Dynamic Network Analysis 00

July 14, 2016

1 Table of Contents

2 Introduction

2.1 Dynamic Network Analysis of Enron Email Network Data

I use the Enron email network data from John Hopkins which has time, sender and receiver pair format data.

2.2 Data Preprocessing

From the JHU data, I have done the following in Excel: - The first column represents seconds elapsed since 1 January 1970, so I convert this in to days - I then add these days to the date to get time stamps for all nodes - From the timestamps, I extract the year field - The network can be partitioned by year in a cumulative manner for DNA

2.3 Key Assumption

The key assumptio in this analysis is that the nodes can be appended to the original network at time, t0 with the new nodes from time, t+1.

3 Import Libraries

```
In [1]: import pandas as pd
        import numpy as np
        import networkx as nx
        import seaborn as sns
        import matplotlib.pyplot as plt
        import scipy as sc
        %matplotlib inline
        sns.set(style="whitegrid", color_codes=True, context='paper')
        import random
        random.seed(1111111111111)
        plt.rc('axes', grid=False, titlesize='large', labelsize='medium',labelweigh
        plt.rc('lines', linewidth=4)
        plt.rc('font',family='serif',size=12, serif='Georgia')
        plt.rc('figure', figsize = (15,6),titlesize='large',titleweight='heavy')
        plt.rc('grid',linewidth=3)
        sns.set_palette('husl')
        from scipy.signal import *
```

4 Import Data

```
In [2]: data = pd.read_excel("../Data/execs.email.linesnum.xlsx")

In [3]: data.head()
```

Out[3]:

	sec	to	from	date	year
0	315522000	24	153	1979-12-31 21:00:00	1979
1	315522000	24	153	1979-12-31 21:00:00	1979
2	315522000	29	29	1979-12-31 21:00:00	1979
3	315522000	29	29	1979-12-31 21:00:00	1979
4	315522000	29	29	1979-12-31 21:00:00	1979

```
In [4]: data.min()
```

```
Out[4]: sec                315522000
        to                         0
        from                       0
        date    1979-12-31 21:00:00
        year                    1979
        dtype: object
```

```
In [5]: data.max()
```

```
Out[5]: sec               1024688419
        to                       183
        from                     183
        date    2002-06-21 19:40:19
        year                    2002
        dtype: object
```

5 Data Partition

```
In [6]: #year = data['year'].unique()
        year = sorted(set(data['year']))
        year
```

```
Out[6]: [1979, 1998, 1999, 2000, 2001, 2002]
```

```
In [7]: sorted(set(data['year']))
```

```
Out[7]: [1979, 1998, 1999, 2000, 2001, 2002]
```

```
In [8]: data.drop(["sec", "date"], axis=1,inplace=True)

In [9]: data.head()
```

Out[9]:

	to	from	year
0	24	153	1979
1	24	153	1979
2	29	29	1979
3	29	29	1979
4	29	29	1979

5.1 Break data into years

```
In [10]: G0 = data[data["year"]==year[0]]
         G1 = data[data["year"]==year[1]]
         G2 = data[data["year"]==year[2]]
         G3 = data[data["year"]==year[3]]
         G4 = data[data["year"]==year[4]]
         G5 = data[data["year"]==year[5]]

In [11]: G1.size,G1.shape

Out[11]: (246, (82, 3))

In [12]: G2.size,G1.shape

Out[12]: (11145, (82, 3))

In [13]: G3.size,G1.shape

Out[13]: (132177, (82, 3))

In [14]: G3.size,G1.shape

Out[14]: (132177, (82, 3))

In [15]: G4.size,G1.shape

Out[15]: (206664, (82, 3))

In [16]: G5.size,G1.shape

Out[16]: (25473, (82, 3))

In [17]: G1.head()

Out[17]:      to   from  year
         174  114   169  1998
         175  114   169  1998
         176  114   123  1998
         177  114   123  1998
         178  114   123  1998

In [18]: G1.tail()

Out[18]:      to   from  year
         251  112    65  1998
         252  112   114  1998
         253  112   114  1998
         254  112   145  1998
         255  112   145  1998
```

```
In [19]: G2.head()
```

```
Out[19]:      to   from  year
          256  114    65  1999
          257  114    65  1999
          258  114   169  1999
          259  114   169  1999
          260  114   112  1999
```

```
In [20]: G3.head()
```

```
Out[20]:       to   from  year
          3971   82    51  2000
          3972   82    51  2000
          3973   82    51  2000
          3974   82    51  2000
          3975   82    51  2000
```

5.2 Create networks at different timesteps

5.2.1 t = 0

```
In [21]: G0_ = np.asarray(G0.ix[:,:2])
         Gt0 = nx.Graph()
         Gt0= nx.from_edgelist(G0_)
```

5.2.2 t =1

```
In [22]: G1_ = G1.ix[:,:2]
         G1_ = np.concatenate((G1_,G0_), axis=0)
         Gt1 = nx.Graph()
         Gt1= nx.from_edgelist(G1_)
```

5.2.3 t = 2

```
In [23]: G2_ = G2.ix[:,:2]
         G2_ = np.concatenate((G2_,G1_), axis=0)
         Gt2 = nx.Graph()
         Gt2= nx.from_edgelist(G2_)
```

5.2.4 t = 3

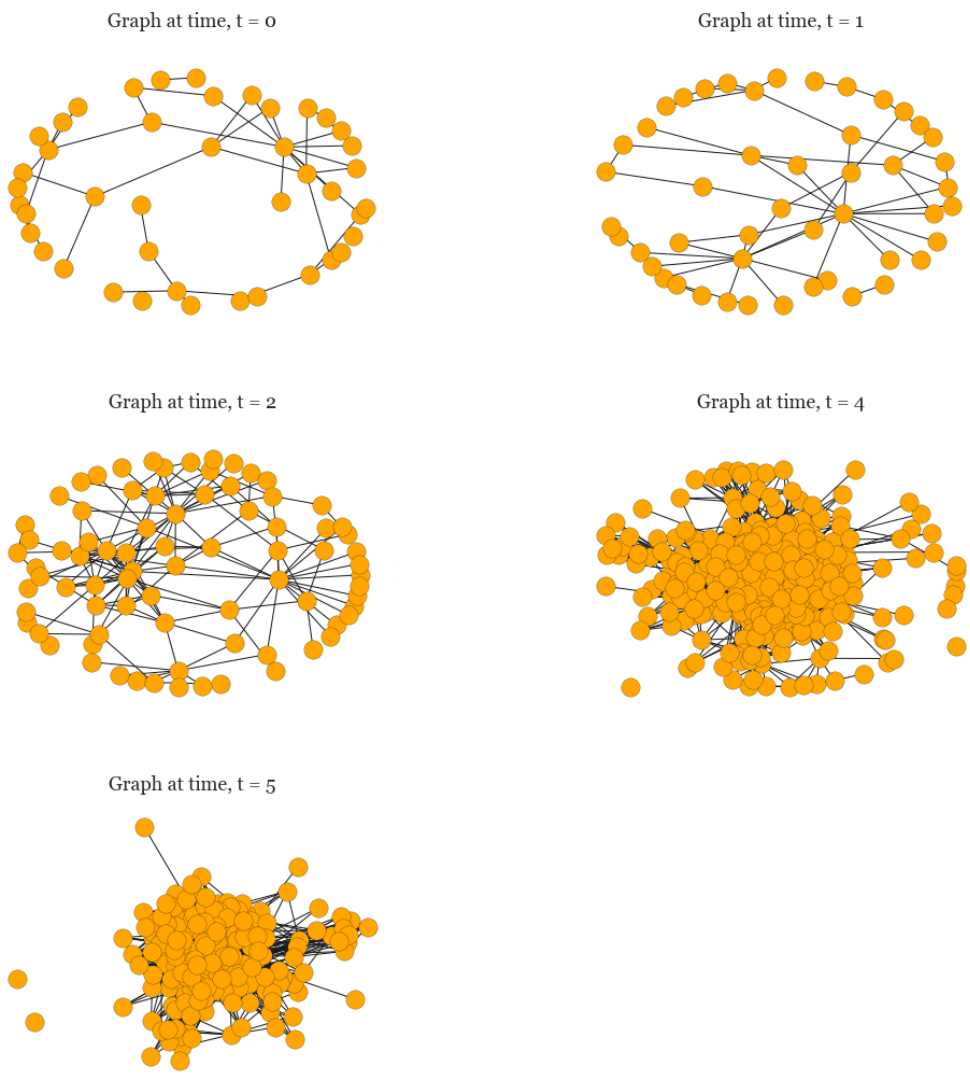
```
In [24]: G3_ = G3.ix[:,:2]
         G3_ = np.concatenate((G3_,G2_), axis=0)
         Gt3 = nx.Graph()
         Gt3= nx.from_edgelist(G3_)
```

5.2.5 t = 4

```
In [25]: G4_ = G4.ix[:,:2]
         G4_ = np.concatenate((G4_,G3_), axis=0)
         Gt4 = nx.Graph()
         Gt4= nx.from_edgelist(G4_)
```

5.2.6 t = 5

```
In [26]: G5_ = G5.ix[:,:2]
         G5_ = np.concatenate((G5_,G4_), axis=0)
         Gt5 = nx.Graph()
         Gt5= nx.from_edgelist(G5_)

In [27]: #Plot graphs together
         plt.figure(figsize=(18,18))
         plt.suptitle('Enron Email Dynamic Network', fontsize=24)
         plt.subplot(321)
         nx.draw_spring(Gt0, cmap=plt.cm.inferno, node_color='#FFA500')
         plt.title("Graph at time, t = 0", fontsize=18)

         plt.subplot(322)
         nx.draw_spring(Gt1, cmap=plt.cm.inferno, node_color='#FFA500')
         plt.title("Graph at time, t = 1", fontsize=18)

         plt.subplot(323)
         nx.draw_spring(Gt2, cmap=plt.cm.inferno, node_color='#FFA500')
         plt.title("Graph at time, t = 2", fontsize=18)

         plt.subplot(324)
         nx.draw_spring(Gt3, cmap=plt.cm.inferno, node_color='#FFA500')
         plt.title("Graph at time, t = 3", fontsize=18)

         plt.subplot(324)
         nx.draw_spring(Gt4, cmap=plt.cm.inferno, node_color='#FFA500')
         plt.title("Graph at time, t = 4", fontsize=18)

         plt.subplot(325)
         nx.draw_spring(Gt5, cmap=plt.cm.inferno, node_color='#FFA500')
         plt.title("Graph at time, t = 5", fontsize=18)

         plt.show()
```



6 Network Statistics

6.1 Centrality analysis without averaging

Define some helper functions here

```
In [28]: def get_cent(net):
             degC = nx.degree_centrality(net)
             cloC = nx.closeness_centrality(net)
             betC = nx.betweenness_centrality(net)
```

```
        eigC = nx.eigenvector_centrality_numpy(net)
        commCC = nx.communicability_centrality(net)
        katzC = nx.katz_centrality_numpy(net)
        loadC = nx.load_centrality(net)

        return [degC,cloC,betC,eigC,commCC,katzC, loadC]
```

In [29]:
```
def get_val(val):
    return sorted(set(val.values()))
```

In [30]:
```
def get_top_keys(dictionary, top):
    items = dictionary.items()
    items.sort(reverse=True, key=lambda x: x[1])
    return map(lambda x: x[0], items[:top])
```

In [31]:
```
def fft_sig(att):
    return sc.fft(get_val(att))

def hilbert_sig(att):
    return hilbert(get_val(att))
```

In [32]:
```
def rms(a, axis=None):
    from numpy import mean, sqrt,square
    rms = sqrt(mean(square(a), axis=axis))
    return rms

def nrms(a,b):
    nrms = rms(a-b)/(rms(a)+ rms(b))
    return nrms
```

In [33]:
```
def cossim(x,y):
    from numpy import dot, sqrt
    csim = dot(x,y) / sqrt(dot(x,x))*sqrt(dot(y,y))
    return csim
```

In [34]:
```
def pairwise_calc(df,func):
    val = []
    for x,y in df.iteritems():
        for z,y in df.iteritems():
            i = 0
            #print(y[i], y[i+1])
            val.append(func(y[i],y[i+1]))
            i=i+1

    return val[:df.shape[0]]
```

In [35]:
```
def avg_cent(cent):
    avg = sum(set(cent.values()))/len(cent)
    return avg
```

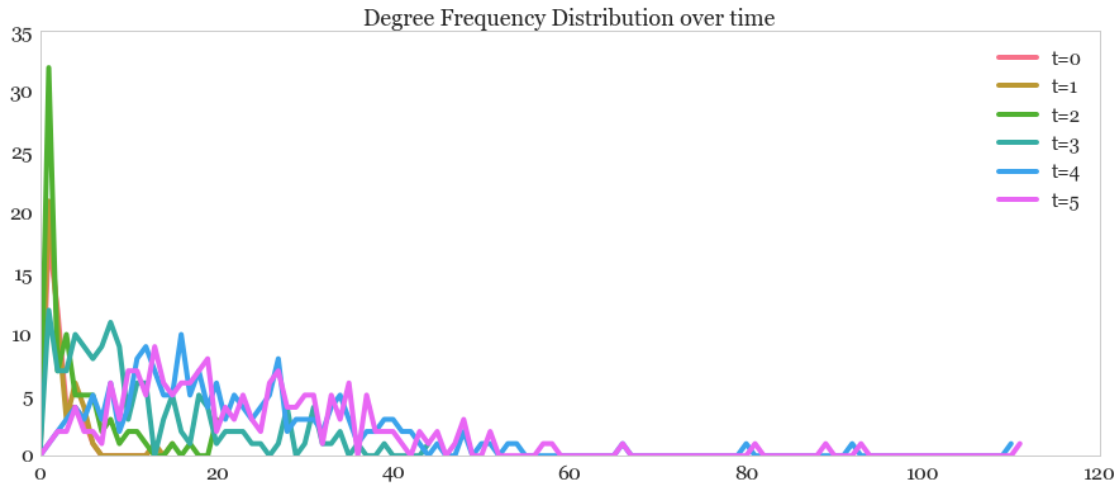
```
In [36]: def cal_stat(net):
             degC = nx.degree_centrality(net)
             cloC = nx.closeness_centrality(net)
             betC = nx.betweenness_centrality(net)
             eigC = nx.eigenvector_centrality_numpy(net)
             algC = nx.algebraic_connectivity(net)
             clustC = nx.average_clustering(net)
             commC = nx.communicability_centrality(net)
             katzC = nx.katz_centrality_numpy(net)
             loadC= nx.load_centrality(net)

             return [avg_cent(degC),avg_cent(cloC),\
                     avg_cent(betC),avg_cent(eigC),\
                     algC, clustC, avg_cent(commC),\
                     avg_cent(katzC), avg_cent(loadC)]
```

6.1.1 Calculate all centralities in one go

```
In [37]: degC0, cloC0, betC0, eigC0, commuC0, katzC0, loadC0 = get_cent(Gt0)
         degC1, cloC1, betC1, eigC1, commuC1, katzC1, loadC1 = get_cent(Gt1)
         degC2, cloC2, betC2, eigC2, commuC2, katzC2, loadC2 = get_cent(Gt2)
         degC3, cloC3, betC3, eigC3, commuC3, katzC3, loadC3 = get_cent(Gt3)
         degC4, cloC4, betC4, eigC4, commuC4, katzC4, loadC4 = get_cent(Gt4)
         degC5, cloC5, betC5, eigC5, commuC5, katzC5, loadC5 = get_cent(Gt5)
```

6.1.2 Degree Centrality

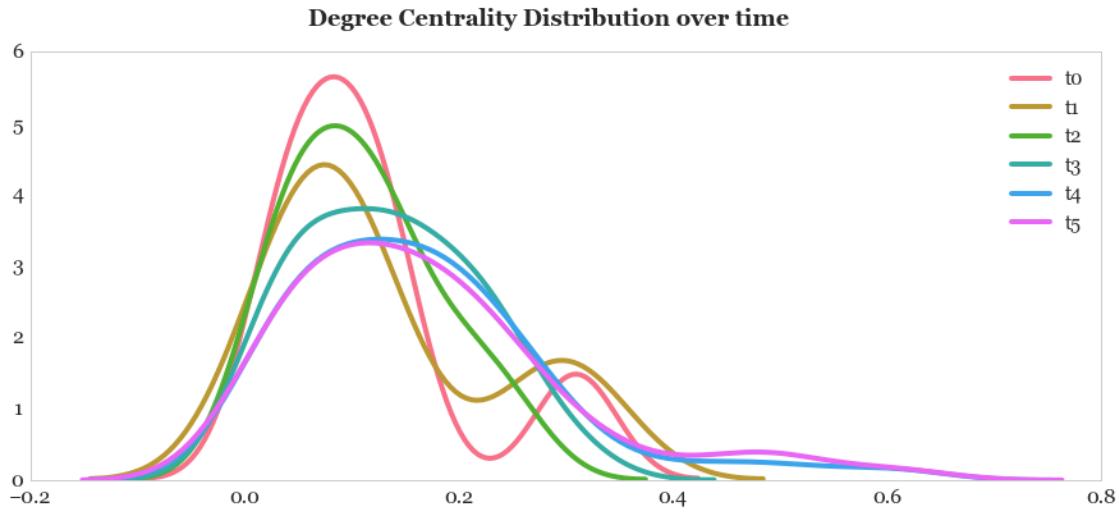
```
In [38]: plt.title("Degree Frequency Distribution over time", fontsize=18)
         plt.plot(nx.degree_histogram(Gt0), label='t=0')
         plt.plot(nx.degree_histogram(Gt1), label='t=1')
         plt.plot(nx.degree_histogram(Gt2), label='t=2')
         plt.plot(nx.degree_histogram(Gt3), label='t=3')
         plt.plot(nx.degree_histogram(Gt4), label='t=4')
         plt.plot(nx.degree_histogram(Gt5), label='t=5')

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)

Out[38]: <matplotlib.legend.Legend at 0x1ad9e8044e0>
```




```
In [39]: plt.suptitle('Degree Centrality Distribution over time', fontsize=18)
         sns.distplot(get_val(degC0), hist=False, label='t0')
         sns.distplot(get_val(degC1), hist=False, label='t1')
         sns.distplot(get_val(degC2), hist=False, label='t2')
         sns.distplot(get_val(degC3), hist=False, label='t3')
         sns.distplot(get_val(degC4), hist=False, label='t4')
         sns.distplot(get_val(degC5), hist=False, label='t5')

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

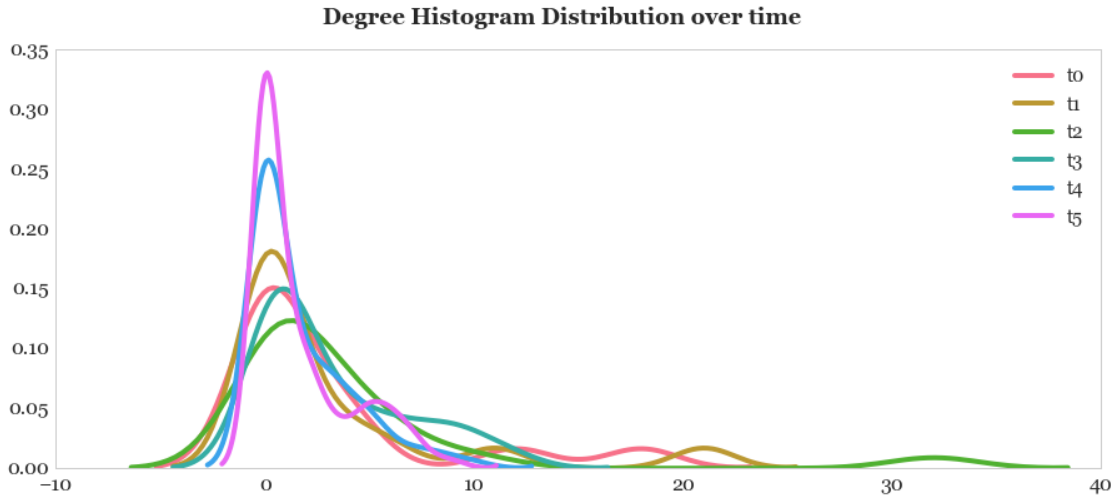
```
Out[39]: <matplotlib.legend.Legend at 0x1ad9f4dbc88>
```



In [40]:
```
plt.suptitle('Degree Histogram Distribution over time', fontsize=18)
sns.distplot(nx.degree_histogram(Gt0), hist=False, label='t0')
sns.distplot(nx.degree_histogram(Gt1), hist=False, label='t1')
sns.distplot(nx.degree_histogram(Gt2), hist=False, label='t2')
sns.distplot(nx.degree_histogram(Gt3), hist=False, label='t3')
sns.distplot(nx.degree_histogram(Gt4), hist=False, label='t4')
sns.distplot(nx.degree_histogram(Gt5), hist=False, label='t5')

plt.yticks(fontsize=16)
plt.xticks(fontsize=16)
plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

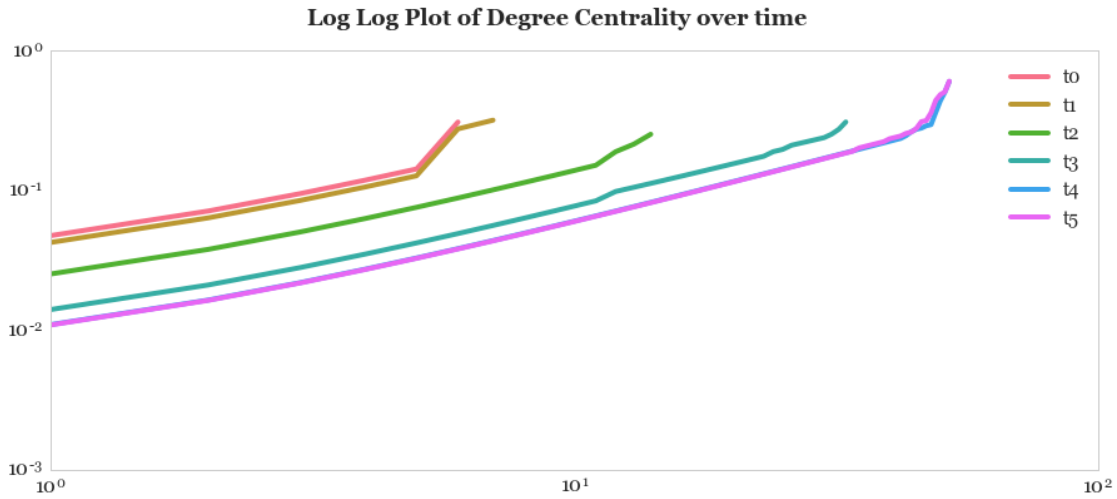
Out[40]: <matplotlib.legend.Legend at 0x1ad9f55d320>



In [41]:
```
plt.suptitle('Log Log Plot of Degree Centrality over time', fontsize=18)

plt.loglog(get_val(degC0), label='t0')
plt.loglog(get_val(degC1), label='t1')
plt.loglog(get_val(degC2), label='t2')
plt.loglog(get_val(degC3), label='t3')
plt.loglog(get_val(degC4), label='t4')
plt.loglog(get_val(degC5), label='t5')

plt.yticks(fontsize=16)
```

