CS 851: Assignment #4

Due on Friday, MAY 1, 2015

DR NELSON 4:20pm

VICTOR NWALA

Contents

Problem 1

Using the pages from A3 that boilerpipe successfully processed, download those representations again & reprocess them with boilerpipe.
Time(A3) (Date Old files were downloaded) = 04/1/2015
Time(A4) (Date New files were downloaded) = 04/22/2015
Time difference = 21 days

Listing 1: Script To Download Page For New Files

```
import hashlib
from hashlib import md5
import os

fh = open("sample.txt",'r')

for line in fh:
     url=line
     url=url.replace('\n','')

     def computeMD5hash(message):
          m = hashlib.md5()
          m.update(message)
          return m.hexdigest()


     hashMessage = computeMD5hash(url)


     os.system("lynx -dump -force_html " + url+ "  > /home/vnwala/NTEXT/" +
         hashMessage  +".processed"+ ".txt ")
```

Listing 2: Script To Calculate Jaccard Index Between File Pairs (In This Case 3-ngram For Exam)

```
import collections
import itertools
import numpy as np
from sklearn.metrics import jaccard_similarity_score
import glob
import os

def find_ngrams(input_list, n):
  return zip(*[input_list[i:] for i in range(n)])


def jack(a,b):
    x=a.split()
    y=b.split()
    k=float(len(list(set(x)&set(y))))/float(len(list(set(x) | set(y))))
    return k

path1 = '/home/vnwala/NTEXT/'

path2 = '/home/vnwala/TEXT/'
```

Problem 1 continued on next page. . .

```
for filename1 in glob.glob(os.path.join(path1, '*.txt')):
    for filename2 in glob.glob(os.path.join(path2, '*.txt')):
     filename1.strip('/home/vnwala/NTEXT/')
     filename2.strip('/home/vnwala/TEXT/')
     if filename1.strip('/home/vnwala/NTEXT/') == filename2.strip('/home/vnwala/
         TEXT/'):
          infile = open(filename1)
          words = collections.Counter()

          array = []
          for line in infile:
                      words.update(line.split())


          for word, count in words.iteritems():
                      array.append(word)



          infile = open(filename2)
          words = collections.Counter()
          array2 = []
          for line in infile:
                      words.update(line.split())

          for word, count in words.iteritems():
                      array2.append(word)
          array = find_ngrams(array,3)
          array2 = find_ngrams(array2,3)
          index  = jack(str(array),str(array2))
          print index
          saveFile = open("3gram_jacc.txt",'a')
          saveFile.write(str(index) + '\n')
          saveFile.close()
```

Figure 1: ECDF OF JACCARD INDEX OF UNIGRAMS

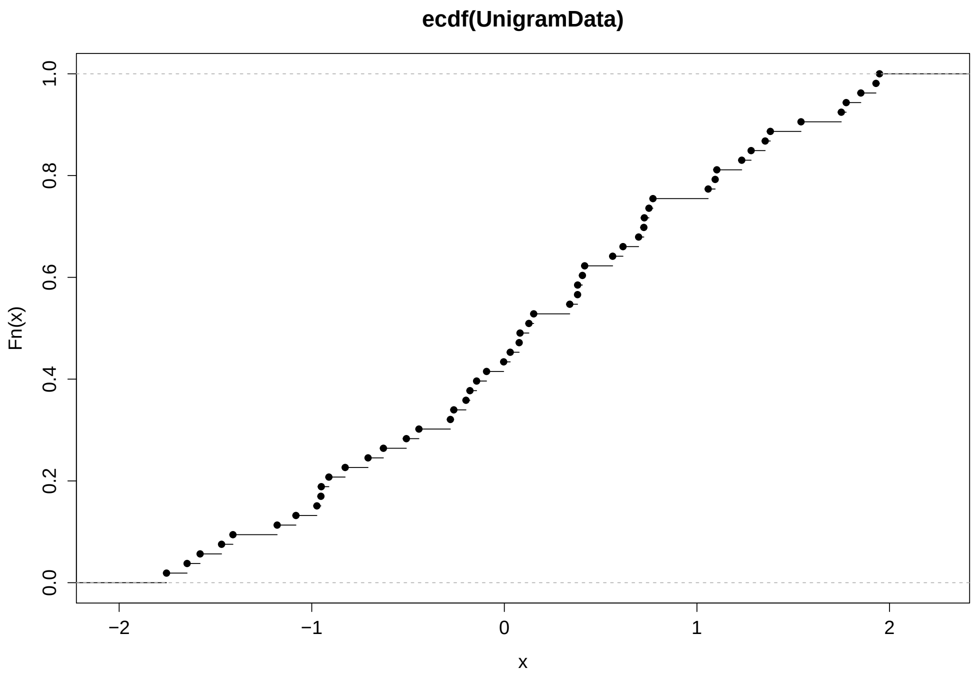
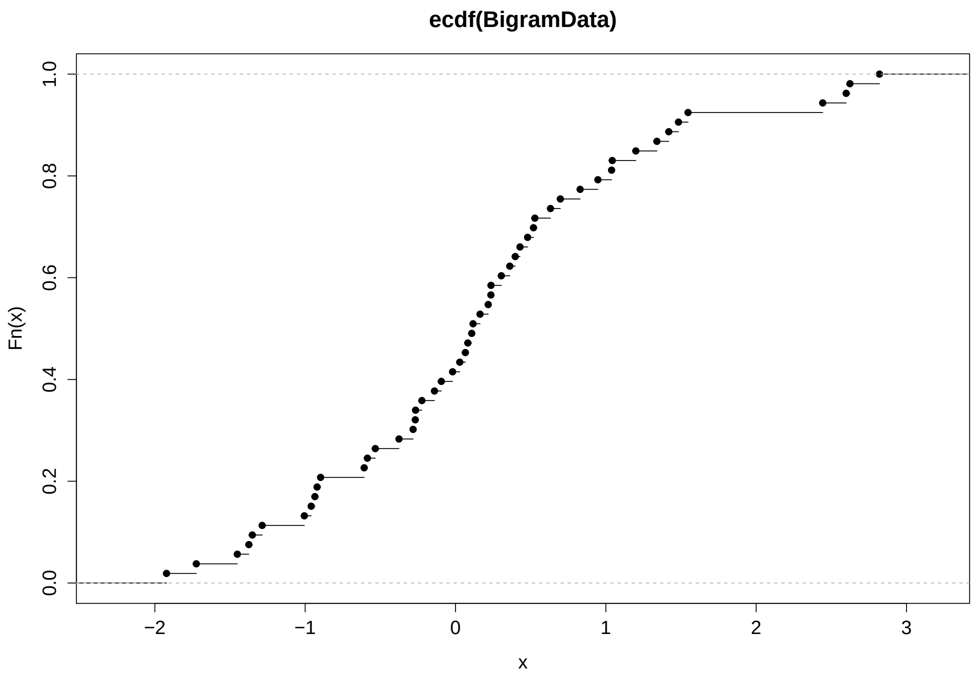
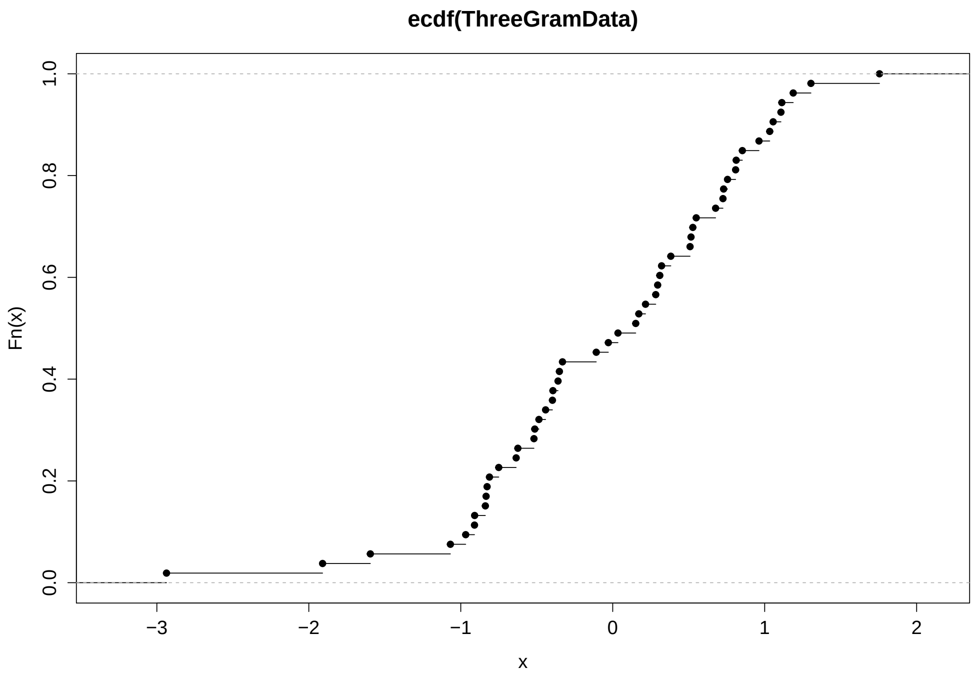


