_samples.\
string(twitter_samples.fileids()[-1])
>>> porter_stemmer = PorterStemmer()
>>> cleaned_tweets = [clean_text(tweet) for tweet
in tweets]
>>> tweet_corpus = ' '.join(cleaned_tweets)
```

Processing Raw Text

```
#lecture3_example4.py (continued)

>>> tweet_tokens = nltk.word_\
tokenizer(tweet_corpus)
>>> stemmed_tweets = [porter_stemmer.stem(tweet)
for tweet in tweet_tokens]
>>> fdist = nltk.FreqDist(stemmed_tweets)
>>> print fdist.most_common(5)
```

Processing Raw Text

```
#lecture3_example4.py (continued)
```

1. (u'miliband', 6308)
2. (u'tori', 5948)]

Processing Raw Text

```
#lecture3_example4.py (continued)

>>> bigram = ngrams(stemmed_tweets, 2)
>>> grams = [gram for gram in bigram]
>>> bifdist = nltk.FreqDist(grams)
>>> print bifdist.most_common(2)
```

Processing and understanding text

Processing Raw Text

```
#lecture3_example4.py (continued)
```

1. ((u'ed', u'miliband'), 1924)
2. ((u'david', u'cameron'), 1658)

Processing Raw Text

```
#lecture3_example4.py (continued)

>>> trigram = ngrams(stemmed_tweets, 3)
>>> grams = [gram for gram in trigram]
>>> trifdist = nltk.FreqDist(grams)
>>> print trifdist.most_common(2)
```

Processing Raw Text

```
#lecture3_example4.py (continued)
```

1. ((u'miliband', u'preoccupi', u'inequ'), 636)
2. ((u'come', u'support', u'tori'), 631)

Text classification

Overview

- Pre-processing is first step in text analysis pipeline
- Text classification is often performed next
  - Spam identification
  - News topic categorization
  - Sentiment analysis
- Large text corpus requires machine learning techniques
  - Supervised learning
  - Unsupervised learning

Text classification

Overview

- Unsupervised Learning
  - No target
  - Clustering
  - Pattern detection
- Supervised Learning
  - $Y = \beta X + \epsilon$
  - Continuous target $\rightarrow$ regression analysis
  - Discrete target $\rightarrow$ classification

Text classification

Classifying Articles

```
#lecture3_example5.py

$user python
>>> import nltk
>>> from nltk.corpus import reuters
>>> from nltk.corpus import stopwords
>>> from nltk.stem.porter import PorterStemmer
>>> from nltk import ngrams
>>> import re
```

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)

>>> import numpy as np
>>> from sklearn.feature_extraction.text import
TfidfVectorizer
>>> from sklearn.naive_bayes import GaussianNB
>>> from sklearn.metrics import confusion_matrix
>>> from sklearn import linear_model
>>> porter_stemmer = PorterStemmer()
>>> print reuters.categories()
```

Text classification

Classifying Articles

```
[u'acq', u'alum', u'barley', u'bop', u'carcass',
u'castor-oil', u'cocoa', u'coconut', u'coconut-oil',
u'coffee', u'copper', u'copra-cake', u'corn',
u'cotton', u'cotton-oil', u'cpi', u'cpu', u'crude',
u'dfl', u'dlr', u'dmk', u'earn', u'fuel', u'gas',
u'gnp', u'gold', u'grain', u'groundnut',
u'groundnut-oil', u'heat', u'hog', ...  ,
u'reserves', u'retail', u'rice', u'rubber', u'rye',
u'ship', u'silver', u'sorghum', u'soy-meal',
u'soy-oil', u'soybean', u'strategic-metal',
u'sugar', u'sun-meal', u'sun-oil', u'sunseed',
u'tea', u'tin', u'trade', u'veg-oil', u'wheat',
u'wpi', u'yen', u'zinc']
```

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)
>>> print reuters.fileids(['corn'])
```

Text classification

Classifying Articles

```
[u'test/14832', u'test/14858', u'test/15033',
u'test/15043', u'test/15106', u'test/15287',
u'test/15341', u'test/15618', u'test/15648',
u'test/15676', u'test/15686', u'test/15720',
u'test/15845', u'test/15856', u'test/15860',
u'test/15863', u'test/15871',...,u'training/8535',
u'training/855', u'training/8759', u'training/8941',
u'training/8983', u'training/8993',
u'training/9058', u'training/9093',
u'training/9094', u'training/934', u'training/9470',
u'training/9521', u'training/9667', u'training/97',
u'training/9865', u'training/9958',
u'training/9989']
```

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)
>>> print reuters.fileids(['wheat'])
```

Text classification

Classifying Articles

```
[u'test/14841', u'test/15043', u'test/15097',
u'test/15132', u'test/15271', u'test/15273',
u'test/15341', u'test/15388', u'test/15472',
u'test/15500', u'test/15567', u'test/15572',
u'test/15582',..., u'training/8179',
u'training/8273', u'training/8413',
u'training/8535', u'training/856', u'training/8604',
u'training/874', u'training/8759', u'training/8993',
u'training/9021', u'training/9095', u'training/97',
u'training/9773', u'training/9782',
u'training/9793', u'training/9865']
```

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)
>>> corn = reuters.fileids(['corn'])
>>> wheat = reuters.fileids(['wheat'])
>>> common = set(corn).intersection(wheat)
>>> corn = [id for id in corn if id not in common]
>>> wheat = [id for id in wheat if id not in
common]
```

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)

>>> train_corn_ids = [train for train in corn if
train.find('train')>-1]
>>> test_corn_ids = [test for test in corn if
test.find('test')>-1]
>>> train_wheat_ids = [train for train in wheat if
train.find('train')>-1]
>>> test_wheat_ids = [test for test in wheat if
test.find('test')>-1]
```

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)
>>> train_corn_target = []
>>> test_corn_target = []
>>> train_wheat_target = []
>>> test_wheat_target = []
>>> train_corn = []
>>> test_corn = []
>>> train_wheat = []
>>> test_wheat = []
```

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)
>>> def load_train_data():
        train = []
        train_target = []
        for id in train_corn_ids:
          train_corn_target.append(0)
          train_corn.append(reuters.raw(id))
        for id in train_wheat_ids:
          train_wheat_target.append(1)
          train_wheat.append(reuters.raw(id))
        train = train_corn + train_wheat
        train_target = train_corn_target +
        train_wheat_target
        return train, train_target
```