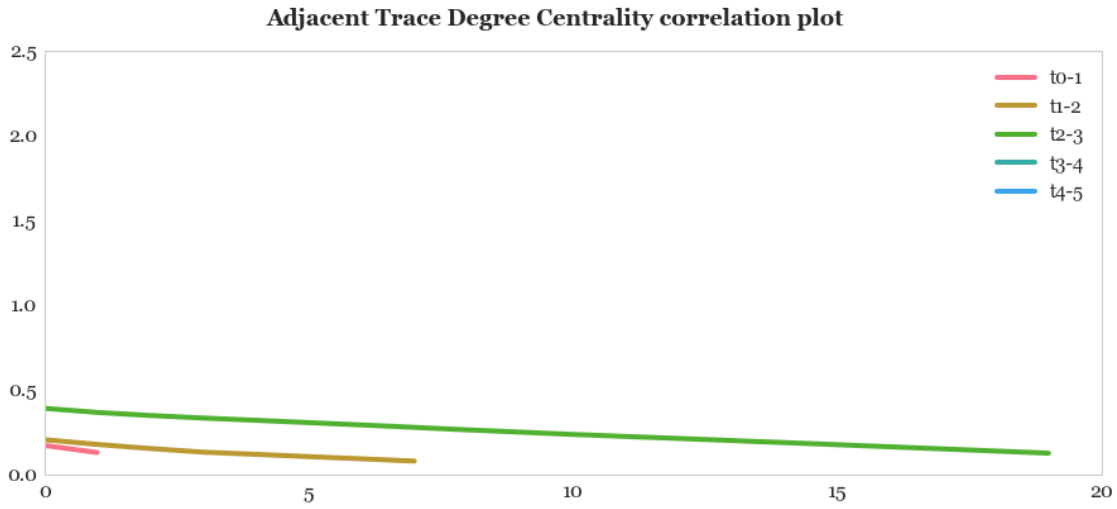
```
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

```
Out[41]: <matplotlib.legend.Legend at 0x1ad9f7da940>
```



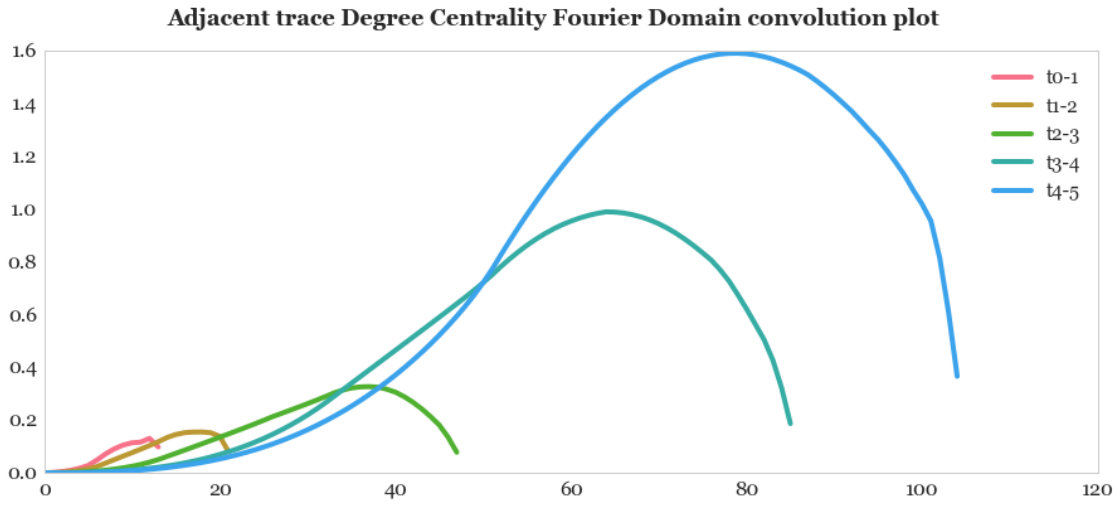

```
In [42]: plt.suptitle('Adjacent Trace Degree Centrality correlation plot', fontsize

        plt.plot(np.correlate(get_val(degC0),get_val(degC1)),label='t0-1')
        plt.plot(np.correlate(get_val(degC1),get_val(degC2)),label='t1-2')
        plt.plot(np.correlate(get_val(degC2),get_val(degC3)),label='t2-3')
        plt.plot(np.correlate(get_val(degC3),get_val(degC3)),label='t3-4')
        plt.plot(np.correlate(get_val(degC4),get_val(degC5)),label='t4-5')

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

```
Out[42]: <matplotlib.legend.Legend at 0x1ad9f77c8d0>
```



In [43]:
```
plt.suptitle('Adjacent trace Degree Centrality Fourier Domain convolution

plt.plot(fftconvolve(get_val(degC0),get_val(degC1)),label='t0-1')
plt.plot(fftconvolve(get_val(degC1),get_val(degC2)),  label='t1-2')
plt.plot(fftconvolve(get_val(degC2),get_val(degC3)), label='t2-3')
plt.plot(fftconvolve(get_val(degC3),get_val(degC4)), label='t3-4')
plt.plot(fftconvolve(get_val(degC4),get_val(degC5)),  label='t4-5')

plt.yticks(fontsize=16)
plt.xticks(fontsize=16)
plt.legend(loc=1, fontsize=15)
```

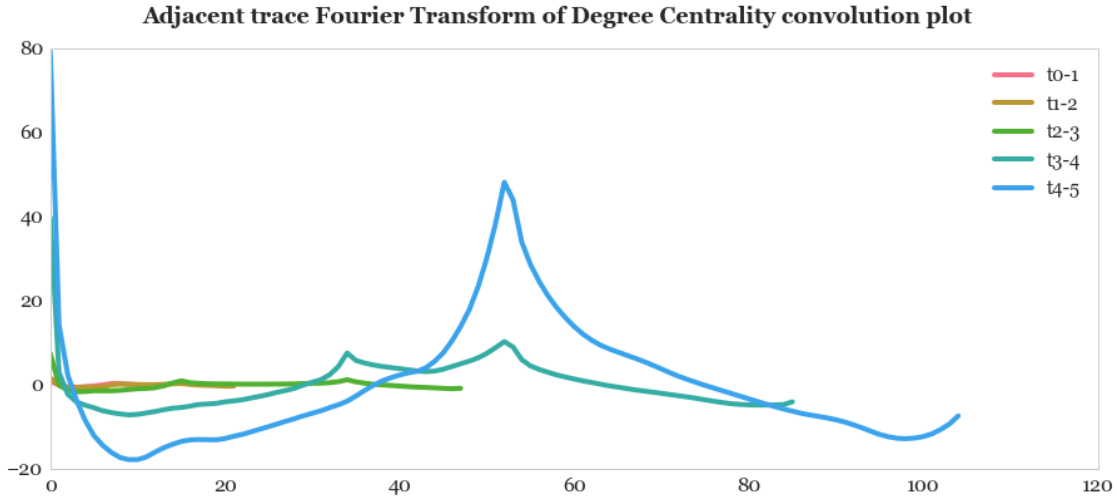
Out[43]:
```
<matplotlib.legend.Legend at 0x1ad9de74cc0>
```



In [44]:
```
plt.suptitle('Adjacent trace Fourier Transform of Degree Centrality convol

        plt.plot(np.convolve(fft_sig(degC0),fft_sig(degC1)),label='t0-1')
        plt.plot(np.convolve(fft_sig(degC1),fft_sig(degC2)),  label='t1-2')
        plt.plot(np.convolve(fft_sig(degC2),fft_sig(degC3)), label='t2-3')
        plt.plot(np.convolve(fft_sig(degC3),fft_sig(degC4)), label='t3-4')
        plt.plot(np.convolve(fft_sig(degC4),fft_sig(degC5)),  label='t4-5')

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)
```

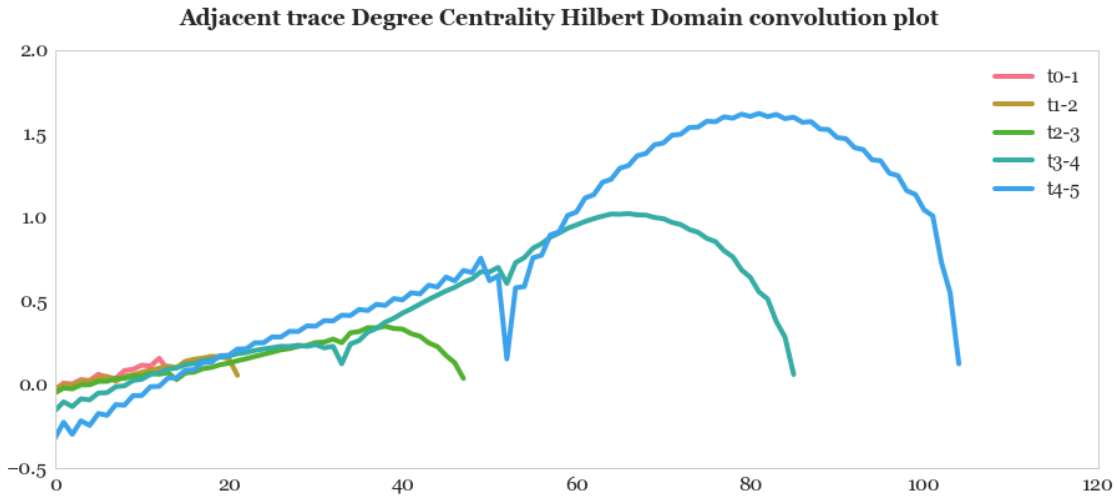
Out[44]: <matplotlib.legend.Legend at 0x1ad9f1f19b0>



In [45]:
```
plt.suptitle('Adjacent trace Degree Centrality Hilbert Domain convolution

        plt.plot(np.convolve(hilbert(get_val(degC0)), hilbert(get_val(degC1))),  l
        plt.plot(np.convolve(hilbert(get_val(degC1)), hilbert(get_val(degC2))),  l
        plt.plot(np.convolve(hilbert(get_val(degC2)), hilbert(get_val(degC3))), la
        plt.plot(np.convolve(hilbert(get_val(degC3)), hilbert(get_val(degC4))), la
        plt.plot(np.convolve(hilbert(get_val(degC4)), hilbert(get_val(degC5))),  l

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)
```

```
Out[45]: <matplotlib.legend.Legend at 0x1ad9f191438>
```



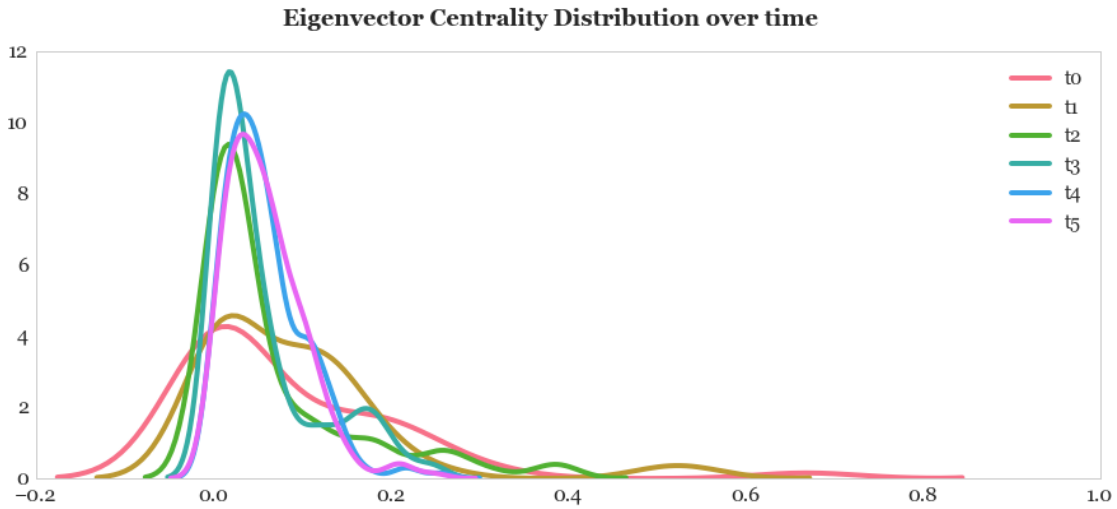
6.1.3 Eigenvector Centrality Histograms

Plotting the Eigenvector Centrality for the different timesteps here. For the first plot it is difficult to discern the trends when all the 6 distributions are plotted together. So in the next series of plots I look at a few a time and its easier to see the change over time. The signal essentially becomes more spiked and squashed over time.

```
In [46]: plt.suptitle('Eigenvector Centrality Distribution over time', fontsize=18)
         sns.distplot(get_val(eigC0), hist=False, label='t0')
         sns.distplot(get_val(eigC1), hist=False, label='t1')
         sns.distplot(get_val(eigC2), hist=False, label='t2')
         sns.distplot(get_val(eigC3), hist=False, label='t3')
         sns.distplot(get_val(eigC4), hist=False, label='t4')
         sns.distplot(get_val(eigC5), hist=False, label='t5')

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)
```

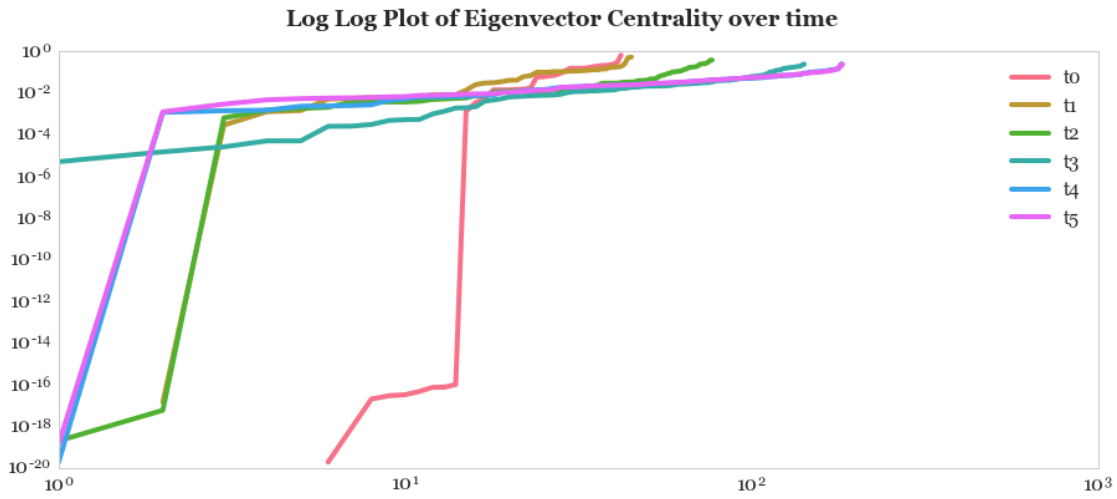
```
C:\Users\arsha_000\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

```
Out[46]: <matplotlib.legend.Legend at 0x1ad9ef9a978>
```



In [47]:
```
plt.suptitle('Log Log Plot of Eigenvector Centrality over time', fontsize=

plt.loglog(get_val(eigC0), label='t0')
plt.loglog(get_val(eigC1), label='t1')
plt.loglog(get_val(eigC2), label='t2')
plt.loglog(get_val(eigC3), label='t3')
plt.loglog(get_val(eigC4), label='t4')
plt.loglog(get_val(eigC5), label='t5')

plt.yticks(fontsize=16)
plt.xticks(fontsize=16)
plt.legend(loc=1, fontsize=15)
```

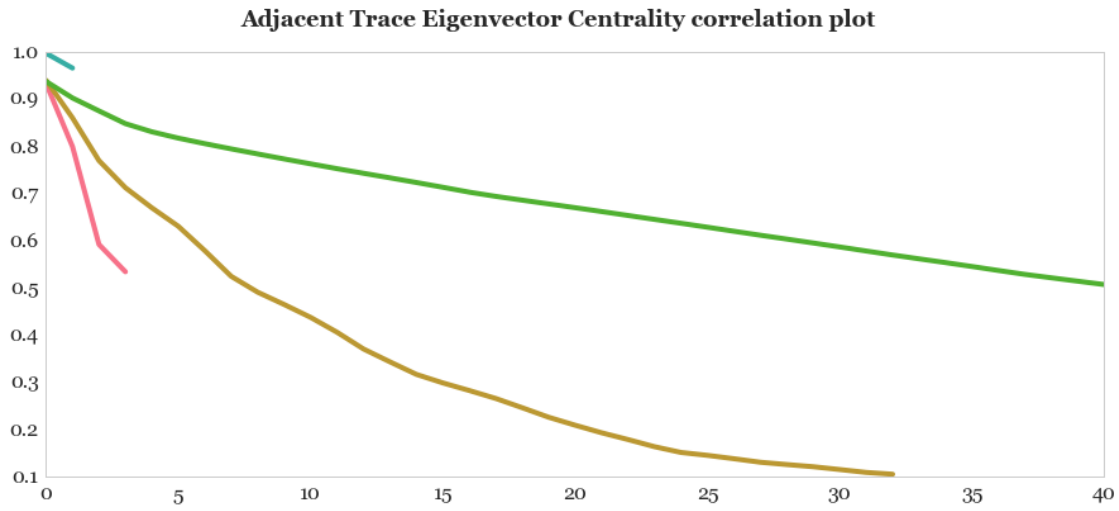
Out[47]: <matplotlib.legend.Legend at 0x1ad9f2b4c18>



```
In [48]: eigC0_a = np.asarray(get_val(eigC0))
         eigC1_a = np.asarray(get_val(eigC1))
```

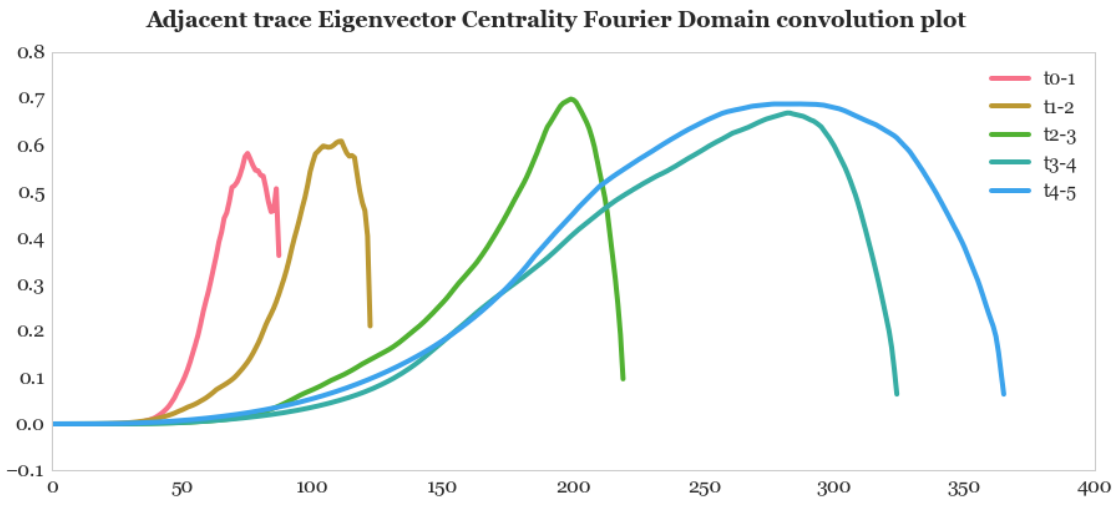
```
In [49]: plt.suptitle('Adjacent Trace Eigenvector Centrality correlation plot', fon

         plt.plot(np.correlate(eigC0_a,eigC1_a))
         plt.plot(np.correlate(get_val(eigC1),get_val(eigC2)))
         plt.plot(np.correlate(get_val(eigC3),get_val(eigC4)))
         plt.plot(np.correlate(get_val(eigC4),get_val(eigC5)))

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:519: UserWa
  warnings.warn("No labelled objects found. "
```




```
In [50]: plt.suptitle('Adjacent trace Eigenvector Centrality Fourier Domain convolu

         plt.plot(fftconvolve(get_val(eigC0),get_val(eigC1)),  label='t0-1')
         plt.plot(fftconvolve(get_val(eigC1),get_val(eigC2)),  label='t1-2')
         plt.plot(fftconvolve(get_val(eigC2),get_val(eigC3)), label='t2-3')
         plt.plot(fftconvolve(get_val(eigC3),get_val(eigC4)), label='t3-4')
         plt.plot(fftconvolve(get_val(eigC4),get_val(eigC5)),  label='t4-5')

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)
```

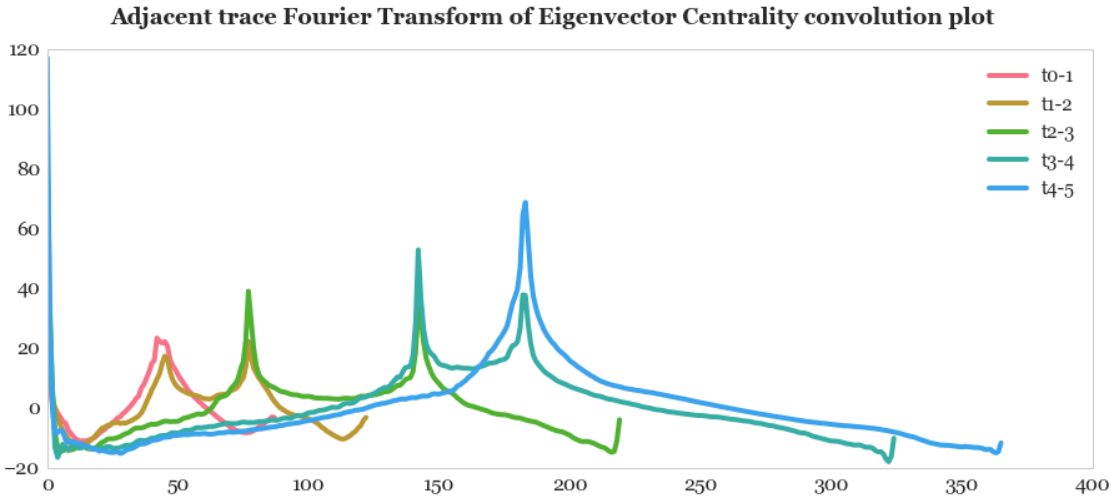
Out[50]: <matplotlib.legend.Legend at 0x1ada132cb38>



In [51]:
```
plt.suptitle('Adjacent trace Fourier Transform of Eigenvector Centrality c

        plt.plot(np.convolve(fft_sig(eigC0),fft_sig(eigC1)),label='t0-1')
        plt.plot(np.convolve(fft_sig(eigC1),fft_sig(eigC2)),  label='t1-2')
        plt.plot(np.convolve(fft_sig(eigC2),fft_sig(eigC3)), label='t2-3')
        plt.plot(np.convolve(fft_sig(eigC3),fft_sig(eigC4)), label='t3-4')
        plt.plot(np.convolve(fft_sig(eigC4),fft_sig(eigC5)),  label='t4-5')

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)
```

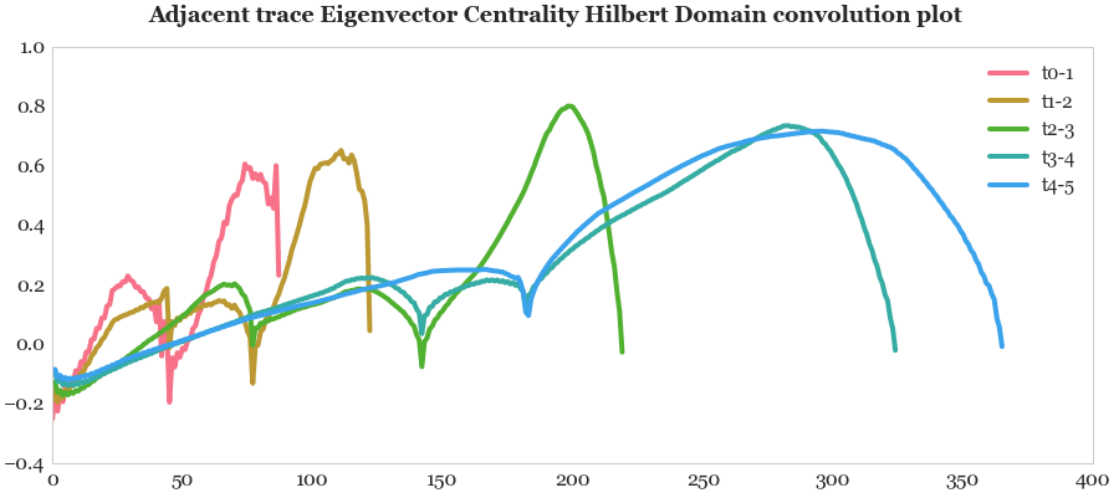
Out[51]: <matplotlib.legend.Legend at 0x1ada15b7a58>