Figure 2: ECDF OF JACCARD INDEX OF BIGRAMS



Problem 1 continued on next page. . .

Figure 3: ECDF OF JACCARD INDEX OF 3-GRAMS



Problem 2

Using the pages from Q1 (A4), download all TimeMaps (including TimeMaps with 404 responses, i.e. empty or null TimeMaps)

Listing 3: Script To Download TimeMapsPages and Count Mementos(It is used to count all mementos now.)

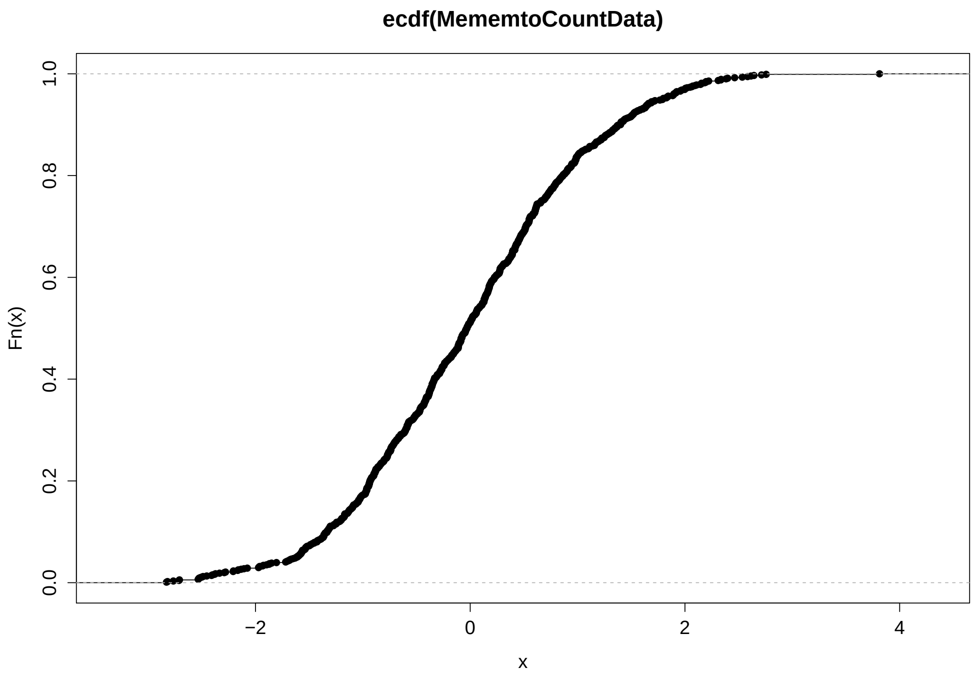
```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from getConfig import getConfigParameters
import commands
import time
import datetime
import sys
import argparse, os
import subprocess
import hashlib
import tldextract
import urlparse
import glob
import json
import requests


globalMementoUrlDateTimeDelimeter = "*+*+*"

def getMementosPages(url):
    pages = []
    url = url.strip()
    if(len(url)>0):

        firstChoiceAggregator = getConfigParameters('mementoAggregator')
        timemapPrefix = firstChoiceAggregator + url
        #timemapPrefix = 'http://mementoproxy.cs.odu.edu/aggr/timemap/link/1/' +
            url

        '''
            The CS memento aggregator payload format:
                [memento, ..., memento, timemap1]; timemap1 points to next
                    page
            The LANL memento aggregator payload format:
                1. [timemap1, ..., timemapN]; timemapX points to mementos list
                2. [memento1, ..., mementoN]; for small payloads
            For LANL Aggregator: The reason the link format is used after
                retrieving the payload
                                  with json format is due to the fact that
                                      the underlying code is based
                                  on the link format structure. json format
                                      was not always the norm
        '''



        #select an aggregator - start
        aggregatorSelector = ''

        co = 'curl --silent -I ' + timemapPrefix
        head = commands.getoutput(co)

        indexOfFirstNewLine = head.find('\n')
        if( indexOfFirstNewLine > -1 ):

            if( head[:indexOfFirstNewLine].split(' ')[1] != '200' ):
                firstChoiceAggregator = getConfigParameters('
                    latentMementoAggregator')
                timemapPrefix = firstChoiceAggregator + url

        if( firstChoiceAggregator.find('cs.odu.edu') > -1 ):
            aggregatorSelector = 'CS'
        else:
            aggregatorSelector = 'LANL'

        print '...using aggregator:', aggregatorSelector
        #select an aggregator - end

        #CS aggregator
        if( aggregatorSelector == 'CS' ):
            while( True ):
                #old: co = 'curl --silent ' + timemapPrefix
                #old: page = commands.getoutput(co)


                page = ''
                r = requests.get(timemapPrefix)
                print 'status code:', r.status_code
                if( r.status_code == 200 ):
                    page = r.text

                pages.append(page)
                indexOfRelTimemapMarker = page.rfind('>;rel="timemap"')

                if( indexOfRelTimemapMarker == -1 ):
                    break
                else:
                    #retrieve next timemap for next page of mementos e.g retrieve url from <http://mementoproxy.cs.odu.edu/aggr/timemap/link/10001/http://www.cnn.com>;rel="timemap"
                    i = indexOfRelTimemapMarker -1
                    timemapPrefix = ''
                    while( i > -1 ):
                        if(page[i] != '<'):
                            timemapPrefix = page[i] + timemapPrefix
                        else:
                            break
                        i = i - 1
        else:
            #LANL Aggregator
            #old: co = 'curl --silent ' + timemapPrefix
            #old: page = commands.getoutput(co)

            page = ''
            r = requests.get(timemapPrefix)
            if( r.status_code == 200 ):
                page = r.text

            try:
                payload = json.loads(page)

                if 'timemap_index' in payload:

                    for timemap in payload['timemap_index']:

                        timemapLink = timemap['uri'].replace('/timemap/json/', '/timemap/link/')
                        #old: co = 'curl --silent ' + timemapLink
                        #old: page = commands.getoutput(co)
                        #old: pages.append(page)
                        r = requests.get(timemapLink)
                        if( r.status_code == 200 ):
                            pages.append(r.text)
```

