Anamitra Bhattacharyya
Predict 420-DL, Section 55
Assignment 6 (June 5, 2016)

1) Python Code

```
import pandas as pd # DataFrame object work
import numpy as np
from datetime import datetime # textdate manipulation

print('First connect to the Northwestern VPNn')

# prompt for user's NetID and password
my_netid = raw_input('Enter your NetID: ')
my_password = raw_input('Enter your password: ')

try:
client = MongoClient("129.105.208.225")
client.enron.authenticate(my_netid, my_password,source='$external', mechanism='PLAIN')
print('\nConnected to MongoDB enron database\n')
success = True
except:
print('\nUnable to connect to the enron database')

# if connection is successful, work with the database

#print('\nCollections in the enron database')
cols = client.enron.collection_names()

for col in cols:
print(col)
# work with documents in the messages collection
workdocs = client.enron.messages

# inquire about the documents in messages collection
print('\nNumber of documents ', workdocs.count())
print('\nOne such document\n', workdocs.find_one())
one_dict = workdocs.find_one() # create dictionary
print('\nType of object workdocs ', type(one_dict))

#Q1) how many documents contain 'Ken Lay' emails in the text field
print(workdocs.find({'$text':{'$search':'"klay@enron.com"'}}).count())#No. of Ken Lay emails = 2335
kenlayemails = list(workdocs.find({'$text':{'$search':'"klay@enron.com"'}}))
print('\nCreated Python object with Ken Lays emails')
print('\nType of object kenlayemails', type(kenlayemails))
print('\nNumber of items in kenlayemails ', len(kenlayemails))

# flatten the nested dictionaries in kenlayemails
# and remove _id field
list_of_kenlayemails_dict_data = []
for message in kenlayemails:
tmp_message_flattened_parent_dict = message
tmp_message_flattened_child_dict = message['headers']
del tmp_message_flattened_parent_dict['headers']
del tmp_message_flattened_parent_dict['_id']
tmp_message_flattened_parent_dict. \
update(tmp_message_flattened_child_dict)
list_of_kenlayemails_dict_data. \
append(tmp_message_flattened_parent_dict.copy())

print('\nType of object list_of_kenlayemails_dict_data',
type(list_of_kenlayemails_dict_data))
print('\nNumber of items in list_of_kenlayemails_dict_data ',
len(list_of_kenlayemails_dict_data))

# we can use Python pandas to explore and analyze these data
# create pandas DataFrame object to begin analysis
kenlay_email_df = pd.DataFrame(list_of_kenlayemails_dict_data)
print('\nType of object kenlay_email_df', type(kenlay_email_df))

# set missing data fields
```

```
kenlay_email_df.fillna("", inplace=True)

# user-defined function to create simple date object (no time)
def convert_date_string (date_string):
        try:
        return(datetime.strptime(str(date_string)[:16].
        lstrip().rstrip(), '%a, %d %b %Y'))
        except:
        return(None)

# apply function to convert string Date to date object
kenlay_email_df['Date'] = kenlay_email_df['Date'].apply(lambda d: convert_date_string(d))

# date of Enron bankruptcy
BANKRUPTCY = datetime.strptime(str('Sun, 2 Dec 2001'), '%a, %d %b %Y')

print('\nExamine kenlay_email_df ', type(kenlay_email_df))
print('\nNumber of observations', len(kenlay_email_df))
print('\nBeginning observations')
print(kenlay_email_df.head())

# Work with kenlay_email_df
print ("The columns of kenlay_email_df are: ")
print (kenlay_email_df.columns)

print ("\nThe number of emails: ")
print (kenlay_email_df.count())

print ("\nThe senders were:")
print (kenlay_email_df.From)

print ("\nThe receivers were:")