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)
>>> def load_test_data():
        test = []
        test_target = []
        for id in test_corn_ids:
            test_corn_target.append(0)
            test_corn.append(reuters.raw(id))
        for id in test_wheat_ids:
            test_wheat_target.append(1)
            test_wheat.append(reuters.raw(id))
        test = test_corn + test_wheat
        test_target = test_corn_target +
        test_wheat_target
        return test, test_target
```

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)

>>> train, train_target = load_train_data()
>>> test, test_target = load_test_data()
```

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)
>>> def preprocess_text(text):
        text = re.sub('[^A-Za-z]+', ' ', text)
        wordTokens = nltk.word_tokenize(text)
        wordTokens = [token.lower() for token in
        wordTokens if len(token)>1]
        stops = set(stopwords.words("english"))
        wordTokens = [token for token in wordTokens
        if token not in stops]
        stemmedTokens = [porter_stemmer.stem(token)
        for token in wordTokens]
        cleanedText = ' '.join(stemmedTokens)
        return cleanedText
```

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)

>>> train = [preprocess_text(doc) for doc in train]
>>> test = [preprocess_text(doc) for doc in test]
>>> train = [doc for doc in train if len(doc)>0]
>>> test = [doc for doc in test if len(doc)>0]
```

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)

>>> vectorizer = TfidfVectorizer()
>>> train_weights =
vectorizer.fit_transform(train).toarray()
>>> test_weights =
vectorizer.fit_transform(test).toarray()
>>> nb_0 = GaussianNB().fit(train_weights,
train_target)
>>> train_pred = nb_0.predict(train_weights)
>>> confusion_matrix(train_target, train_pred)
```

Text classification

Classifying Articles

	corn	wheat
corn	119	3
wheat	1	152

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)
>>> test_pred = nb_0.predict(test_weights)
```

Text classification

Classifying Articles

```
ValueError:  operands could not be broadcast
together with shapes (127,1640) (3193,)
```

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)

>>> weights = vector-
izer.fit_transform(np.hstack([train,test])).toarray()
>>> train_weights = weights[:len(train_weights),:]
>>> test_weights = weights[len(train_weights):,:]
```

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)

>>> nb_1 = GaussianNB().fit(train_weights,
train_target)
>>> train_pred = nb_1.predict(train_weights)
>>> test_pred = nb_1.predict(test_weights)
>>> confusion_matrix(train_target, train_pred)
```

Text classification

Classifying Articles

	corn	wheat
corn	119	3
wheat	1	152

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)
>>> confusion_matrix(test_target, test_pred)
```

Text classification

Pre-Processing Data

	corn	wheat
corn	20	14
wheat	7	42

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)

>>> lr = linear_model.LogisticRegression().\
fit(train_weights, train_target)
>>> train_pred = lr.predict(train_weights)
>>> test_pred = lr.predict(test_weights)
>>> confusion_matrix(train_target, train_pred)
```

Text classification

Classifying Articles

	corn	wheat
corn	119	3
wheat	1	152

Text classification

Classifying Articles

```
#lecture3_example5.py (continued)
>>> confusion_matrix(test_target, test_pred)
```

Text classification

Classifying Articles

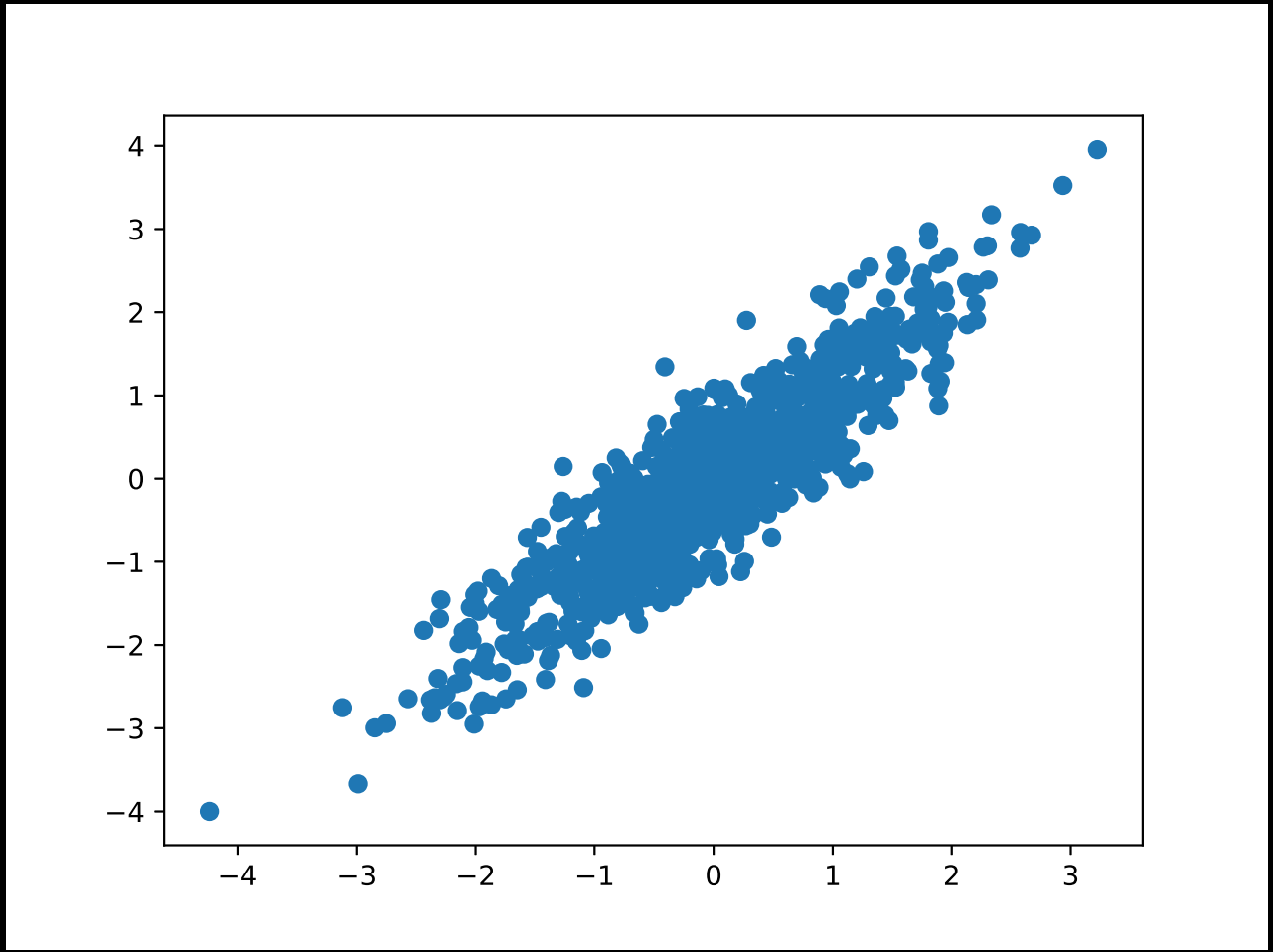
	corn	wheat
corn	28	6
wheat	2	47

Matplotlib

Generating Scatterplots

```
$user ipython qtconsole -pylab=inline
>>> import matplotlib
>>> matplotlib.use(`TkAgg')
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> %pylab inline
>>> x = np.random.normal(0,1,1000)
>>> y = x + np.random.normal(0,0.50,1000)
>>> plt.scatter(x,y)
>>> plt.savefig(`../scatter.png')
>>> plt.savefig(`../scatter.eps')
```