Problem 2 continued on next page. . .

```
                elif 'mementos' in payload:
                    #untested block
                    timemapLink = payload['timemap_uri']['json_format'].replace('/timemap/json/', '/timemap/link/')
                    #old: co = 'curl --silent ' + timemapLink
                    #old: page = commands.getoutput(co)
                    #old: pages.append(page)

                    print 'timemap:', timemapLink
                    r = requests.get(timemapLink)
                    if( r.status_code == 200 ):
                        pages.append(r.text)



            except:
                exc_type, exc_obj, exc_tb = sys.exc_info()
                fname = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
                print(fname, exc_tb.tb_lineno, sys.exc_info() )



    return pages


def getItemGivenSignature(page):

    listOfItems = []
    if( len(page) > 0 ):
        page = page.splitlines()
        for line in page:
            if(line.find('memento";') != -1):
                #uriRelDateTime: ['<http://www.webcitation.org/64ta04WpM>', ' rel="first memento"', ' datetime="Mon, 23 Jan 2012 02:01:29 GMT",']
                uriRelDateTime = line.split(';')
                if( len(uriRelDateTime) > 2 ):
                    if( uriRelDateTime[0].find('://') != -1 ):
                        if( uriRelDateTime[2].find('datetime="') != -1 ):


                            uri = ''
                            uri = uriRelDateTime[0].split('<')
                            #print uri
                            if( len(uri) > 1 ):
                                uri = uri[1].replace('>', '')
                                uri = uri.strip()

                            datetime = ''
                            datetime = uriRelDateTime[2].split('"')
                            if( len(datetime) > 1 ):
                                datetime = datetime[1]
```

```
                                        if( len(uri) != 0 and len(datetime) != 0 ):
                                            #print uri, '---', datetime
                                            listOfItems.append(uri +
                                               globalMementoUrlDateTimeDelimeter +
                                               datetime)

     return listOfItems

fh = open("NonDup.txt",'r')
count = 0
count1 = 0
for line in fh:
     url=line
     url=url.replace('\n','')

     print "...getting timemaps pages"
     pages = getMementosPages(url)
     print "...done getting timemaps pages"
     #saveFile = open("/home/vnwala/TFile/"+str(count1)+".txt",'a')
     #saveFile.write(str(pages) + '\n')
     #count1 = count1 + 1
     #pages has all timemaps
     for i in range(0,len(pages)):
          mementos = getItemGivenSignature(pages[i])
          print mementos
          count += len(mementos)
          #print 'mementos:', mementos[0]
          saveFile = open("/home/vnwala/TFile/"+str(count1)+".txt",'a')
          for mementos in range(0,len(mementos)):

          saveFile.write(str(mementos) + '\n')
     print 'total count of mementos:', count
     saveFile1 = open("count.txt",'a')
     saveFile1.write(str(count) + '\n')
     saveFile1.close()
     count1 = count1 + 1
     count = 0
```

Figure 4: ECDF OF Total Memento Count



Problem 3

Using 20 links that have TimeMaps With greater than or = 20 mementos Have existed greater than or = 2 years (i.e., Memento-Datetime of first memento is April XX, 2013 or older) Note: select from Q1/Q2 links, else choose them by hand For each link, create a graph that shows Jaccard Distance, relative to the first memento, through time x-axis: continuous time, y-axis: Jaccard Distance relative to the first memento

Listing 4: Script For Q3

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from getConfig import getConfigParameters
import commands
import time
from datetime import datetime
import sys
import argparse, os
import subprocess
import hashlib
import tldextract
import urlparse
import glob
import json
import requests
import urllib2
import justext
import collections
import itertools
```

Problem 3 continued on next page. . .

```
import numpy as np
from sklearn.metrics import jaccard_similarity_score
import glob



globalMementoUrlDateTimeDelimeter = "*+*+*"


def jack(a,b):
    x=a.split()
    y=b.split()
    k=float(len(list(set(x)&set(y))))/float(len(list(set(x) | set(y))))
    return k




def getUriText(url):
     array = []
     try:
          response = requests.get(url)
          code = str(response.status_code)
          if code == '200':
               paragraphs = justext.justext(response.content, justext.get_stoplist
                   ("English"))
               for paragraph in paragraphs:
                    if not paragraph.is_boilerplate:
                         line = paragraph.text.encode('utf-8')
                         if line != "":
                              words = collections.Counter()
                              words.update(line.split())
               for word, count in words.iteritems():
                    array.append(word)
          return array
     except Exception as e:
          print str(e)
```

Problem 3 continued on next page...

```
def getMementosPages(url):

     pages = []
     url = url.strip()
     if(len(url)>0):

          firstChoiceAggregator = getConfigParameters('mementoAggregator')
          timemapPrefix = firstChoiceAggregator + url
          #timemapPrefix = 'http://mementoproxy.cs.odu.edu/aggr/timemap/link/1/' +
               url

          '''
               The CS memento aggregator payload format:
                    [memento, ..., memento, timemap1]; timemap1 points to next
                        page
               The LANL memento aggregator payload format:
                    1. [timemap1, ..., timemapN]; timemapX points to mementos list
                    2. [memento1, ..., mementoN]; for small payloads
               For LANL Aggregator: The reason the link format is used after
                   retrieving the payload
                                        with json format is due to the fact that
                                            the underlying code is based
                                        on the link format structure. json format
                                             was not always the norm
          '''



          #select an aggregator - start
          aggregatorSelector = ''

          co = 'curl --silent -I ' + timemapPrefix
          head = commands.getoutput(co)

          indexOfFirstNewLine = head.find('\n')
          if( indexOfFirstNewLine > -1 ):

               if( head[:indexOfFirstNewLine].split(' ')[1] != '200' ):
                    firstChoiceAggregator = getConfigParameters('
                        latentMementoAggregator')
                    timemapPrefix = firstChoiceAggregator + url

          if( firstChoiceAggregator.find('cs.odu.edu') > -1 ):
               aggregatorSelector = 'CS'
```

Problem 3 continued on next page. . .

```
        else:
            aggregatorSelector = 'LANL'

        print '...using aggregator:', aggregatorSelector
        #select an aggregator - end

        #CS aggregator
        if( aggregatorSelector == 'CS' ):
            while( True ):
                #old: co = 'curl --silent ' + timemapPrefix
                #old: page = commands.getoutput(co)


                page = ''
                r = requests.get(timemapPrefix)
                print 'status code:', r.status_code
                if( r.status_code == 200 ):
                    page = r.text

                pages.append(page)
                indexOfRelTimemapMarker = page.rfind('>;rel="timemap"')

                if( indexOfRelTimemapMarker == -1 ):
                    break
                else:
                    #retrieve next timemap for next page of mementos e.g
                        retrieve url from <http://mementoproxy.cs.odu.edu/
                        aggr/timemap/link/10001/http://www.cnn.com>;rel="
                        timemap"
                    i = indexOfRelTimemapMarker -1
                    timemapPrefix = ''
                    while( i > -1 ):
                        if(page[i] != '<'):
                            timemapPrefix = page[i] + timemapPrefix
                        else:
                            break
                        i = i - 1
        else:
            #LANL Aggregator
            #old: co = 'curl --silent ' + timemapPrefix
            #old: page = commands.getoutput(co)

            page = ''
            r = requests.get(timemapPrefix)
            if( r.status_code == 200 ):
                page = r.text

            try:
                payload = json.loads(page)

                if 'timemap_index' in payload:

                    for timemap in payload['timemap_index']:
```

Problem 3 continued on next page. . .