print (kenlay_email_df.To)
#Q1
#Emails from Ken Lay
fromLayDF = kenlay_email_df.loc[kenlay_email_df['From']=='klay@enron.com']
len(fromLayDF)# Zero emails 'From' Ken Lay (sent mail)

#Emails to Ken Lay
toLayDF = kenlay_email_df.loc[kenlay_email_df['To']=='klay@enron.com']#emails to Ken Lay
len(toLayDF)#1601 emails 'To' Ken Lay (received mail)

#Bcc and cc emails for Ken Lay
bccLayDF = kenlay_email_df[kenlay_email_df['X-bcc'].str.strip()=='klay@enron.com']
len(bccLayDF)# 0 emails that Ken Lay is Bcc'd
ccLayDF = kenlay_email_df[kenlay_email_df['X-cc'].str.strip()=='klay@enron.com']
len(ccLayDF)# 7 emails that Ken Lay is Cc'd

#Q2 Focused on the 1601 emails to Ken Lay (since none from him)
toLayDF.sort_values(['From'], ascending=True)#sort based on 'From' column then 'To'
toLayGrp = toLayDF.groupby('From')#grouping by the sender (i.e. 'From' category)
#There appears numerous instances of no more than 1 email per sender to Ken Lay
toLayGrp['From'].describe()
toLayResult = toLayGrp['From'].aggregate(np.sum)
print(toLayResult)

#Q3 Bankruptcy date
#Since sent emails are zero from Ken Lay in this data set irrespective of bankruptcy date
#So I checked for the frequency of emails to Ken Lay before/after bankruptcy date
toLayDF.sort_values(['Date'], ascending=True)

#At index=0, 2002-01-30 date is after bankruptcy 2001-12-02 (i.e. Sun, 2 Dec 2001)
toLayDF.head()

#At index=0, 2002-01-30 date is after bankruptcy 2001-12-02 (i.e. Sun, 2 Dec 2001)
toLayDF.tail()

tseries = pd.to_datetime(toLayDF['Date'])#
orderedbyDate = tseries.order()
orderedbyDate.rename(index=True, columns='Date', inplace=True)
orderedbyDate[::100]#Narrowed down sorted index to between 400-500
#index=437 date is 2001-11-30 (before); index of 438=2001-12-03 (1 day after bankruptcy)
orderedbyDate[437:439]
```

```
len(orderedbyDate)#1601 emails total
#Therefore from a total of 1601, 438 emails were before bankruptcy date and 1163 emails
#(i.e.1601-438) after bankruptcy

#Q4 Searched for 'Arthur Andersen' in the original workdocs
#Connected to the MongoDB via SSCC and extracted the search query results
print(workdocs.find({'$text':{'$search':'"Arthur Andersen"'}}).count())#Arthur Andersen in emails = 887
andersenemails = list(workdocs.find({'$text':{'$search':'"Arthur Andersen"'}}))
len(andersenemails)#list of 887 docs mentioning 'Arthur Andersen'
#When I searched for 'Arthur Andersen' in the kenlayemailsDF found 0 emails
andersenDF = kenlay_email_df[kenlay_email_df['body'].str.strip()==’Arthur Andersen’]


#EXTRA CREDIT
#Used modified code from URL source:
#http://stackoverflow.com/questions/30850688/construct-bipartite-graph-from-columns-of-python-dataframe

        import networkx as nx
        import matplotlib.pyplot as plt

        B = nx.Graph()
        B.add_nodes_from(kenlay_email_df['From'], bipartite=0)
        B.add_nodes_from(kenlay_email_df['To'], bipartite=1)
        B.add_weighted_edges_from(
        [(row['From'], row['To'], 1) for idx, row in kenlay_email_df.iterrows()])

        pos = {node:[0, i] for i,node in enumerate(kenlay_email_df['From'])}
        pos.update({node:[1, i] for i,node in enumerate(kenlay_email_df['To'])})
        nx.draw(B, pos, with_labels=False)
        plt.show()
```

Example output:

Part #3 Created a sorted ordered series and identified transition indexes between before and after bankruptcy date

```
In [1703]: len(orderedbyDate)
Out[1703]: 1601

In [1704]: orderedbyDate[437:439]
Out[1704]:
557   2001-11-30
855   2001-12-03
Name: True, dtype: datetime64[ns]
```

Extra credit output (undirected graph)