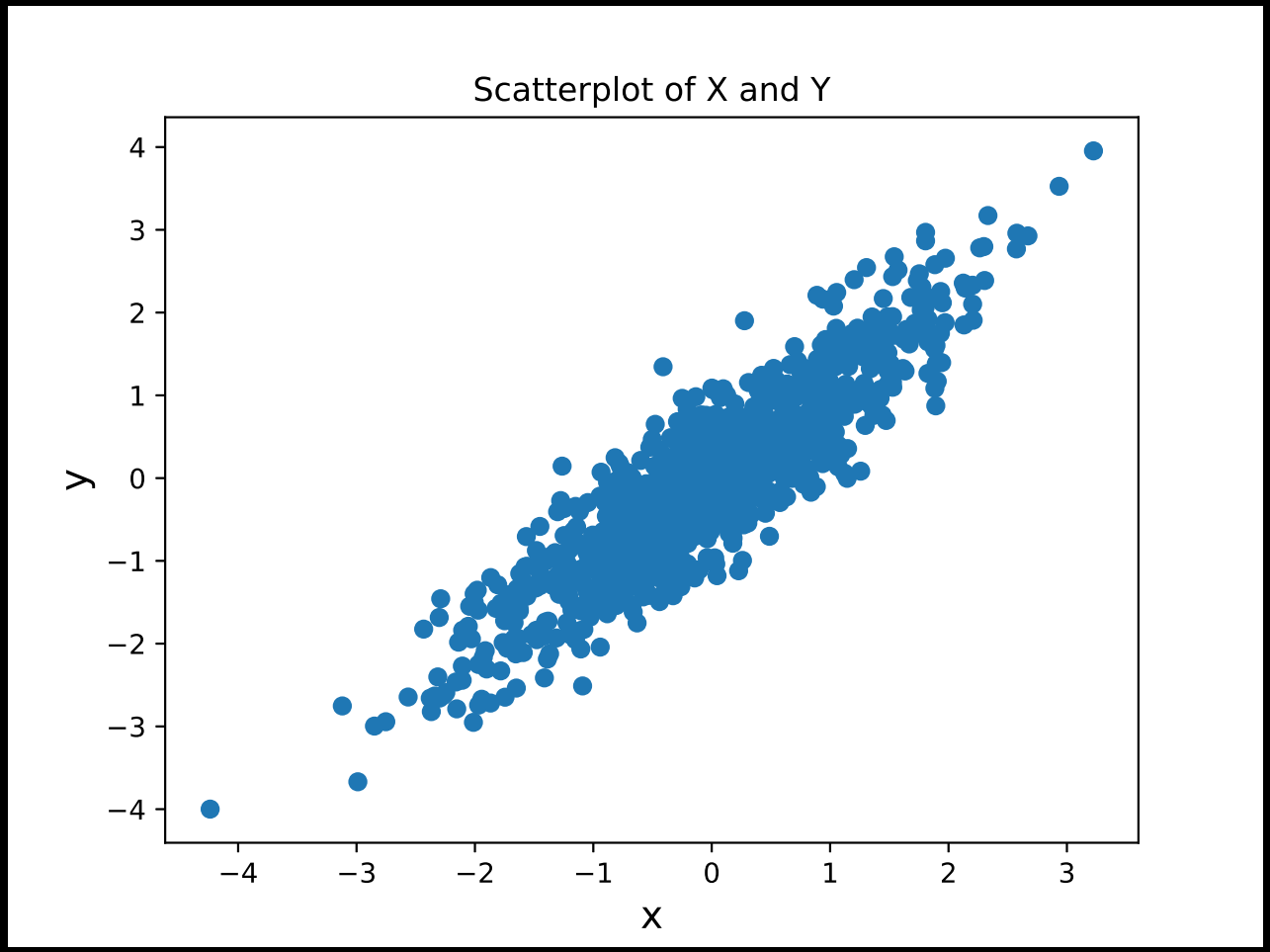
Generating Scatterplots



Generating Scatterplots

```
>>> plt.xlabel('x', fontsize=14, color='black')
>>> plt.ylabel('y', fontsize=14, color='black')
>>> plt.title('Scatterplot of X and Y')
>>> plt.savefig(`../scatter_labels.eps')
>>> plt.close()
```