```
                              timemapLink = timemap['uri'].replace('/timemap/json/
                                  ', '/timemap/link/')
                              #old: co = 'curl --silent ' + timemapLink
                              #old: page = commands.getoutput(co)
                              #old: pages.append(page)
                              r = requests.get(timemapLink)
                              if( r.status_code == 200 ):
                                   pages.append(r.text)

                    elif 'mementos' in payload:
                         #untested block
                         timemapLink = payload['timemap_uri']['json_format'].
                             replace('/timemap/json/', '/timemap/link/')
                         #old: co = 'curl --silent ' + timemapLink
                         #old: page = commands.getoutput(co)
                         #old: pages.append(page)

                         print 'timemap:', timemapLink
                         r = requests.get(timemapLink)
                         if( r.status_code == 200 ):
                              pages.append(r.text)



               except:
                    exc_type, exc_obj, exc_tb = sys.exc_info()
                    fname = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
                    print(fname, exc_tb.tb_lineno, sys.exc_info() )



     return pages


def getItemGivenSignature(page):

     listOfItems = []
     if( len(page) > 0 ):
          page = page.splitlines()
          for line in page:
               if(line.find('memento";') != -1):
                    #uriRelDateTime: ['<http://www.webcitation.org/64ta04WpM>', '
                        rel="first memento"', ' datetime="Mon, 23 Jan 2012
                        02:01:29 GMT",']
                    uriRelDateTime = line.split(';')
                    if( len(uriRelDateTime) > 2 ):
                         if( uriRelDateTime[0].find('://') != -1 ):
                              if( uriRelDateTime[2].find('datetime="') != -1 ):


                                   uri = ''
                                   uri = uriRelDateTime[0].split('<')
                                   #print uri
                                   if( len(uri) > 1 ):
                                        uri = uri[1].replace('>', '')
                                        uri = uri.strip()

                                   datetimeValue = ''
                                   datetimeValue = uriRelDateTime[2].split('"')
                                   if( len(datetimeValue) > 1 ):
                                        datetimeValue = datetimeValue[1]

                                   if( len(uri) != 0 and len(datetimeValue) != 0 )
                                       :
                                        #print uri, '---', datetime

                                        #print uri
                                        getUriText(uri)


                                        datetimeValue = datetime.strptime(
                                            datetimeValue, '%a, %d %b %Y %H:%M:%S
                                            %Z')

                                        abcd = dict()
                                        abcd['uri'] = uri
                                        abcd['date'] = datetimeValue
                                        listOfItems.append(abcd)

     return listOfItems


fh = open("mem.txt",'r')
count2 = 0
count1 = 0
for line in fh:
     url=line
     url=url.replace('\n','')

     print "...getting timemaps pages"
     pages = getMementosPages(url)
     print "...done getting timemaps pages"
     #saveFile = open("/home/vnwala/TFile/"+str(count1)+".txt",'a')
     #saveFile.write(str(pages) + '\n')
     #count1 = count1 + 1
     #pages has all timemaps
     array = []
     array2 = []
     abcd = []
     for i in range(0,len(pages)):
          abcd += getItemGivenSignature(pages[i])

     abcd2 = sorted(abcd, key=lambda k: k['date'])

     uri_first = str(abcd2[0]['uri'])
```

Problem 3 continued on next page. . .