```
In [52]: plt.suptitle('Adjacent trace Eigenvector Centrality Hilbert Domain convolu

         plt.plot(np.convolve(hilbert(get_val(eigC0)), hilbert(get_val(eigC1))),  l
         plt.plot(np.convolve(hilbert(get_val(eigC1)), hilbert(get_val(eigC2))),  l
         plt.plot(np.convolve(hilbert(get_val(eigC2)), hilbert(get_val(eigC3))), la
         plt.plot(np.convolve(hilbert(get_val(eigC3)), hilbert(get_val(eigC4))), la
         plt.plot(np.convolve(hilbert(get_val(eigC4)), hilbert(get_val(eigC5))),  l

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)
```

```
Out[52]: <matplotlib.legend.Legend at 0x1ada1832cc0>
```

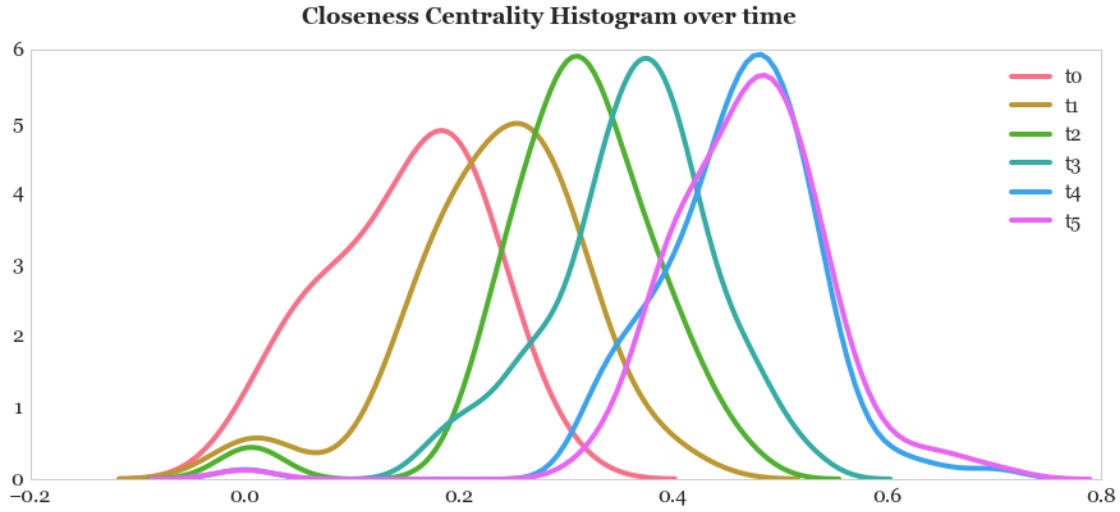


```
In [53]: print(np.correlate(eigC0_a,eigC1_a))
         print(np.correlate(eigC0_a,eigC1_a)/np.sqrt(np.correlate(eigC0_a,eigC1_a))

[ 0.934373    0.8003439   0.59281644  0.53482695]
[ 0.96662971  0.89461941  0.76994574  0.73131864]
```

6.1.4 Closeness Centrality Histograms

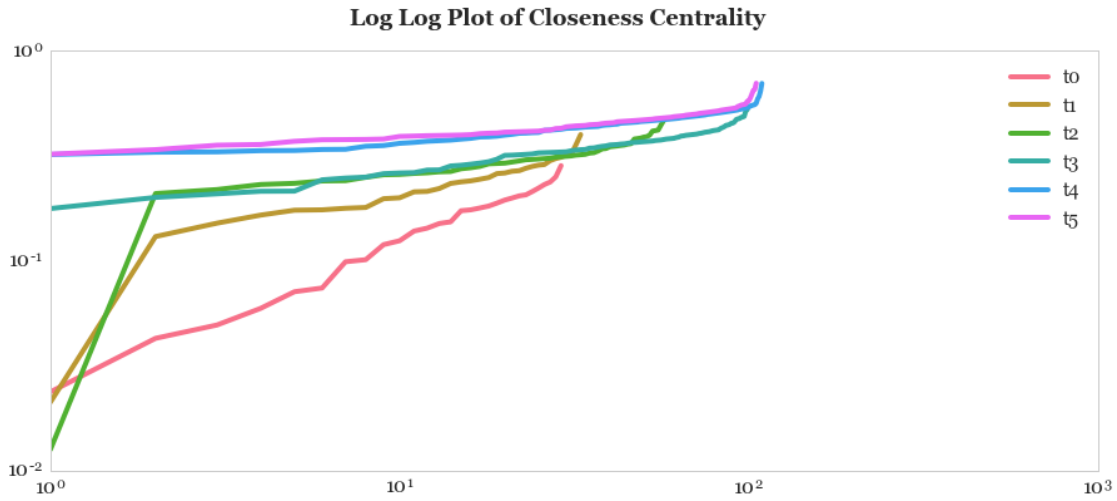
The Closeness Centrality shows a much better evolution over time than the Eigenvector Centrality Histograms

```
In [54]: plt.suptitle('Closeness Centrality Histogram over time', fontsize=18)
         sns.distplot(get_val(cloC0), hist=False, label='t0')
         sns.distplot(get_val(cloC1), hist=False, label='t1')
         sns.distplot(get_val(cloC2), hist=False, label='t2')
         sns.distplot(get_val(cloC3), hist=False, label='t3')
         sns.distplot(get_val(cloC4), hist=False, label='t4')
         sns.distplot(get_val(cloC5), hist=False, label='t5')

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)

C:\Users\arsha_000\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j


Out[54]: <matplotlib.legend.Legend at 0x1ad9fb6cf60>
```



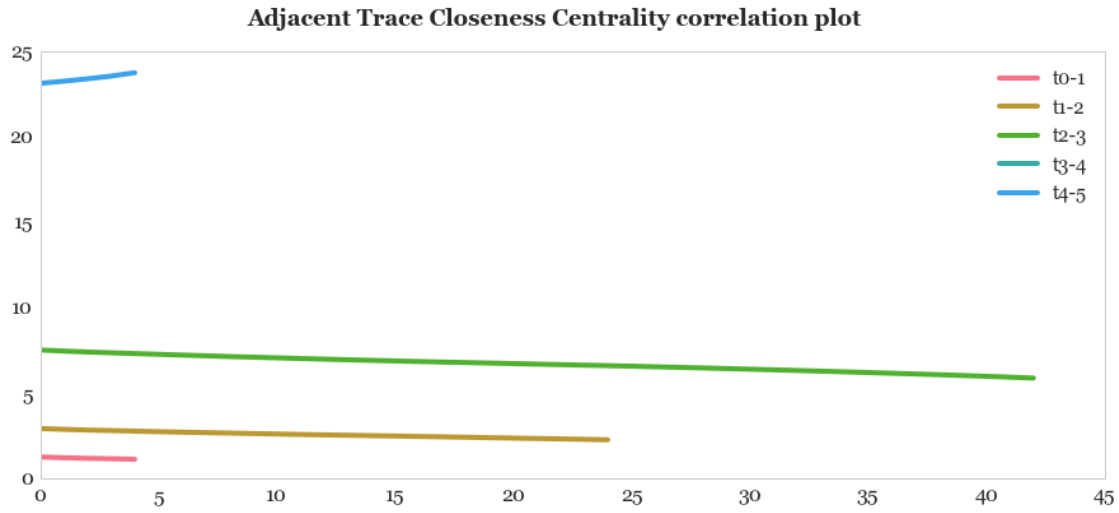
In [55]:
```
plt.suptitle('Log Log Plot of Closeness Centrality', fontsize=18)

plt.loglog(get_val(cloC0),label='t0')
plt.loglog(get_val(cloC1),label='t1')
plt.loglog(get_val(cloC2),label='t2')
plt.loglog(get_val(cloC3),label='t3')
plt.loglog(get_val(cloC4),label='t4')
plt.loglog(get_val(cloC5),label='t5')

plt.yticks(fontsize=16)
plt.xticks(fontsize=16)
plt.legend(loc=1, fontsize=15)
```

Out[55]: <matplotlib.legend.Legend at 0x1ada186fa90>

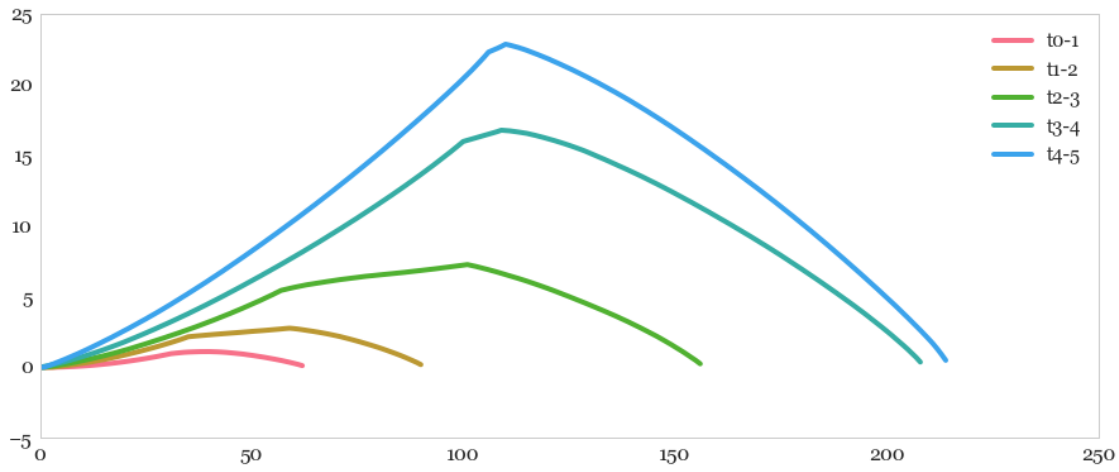


In [56]:
```
plt.suptitle('Adjacent Trace Closeness Centrality correlation plot', fonts

plt.plot(np.correlate(get_val(cloC0),get_val(cloC1)),label='t0-1')
plt.plot(np.correlate(get_val(cloC1),get_val(cloC2)),label='t1-2')
plt.plot(np.correlate(get_val(cloC2),get_val(cloC3)),label='t2-3')
plt.plot(np.correlate(get_val(cloC3),get_val(cloC3)),label='t3-4')
plt.plot(np.correlate(get_val(cloC4),get_val(cloC5)),label='t4-5')

plt.yticks(fontsize=16)
plt.xticks(fontsize=16)
plt.legend(loc=1, fontsize=15)
```

Out[56]: <matplotlib.legend.Legend at 0x1ad9facf390>



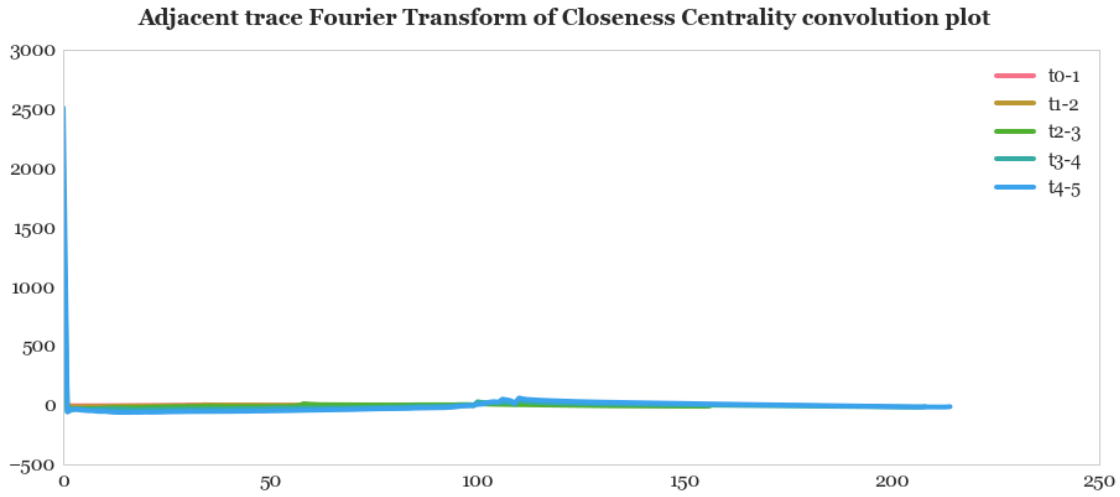

```
In [57]: plt.suptitle('Adjacent trace Closeness Centrality Fourier Domain convoluti

         plt.plot(fftconvolve(get_val(cloC0),get_val(cloC1)),label='t0-1')
         plt.plot(fftconvolve(get_val(cloC1),get_val(cloC2)),  label='t1-2')
         plt.plot(fftconvolve(get_val(cloC2),get_val(cloC3)), label='t2-3')
         plt.plot(fftconvolve(get_val(cloC3),get_val(cloC4)), label='t3-4')
         plt.plot(fftconvolve(get_val(cloC4),get_val(cloC5)),  label='t4-5')

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)

Out[57]: <matplotlib.legend.Legend at 0x1ada18e9c50>
```



In [58]:
```
plt.suptitle('Adjacent trace Fourier Transform of Closeness Centrality cor

        plt.plot(np.convolve(fft_sig(cloC0),fft_sig(cloC1)),label='t0-1')
        plt.plot(np.convolve(fft_sig(cloC1),fft_sig(cloC2)),label='t1-2')
        plt.plot(np.convolve(fft_sig(cloC2),fft_sig(cloC3)),label='t2-3')
        plt.plot(np.convolve(fft_sig(cloC3),fft_sig(cloC4)),label='t3-4')
        plt.plot(np.convolve(fft_sig(cloC4),fft_sig(cloC5)),label='t4-5')

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)
```

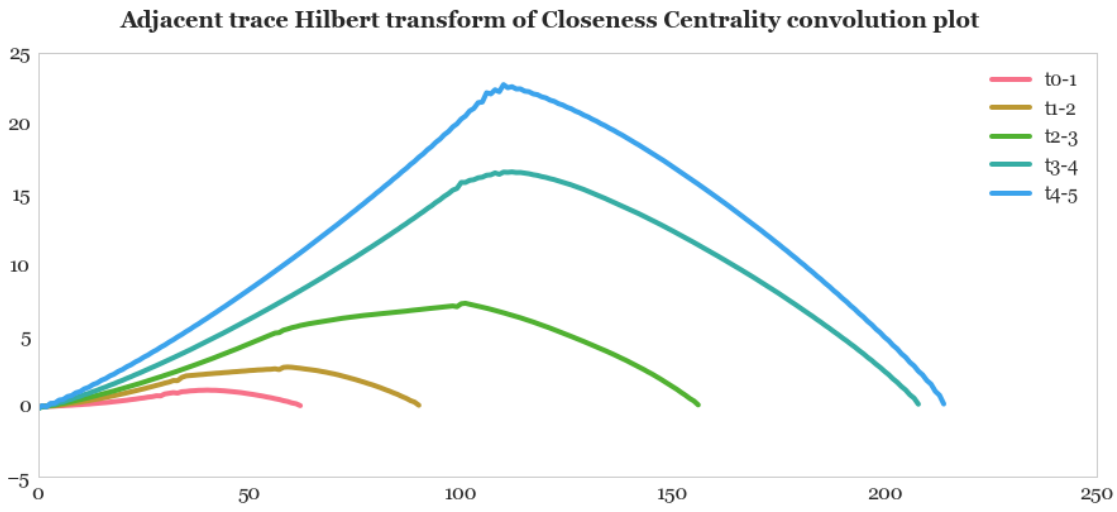
Out[58]: <matplotlib.legend.Legend at 0x1ada1927358>



In [59]:
```
plt.suptitle('Adjacent trace Hilbert transform of Closeness Centrality cor

        plt.plot(np.convolve(hilbert_sig(cloC0), hilbert_sig(cloC1)), label='t0-1'
        plt.plot(np.convolve(hilbert_sig(cloC1), hilbert_sig(cloC2)), label='t1-2'
        plt.plot(np.convolve(hilbert_sig(cloC2), hilbert_sig(cloC3)),label='t2-3')
        plt.plot(np.convolve(hilbert_sig(cloC3), hilbert_sig(cloC4)), label='t3-4'
        plt.plot(np.convolve(hilbert_sig(cloC4), hilbert_sig(cloC5)), label='t4-5'

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)
```

Out[59]: <matplotlib.legend.Legend at 0x1ada19fbf98>

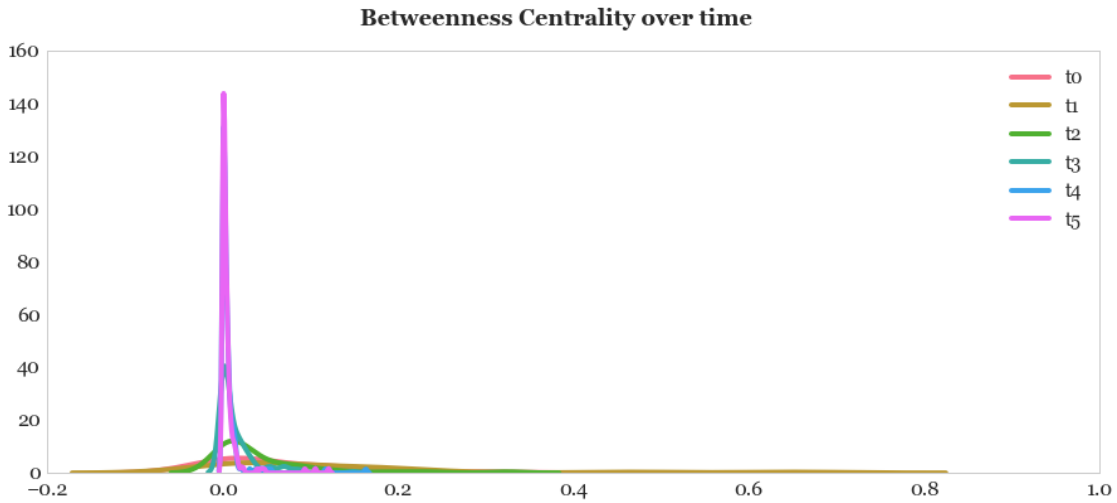


6.1.5 Betweenness Centrality Histogram

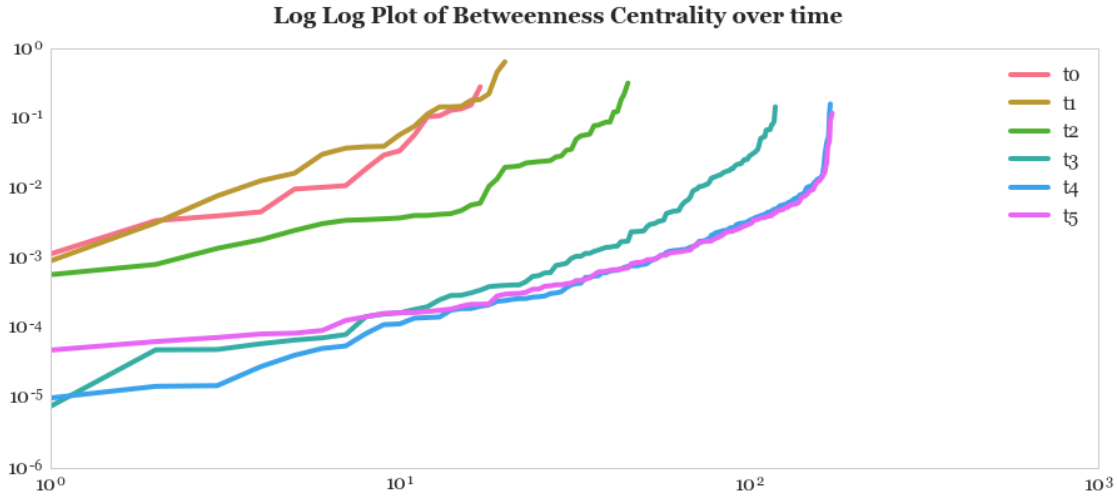
```
In [60]: plt.suptitle('Betweenness Centrality over time', fontsize=18)
         sns.distplot(get_val(betC0), hist=False, label='t0')
         sns.distplot(get_val(betC1), hist=False, label='t1')
         sns.distplot(get_val(betC2), hist=False, label='t2')
         sns.distplot(get_val(betC3), hist=False, label='t3')
         sns.distplot(get_val(betC4), hist=False, label='t4')
         sns.distplot(get_val(betC5), hist=False, label='t5')

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

Out[60]: <matplotlib.legend.Legend at 0x1ada1fe51d0>

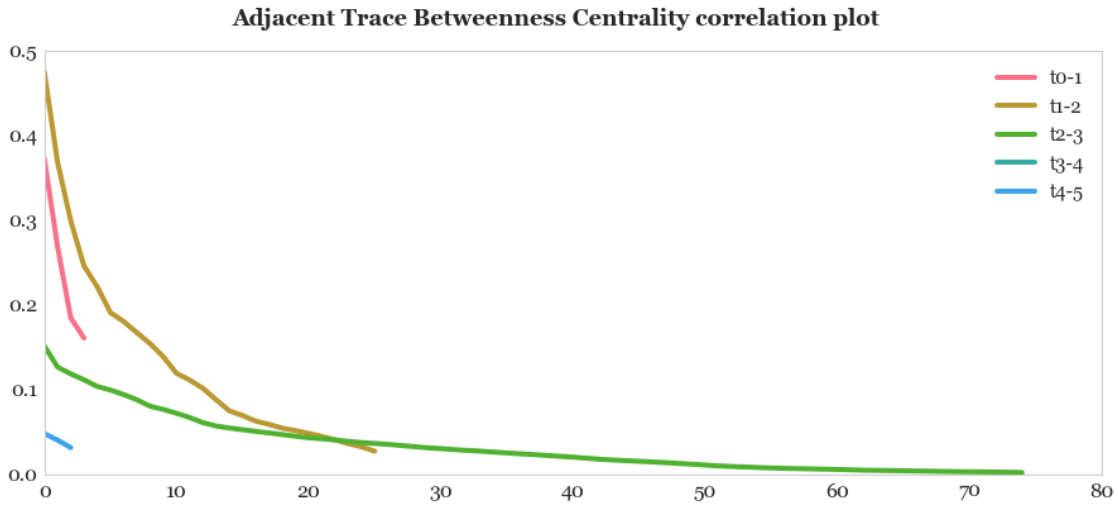



```
In [61]: plt.suptitle('Log Log Plot of Betweenness Centrality over time', fontsize=

         plt.loglog(get_val(betC0), label='t0')
         plt.loglog(get_val(betC1), label='t1')
         plt.loglog(get_val(betC2), label='t2')
         plt.loglog(get_val(betC3), label='t3')
         plt.loglog(get_val(betC4), label='t4')
         plt.loglog(get_val(betC5), label='t5')

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)

Out[61]: <matplotlib.legend.Legend at 0x1ada254de80>
```