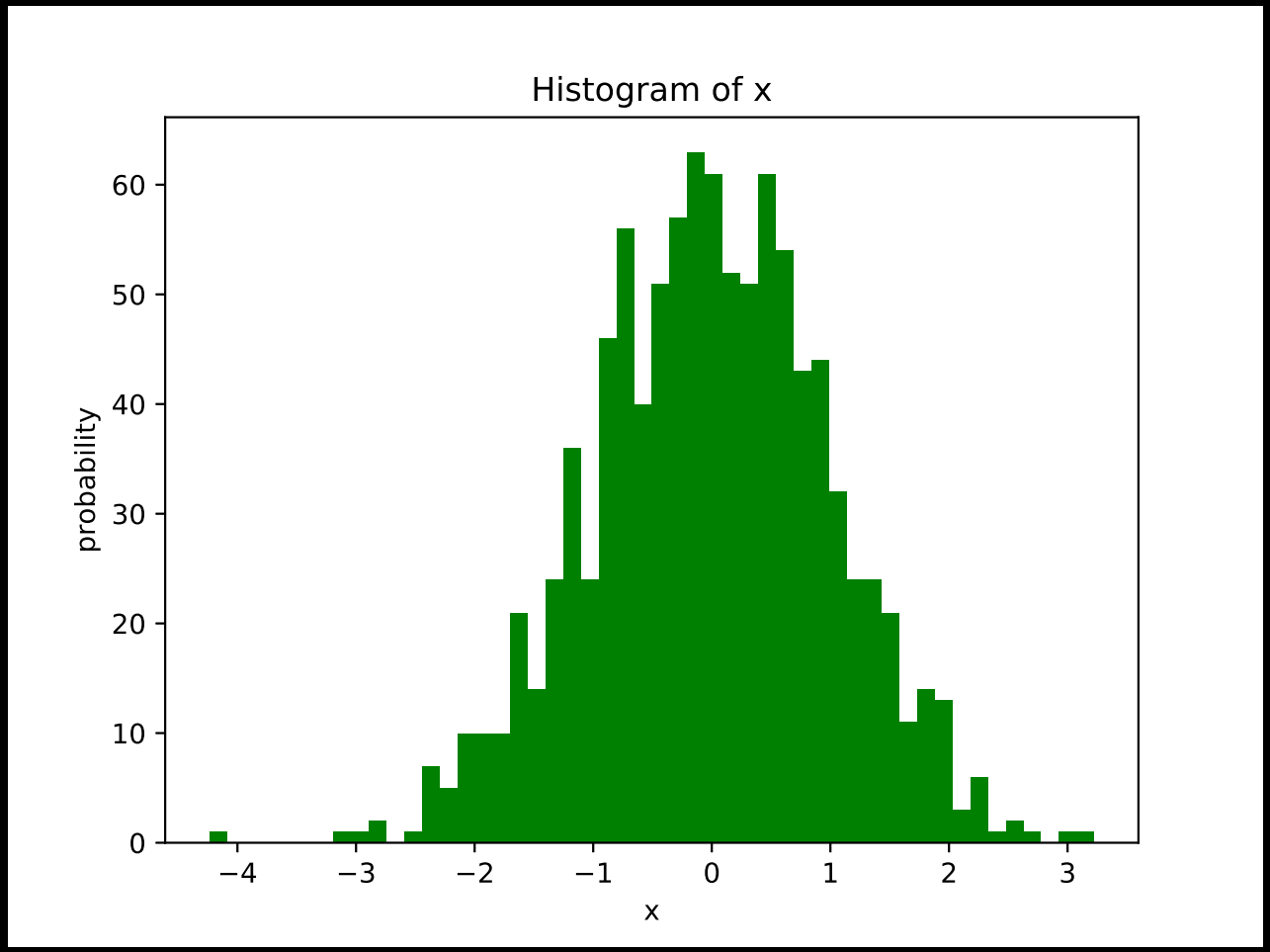
Generating Scatterplots



Generating Histograms

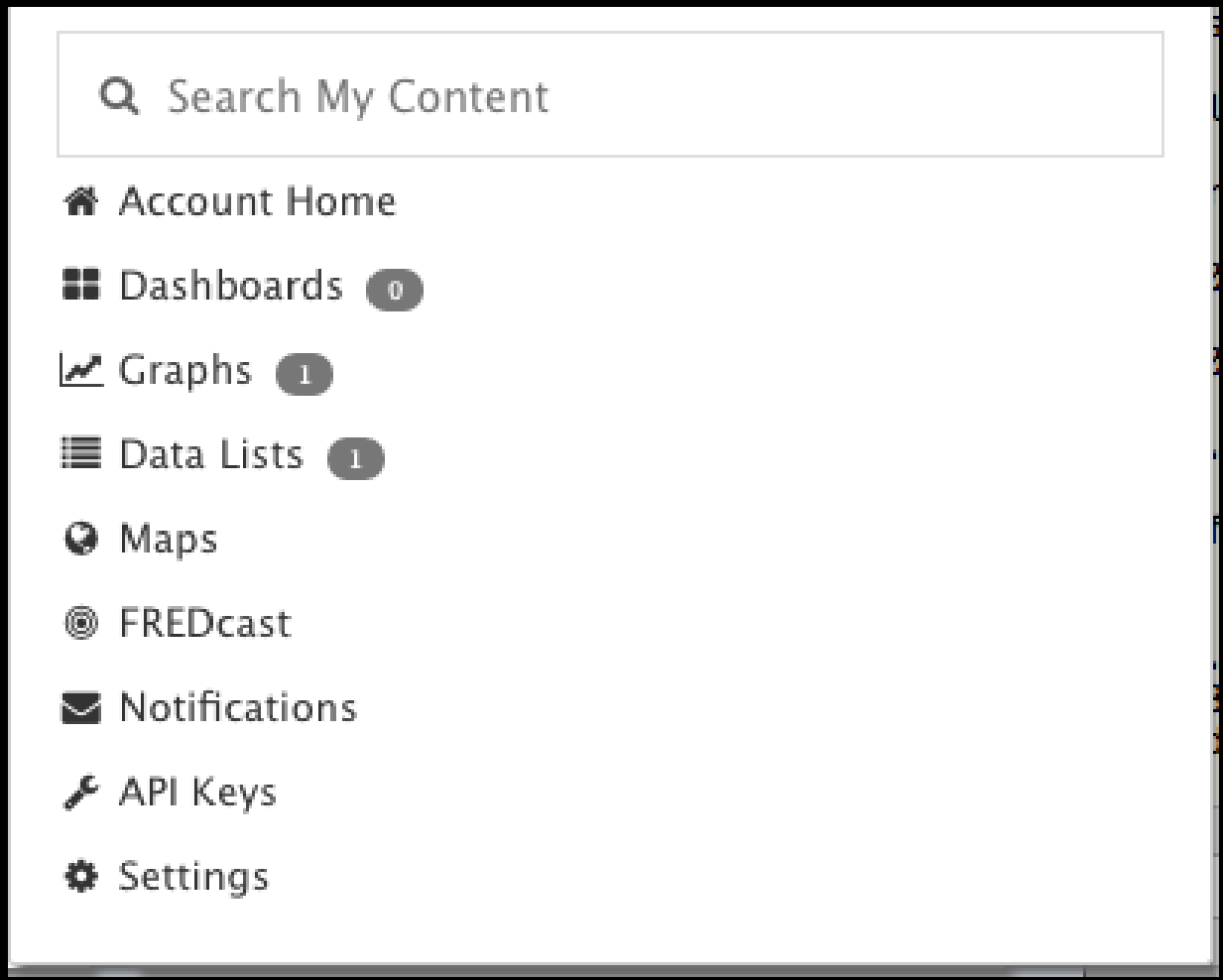
```
>>> n, bins, patches = plt.hist(x, 50, normed=1,
facecolor=`g', alpha=0.75)
>>> plt.xlabel('x')
>>> plt.ylabel('y')
>>> plt.title('Histogram of X', fontsize=18,
color=`black')
>>> plt.savefig(`../histogram_labels.eps')
>>> plt.close()
```