```
        print uri_first

        array = getUriText(uri_first)
        #print len(array)
        for i in range(1,len(abcd2)):
             uri_next = str(abcd2[i]['uri'])
             array2 = getUriText(uri_next)
             if (array) is not None and (array2) is not None:
                  if len(array) != 0 and len(array2) != 0:
                       index  = jack(str(array),str(array2))
                       saveFile = open("/home/vnwala/JAC/"+str(count2)+".txt",'a')
                       print  index
                       saveFile.write(str(index))
                       saveFile.write('\n')
                       saveFile.close()
             else:
                  print "empty"
             array = array2
        count2 = count2 + 1
```

Figure 5: JACCARD INDEX Assuming Contant Time for First URI

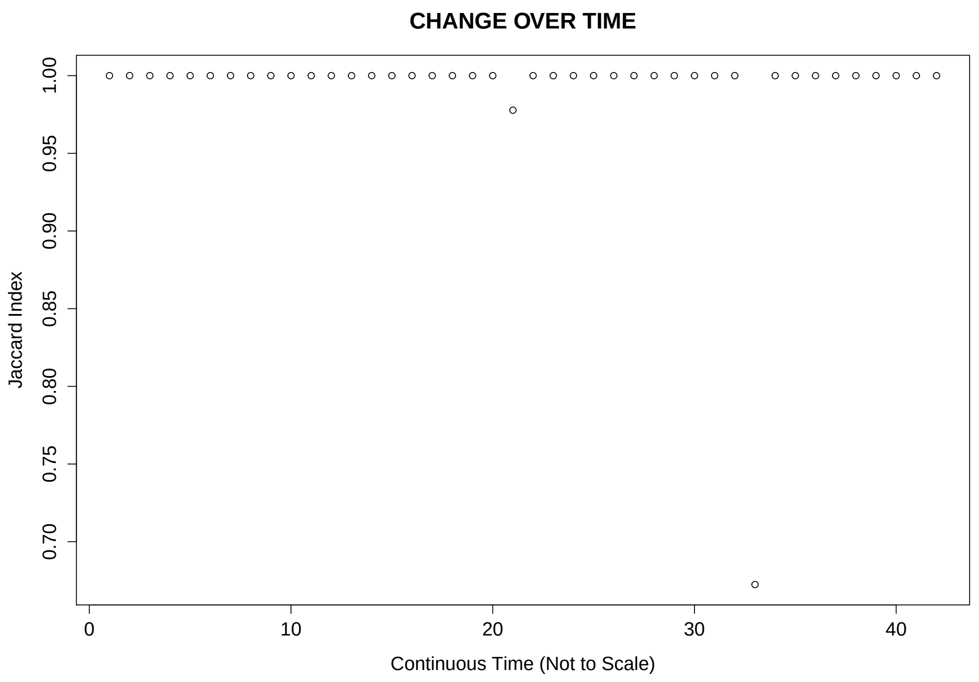