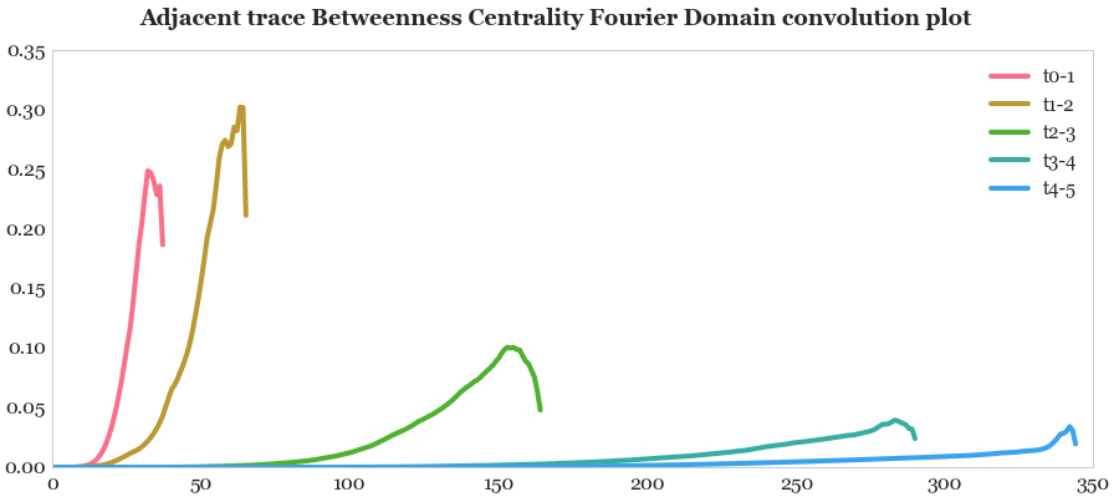
In [62]:
```
plt.suptitle('Adjacent Trace Betweenness Centrality correlation plot', for

        plt.plot(np.correlate(get_val(betC0),get_val(betC1)),label='t0-1')
        plt.plot(np.correlate(get_val(betC1),get_val(betC2)),label='t1-2')
        plt.plot(np.correlate(get_val(betC2),get_val(betC3)),label='t2-3')
        plt.plot(np.correlate(get_val(betC3),get_val(betC3)),label='t3-4')
        plt.plot(np.correlate(get_val(betC4),get_val(betC5)),label='t4-5')

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

Out[62]:
```
<matplotlib.legend.Legend at 0x1ada5433e10>
```



In [63]:
```
plt.suptitle('Adjacent trace Betweenness Centrality Fourier Domain convolu

        plt.plot(fftconvolve(get_val(betC0),get_val(betC1)),label='t0-1')
        plt.plot(fftconvolve(get_val(betC1),get_val(betC2)),  label='t1-2')
        plt.plot(fftconvolve(get_val(betC2),get_val(betC3)), label='t2-3')
        plt.plot(fftconvolve(get_val(betC3),get_val(betC4)), label='t3-4')
        plt.plot(fftconvolve(get_val(betC4),get_val(betC5)),  label='t4-5')

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

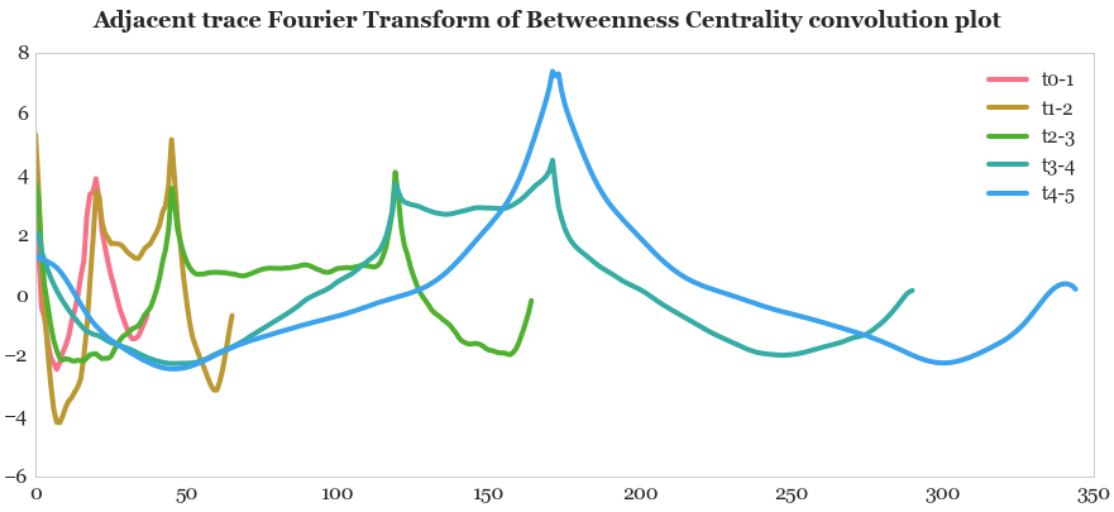
Out[63]:
```
<matplotlib.legend.Legend at 0x1ada55ea080>
```



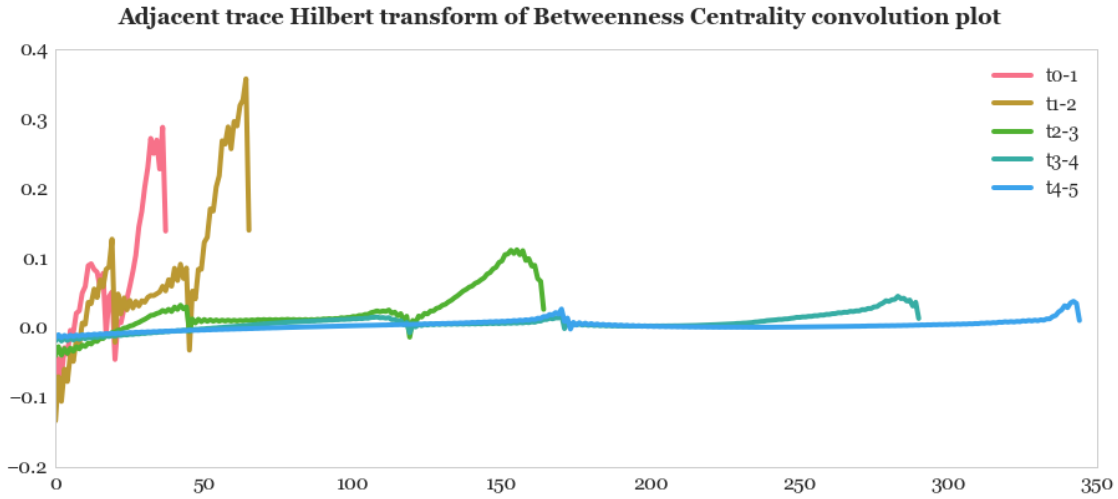

```
In [64]: plt.suptitle('Adjacent trace Fourier Transform of Betweenness Centrality c

         plt.plot(np.convolve(fft_sig(betC0),fft_sig(betC1)),label='t0-1')
         plt.plot(np.convolve(fft_sig(betC1),fft_sig(betC2)),label='t1-2')
         plt.plot(np.convolve(fft_sig(betC2),fft_sig(betC3)),label='t2-3')
         plt.plot(np.convolve(fft_sig(betC3),fft_sig(betC4)),label='t3-4')
         plt.plot(np.convolve(fft_sig(betC4),fft_sig(betC5)),label='t4-5')

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)
```

```
Out[64]: <matplotlib.legend.Legend at 0x1ada5671400>
```



```
In [65]: plt.suptitle('Adjacent trace Hilbert transform of Betweenness Centrality c

         plt.plot(np.convolve(hilbert_sig(betC0), hilbert_sig(betC1)), label='t0-1'
         plt.plot(np.convolve(hilbert_sig(betC1), hilbert_sig(betC2)), label='t1-2'
         plt.plot(np.convolve(hilbert_sig(betC2), hilbert_sig(betC3)),label='t2-3')
         plt.plot(np.convolve(hilbert_sig(betC3), hilbert_sig(betC4)), label='t3-4'
         plt.plot(np.convolve(hilbert_sig(betC4), hilbert_sig(betC5)), label='t4-5'

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)

C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)


Out[65]: <matplotlib.legend.Legend at 0x1ada4fe6ba8>
```



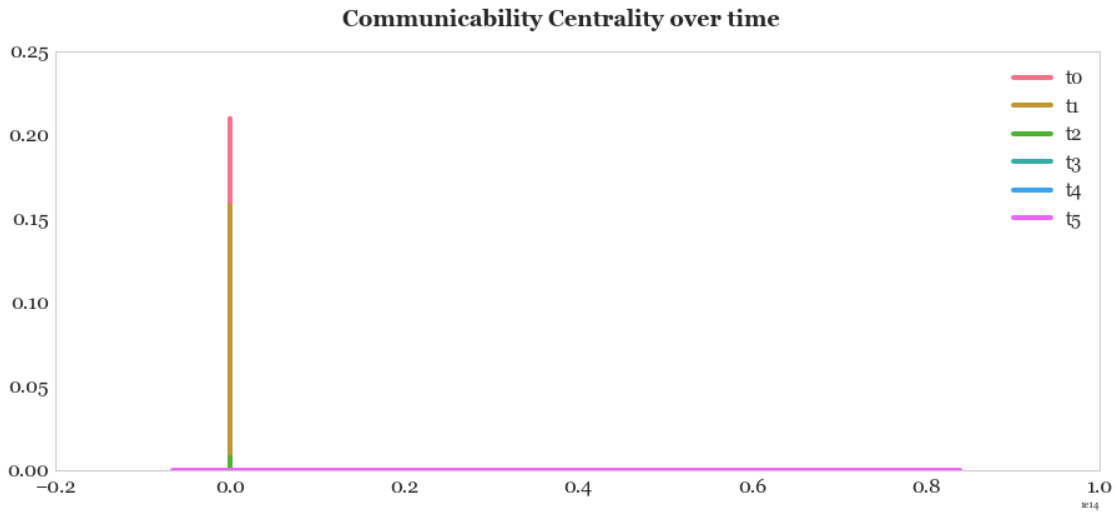
6.1.6 Communicability Centrality Histograms

```
In [66]: plt.suptitle('Communicability Centrality over time', fontsize=18)
         sns.distplot(get_val(commuC0), hist=False, label='t0')
         sns.distplot(get_val(commuC1), hist=False, label='t1')
         sns.distplot(get_val(commuC2), hist=False, label='t2')
         sns.distplot(get_val(commuC3), hist=False, label='t3')
         sns.distplot(get_val(commuC4), hist=False, label='t4')
         sns.distplot(get_val(commuC5), hist=False, label='t5')
```

```
        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

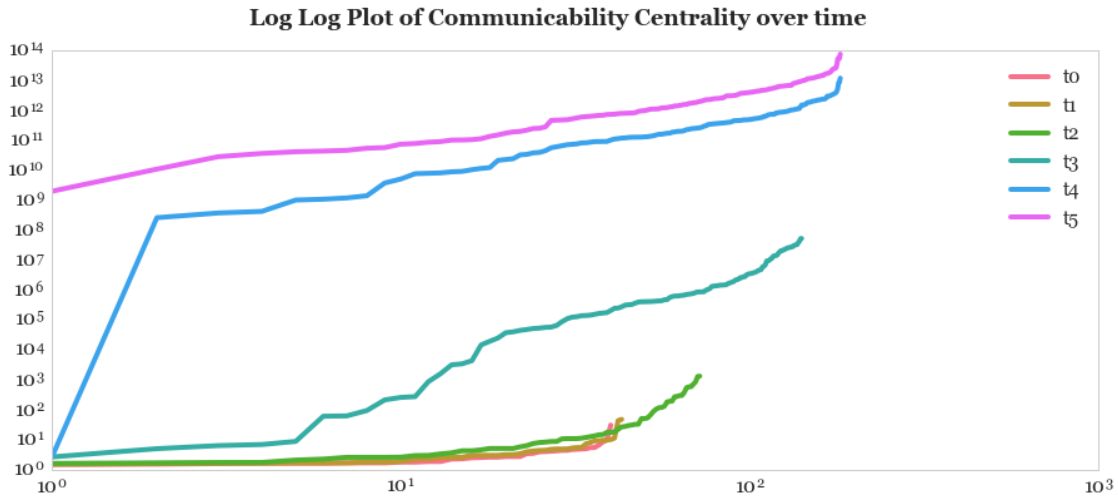
```
C:\Users\arsha_000\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

Out[66]: <matplotlib.legend.Legend at 0x1ada4f08240>



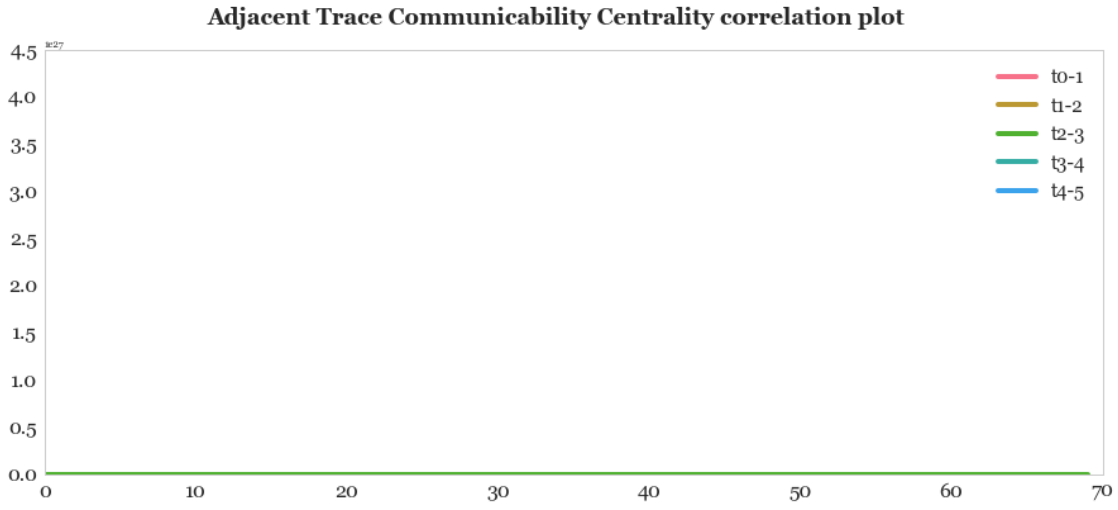
In [67]:
```
plt.suptitle('Log Log Plot of Communicability Centrality over time', fonts

        plt.loglog(get_val(commuC0), label='t0')
        plt.loglog(get_val(commuC1), label='t1')
        plt.loglog(get_val(commuC2), label='t2')
        plt.loglog(get_val(commuC3), label='t3')
        plt.loglog(get_val(commuC4), label='t4')
        plt.loglog(get_val(commuC5), label='t5')

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

Out[67]: <matplotlib.legend.Legend at 0x1ada514ccc0>



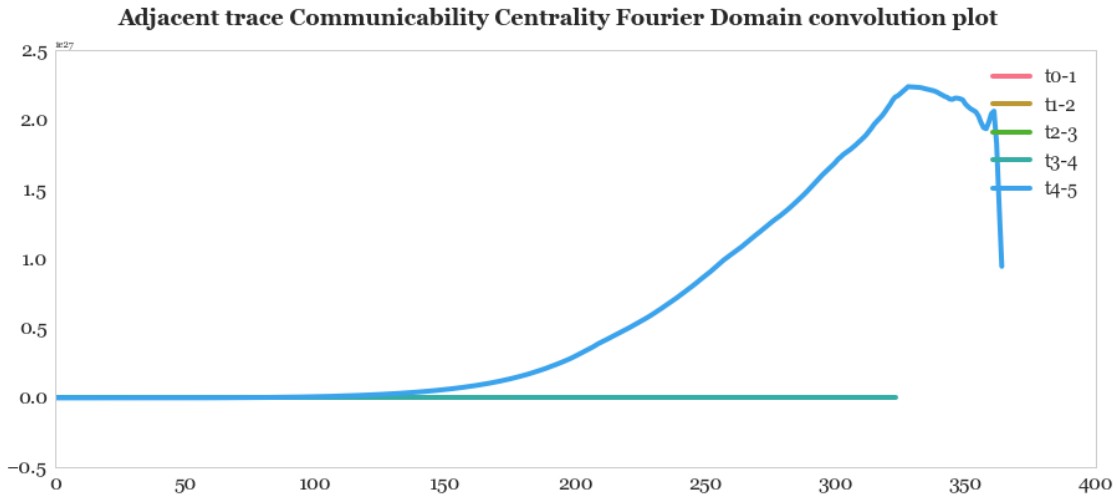
In [68]:
```
plt.suptitle('Adjacent Trace Communicability Centrality correlation plot',

        plt.plot(np.correlate(get_val(commuC0),get_val(commuC1)),label='t0-1')
        plt.plot(np.correlate(get_val(commuC1),get_val(commuC2)),label='t1-2')
        plt.plot(np.correlate(get_val(commuC2),get_val(commuC3)),label='t2-3')
        plt.plot(np.correlate(get_val(commuC3),get_val(commuC3)),label='t3-4')
        plt.plot(np.correlate(get_val(commuC4),get_val(commuC5)),label='t4-5')

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

Out[68]:
```
<matplotlib.legend.Legend at 0x1ada1d7aa90>
```



In [69]:
```
plt.suptitle('Adjacent trace Communicability Centrality Fourier Domain cor

        plt.plot(fftconvolve(get_val(commuC0),get_val(commuC1)),label='t0-1')
        plt.plot(fftconvolve(get_val(commuC1),get_val(commuC2)),  label='t1-2')
        plt.plot(fftconvolve(get_val(commuC2),get_val(commuC3)), label='t2-3')
        plt.plot(fftconvolve(get_val(commuC3),get_val(commuC4)), label='t3-4')
        plt.plot(fftconvolve(get_val(commuC4),get_val(commuC5)),  label='t4-5')

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

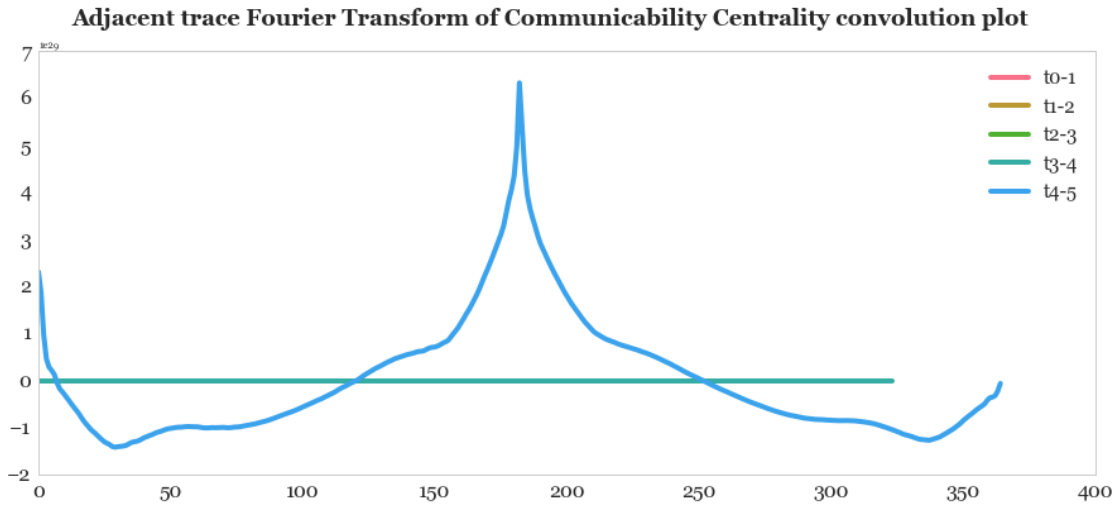
Out[69]: <matplotlib.legend.Legend at 0x1ada1bdecc0>



In [70]:
```
plt.suptitle('Adjacent trace Fourier Transform of Communicability Centrali

        plt.plot(np.convolve(fft_sig(commuC0),fft_sig(commuC1)),label='t0-1')
        plt.plot(np.convolve(fft_sig(commuC1),fft_sig(commuC2)),label='t1-2')
        plt.plot(np.convolve(fft_sig(commuC2),fft_sig(commuC3)),label='t2-3')
        plt.plot(np.convolve(fft_sig(commuC3),fft_sig(commuC4)),label='t3-4')
        plt.plot(np.convolve(fft_sig(commuC4),fft_sig(commuC5)),label='t4-5')

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)
```

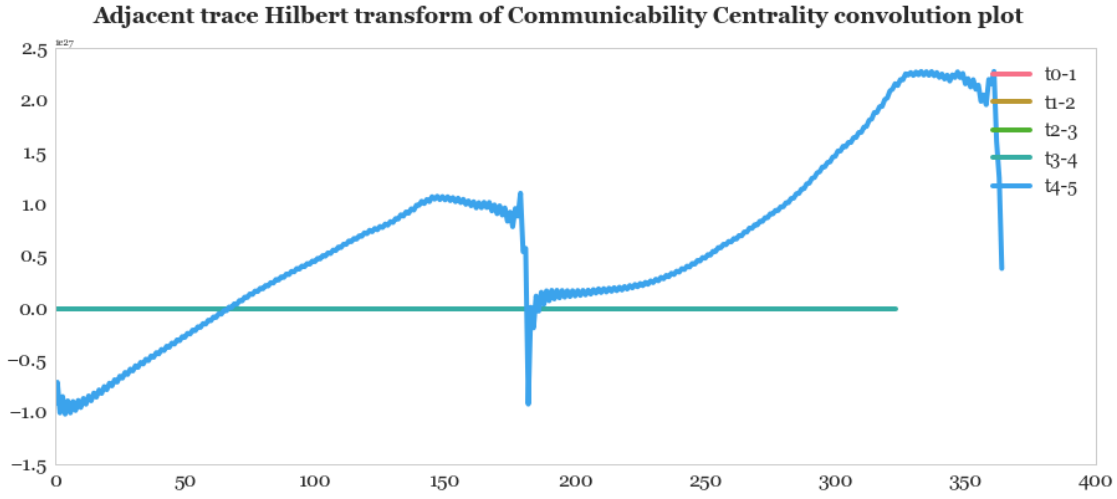
Out[70]: <matplotlib.legend.Legend at 0x1ada4e09ac8>



In [71]:
```
plt.suptitle('Adjacent trace Hilbert transform of Communicability Centrali

        plt.plot(np.convolve(hilbert_sig(commuC0), hilbert_sig(commuC1)), label='t
        plt.plot(np.convolve(hilbert_sig(commuC1), hilbert_sig(commuC2)), label='t
        plt.plot(np.convolve(hilbert_sig(commuC2), hilbert_sig(commuC3)),label='t2
        plt.plot(np.convolve(hilbert_sig(commuC3), hilbert_sig(commuC4)), label='t
        plt.plot(np.convolve(hilbert_sig(commuC4), hilbert_sig(commuC5)), label='t