Generating Histograms



Generating Time Series Plots

Generating Time Series Plots



Generating Time Series Plots

```
$user pip install fredapi
>>> from fredapi import Fred
>>> fred = Fred(api_key=`REPLACE_WITH_YOUR_API_KEY'
>>> series = fred.search(`ukraine')
>>> print len(series)
373
```

Generating Time Series Plots

```
$user pip install fredapi
>>> for title in series['title']:
        print title
```

Generating Time Series Plots

```
Gross Domestic Product for Ukraine
GINI Index for the Ukraine
...
Bank's Cost to Income Ratio for Ukraine
Bank Capital to Total Assets for Ukraine
...
Number of Identified Exporters to Ukraine from North
Carolina
```

Generating Time Series Plots

```
>>> search_string = series[`id'][50]
>>> title_string = series[`title'][50]
>>> data = fred.get_series(search_string)
```

Matplotlib

Generating Time Series Plots

```
>>> plt.plot(data)
>>> plt.xlabel(`Year')
>>> plt.ylabel(`Ratio')
>>> plt.title(title_string)
>>> plt.savefig(`../time_series.eps')
>>> plt.close()
```

Generating Time Series Plots

```
# Generate time series plot.
```

Text Analysis and Data Visualization

1. Load tweets about Brexit
2. Pre-process tweets
3. Apply a topic model
4. Apply a sentiment model
5. Sort tweets into topic groups
6. Plot topic group distributions

Text Analysis and Data Visualization

```
#lecture3_example6.py (continued)

$user python
>>> import nltk
>>> from nltk.corpus import twitter_samples
>>> from nltk.corpus import stopwords
>>> from nltk.stem.porter import PorterStemmer
>>> import re
>>> import numpy as np
>>> import pandas as pd
```

Bokeh

Text Analysis and Data Visualization

```
#lecture3_example6.py (continued)

>>> from nltk.sentiment.vader import
SentimentIntensityAnalyzer as SIA
>>> from sklearn.feature_extraction.text import
TfidfVectorizer
>>> from sklearn.decomposition import NMF,
LatentDirichletAllocation
>>> from bokeh.plotting import figure, show,
output_file
```

Text Analysis and Data Visualization

```
#lecture3_example6.py (continued)

>>> porter_stemmer = PorterStemmer()
>>> sia = SIA()
>>> tweets = twitter_samples.\
strings(twitter_samples.fileids()[-1])
```

Text Analysis and Data Visualization

```
#lecture3_example6.py (continued)

>>> def clean_text(tweet):
        tweet = tweet.strip().lower()
        pattern = "(@[A-Za-z0-9]+)|([A-Za-z0-9]+)|
        ([^0-9A-Za-z \t])|(\w+:\/\/\S+)|(rt)"
        cleanedTweet = ' '.join(re.sub(pattern, ' ',
        tweet).split())
        wordTokens = nltk.word_tokenize(cleanedTweet)
        wordTokens = [token for token in wordTokens if
        len(token)>1]
        stops = set(stopwords.words("english")
        wordTokens = [token for token in wordTokens if token
        not in stops]
        stemmedTweet = [porter_stemmer.stem(tweet) for tweet
        in wordTokens]
        cleanedTweet = ' '.join(stemmedTweet)
        return cleanedTweet
```

Text Analysis and Data Visualization

```
#lecture3_example6.py (continued)

>>> cleanedTweets = [clean_text(tweet) for tweet in
tweets]
>>> lda = LatentDirichletAllocation(
                n_topics=5, max_iter=5,
                learning_method=`online′,
                learning_offset=50.,
                random_state=0)
```

Text Analysis and Data Visualization

```
#lecture3_example6.py (continued)

>>> vectorizer = TfidfVectorizer()
>>> tf = vectorizer.fit_transform(cleanedTweets)
>>> feature_names = vectorizer.get_feature_names()
>>> lda.fit(tf)
>>> topic_words = []
```

Text Analysis and Data Visualization

```
#lecture3_example6.py (continued)

>>> for topic in lda.components_:
        word_idx = np.argsort(topic)[::-1][0:1]
        topic_words.append([feature_names[i] for i
        in word_idx][0])

>>> print topic_words
```

Bokeh

Text Analysis and Data Visualization

```
[u`cameron´,
u`farage´,
u`claim´,
u`ukip´,
u`snp´]
```

Text Analysis and Data Visualization

```
#lecture3_example6.py (continued)

>>> cameron = [tweet for tweet in cleanedTweets if
tweet.find('cameron')>-1]
>>> farage = [tweet for tweet in cleanedTweets if
tweet.find('farage')>-1]
>>> claim = [tweet for tweet in cleanedTweets if
tweet.find('claim')>-1]
>>> ukip = [tweet for tweet in cleanedTweets if
tweet.find('ukip')>-1]
>>> snp = [tweet for tweet in cleanedTweets if
tweet.find('snp')>-1]
```

Text Analysis and Data Visualization

```
#lecture3_example6.py (continued)

>>> cameron_compound = [sia.polarity_
scores(tweet)[`compound´] for tweet in cameron]
>>> farage_compound = [sia.polarity_
scores(tweet)[`compound´] for tweet in farage]
>>> claim_compound = [sia.polarity_
scores(tweet)[`compound´] for tweet in claim]
>>> ukip_compound = [sia.polarity_
scores(tweet)[`compound´] for tweet in ukip]
>>> snp_compound = [sia.polarity_
scores(tweet)[`compound´] for tweet in snp]
```

Text Analysis and Data Visualization

```
#lecture3_example6.py (continued)

>>> cameron_name= np.array(["cameron"]*
len(cameron_compound)
>>> farage_name = np.array(["farage"]*
len(farage_compound))
>>> claim_name = np.array(["claim"]*
len(claim_compound))
>>> ukip_name = np.array(["ukip"]*
len(ukip_compound))
>>> snp_name = np.array(["snp"]*
len(snp_compound))
```

Text Analysis and Data Visualization

```
#lecture3_example6.py (continued)
>>> cats = [“cameron”, “claim”, “farage”, “snp”,
“ukip”]
>>> compound = np.hstack([cameron_compound,
claim_compound, farage_compound, snp_compound,
ukip_compound])
>>> name = np.hstack([cameron_name, claim_name,
farage_name, snp_name, ukip_name])
>>> df = pd.DataFrame(np.vstack([name,
compound]).T, columns=[`group′, `score′])
>>> df[`score′] = df[`score′].astype(`float′)
```