Figure 6: JACCARD INDEX Assuming Contant Time for Second URI

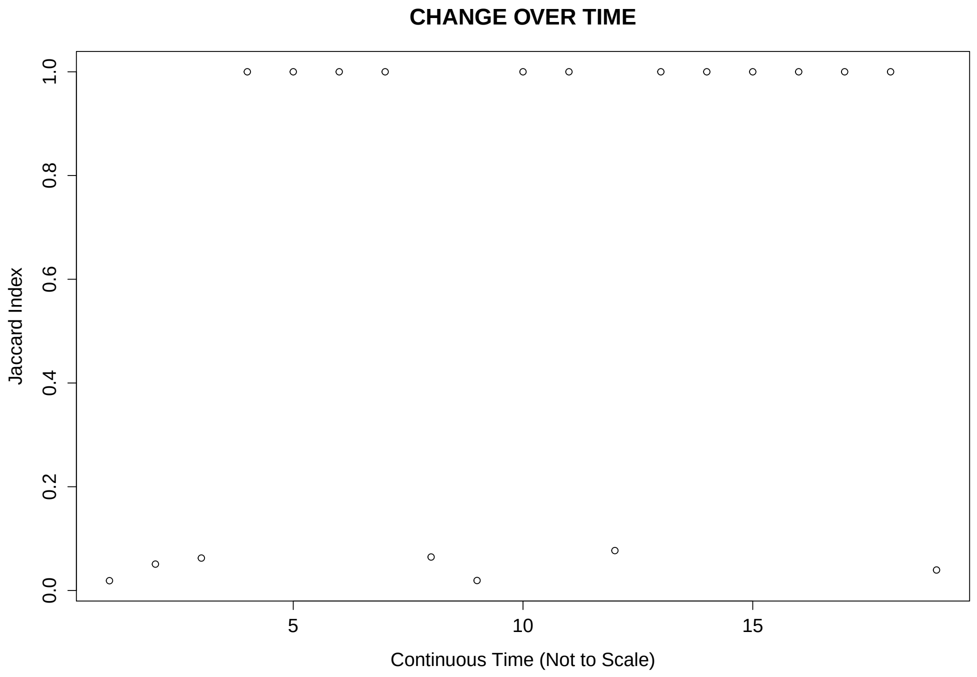


Figure 7: JACCARD INDEX Assuming Contant Time for Third URI

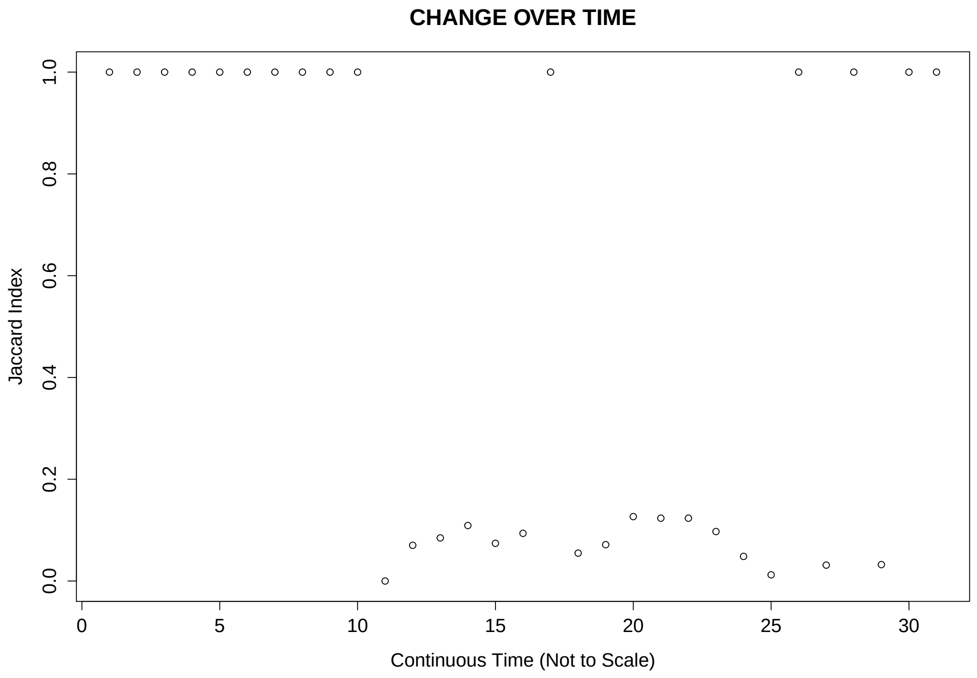


Figure 8: JACCARD INDEX Assuming Contant Time for Fourth URI

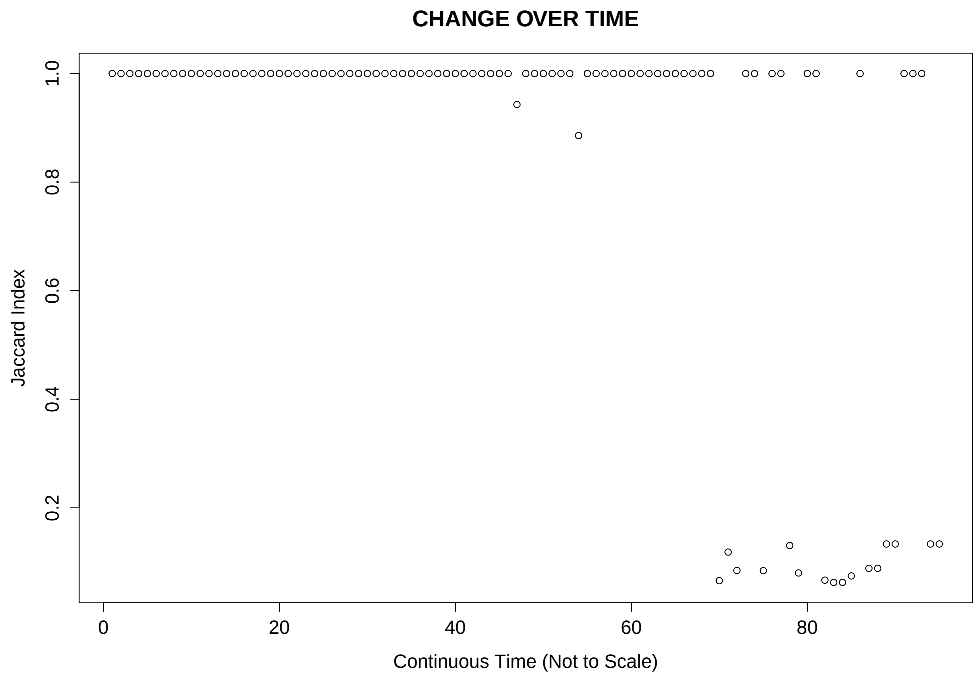


Figure 9: JACCARD INDEX Assuming Contant Time for Fifth URI

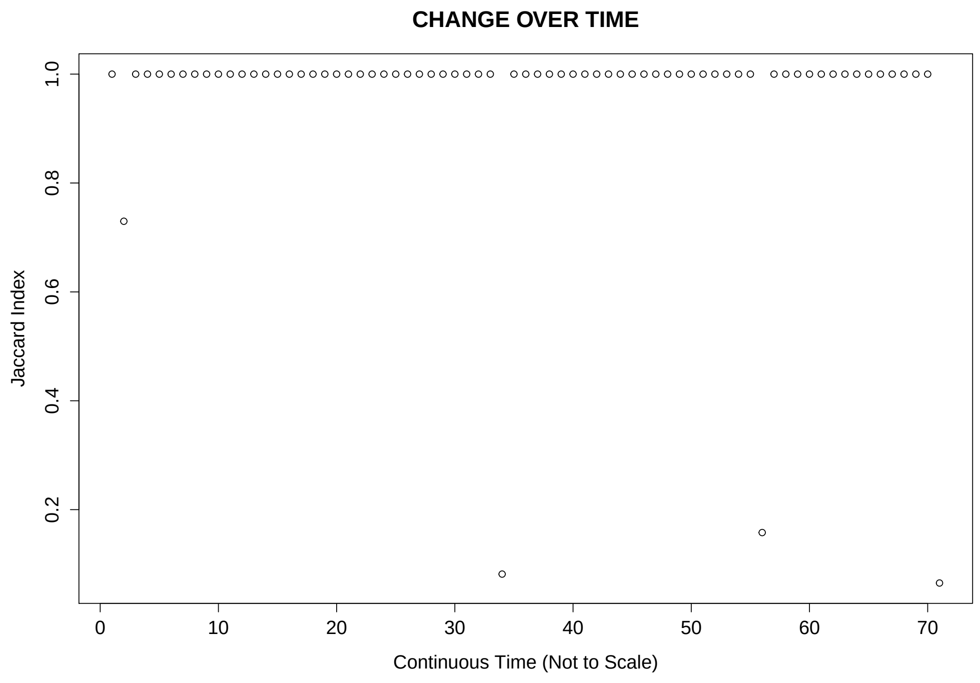


Figure 10: JACCARD INDEX Assuming Contant Time for Sixth URI

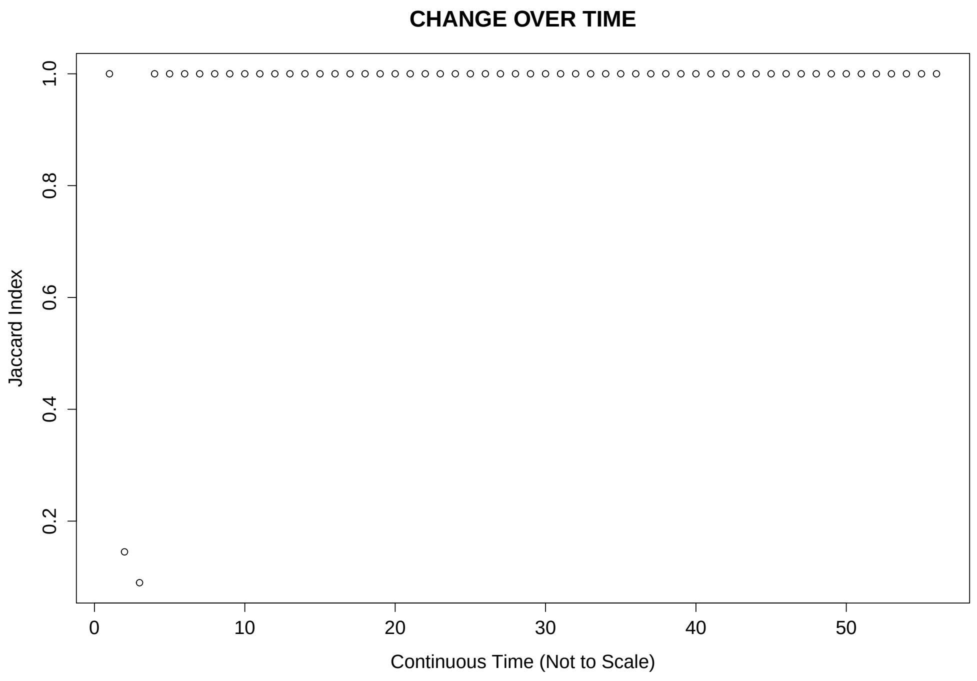
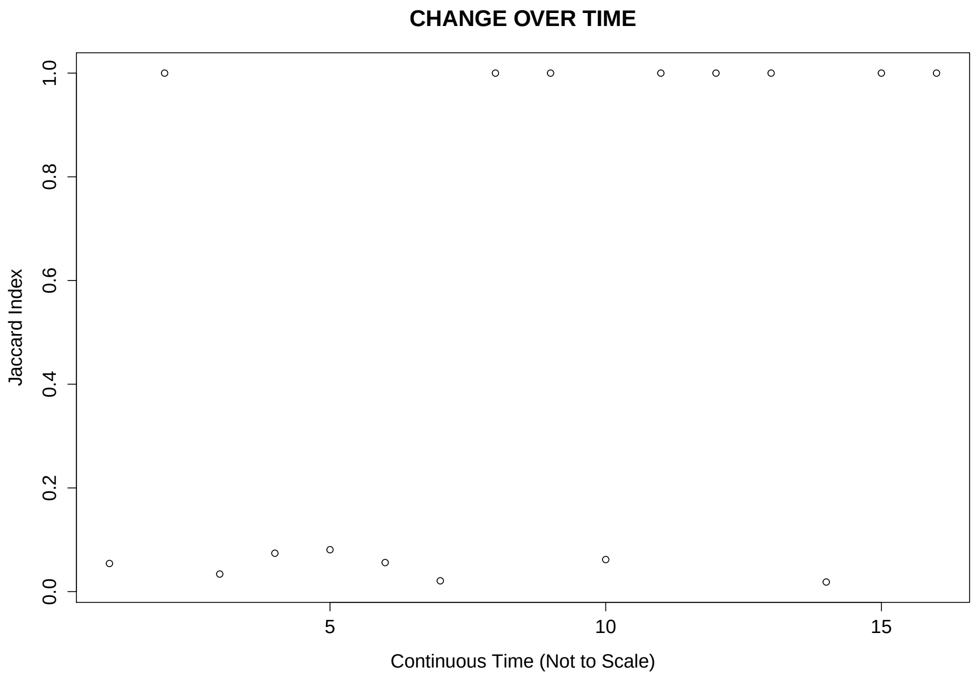


Figure 11: JACCARD INDEX Assuming Contant Time for Seventh URI



Problem 3 continued on next page. . .

Figure 12: JACCARD INDEX Assuming Contant Time for Eight URI

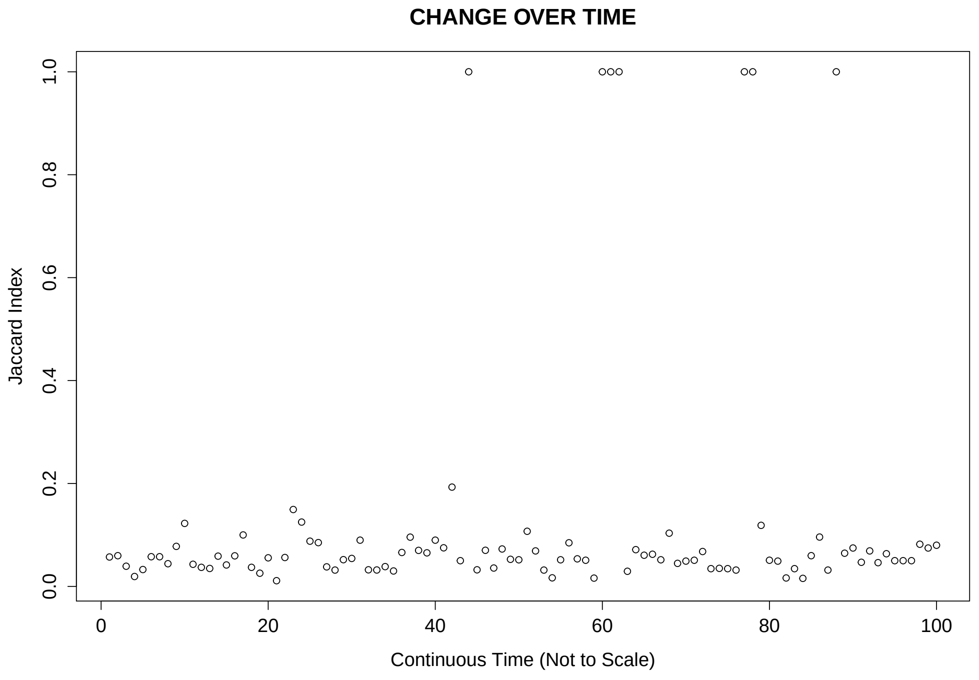
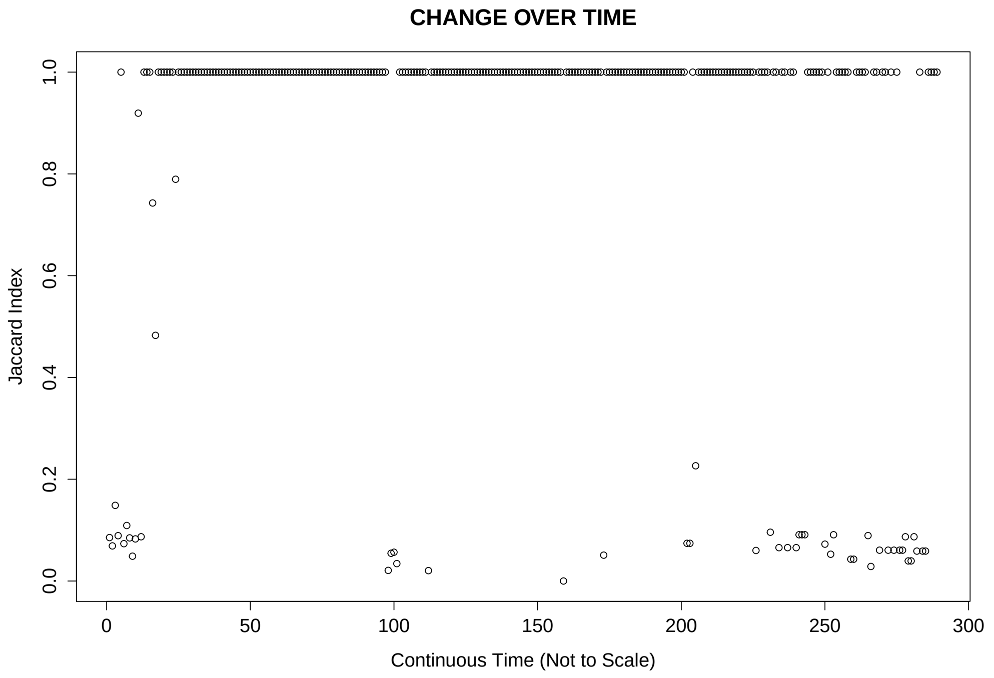


Figure 13: JACCARD INDEX Assuming Contant Time for Ninth URI