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)
```

Out[71]:
```
<matplotlib.legend.Legend at 0x1ada4e99ba8>
```

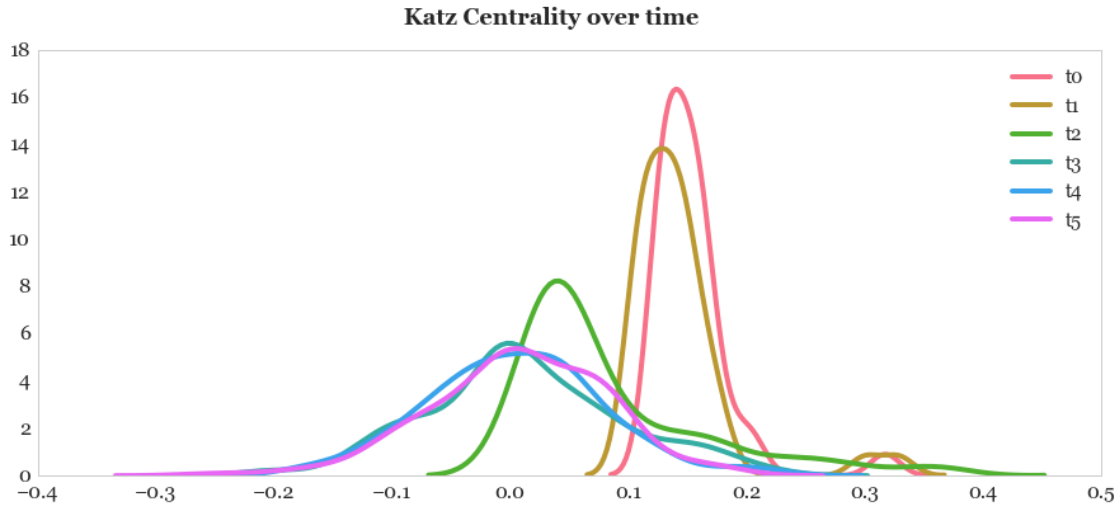


6.1.7 Katz Centrality Histograms

```
In [72]: plt.suptitle('Katz Centrality over time', fontsize=18)
         sns.distplot(get_val(katzC0), hist=False, label='t0')
         sns.distplot(get_val(katzC1), hist=False, label='t1')
         sns.distplot(get_val(katzC2), hist=False, label='t2')
         sns.distplot(get_val(katzC3), hist=False, label='t3')
         sns.distplot(get_val(katzC4), hist=False, label='t4')
         sns.distplot(get_val(katzC5), hist=False, label='t5')

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

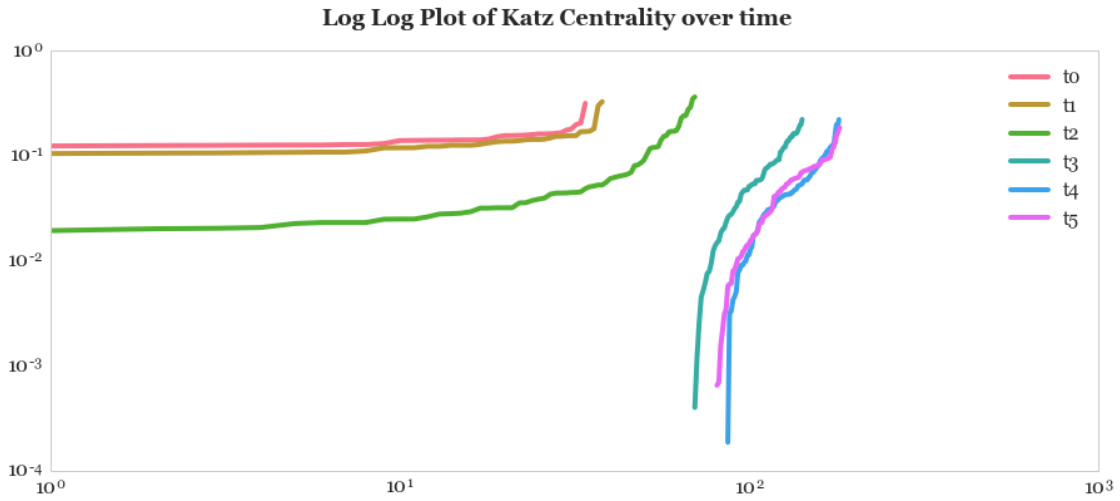
```
Out[72]: <matplotlib.legend.Legend at 0x1ada5d077b8>
```




```
In [73]: plt.suptitle('Log Log Plot of Katz Centrality over time', fontsize=18)

         plt.loglog(get_val(katzC0), label='t0')
         plt.loglog(get_val(katzC1), label='t1')
         plt.loglog(get_val(katzC2), label='t2')
         plt.loglog(get_val(katzC3), label='t3')
         plt.loglog(get_val(katzC4), label='t4')
```

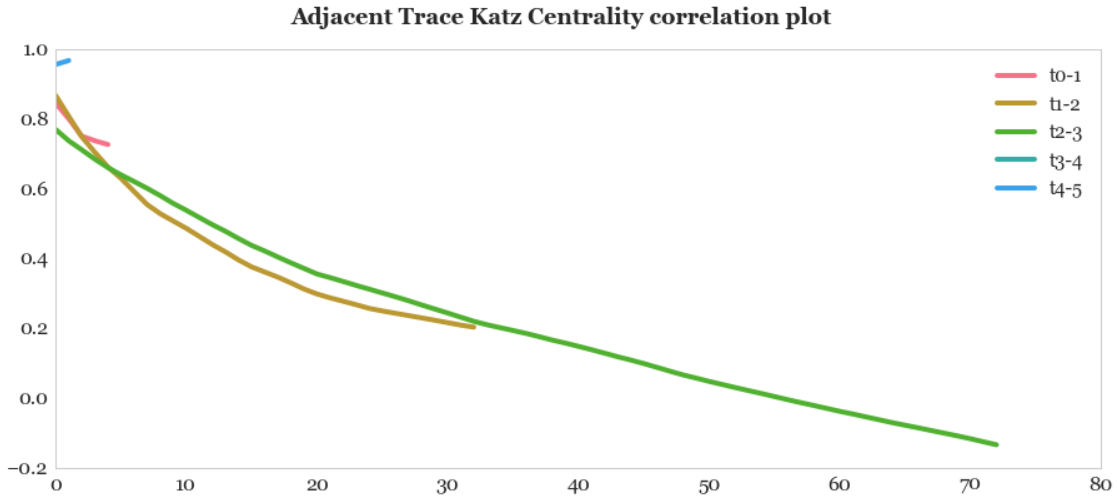
```
        plt.loglog(get_val(katzC5), label='t5')

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

```
Out[73]: <matplotlib.legend.Legend at 0x1ada76f8e10>
```



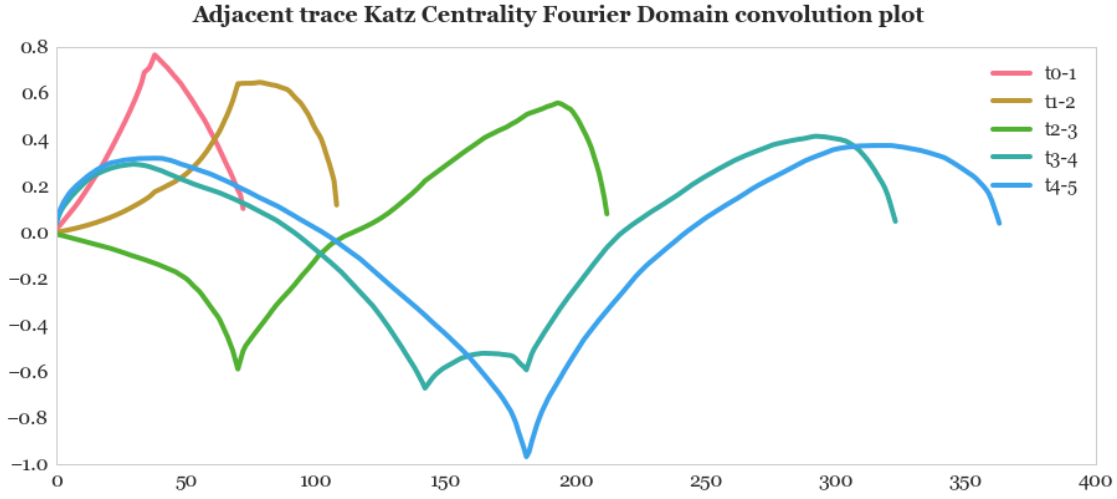

```
In [74]: plt.suptitle('Adjacent Trace Katz Centrality correlation plot', fontsize=1

        plt.plot(np.correlate(get_val(katzC0),get_val(katzC1)),label='t0-1')
        plt.plot(np.correlate(get_val(katzC1),get_val(katzC2)),label='t1-2')
        plt.plot(np.correlate(get_val(katzC2),get_val(katzC3)),label='t2-3')
        plt.plot(np.correlate(get_val(katzC3),get_val(katzC3)),label='t3-4')
        plt.plot(np.correlate(get_val(katzC4),get_val(katzC5)),label='t4-5')

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

```
Out[74]: <matplotlib.legend.Legend at 0x1ada7c6fb38>
```



In [75]:
```
plt.suptitle('Adjacent trace Katz Centrality Fourier Domain convolution pl

        plt.plot(fftconvolve(get_val(katzC0),get_val(katzC1)),label='t0-1')
        plt.plot(fftconvolve(get_val(katzC1),get_val(katzC2)),  label='t1-2')
        plt.plot(fftconvolve(get_val(katzC2),get_val(katzC3)), label='t2-3')
        plt.plot(fftconvolve(get_val(katzC3),get_val(katzC4)), label='t3-4')
        plt.plot(fftconvolve(get_val(katzC4),get_val(katzC5)),  label='t4-5')

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

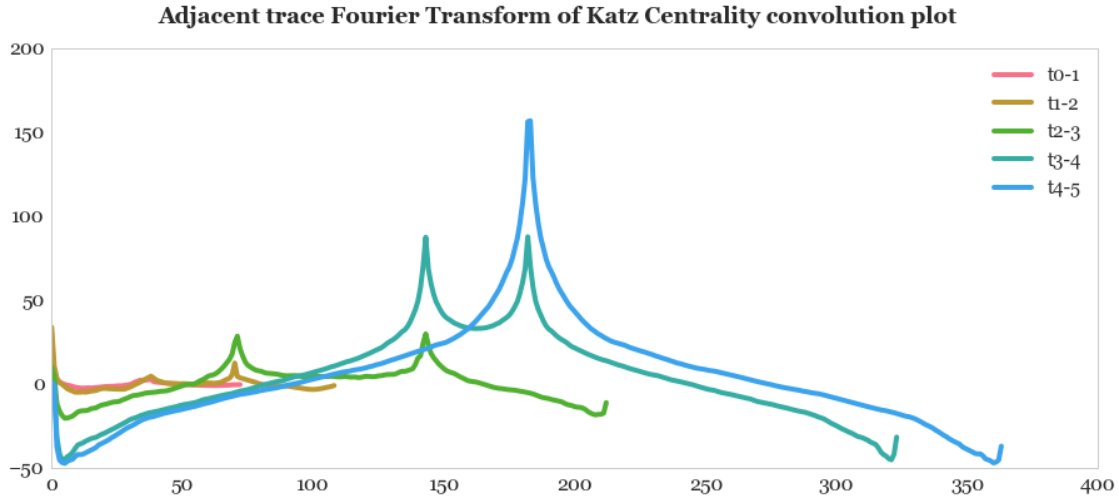
Out[75]: <matplotlib.legend.Legend at 0x1ada7e2d438>



In [76]:
```
plt.suptitle('Adjacent trace Fourier Transform of Katz Centrality convolut

        plt.plot(np.convolve(fft_sig(katzC0),fft_sig(katzC1)),label='t0-1')
        plt.plot(np.convolve(fft_sig(katzC1),fft_sig(katzC2)),label='t1-2')
        plt.plot(np.convolve(fft_sig(katzC2),fft_sig(katzC3)),label='t2-3')
        plt.plot(np.convolve(fft_sig(katzC3),fft_sig(katzC4)),label='t3-4')
        plt.plot(np.convolve(fft_sig(katzC4),fft_sig(katzC5)),label='t4-5')

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)
```

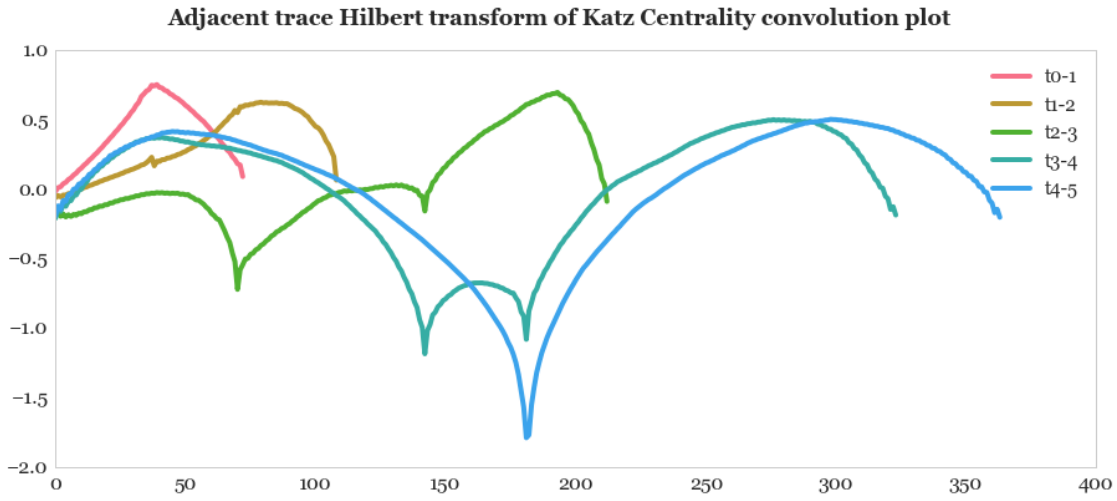
Out[76]: <matplotlib.legend.Legend at 0x1ada7ea71d0>



In [77]:
```
plt.suptitle('Adjacent trace Hilbert transform of Katz Centrality convolut

        plt.plot(np.convolve(hilbert_sig(katzC0), hilbert_sig(katzC1)), label='t0-
        plt.plot(np.convolve(hilbert_sig(katzC1), hilbert_sig(katzC2)), label='t1-
        plt.plot(np.convolve(hilbert_sig(katzC2), hilbert_sig(katzC3)),label='t2-3
        plt.plot(np.convolve(hilbert_sig(katzC3), hilbert_sig(katzC4)), label='t3-
        plt.plot(np.convolve(hilbert_sig(katzC4), hilbert_sig(katzC5)), label='t4-

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)
```

```
Out[77]: <matplotlib.legend.Legend at 0x1ada827ee48>
```

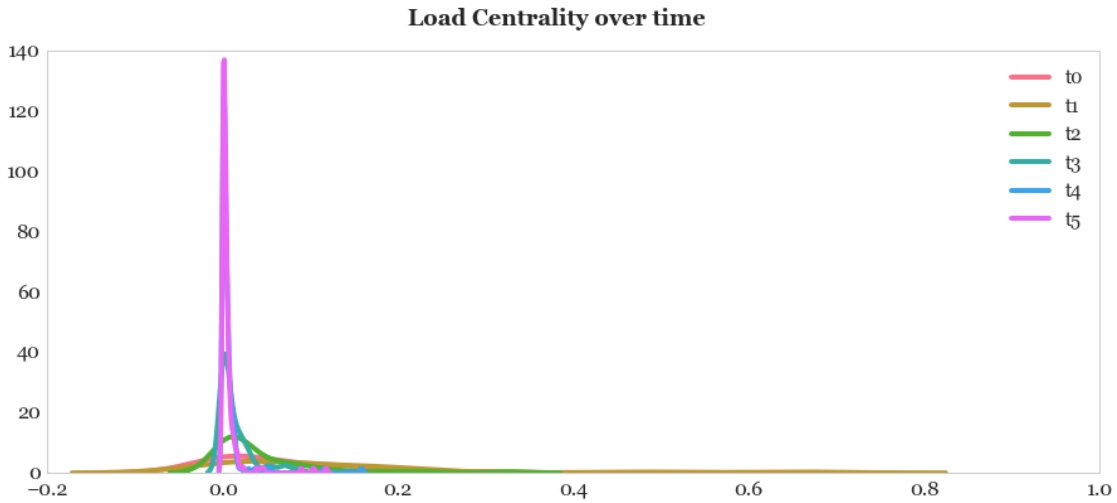


6.1.8 Load Centrality

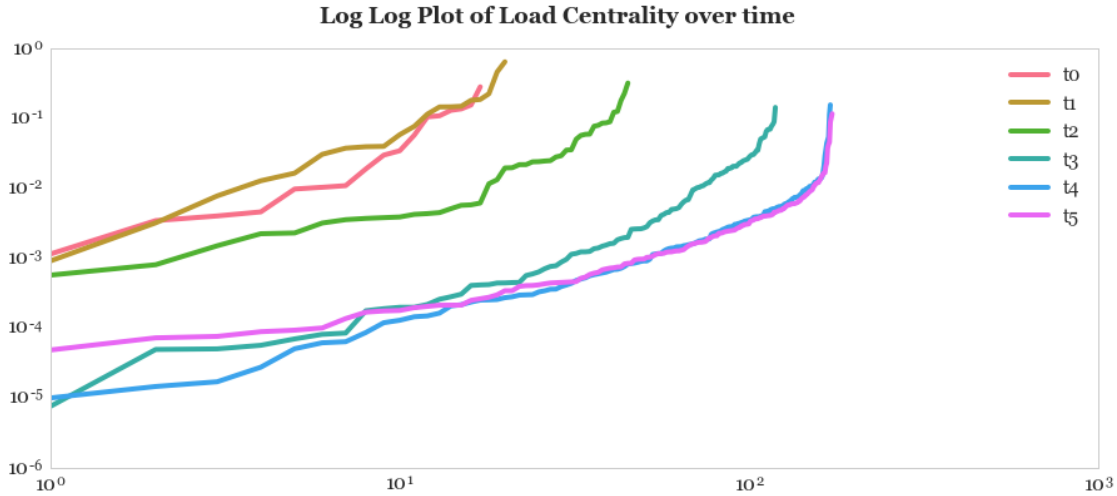
```
In [78]: plt.suptitle('Load Centrality over time', fontsize=18)

         sns.distplot(get_val(loadC0), hist=False, label='t0')
         sns.distplot(get_val(loadC1), hist=False, label='t1')
         sns.distplot(get_val(loadC2), hist=False, label='t2')
         sns.distplot(get_val(loadC3), hist=False, label='t3')
         sns.distplot(get_val(loadC4), hist=False, label='t4')
         sns.distplot(get_val(loadC5), hist=False, label='t5')

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

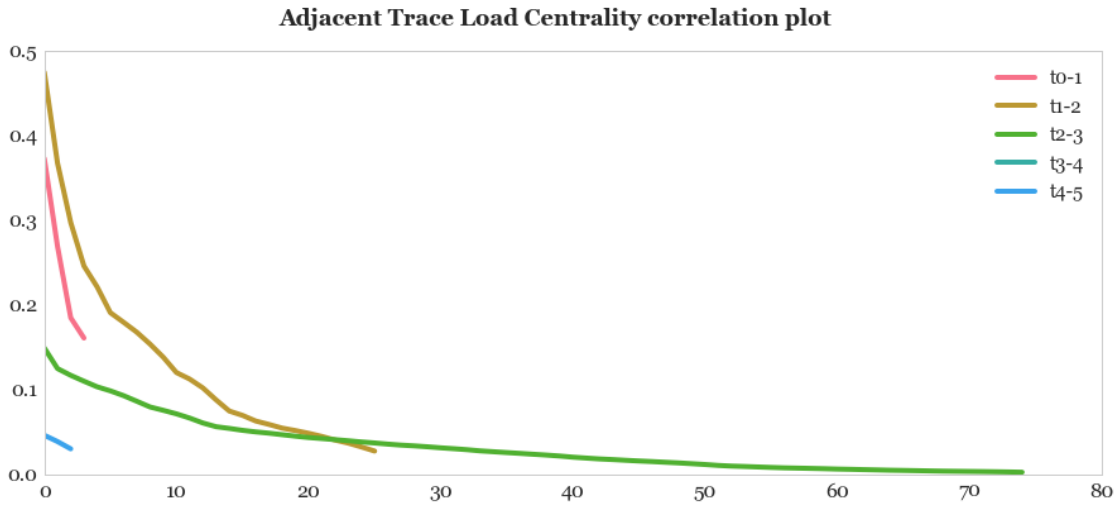
```
Out[78]: <matplotlib.legend.Legend at 0x1ada8453b70>
```



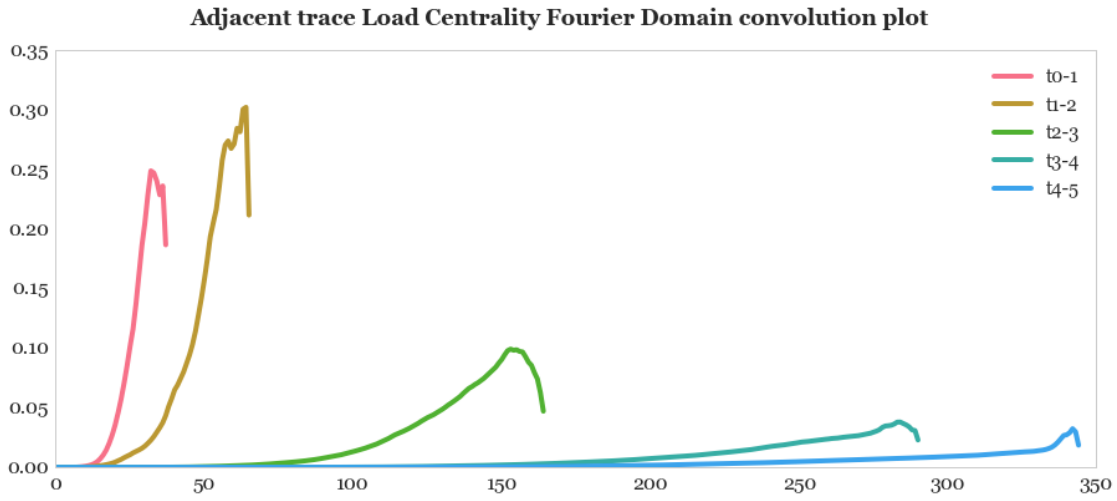

```
In [79]: plt.suptitle('Log Log Plot of Load Centrality over time', fontsize=18)

         plt.loglog(get_val(loadC0), label='t0')
         plt.loglog(get_val(loadC1), label='t1')
         plt.loglog(get_val(loadC2), label='t2')
         plt.loglog(get_val(loadC3), label='t3')
         plt.loglog(get_val(loadC4), label='t4')
         plt.loglog(get_val(loadC5), label='t5')

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)

Out[79]: <matplotlib.legend.Legend at 0x1ada869cdd8>
```



```
In [80]: plt.suptitle('Adjacent Trace Load Centrality correlation plot', fontsize=1

         plt.plot(np.correlate(get_val(loadC0),get_val(loadC1)),label='t0-1')
         plt.plot(np.correlate(get_val(loadC1),get_val(loadC2)),label='t1-2')
         plt.plot(np.correlate(get_val(loadC2),get_val(loadC3)),label='t2-3')
         plt.plot(np.correlate(get_val(loadC3),get_val(loadC3)),label='t3-4')
         plt.plot(np.correlate(get_val(loadC4),get_val(loadC5)),label='t4-5')

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)

Out[80]: <matplotlib.legend.Legend at 0x1ada8ab1a90>
```