Bokeh

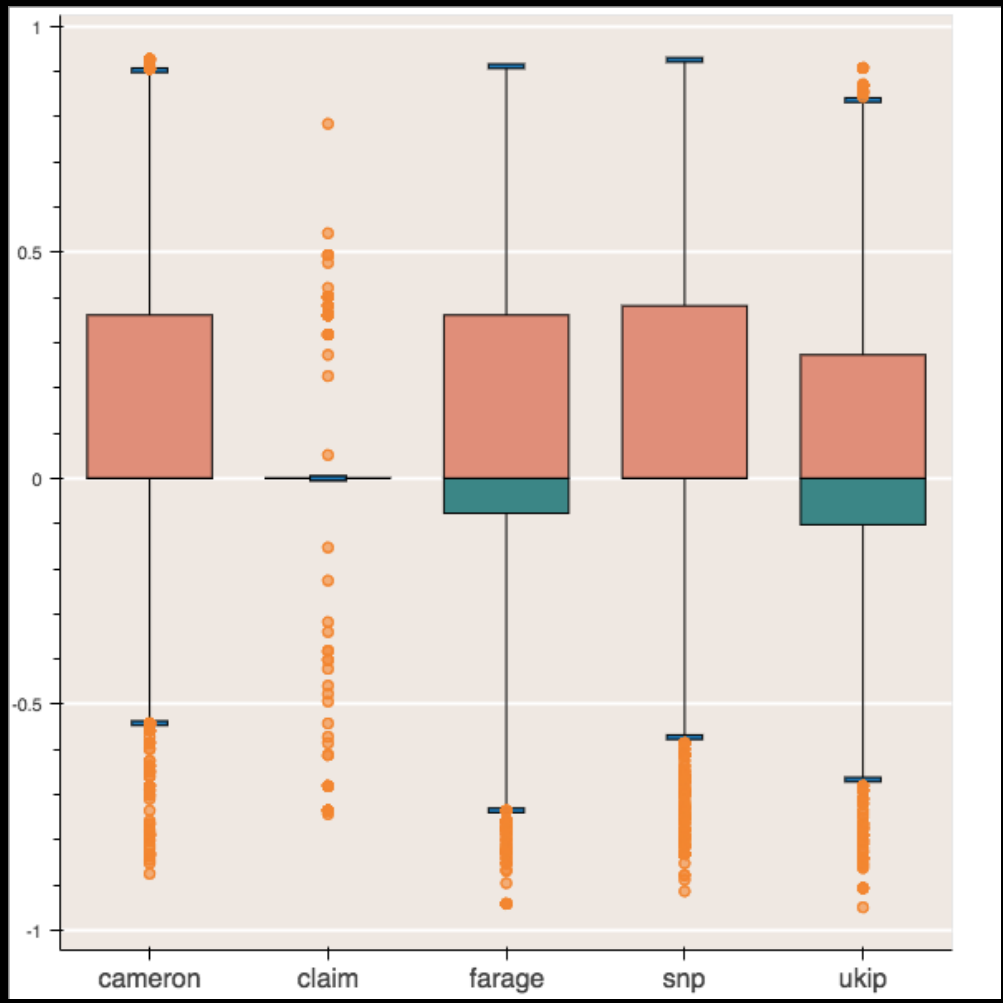
Text Analysis and Data Visualization

- Bokeh Boxplot

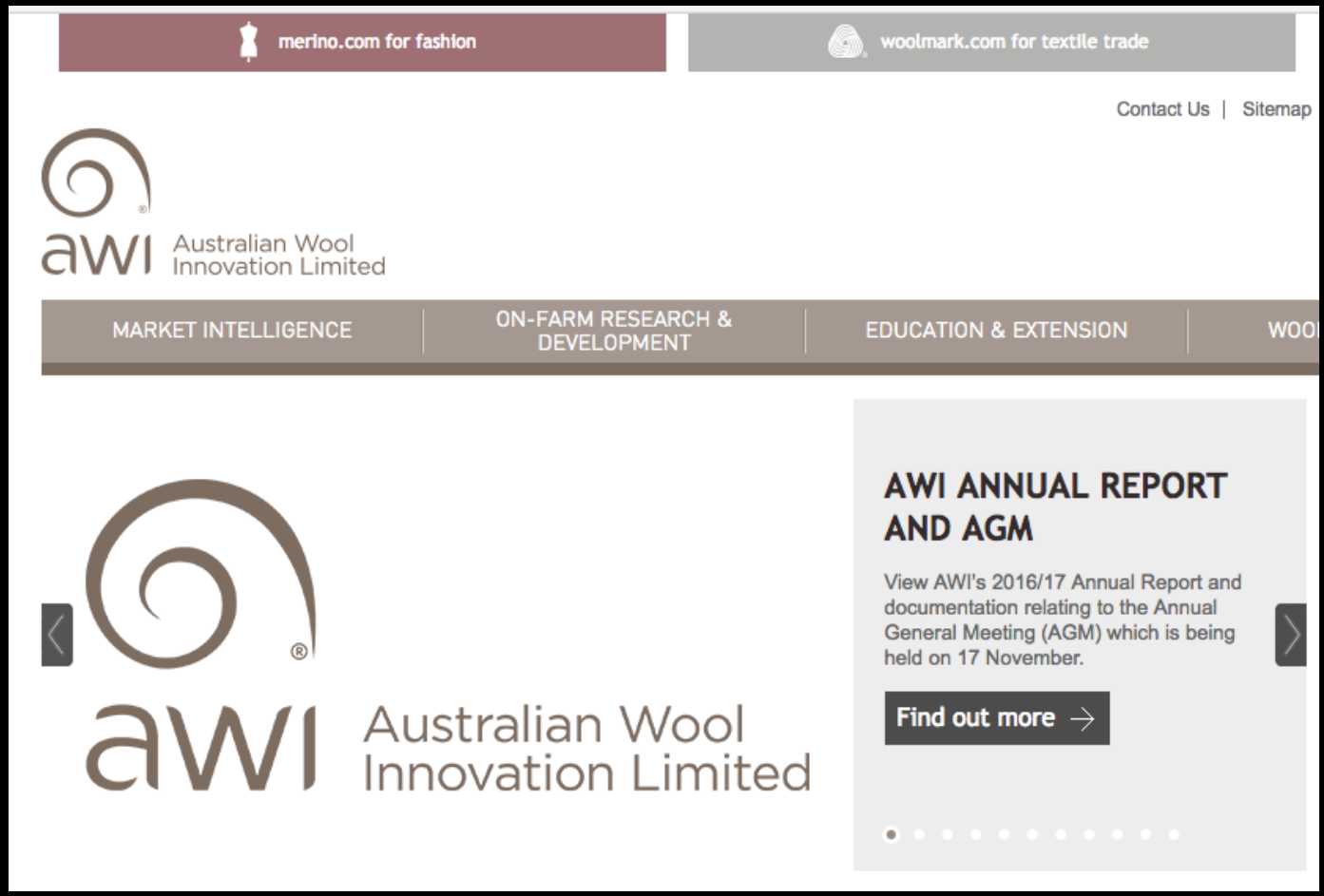
https://bokeh.pydata.org/en/latest/docs/gallery/boxplot.html