Problem 3 continued on next page. . .

Figure 14: JACCARD INDEX Assuming Contant Time for Tenth URI

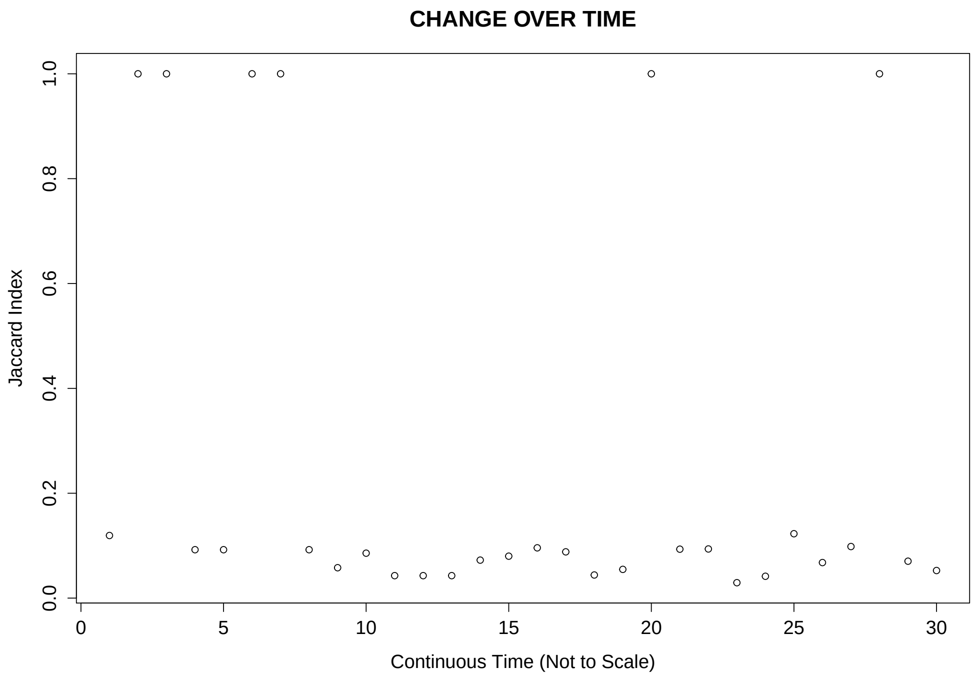
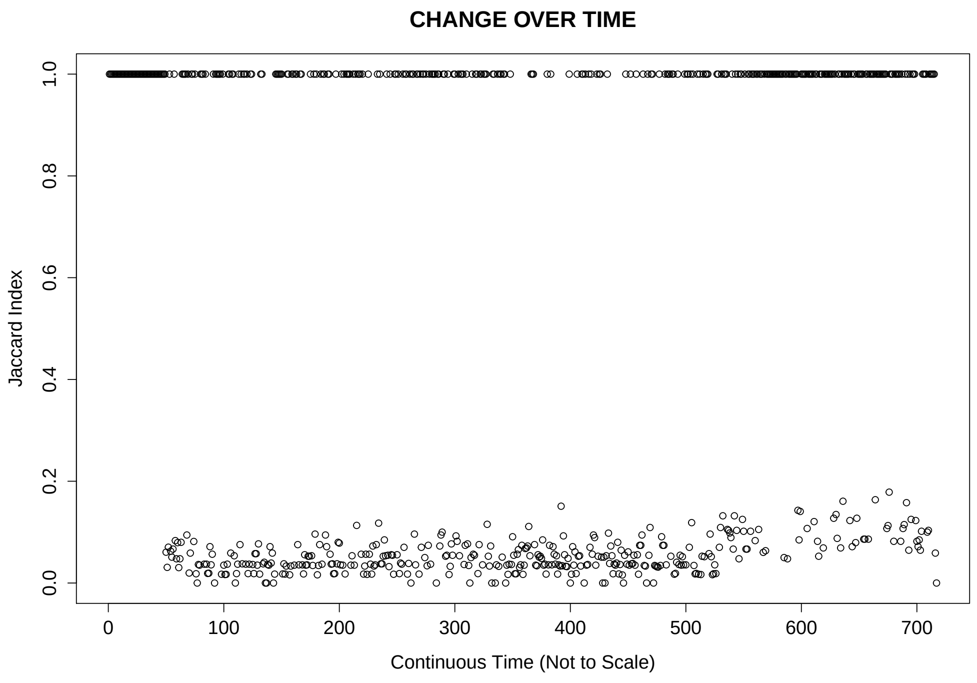


Figure 15: JACCARD INDEX Assuming Contant Time for Eleventh URI



Problem 3 continued on next page. . .

Figure 16: JACCARD INDEX Assuming Contant Time for Twelveth URI

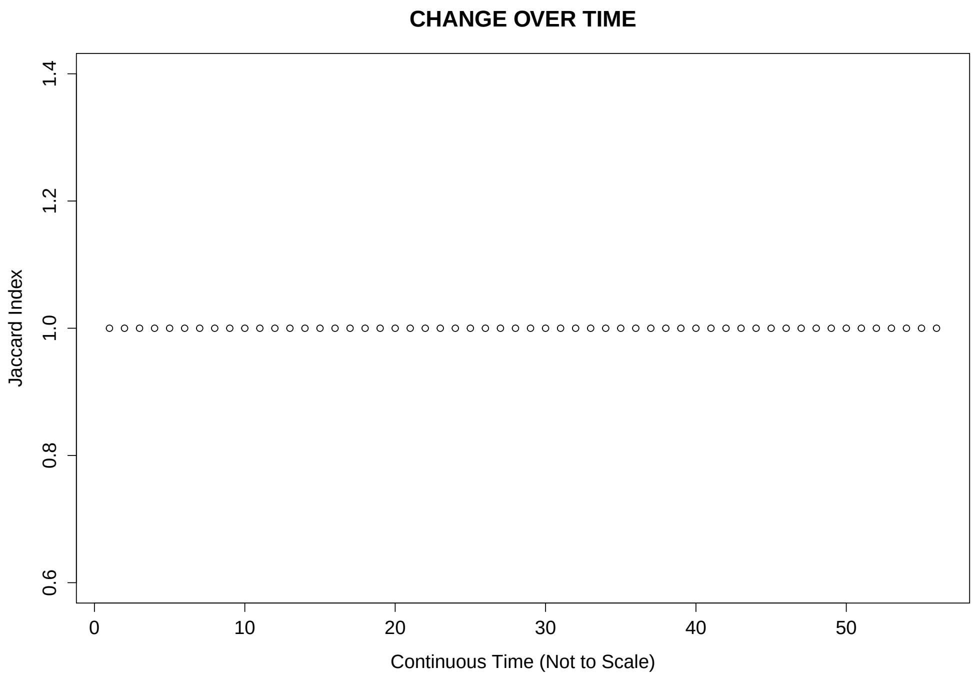


Figure 17: JACCARD INDEX Assuming Contant Time for Thirteenth URI

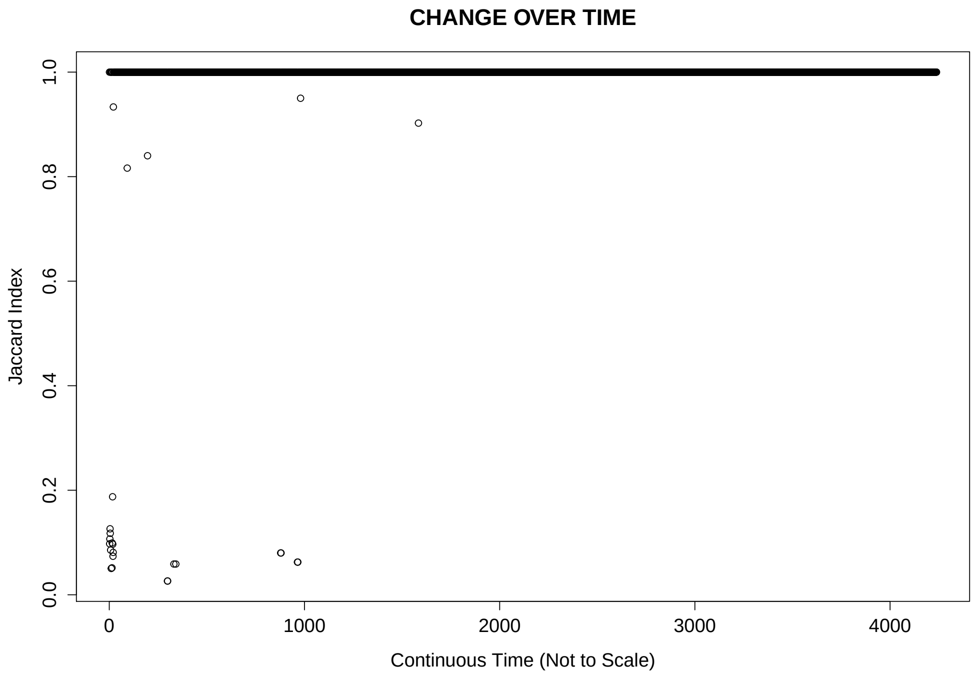


Figure 18: JACCARD INDEX Assuming Contant Time for Fourteenth URI

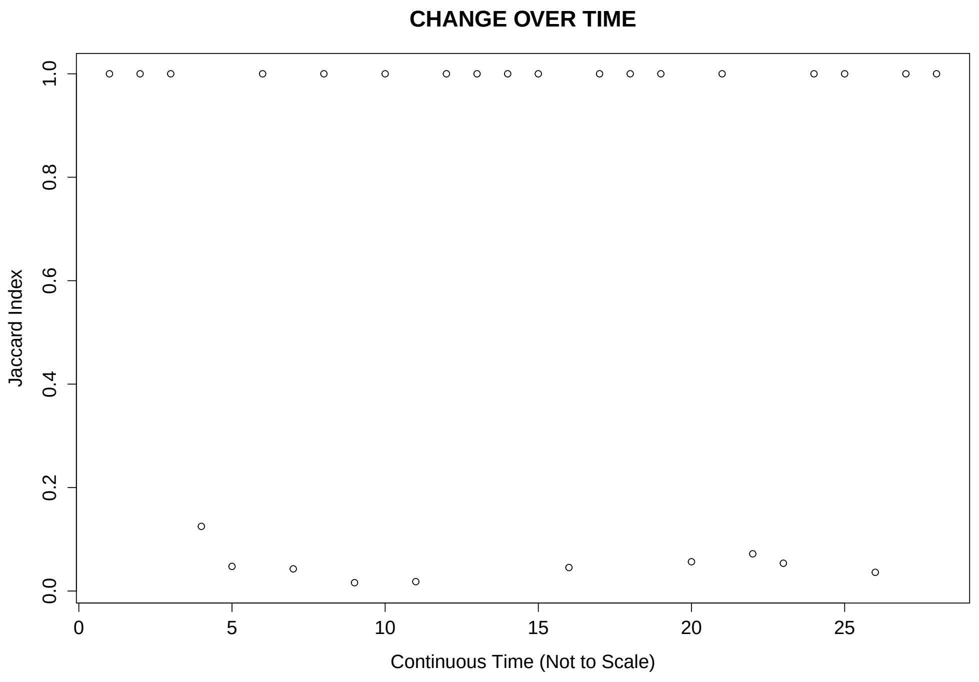
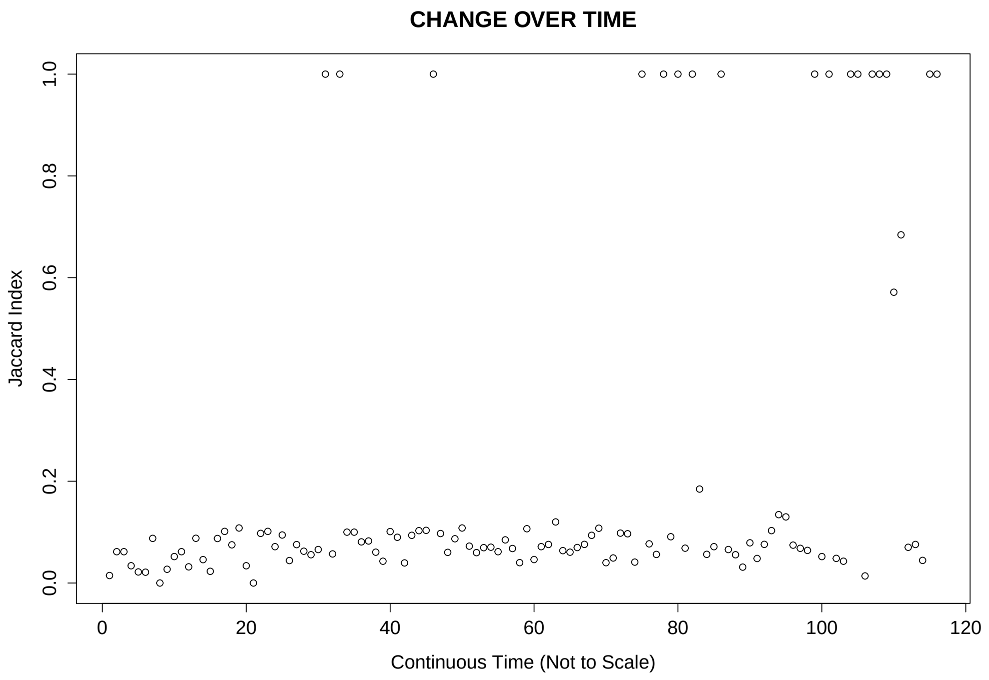


Figure 17: JACCARD INDEX Assuming Contant Time for Fifteenth URI



CONCLUSION

For question 3 I was able to use 15 of 20 links that fulfilled the criteria, I also assumed a constant time and the jaccard index is placed in the same sequence as they occured, my intention was not to show the variations with respect to real(original) time of occurrence.

References

[1] anwala. timelapse.py. "https://github.com/anwala/wdill/blob/master/timelapse.py", 2015.

[2] Locally Optimal. Executable Python Scripts via Entry Points. "http://locallyoptimal.com/blog/2013/01/20/elegant-n-gram-generation-in-python/", JAN 20TH 2015.

[3] R-bloggers. Exploratory data analysis: 2 ways of plotting empirical cumulative distribution functions in r. "http://www.r-bloggers.com/exploratory-data-analysis-2-ways-of-plotting-empirical-cumulative-distribution-functions-in-r", 2015.

Table 1: Jaccard Index Differences

LINK	1-GramJaccardIndex	2-GramJaccardIndex	3-GramJaccardIndex
1	0.354793561931	0.354804831087	0.354816112084
2	1.0	1.0	1.0
3	1.0	1.0	1.0
4	0.209553158706	0.209768637532	0.209984559959