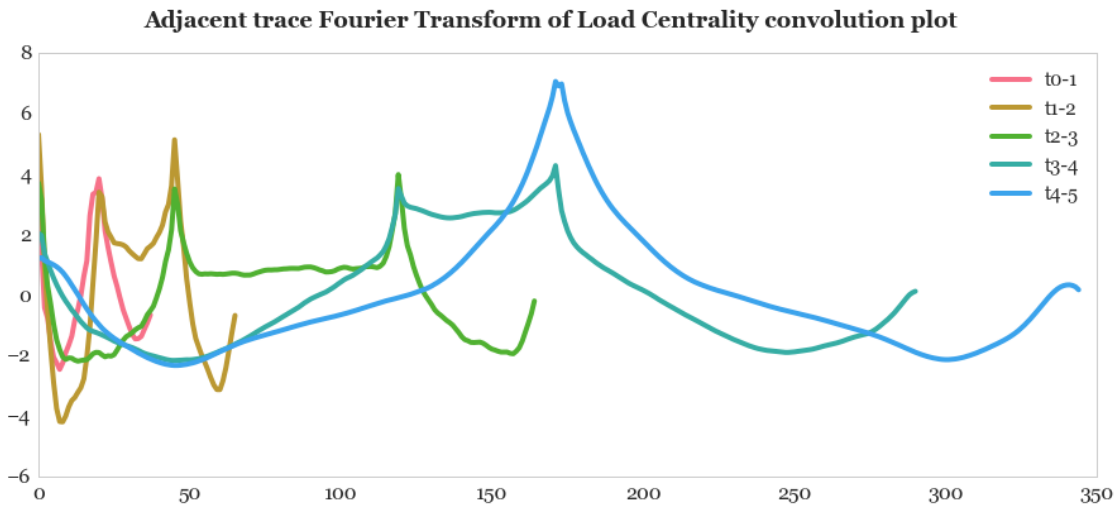

```
In [81]: plt.suptitle('Adjacent trace Load Centrality Fourier Domain convolution pl

         plt.plot(fftconvolve(get_val(loadC0),get_val(loadC1)),label='t0-1')
         plt.plot(fftconvolve(get_val(loadC1),get_val(loadC2)),  label='t1-2')
         plt.plot(fftconvolve(get_val(loadC2),get_val(loadC3)), label='t2-3')
         plt.plot(fftconvolve(get_val(loadC3),get_val(loadC4)), label='t3-4')
         plt.plot(fftconvolve(get_val(loadC4),get_val(loadC5)),  label='t4-5')

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)

Out[81]: <matplotlib.legend.Legend at 0x1ada8cb6cc0>
```



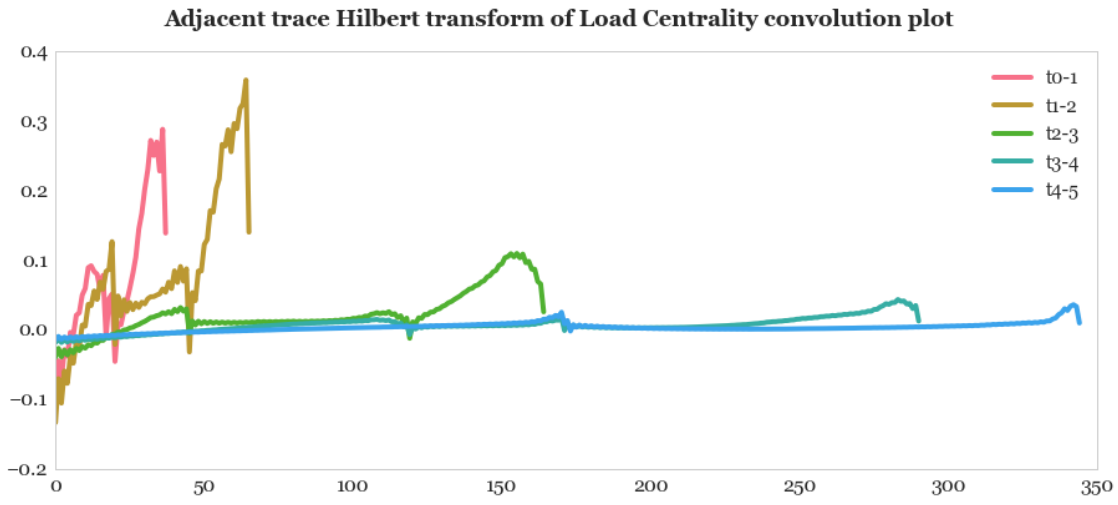
In [82]:
```
plt.suptitle('Adjacent trace Fourier Transform of Load Centrality convolut

        plt.plot(np.convolve(fft_sig(loadC0),fft_sig(loadC1)),label='t0-1')
        plt.plot(np.convolve(fft_sig(loadC1),fft_sig(loadC2)),label='t1-2')
        plt.plot(np.convolve(fft_sig(loadC2),fft_sig(loadC3)),label='t2-3')
        plt.plot(np.convolve(fft_sig(loadC3),fft_sig(loadC4)),label='t3-4')
        plt.plot(np.convolve(fft_sig(loadC4),fft_sig(loadC5)),label='t4-5')

        plt.yticks(fontsize=16)
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)
```

Out[82]: <matplotlib.legend.Legend at 0x1ada8ec6b70>



```
In [83]: plt.suptitle('Adjacent trace Hilbert transform of Load Centrality convolut

         plt.plot(np.convolve(hilbert_sig(loadC0), hilbert_sig(loadC1)), label='t0-
         plt.plot(np.convolve(hilbert_sig(loadC1), hilbert_sig(loadC2)), label='t1-
         plt.plot(np.convolve(hilbert_sig(loadC2), hilbert_sig(loadC3)),label='t2-3
         plt.plot(np.convolve(hilbert_sig(loadC3), hilbert_sig(loadC4)), label='t3-
         plt.plot(np.convolve(hilbert_sig(loadC4), hilbert_sig(loadC5)), label='t4-

         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(loc=1, fontsize=15)

C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)


Out[83]: <matplotlib.legend.Legend at 0x1ada8f4ce10>
```



6.2 Centrality Analysis with averaging

6.2.1 Calculate Centrality Statistics at different time steps

```
In [84]: deg0, clo0, bet0, eig0, alg0, clust0, commC0, katz0,load0 = cal_stat(Gt0)
         deg1, clo1, bet1, eig1, alg1, clust1, commC1, katz1,load1 = cal_stat(Gt1)
         deg2, clo2, bet2, eig2, alg2, clust2, commC2, katz2, load2 = cal_stat(Gt2)
         deg3, clo3, bet3, eig3, alg3, clust3, commC3, katz3,load3 = cal_stat(Gt3)
         deg4, clo4, bet4, eig4, alg4, clust4, commC4, katz4,load4 = cal_stat(Gt4)
         deg5, clo5, bet5, eig5, alg5, clust5, commC5, katz5, load5 = cal_stat(Gt5)
```

```
In [85]: print('Avg Deg Cent',deg0)
         print('Avg Clo Cent',clo0)
         print('Avg Bet Cent',bet0)
         print('Avg Eig Cent',eig0)
         print('Avg Alg Connectivity',alg0)
         print('Avg Clustering Coeff',clust0)
         print('Avg Communicability Cent',commC0)
         print('Avg Katz Cent',katz0)
         print('Avg Load Cent', load0)

Avg Deg Cent 0.018826135105204873
Avg Clo Cent 0.10466121433401401
Avg Bet Cent 0.02598384787834589
Avg Eig Cent 0.07773852412753667
Avg Alg Connectivity 0.0
Avg Clustering Coeff 0.026004228329809725
Avg Communicability Cent 3.9809326044505062
Avg Katz Cent 0.12524033960814704
Avg Load Cent 0.02598384787834589


In [86]: print('Avg Deg Cent',deg1)
         print('Avg Clo Cent',clo1)
         print('Avg Bet Cent',bet1)
         print('Avg Eig Cent',eig1)
         print('Avg Alg Connectivity',alg1)
         print('Avg Clustering Coeff',clust1)
         print('Avg Communicability Cent',commC1)
         print('Avg Katz Cent',katz1)

Avg Deg Cent 0.021719858156028365
Avg Clo Cent 0.16550053458700076
Avg Bet Cent 0.05436709836571077
Avg Eig Cent 0.08950640258093145
Avg Alg Connectivity 0.0
Avg Clustering Coeff 0.08708721833721834
Avg Communicability Cent 5.874869893519794
Avg Katz Cent 0.11654349936679959


In [87]: stat_df = pd.DataFrame([deg0,deg1,deg2,deg3,deg4,deg5])

In [88]: #calculate density
         den0 = nx.density(Gt0)
         den1 = nx.density(Gt1)
         den2 = nx.density(Gt2)
         den3 = nx.density(Gt3)
         den4 = nx.density(Gt4)
         den5 = nx.density(Gt5)
```

```
In [89]: stat_df['Closeness'] = pd.DataFrame([clo0,clo1,clo2,clo3,clo4,clo5])
         stat_df['Betweeness'] = pd.DataFrame([bet0,bet1,bet2,bet3,bet4,bet5])
         stat_df['Eig'] = pd.DataFrame([eig0,eig1,eig2,eig3,eig4,eig5])
         stat_df['AlgConnect'] = pd.DataFrame([alg0,alg1,alg2,alg3,alg4,alg5])
         stat_df['ClustCoeff'] = pd.DataFrame([clust0,clust1,clust2,clust3,clust4,c
         stat_df['Communicability'] = pd.DataFrame([commC0,commC1,commC2,commC3,com
         stat_df['Katz'] = pd.DataFrame([katz0,katz1,katz2,katz3,katz4,katz5])
         stat_df['Load']=pd.DataFrame([load0,load1,load2,load3,load4,load5])
         stat_df['Density'] = pd.DataFrame([den0,den1,den2,den3,den4,den5])

In [90]: stat_df.columns.values[0]='Deg'

In [91]: stat_df.head()

Out[91]:         Deg  Closeness  Betweeness       Eig  AlgConnect  ClustCoeff  \
         0  0.018826   0.104661    0.025984  0.077739    0.000000    0.026004
         1  0.021720   0.165501    0.054367  0.089506    0.000000    0.087087
         2  0.020570   0.223937    0.025495  0.066241    0.000000    0.185696
         3  0.031813   0.253585    0.012739  0.056478    0.121915    0.462544
         4  0.047889   0.276022    0.006152  0.058781    0.000000    0.493717

            Communicability      Katz      Load   Density
         0     3.980933e+00  0.125240  0.025984  0.057586
         1     5.874870e+00  0.116543  0.054367  0.057624
         2     1.206113e+02  0.076152  0.025495  0.054430
         3     6.140791e+06  0.011703  0.012739  0.081651
         4     1.009380e+12  0.003815  0.006151  0.123401

In [92]: stat_df.mean()

Out[92]: Deg                3.167631e-02
         Closeness          2.157075e-01
         Betweeness         2.176173e-02
         Eig                6.800322e-02
         AlgConnect         2.031923e-02
         ClustCoeff         2.911402e-01
         Communicability    1.302104e+12
         Katz               5.675227e-02
         Load               2.176157e-02
         Density            8.438583e-02
         dtype: float64

In [93]: stat_df.std()

Out[93]: Deg                1.384896e-02
         Closeness          6.782144e-02
         Betweeness         1.829567e-02
         Eig                1.307165e-02
         AlgConnect         4.977175e-02
```

```
ClustCoeff          2.162089e-01
Communicability     2.725070e+12
Katz                5.646791e-02
Load                1.829584e-02
Density             3.490777e-02
dtype: float64
```

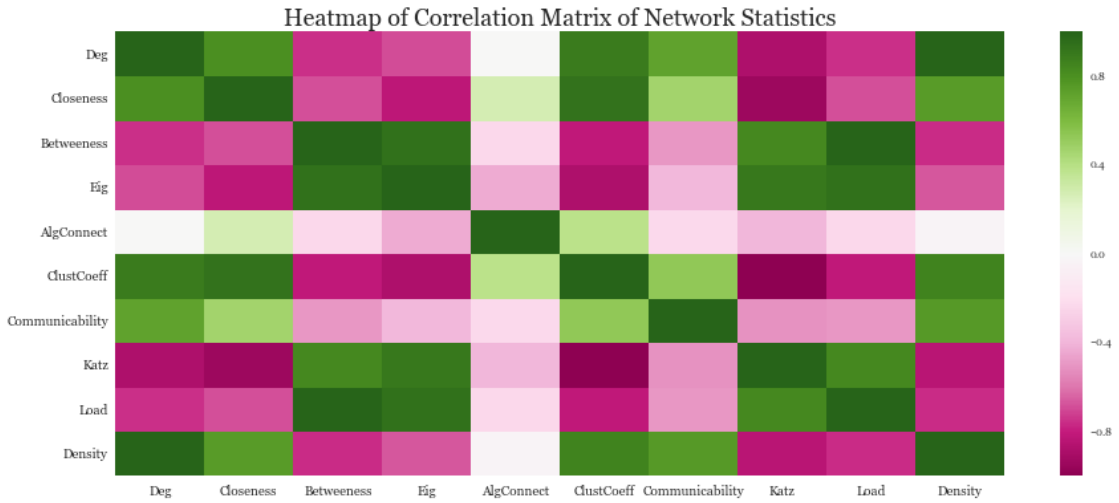
In [94]:
```
corr_stat = stat_df.corr()
corr_stat
```

Out[94]:

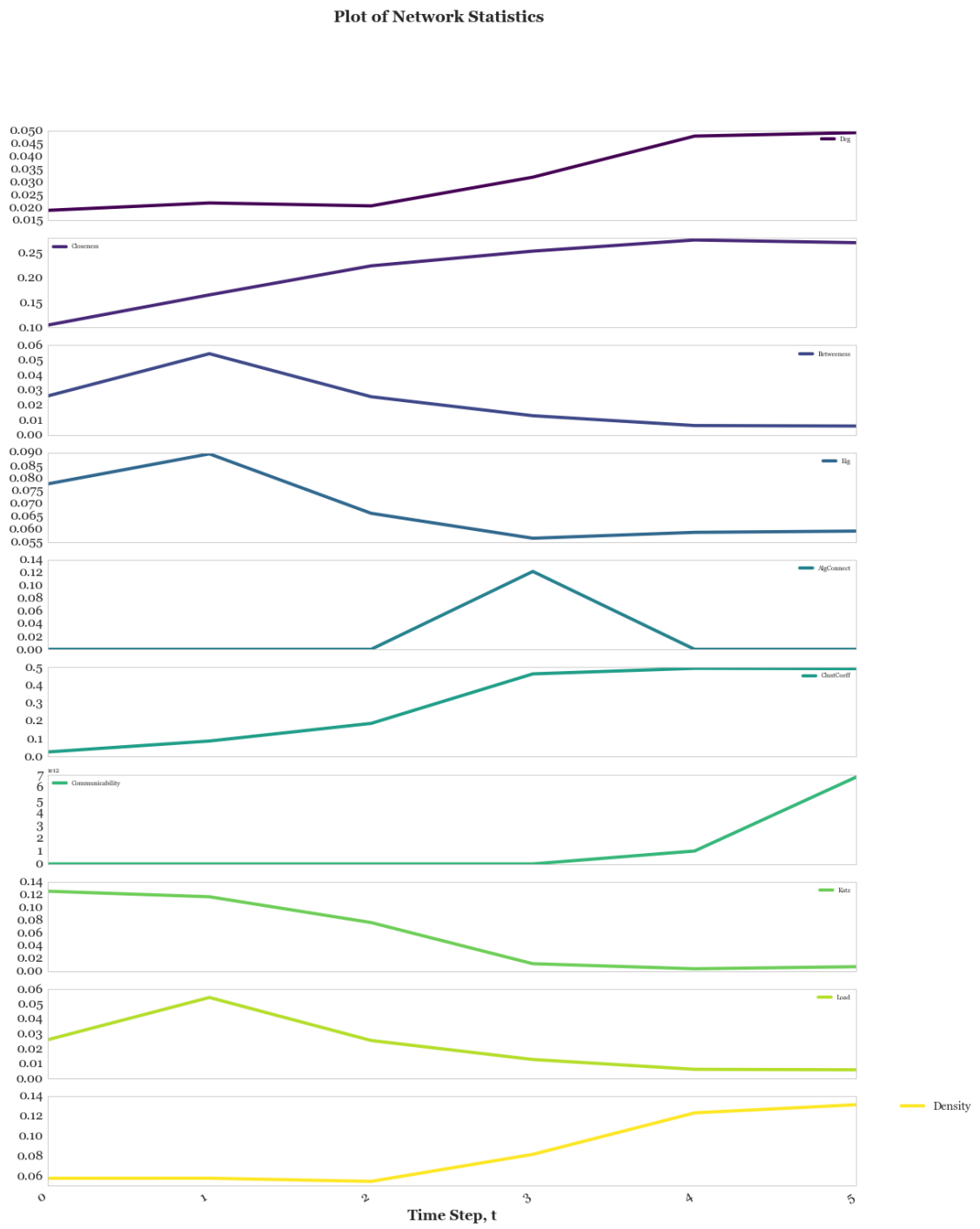
	Deg	Closeness	Betweeness	Eig	AlgConnect	ClustCoeff	Communicability	Katz	Load	Density
Deg	1.000000	0.805217	-0.753366	-0.689411	0.004844	0.897836	0.719954	-0.876722	-0.753372	0.995111
Closeness	0.805217	1.000000	-0.682081	-0.822651	0.273602	0.934724	0.469556	-0.943750	-0.682084	0.752929
Betweeness	-0.753366	-0.682081	1.000000	0.938821	-0.241596	-0.812555	-0.497897	0.839290	1.000000	-0.762315
Eig	-0.689411	-0.822651	0.938821	1.000000	-0.431955	-0.875959	-0.385656	0.910251	0.938820	-0.668044
AlgConnect	0.004844	0.273602	-0.241596	-0.431955	1.000000	0.388376	-0.234084	-0.390831	-0.241589	-0.038384
ClustCoeff	0.897836	0.934724	-0.812555	-0.875959	0.388376	1.000000	0.532793	-0.996347	-0.812557	0.864887
Communicability	0.719954	0.469556	-0.497897	-0.385656	-0.234084	0.532793	1.000000	-0.508849	-0.497902	0.758456
Katz	-0.876722	-0.943750	0.839290	0.910251	-0.390831	-0.996347	-0.508849	1.000000	0.839292	-0.842771
Load	-0.753372	-0.682084	1.000000	0.938820	-0.241589	-0.812557	-0.497902	0.839292	1.000000	-0.762321
Density	0.995111	0.752929	-0.762315	-0.668044	-0.038384	0.864887	0.758456	-0.842771	-0.762321	1.000000

In [95]:
```
sns.heatmap(corr_stat, cmap='PiYG')
plt.title('Heatmap of Correlation Matrix of Network Statistics', fontsize=
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
```

Out[95]:
```
(array([ 0.5,  1.5,  2.5,  3.5,  4.5,  5.5,  6.5,  7.5,  8.5,  9.5]),
 <a list of 10 Text yticklabel objects>)
```



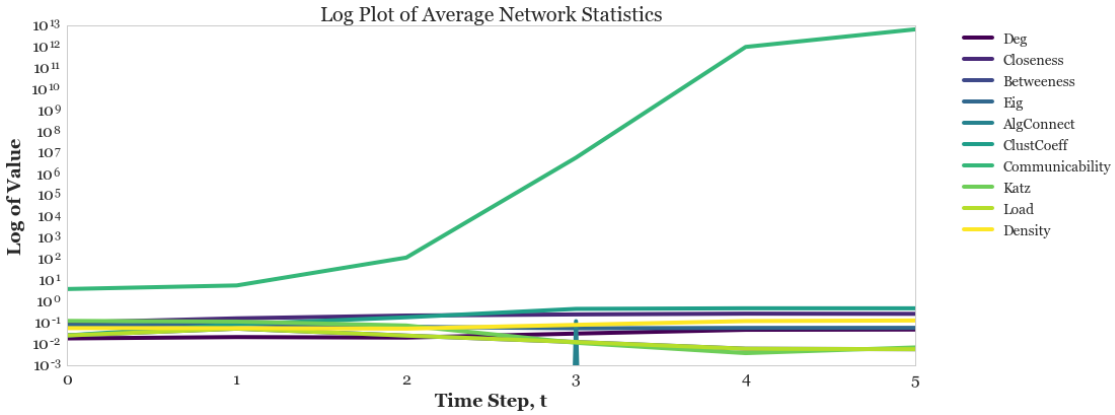

```
In [163]: stat_df.plot(colormap='viridis', fontsize=16, subplots=True, figsize=(18,
          plt.suptitle('Plot of Network Statistics', fontsize=20)
          plt.yticks(fontsize=16)
          plt.xticks(fontsize=16)
          plt.legend(fontsize=14, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
          plt.xlabel("Time Step, t", fontsize=18)

Out[163]: <matplotlib.text.Text at 0x1adc8471eb8>
```




```
In [97]: stat_df.plot(colormap='viridis',logy=True, fontsize=14)
         plt.title('Log Plot of Average Network Statistics', fontsize=20)
         plt.yticks(fontsize=16)
         plt.xticks(fontsize=16)
         plt.legend(fontsize=14, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

```
        plt.xlabel("Time Step, t", fontsize=18)
        plt.ylabel("Log of Value", fontsize=18)
```

```
Out[97]: <matplotlib.text.Text at 0x1ada53571d0>
```



7 Assortativity Analysis

```
In [98]: def cal_assort_att(net):
             dac = nx.degree_assortativity_coefficient(net)
             dpc = nx.degree_pearson_correlation_coefficient(net)
             avg_neigdeg = nx.average_neighbor_degree(net)
             avg_degconnect = nx.average_degree_connectivity(net)
             triangles = nx.triangles(net)

             return [dac,dpc,avg_neigdeg, avg_degconnect,triangles]
```

```
In [99]: dac0 = nx.degree_assortativity_coefficient(Gt0)
```

```
In [100]: nx.degree_pearson_correlation_coefficient(Gt0)
```

```
Out[100]: -0.25126403290770427
```

7.1 Calculate Assortativity statistics for each time step

```
In [101]: dac0,dpc0,avg_neigdeg0, avg_degconnect0, triangles0= cal_assort_att(Gt0)
          dac1,dpc1,avg_neigdeg1, avg_degconnect1, triangles1= cal_assort_att(Gt1)
          dac2,dpc2,avg_neigdeg2, avg_degconnect2, triangles2= cal_assort_att(Gt2)
          dac3,dpc3,avg_neigdeg3, avg_degconnect3, triangles3= cal_assort_att(Gt3)
          dac4,dpc4,avg_neigdeg4, avg_degconnect4, triangles4= cal_assort_att(Gt4)
          dac5,dpc5,avg_neigdeg5, avg_degconnect5, triangles5= cal_assort_att(Gt5)
```