Bokeh

Text Analysis and Data Visualization



Text Analysis and Data Visualization



https://www.wool.com/

Bokeh

Text Analysis and Data Visualization

1. Collect links to monthly reports on wool.
2. Download pdf at each link.
3. Convert pdf to txt and pre-process text.
4. Apply Loughran-McDonald (2011) sentiment model.
5. Generate monthly intensity plot in Bokeh.

Bokeh

Text Analysis and Data Visualization

Text Analysis and Data Visualization

Recent wool prices have been a welcome start to the 2017 calendar year, with the EMI (Eastern Market Indicator) reaching well above 1400 ac clean/kg. The market has been marked by record highs and even 'peak' wool values, yet it is important to understand the previous times wool has achieved these values. This month we place some historical context around the current market and explore supply versus demand.

It has been great to see the current wool market give such a well-deserved return for Merino woolgrowers' efforts, however it is important to note that these prices are not actually record levels as reported. Prices did achieve record highs but only relative to when AWEX began market reporting in 1994. The highest prices were actually achieved in 1988 while the Reserve Price Scheme was in place. Back then, the market indicator was based around a 22 micron type, before a subsequent re-basing of the indicator to a more production-relevant 21 micron. A high of 1584ac clean/kg was reached in the autumn of 1988. Due to heavy Russian interest in Australian wool during that era, it pushed prices upwards as they secured around 35% of the wool clip, a figure that would nowadays buy the entire Australian clip. The average market indicator for that year was

Bokeh

Text Analysis and Data Visualization

```
#lecture3_example7.py

$user pip install pysentiment
$user python
>>> import os
>>> from urllib2 import urlopen
>>> import nltk
>>> from nltk.corpus import stopwords
>>> import PyPDF2
>>> import re
>>> import time
```

Bokeh

Text Analysis and Data Visualization

```
#lecture3_example7.py (continued)

>>> import pysentiment as ps
>>> import pandas as pd
>>> import numpy as np
>>> from math import pi
>>> from bokeh.io import show
>>> from bokeh.plotting import figure
```

Bokeh

Text Analysis and Data Visualization

```
#lecture3_example7.py (continued)

>>> from bokeh.models (
        ColumnDataSource,
        HoverTool,
        LinearColorMapper,
        BasicTicker,
        PrintfTickFormatter,
        ColorBar,
)
```

Text Analysis and Data Visualization

```
#lecture3_example7.py (continued)

# Set save directory and urls.

>>> save_dir = `/Users/user/Desktop/'
>>> url = `https://www.wool.com'
>>> url0 = 'https://www.wool.com/market-
intelligence/monthly-market-reports/?page=1month='
>>> url1 = '&year='

# Define list.

>>> links, lmonths, lyears = [], [], []
>>> months = [`1', '2', `3', `4', `5', `6', `7',
`8', `9', `10', `11', `12']
>>> years = [`2013', `2014', `2015', `2016',
`2017']
```

Text Analysis and Data Visualization

```
#lecture3_example7.py (continued)

# Get links to reports.

for year in years:
        for month in months:
                html = urlopen(url0+month+url1+year).read()
                soup = BeautifulSoup(html)
                link = soup.findAll("a", "class":"btnPrimary")
                if(len(link)>0):
                        links.append(link)
                else:
                        links.append('')
                lmonths.append(month)
                lyears.append(year)
                print month, year
                time.sleep(3)
```

Text Analysis and Data Visualization

```
#lecture3_example7.py (continued)

# Extract href tags.

>>> for j in range(len(links)):
        try:
            links[j] = url+links[j][0][`href']
        except:
            links[j] = ''

>>> lm = ps.LM()
```

Text Analysis and Data Visualization

```
#lecture3_example7.py (continued)

# Load, save, and delete pdf.

>>> def load_and_process_pdf(link):
        content = urlopen(link).read()
        with open(save_dir+`content.pdf', `wb') as f:
            f.write(content)
        time.sleep(120)
        content = PyPDF2.PdfFileReader(
        open(save_dir+`content.pdf'), "rb")
        content_text = [content.getPage(j).extractText()
        for j in range(content.numPages)]
        content_text = ' '.join(content_text)
        os.unlink(save_dir+'content.pdf')
        return content_text
```

Text Analysis and Data Visualization

```
#lecture3_example7.py (continued)

# Compute sentiment scores.

>>> def compute_sentiment_scores(link):
        try:
            page = load_and_process_pdf(link)
            processed_page = lm.tokenize(page)
            sentiment = lm.get_score(processed_page)
            negativity = sentiment[`Negative']
            polarity = sentiment[`Polarity']
            positivity = sentiment[`Positive']
            subjectivity = sentiment[`Subjectivity']
        except:
            sentiment, negativity, polarity,
          positivity, subjectivity = '', '', '', '', ''
        return negativity, polarity, positivity, subjectivity
```

Text Analysis and Data Visualization

```
#lecture3_example7.py (continued)

# Generate sentiment scores.

>>> sent = []
>>> for link in links:
        sent.append(compute_sentiment_scores(link))
        print link
```

Text Analysis and Data Visualization

```
#lecture3_example7.py (continued)

# Extract sentiment scores.

>>> negativity = [x[0] for x in sent]
>>> polarity = [x[1] for x in sent]
>>> positivity = [x[2] for x in sent]
>>> subjectivity = [x[3] for x in sent]
```