```
In [102]: avg_cent(avg_neigdeg0)
```

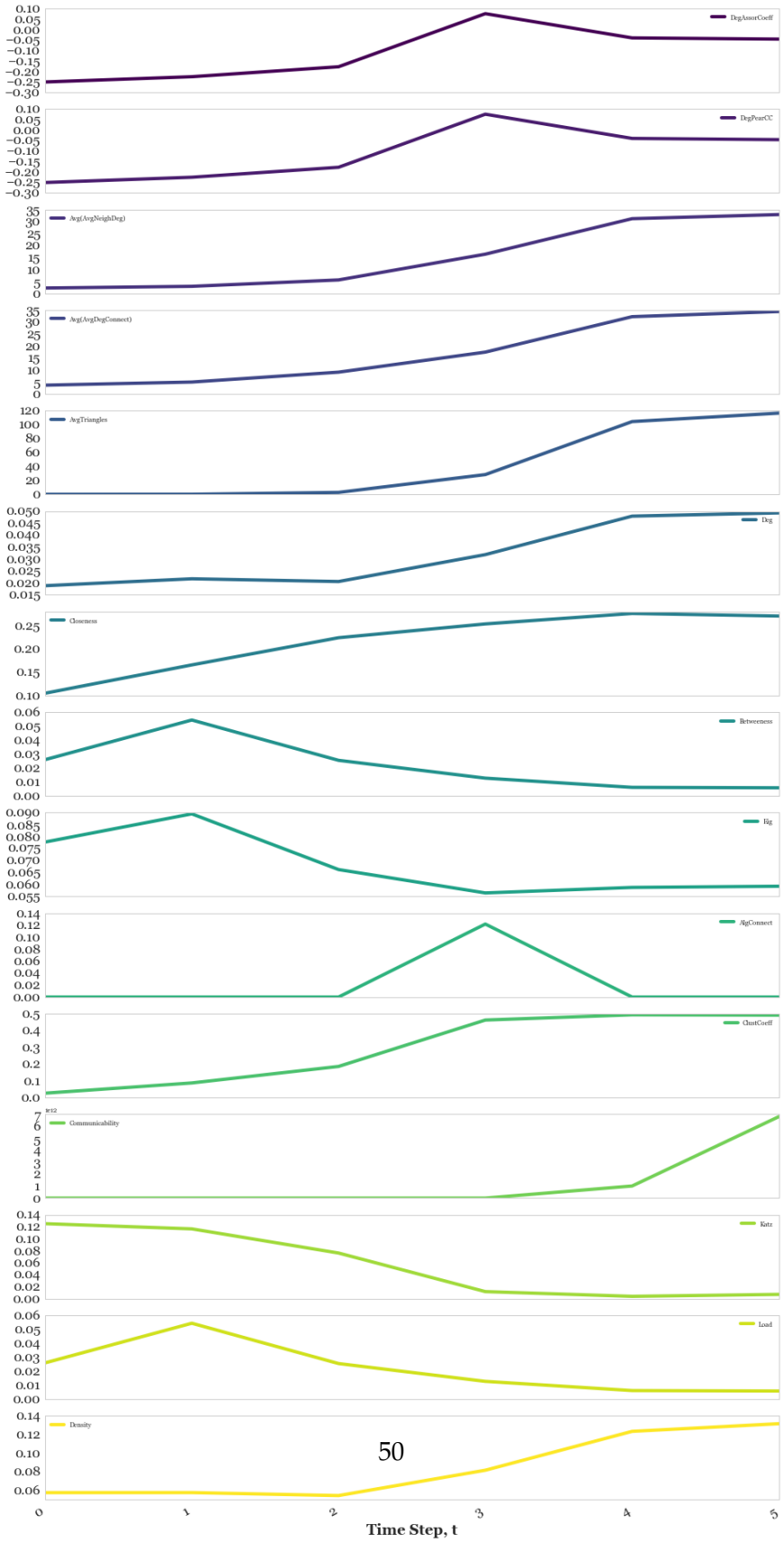
```
Out[102]: 2.190131186642814

In [103]: assor_df = pd.DataFrame([dac0,dac1,dac2,dac3,dac4,dac5])
          assor_df['DegPearCC'] = pd.DataFrame([dpc0,dpc1,dpc2,dpc3,dpc4,dpc5])

In [104]: assor_df['Avg(AvgNeighDeg)'] = pd.DataFrame([avg_cent(avg_neigdeg0),avg_c
                                                       avg_cent(avg_neigdeg2),avg_c
                                                       avg_cent(avg_neigdeg4), avg_

In [105]: assor_df['Avg(AvgDegConnect)'] = pd.DataFrame([avg_cent(avg_degconnect0),
                                                         avg_cent(avg_degconnect2),av
                                                         avg_cent(avg_degconnect4), a

In [106]: assor_df['AvgTriangles'] = pd.DataFrame([avg_cent(triangles0),avg_cent(tr
                                                   avg_cent(triangles2),avg_cen
                                                   avg_cent(triangles4), avg_ce

In [107]: assor_df.columns.values[0]= 'DegAssorCoeff'
          assor_df

Out[107]:    DegAssorCoeff  DegPearCC  Avg(AvgNeighDeg)  Avg(AvgDegConnect)  \
          0      -0.251264  -0.251264          2.190131            3.600092
          1      -0.226108  -0.226108          2.896955            4.906208
          2      -0.178525  -0.178525          5.563177            9.037252
          3       0.074870   0.074870         16.332357           17.412081
          4      -0.040770  -0.040770         31.159027           32.211278
          5      -0.046592  -0.046592         32.873515           34.366099

             AvgTriangles
          0      0.023256
          1      0.125000
          2      2.762500
          3     28.272727
          4    104.180328
          5    116.304348

In [108]: stat_df2 = assor_df.join(stat_df)

In [109]: stat_df2.head()

Out[109]:    DegAssorCoeff  DegPearCC  Avg(AvgNeighDeg)  Avg(AvgDegConnect)  \
          0      -0.251264  -0.251264          2.190131            3.600092
          1      -0.226108  -0.226108          2.896955            4.906208
          2      -0.178525  -0.178525          5.563177            9.037252
          3       0.074870   0.074870         16.332357           17.412081
          4      -0.040770  -0.040770         31.159027           32.211278

             AvgTriangles       Deg  Closeness  Betweeness       Eig  AlgConnect  \
          0      0.023256  0.018826   0.104661    0.025984  0.077739    0.000000
```

```
1      0.125000  0.021720   0.165501    0.054367  0.089506    0.000000
2      2.762500  0.020570   0.223937    0.025495  0.066241    0.000000
3     28.272727  0.031813   0.253585    0.012739  0.056478    0.121915
4    104.180328  0.047889   0.276022    0.006152  0.058781    0.000000

   ClustCoeff  Communicability      Katz      Load   Density
0    0.026004     3.980933e+00  0.125240  0.025984  0.057586
1    0.087087     5.874870e+00  0.116543  0.054367  0.057624
2    0.185696     1.206113e+02  0.076152  0.025495  0.054430
3    0.462544     6.140791e+06  0.011703  0.012739  0.081651
4    0.493717     1.009380e+12  0.003815  0.006151  0.123401
```

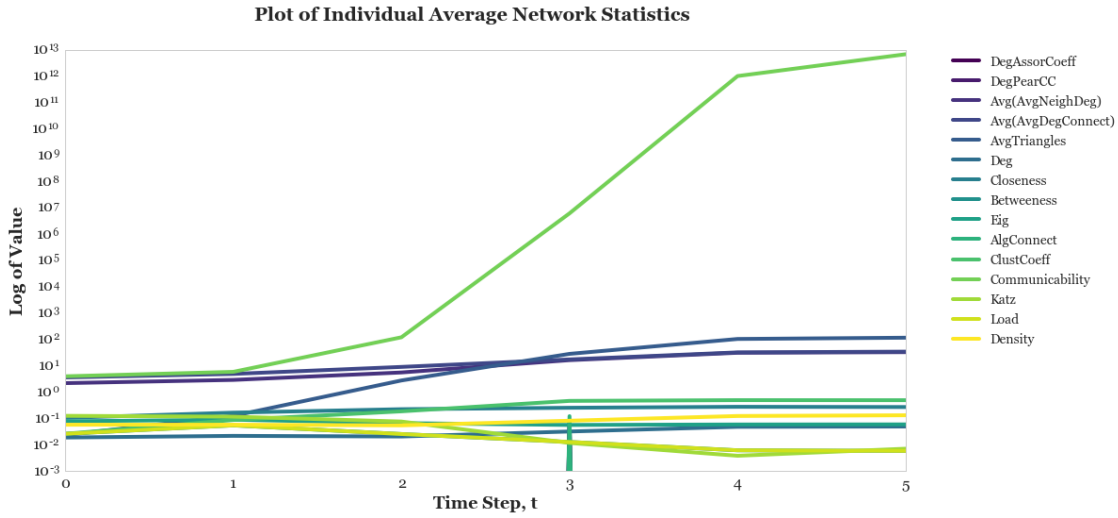
In [168]:
```
stat_df2.plot(colormap='viridis', figsize=(16,36), fontsize=16, sharex=Tr
plt.suptitle('Plot of Individual Average Network Statistics', fontsize=20
#plt.legend(fontsize=14, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0
plt.xlabel("Time Step, t", fontsize=18)
#plt.ylabel("Log of Value", fontsize=18)
```

Out[168]:
```
<matplotlib.text.Text at 0x1adcb3ae278>
```

Plot of Individual Average Network Statistics



```
In [110]: stat_df2.plot(colormap='viridis', figsize=(16,8), fontsize=16, sharex=Tru
          plt.suptitle('Plot of Individual Average Network Statistics', fontsize=20
          plt.legend(fontsize=14, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
          plt.xlabel("Time Step, t", fontsize=18)
          plt.ylabel("Log of Value", fontsize=18)

Out[110]: <matplotlib.text.Text at 0x1adaab3f6d8>
```

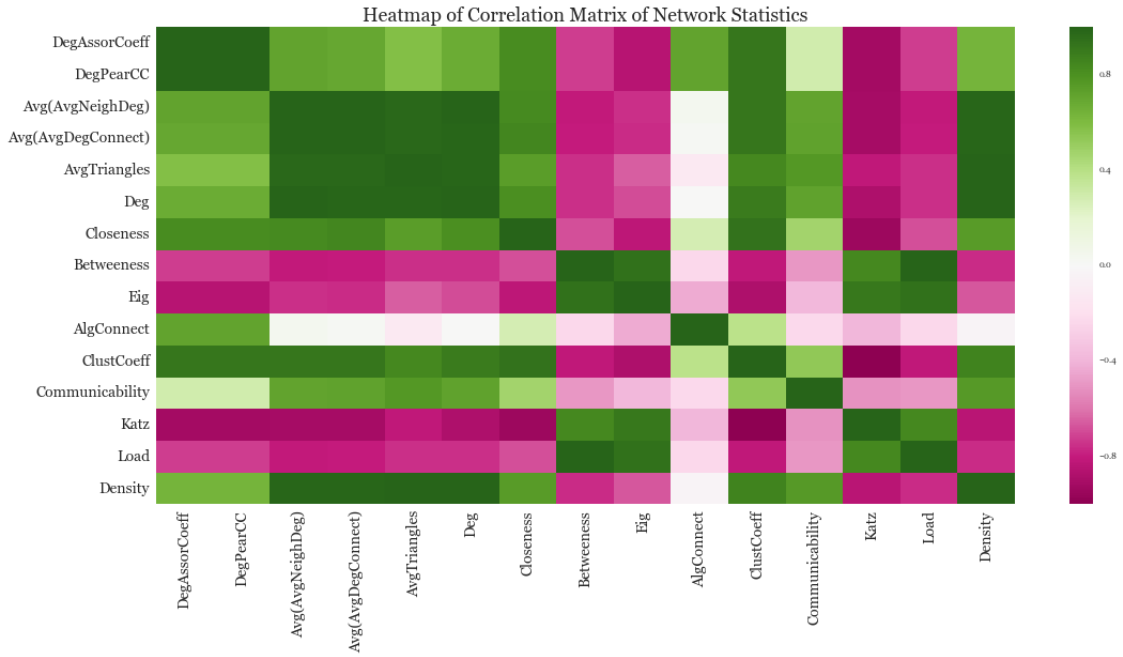



```
In [111]: stat_df2.describe()

Out[111]:        DegAssorCoeff  DegPearCC  Avg(AvgNeighDeg)  Avg(AvgDegConnect)  \
          count       6.000000   6.000000          6.000000           6.000000
          mean       -0.111398  -0.111398         15.169194          16.922168
          std         0.127366   0.127366         14.012924          13.581757
          min        -0.251264  -0.251264          2.190131           3.600092
          25%        -0.214212  -0.214212          3.563511           5.938969
          50%        -0.112558  -0.112558         10.947767          13.224666
          75%        -0.042225  -0.042225         27.452360          28.511479
          max         0.074870   0.074870         32.873515          34.366099

                 AvgTriangles       Deg  Closeness  Betweeness       Eig  AlgConnec
          count      6.000000  6.000000   6.000000    6.000000  6.000000    6.00000
          mean      41.944693  0.031676   0.215707    0.021762  0.068003    0.02031
          std       54.094534  0.013849   0.067821    0.018296  0.013072    0.04977
          min        0.023256  0.018826   0.104661    0.005834  0.056478    0.00000
          25%        0.784375  0.020857   0.180110    0.007798  0.058904    0.00000
          50%       15.517614  0.026767   0.238761    0.019117  0.062758    0.00000
```

```
75%        85.203428   0.043870    0.266301     0.025862   0.074864     0.00000
max       116.304348   0.049240    0.276022     0.054367   0.089506     0.12191

       ClustCoeff  Communicability      Katz      Load   Density
count    6.000000     6.000000e+00  6.000000  6.000000  6.000000
mean     0.291140     1.302104e+12  0.056752  0.021762  0.084386
std      0.216209     2.725070e+12  0.056468  0.018296  0.034908
min      0.026004     3.980933e+00  0.003815  0.005833  0.054430
25%      0.111739     3.455898e+01  0.008220  0.007798  0.057595
50%      0.324120     3.070456e+06  0.043928  0.019117  0.069637
75%      0.484481     7.570362e+11  0.106446  0.025862  0.112964
max      0.493717     6.803237e+12  0.125240  0.054367  0.131623
```

In [112]:
```
plt.figure(figsize=(18,8))
sns.heatmap(stat_df2.corr(), cmap='PiYG')
plt.title('Heatmap of Correlation Matrix of Network Statistics', fontsize
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
```

Out[112]:
```
(array([  0.5,   1.5,   2.5,   3.5,   4.5,   5.5,   6.5,   7.5,   8.5,
          9.5,  10.5,  11.5,  12.5,  13.5,  14.5]),
 <a list of 15 Text yticklabel objects>)
```

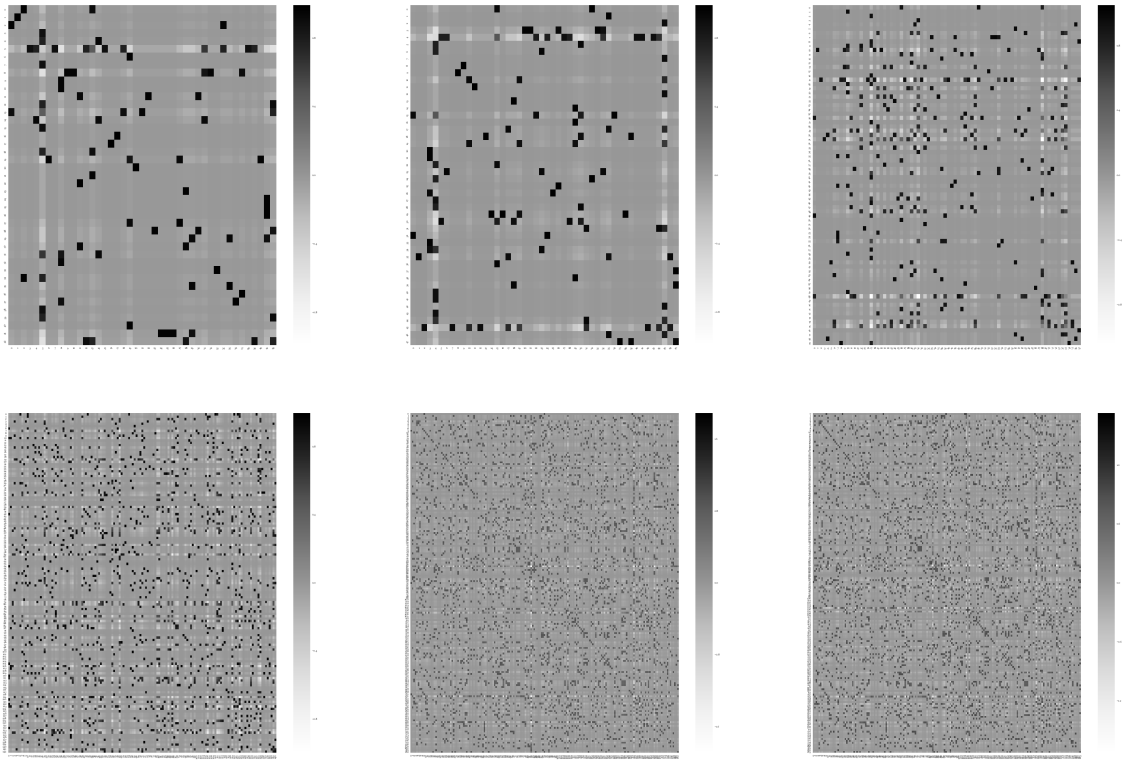


8 Graph Spectra

8.1 Modularity Matrix

```
In [113]: modM0 = nx.modularity_matrix(Gt0)
          modM1 = nx.modularity_matrix(Gt1)
          modM2 = nx.modularity_matrix(Gt2)
          modM3 = nx.modularity_matrix(Gt3)
          modM4 = nx.modularity_matrix(Gt4)
          modM5 = nx.modularity_matrix(Gt5)

In [114]: plt.figure(figsize=(64, 42))

          plt.subplot(231)
          sns.heatmap(modM0,cmap='Greys')
          plt.suptitle('Heatmap of Modularity Matrix, t0-t5', fontsize=80)

          plt.subplot(232)
          sns.heatmap(modM1,cmap='Greys')

          plt.subplot(233)
          sns.heatmap(modM2,cmap='Greys')

          plt.subplot(234)
          sns.heatmap(modM3,cmap='Greys')

          plt.subplot(235)
          sns.heatmap(modM4,cmap='Greys')

          plt.subplot(236)
          sns.heatmap(modM5,cmap='Greys')

Out[114]: <matplotlib.axes._subplots.AxesSubplot at 0x1adaa62d198>
```

Heatmap of Modularity Matrix, to-t5



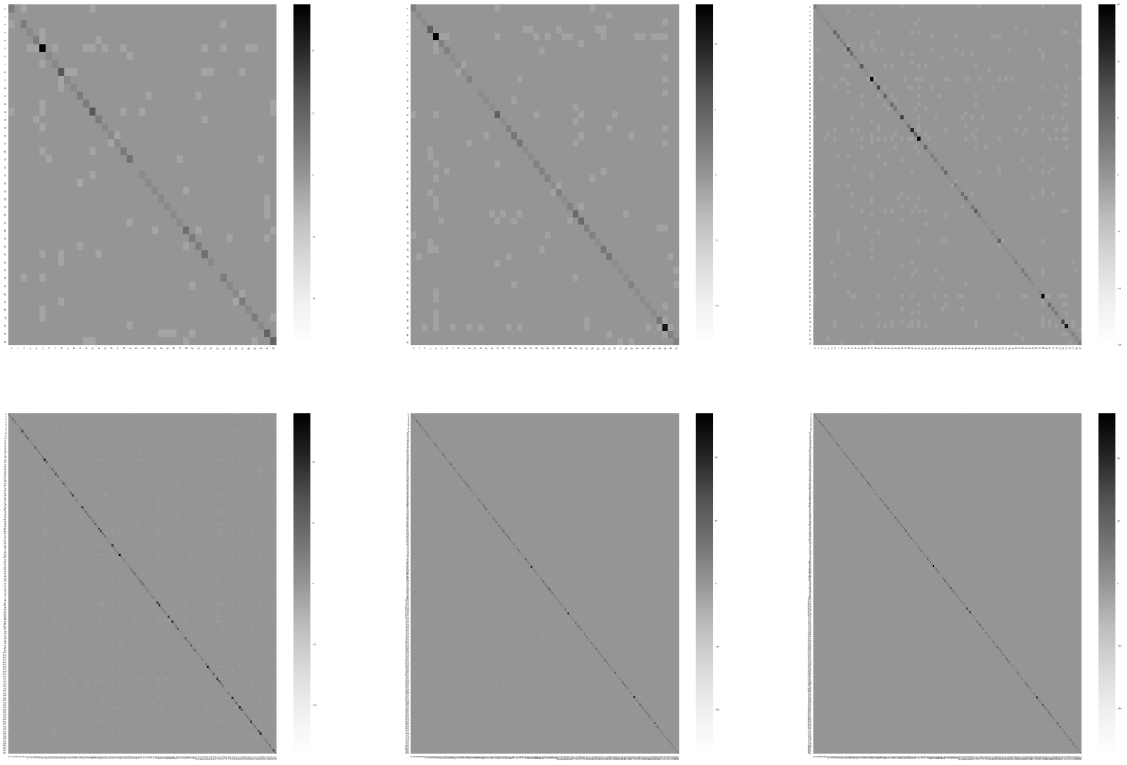
8.2 Laplacian Matrix

```
In [115]: lapM0 = nx.laplacian_matrix(Gt0).todense()
          lapM1 = nx.laplacian_matrix(Gt1).todense()
          lapM2 = nx.laplacian_matrix(Gt2).todense()
          lapM3 = nx.laplacian_matrix(Gt3).todense()
          lapM4 = nx.laplacian_matrix(Gt4).todense()
          lapM5 = nx.laplacian_matrix(Gt5).todense()

In [116]: plt.figure(figsize=(64, 42))

          plt.subplot(231)
          sns.heatmap(lapM0,cmap='Greys')
          plt.suptitle('Heatmap of Laplacian Matrix, t0-t5', fontsize=80)

          plt.subplot(232)
          sns.heatmap(lapM1,cmap='Greys')

          plt.subplot(233)
          sns.heatmap(lapM2,cmap='Greys')
```

```
plt.subplot(234)
sns.heatmap(lapM3,cmap='Greys')

plt.subplot(235)
sns.heatmap(lapM4,cmap='Greys')

plt.subplot(236)
sns.heatmap(lapM5,cmap='Greys')
```

```
Out[116]: <matplotlib.axes._subplots.AxesSubplot at 0x1adade1d7f0>
```

Heatmap of Laplacian Matrix, to-t5



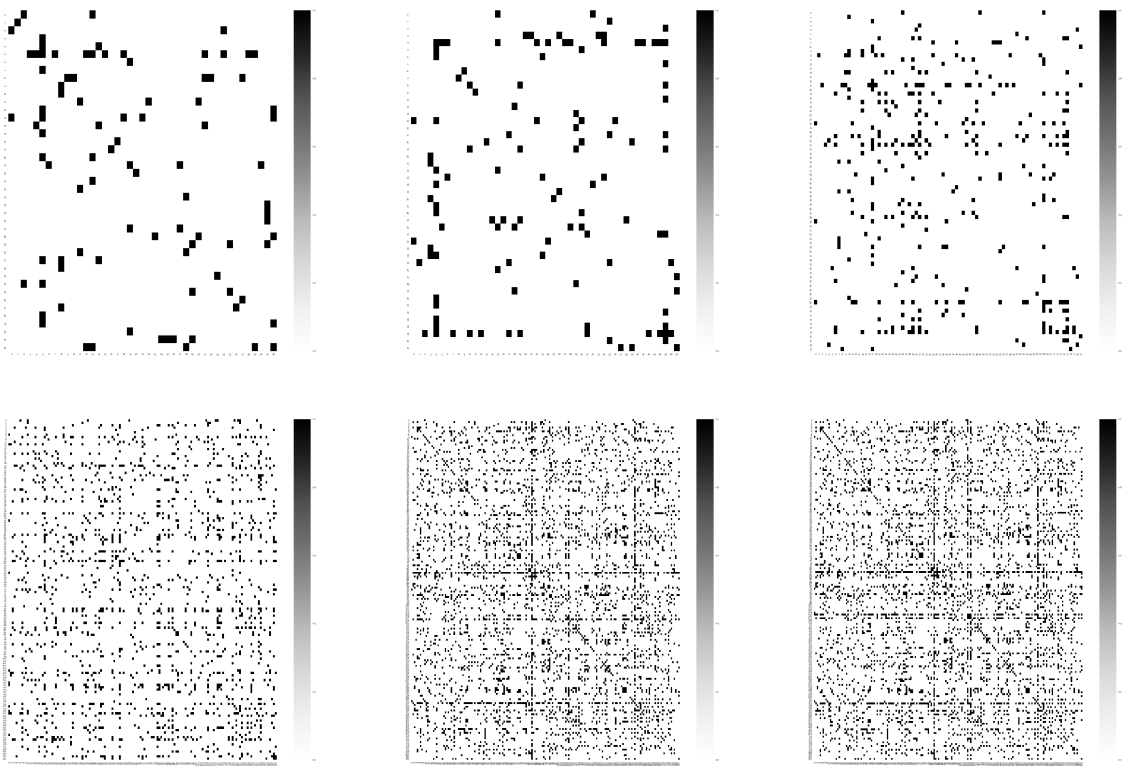
8.3 Adjacency Matrix