Text Analysis and Data Visualization

```
#lecture3_example7.py (continued)

# Arrange polarity into date array.

>>> Jan = polarity[0::12]
>>> Feb = polarity[1::12]
>>> Mar = polarity[2::12]
>>> Apr = polarity[3::12]
>>> May = polarity[4::12]
>>> Jun = polarity[5::12]
>>> Jul = polarity[6::12]
>>> Aug = polarity[7::12]
>>> Sep = polarity[8::12]
>>> Oct = polarity[9::12]
>>> Nov = polarity[10::12]
>>> Dec = polarity[11::12]
```

Text Analysis and Data Visualization

```
#lecture3_example7.py (continued)

# Arrange polarity into date array.

>>> data = pd.DataFrame(np.vstack([Jan, Feb, Mar,
Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec]).T,
columns=[`Jan', `Feb', `Mar', `Apr', `May', `Jun',
`Jul', `Aug', `Sep', `Oct', `Nov', `Dec'])
>>> data = data.apply(lambda x:
x.str.strip()).replace('', np.nan).astype(`float')
>>> data = data.set_index([[`2013', `2014', `2015',
`2016', `2017']])
>>> data.index.name = `Year'
>>> data.columns.name = `Month'
>>> months = ['Jan', 'Feb', 'Mar', 'Apr', 'May',
'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
```

Text Analysis and Data Visualization

```
#lecture3_example7.py (continued)

# Plot data.
>>> df = pd.DataFrame(data.stack(),
columns=['Sentiment']).reset_index()
>>> colors = ["#550b1d", "#933b41", "#cc7878",
"#ddb7b1", "#dfccce", "#e2e2e2", "#c9d9d3",
"#a5bab7", "#75968f"]
>>> mapper = LinearColorMapper(palette=colors,
low=df.Sentiment.min(), high=df.Sentiment.max())
>>> source = ColumnDataSource(df)
>>> TOOLS =
"hover,save,pan,box_zoom,reset,wheel_zoom"
```

Bokeh

Text Analysis and Data Visualization

```
#lecture3_example7.py (continued)

>>> p = figure(title="Wool Sentiment Score
      (0 - 1)".format(years[0], years[-1]),
      x_range=years, y_range=list(reversed(months)),
      x_axis_location="above",
      plot_width=400, plot_height=400, tools=TOOLS,
      toolbar_location='below')
```

Text Analysis and Data Visualization

```
#lecture3_example7.py (continued)

>>> p.grid.grid_line_color = None
>>> p.axis.axis_line_color = None
>>> p.axis.major_tick_line_color = None
>>> p.axis.major_label_text_font_size = “5pt"
>>> p.axis.major_label_standoff = 0
>>> p.xaxis.major_label_orientation = pi / 3
```

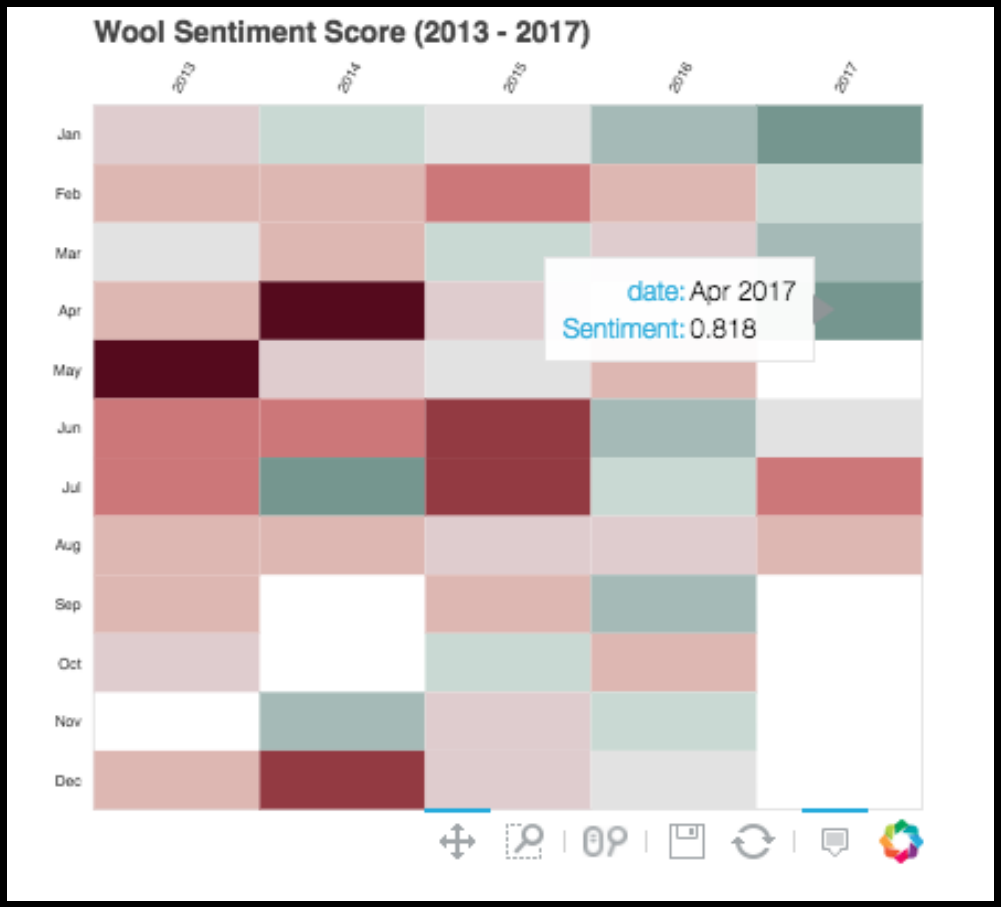
Bokeh

Text Analysis and Data Visualization

```
#lecture3_example7.py (continued)

>>> p.rect(x="Year", y="Month", width=1, height=1,
source=source, fill_color=`field': `Sentiment',
`transform': mapper, line_color=None)

>>> p.select_one(HoverTool).tooltips = [(`date',
`@Month @Year'), (`Sentiment', `@Sentiment'),]

>>> show(p)
```

Bokeh

Text Analysis and Data Visualization



Summary

- Processing and understanding text
  - NLTK
  - Tokenization
  - Cleaning raw text
  - Processing raw text

Summary

- Text classification
  - Classifying articles
  - Tfidf vectorization
  - Cross validation
  - Supervised learning

Summary

- Matplotlib
  - Scatterplots
  - Histograms
  - Time series plots
  - FRED API

Summary

- Bokeh
  - Text analysis and data visualization
  - Loughran-McDonald (2011) sentiment model
  - Monthly intensity plots
  - Sentiment boxplots
  - Latent Dirichlet allocation (LDA)