```
In [117]: adjM0 = nx.adjacency_matrix(Gt0).todense()
          adjM1 = nx.adjacency_matrix(Gt1).todense()
          adjM2 = nx.adjacency_matrix(Gt2).todense()
          adjM3 = nx.adjacency_matrix(Gt3).todense()
          adjM4 = nx.adjacency_matrix(Gt4).todense()
          adjM5 = nx.adjacency_matrix(Gt5).todense()
```

```
In [118]: plt.figure(figsize=(64, 42))
```

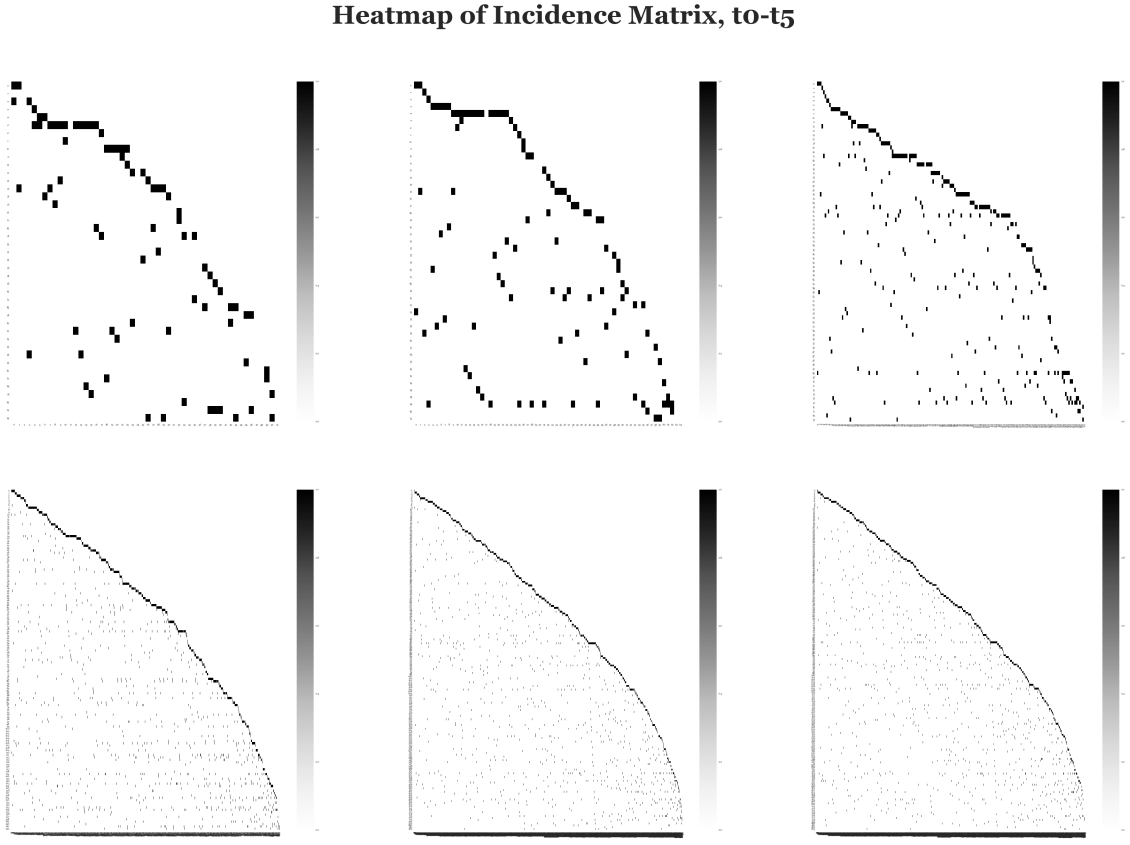
```
plt.subplot(231)
sns.heatmap(adjM0,cmap='Greys')
plt.suptitle('Heatmap of Adjacency Matrix, t0-t5', fontsize=80)

plt.subplot(232)
sns.heatmap(adjM1,cmap='Greys')

plt.subplot(233)
sns.heatmap(adjM2,cmap='Greys')

plt.subplot(234)
sns.heatmap(adjM3,cmap='Greys')

plt.subplot(235)
sns.heatmap(adjM4,cmap='Greys')

plt.subplot(236)
sns.heatmap(adjM5,cmap='Greys')
```

```
Out[118]: <matplotlib.axes._subplots.AxesSubplot at 0x1adb972d898>
```



8.4 Incidence Matrix

```
In [119]: incM0 = nx.incidence_matrix(Gt0).todense()
          incM1 = nx.incidence_matrix(Gt1).todense()
          incM2 = nx.incidence_matrix(Gt2).todense()
          incM3 = nx.incidence_matrix(Gt3).todense()
          incM4 = nx.incidence_matrix(Gt4).todense()
          incM5 = nx.incidence_matrix(Gt5).todense()

In [120]: plt.figure(figsize=(64, 42))

          plt.subplot(231)
          sns.heatmap(incM0,cmap='Greys')
          plt.suptitle('Heatmap of Incidence Matrix, t0-t5', fontsize=80)

          plt.subplot(232)
          sns.heatmap(incM1, cmap='Greys')

          plt.subplot(233)
          sns.heatmap(incM2,cmap='Greys')

          plt.subplot(234)
          sns.heatmap(incM3,cmap='Greys')

          plt.subplot(235)
          sns.heatmap(incM4, cmap='Greys')

          plt.subplot(236)
          sns.heatmap(incM5, cmap='Greys')

Out[120]: <matplotlib.axes._subplots.AxesSubplot at 0x1adb7e95c88>
```



9 Subgraph Stationarity

The subgraph stationarity is computed in 2 steps.

Step 1:

Compute

$$Ct = \frac{A(t0) \cap A(t0+1)}{A(t0) \cup A(t0+1)}$$

This is effectively the autocorrelation of the graph with a time delayed version of itself. A(t) denotes the adjacency matrix

Step 2:

Calculate Subgraph stationarity, ζ

$$\zeta = \frac{\sum_{t=0}^{tmax-1} C(t, t+1)}{tmax - t0 - 1}$$

```
In [121]: def pad_shape(x,ref,offset=0):
              result = np.zeros_like(ref)
              result[0:x.shape[0]+0,0:x.shape[1]+0] = x

              return [result, nx.Graph(result)]
```

```
In [122]: #_,Gt0_pad = pad_shape(adjM0,adjM1)
          Gt0_int_Gt0= Gt0.copy()
          Gt0_int_Gt0.remove_nodes_from(n for n in Gt0 if n not in Gt0)
          Gt0_U_Gt0 = nx.disjoint_union(Gt0,Gt0)
          adjmat_inter = nx.adjacency_matrix(Gt0_int_Gt0).todense()
          adjmat_union = nx.adjacency_matrix(Gt0_U_Gt0).todense()
          adjmat_inter_pad,_  = pad_shape(adjmat_inter,adjmat_union)
          Ct0 = np.divide(np.linalg.norm(adjmat_inter_pad),np.linalg.norm(adjmat_ur
          Ct0

Out[122]: 0.70710678118654757

In [123]: _,Gt0_pad = pad_shape(adjM0,adjM1)
          Gt0_int_Gt1= Gt0.copy()
          Gt0_int_Gt1.remove_nodes_from(n for n in Gt0 if n not in Gt1)
          Gt0_U_Gt1 = nx.disjoint_union(Gt0_pad,Gt1)
          adjmat_inter = nx.adjacency_matrix(Gt0_int_Gt1).todense()
          adjmat_union = nx.adjacency_matrix(Gt0_U_Gt1).todense()
          adjmat_inter_pad,_  = pad_shape(adjmat_inter,adjmat_union)
          Ct1 = np.divide(np.linalg.norm(adjmat_inter_pad),np.linalg.norm(adjmat_ur
          Ct1

Out[123]: 0.66232865352709824

In [124]: _,Gt1_pad = pad_shape(adjM1,adjM2)
          Gt1_int_Gt2= Gt1.copy()
          Gt1_int_Gt2.remove_nodes_from(n for n in Gt1 if n not in Gt2)
          Gt1_U_Gt2 = nx.disjoint_union(Gt0_pad,Gt1)
          adjmat_inter = nx.adjacency_matrix(Gt1_int_Gt2).todense()
          adjmat_union = nx.adjacency_matrix(Gt1_U_Gt2).todense()
          adjmat_inter_pad,_  = pad_shape(adjmat_inter,adjmat_union)
          Ct2 = np.divide(np.linalg.norm(adjmat_inter_pad),np.linalg.norm(adjmat_ur
          Ct2

Out[124]: 0.7492134240101288

In [125]: _,Gt2_pad = pad_shape(adjM2,adjM3)
          Gt2_int_Gt3= Gt2.copy()
          Gt2_int_Gt3.remove_nodes_from(n for n in Gt3 if n not in Gt3)
          Gt2_U_Gt3 = nx.disjoint_union(Gt2_pad,Gt3)
          adjmat_inter = nx.adjacency_matrix(Gt2_int_Gt3).todense()
          adjmat_union = nx.adjacency_matrix(Gt2_U_Gt3).todense()
          adjmat_inter_pad,_  = pad_shape(adjmat_inter,adjmat_union)
          Ct3 = np.divide(np.linalg.norm(adjmat_inter_pad),np.linalg.norm(adjmat_ur
          Ct3

Out[125]: 0.41226013627128677

In [126]: _,Gt3_pad = pad_shape(adjM3,adjM4)
          Gt3_int_Gt4= Gt3.copy()
```

```
Gt3_int_Gt4.remove_nodes_from(n for n in Gt3 if n not in Gt4)
Gt3_U_Gt4 = nx.disjoint_union(Gt3_pad,Gt4)
adjmat_inter = nx.adjacency_matrix(Gt3_int_Gt4).todense()
adjmat_union = nx.adjacency_matrix(Gt3_U_Gt4).todense()
adjmat_inter_pad,_  = pad_shape(adjmat_inter,adjmat_union)
Ct4 = np.divide(np.linalg.norm(adjmat_inter_pad),np.linalg.norm(adjmat_ur
Ct4
```

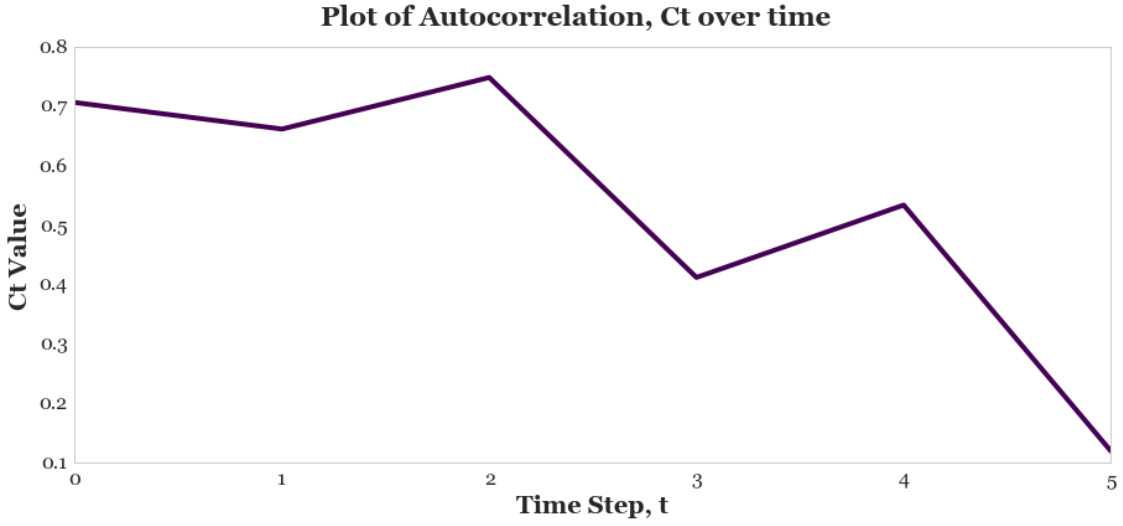
```
Out[126]: 0.53425969593338996
```

```
In [127]: _,Gt4_pad = pad_shape(adjM4,adjM5)
          Gt4_int_Gt5= Gt1.copy()
          Gt4_int_Gt5.remove_nodes_from(n for n in Gt4 if n not in Gt5)
          Gt4_U_Gt5 = nx.disjoint_union(Gt4_pad,Gt5)
          adjmat_inter = nx.adjacency_matrix(Gt4_int_Gt5).todense()
          adjmat_union = nx.adjacency_matrix(Gt4_U_Gt5).todense()
          adjmat_inter_pad,_  = pad_shape(adjmat_inter,adjmat_union)
          Ct5 = np.divide(np.linalg.norm(adjmat_inter_pad),np.linalg.norm(adjmat_ur
          Ct5
```

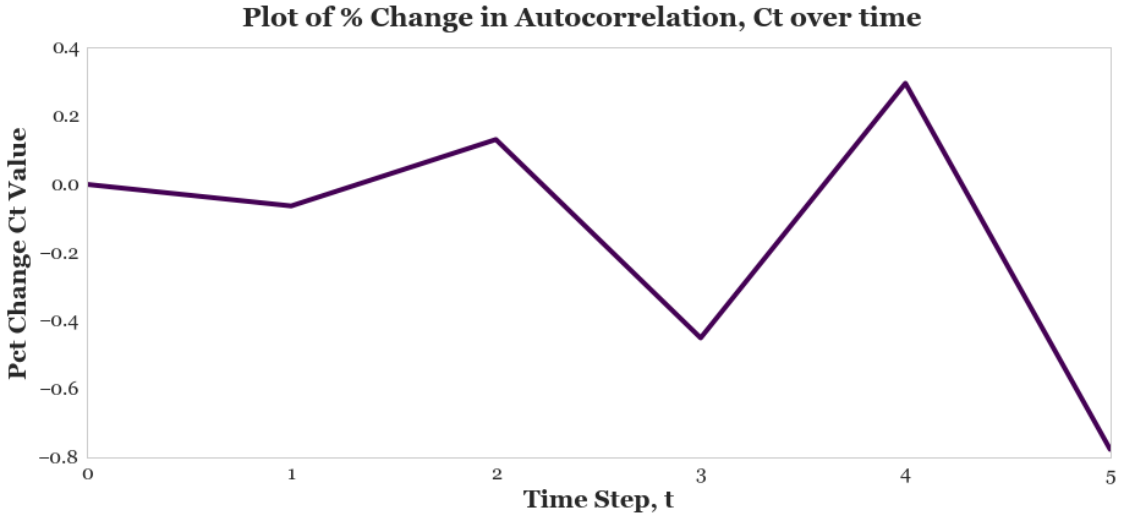
```
Out[127]: 0.11965940514762706
```

```
In [128]: Ct_df = pd.DataFrame([Ct0,Ct1,Ct2,Ct3,Ct4,Ct5])
          Ct_df.plot(fontsize=16, cmap='viridis', legend=False)
          plt.suptitle('Plot of Autocorrelation, Ct over time', fontsize=22)
          plt.xlabel("Time Step, t", fontsize=20)
          plt.ylabel("Ct Value", fontsize=20)
```

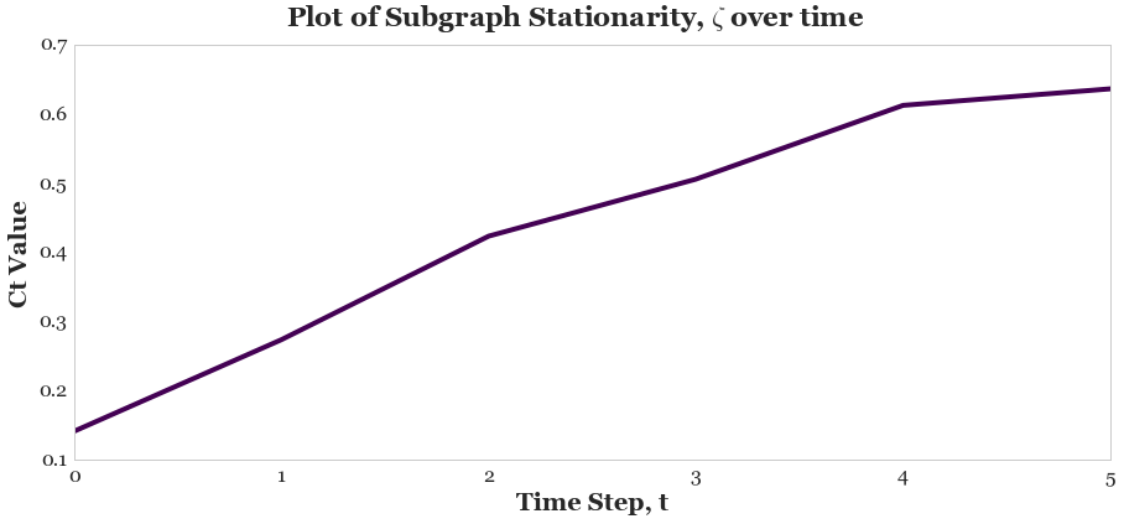
```
Out[128]: <matplotlib.text.Text at 0x1adc6f96320>
```



```
In [129]: Ct_df.pct_change().fillna(0).plot(fontsize=16, cmap='viridis', legend=Fal
          plt.suptitle('Plot of % Change in Autocorrelation, Ct over time', fontsiz
          plt.xlabel("Time Step, t", fontsize=20)
          plt.ylabel("Pct Change Ct Value", fontsize=20)

Out[129]: <matplotlib.text.Text at 0x1adc6cd6128>
```




```
In [130]: zeta = Ct_df.cumsum()/(5)
          zeta.plot(fontsize=16, cmap='viridis', legend=False)
          plt.suptitle('Plot of Subgraph Stationarity, $\zeta$ over time', fontsize
          plt.xlabel("Time Step, t", fontsize=20)
          plt.ylabel("Ct Value", fontsize=20)

Out[130]: <matplotlib.text.Text at 0x1adc6be0f60>
```



```
In [131]: zeta.mad()

Out[131]: 0    0.152857
          dtype: float64
```

10 Graph Stationarity Ratio

The stationarity ratio as used here is inspired by the Matlab Code based on the arXiv:1601.02522

```
In [132]: def stationarity_ratio(G):
              #stationarity ratio with laplian
              L = nx.laplacian_matrix(G).todense()
              U = nx.laplacian_spectrum(G)
              C =np.cov(L)
              CF = np.dot(L,np.dot(np.dot(U.T,C),U))
              r = np.linalg.norm(np.diag(CF))/np.linalg.norm(CF)

              #stationarity with modularity
              M = nx.modularity_matrix(G)
              U_mod = nx.modularity_spectrum(G)
              C_mod =np.cov(M)
              CF_mod = np.dot(M,np.dot(np.dot(U_mod.T,C_mod),U_mod))
              r_mod = np.linalg.norm(np.diag(CF_mod))/np.linalg.norm(CF_mod)


              #adjacency matrix stationarity
              A = nx.incidence_matrix(G).todense()
              U_adj = nx.adjacency_spectrum(G)
              C_adj =np.cov(A)
              CF_adj = np.dot(A,np.dot(np.dot(U_adj.T,C_adj),U_adj))
              r_adj = np.linalg.norm(np.diag(CF_adj))/np.linalg.norm(CF_adj)

              return [r, r_mod,r_adj]


In [133]: st_rat0, mod_st_rat0, adj_st_rat0 = stationarity_ratio(Gt0)
          st_rat1, mod_st_rat1, adj_st_rat1 = stationarity_ratio(Gt1)
          st_rat2, mod_st_rat2, adj_st_rat2 = stationarity_ratio(Gt2)
          st_rat3, mod_st_rat3, adj_st_rat3 = stationarity_ratio(Gt3)
          st_rat4, mod_st_rat4, adj_st_rat4 = stationarity_ratio(Gt4)
          st_rat5, mod_st_rat5, adj_st_rat5 = stationarity_ratio(Gt5)

In [134]: stationarity_df = pd.DataFrame([st_rat0,st_rat1,st_rat2,st_rat3,st_rat4,s
          stationarity_df['ModStatRat'] = pd.DataFrame([mod_st_rat0,mod_st_rat1,mod
          stationarity_df['AdjStatRat'] = pd.DataFrame([adj_st_rat0,adj_st_rat1,adj
```

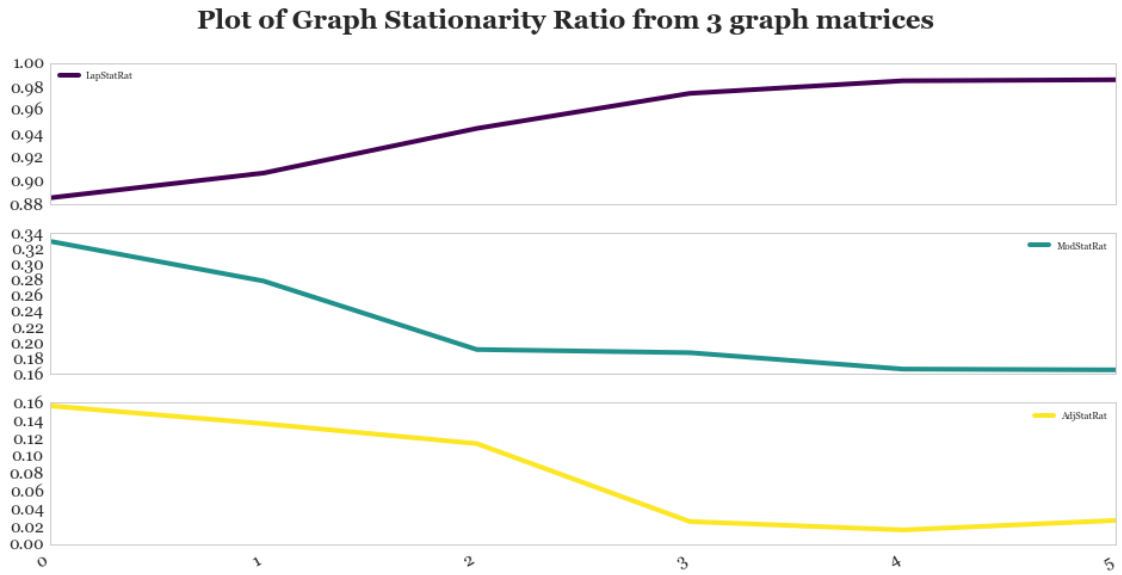
```
In [135]: stationarity_df.columns.values[0]='LapStatRat'
          stationarity_df
```

Out[135]:

	LapStatRat	ModStatRat	AdjStatRat
0	0.885557	0.328867	0.156174
1	0.906522	0.278447	0.136083
2	0.944330	0.191400	0.113592
3	0.974044	0.187360	0.025516
4	0.984701	0.166550	0.016042
5	0.985587	0.165496	0.026745

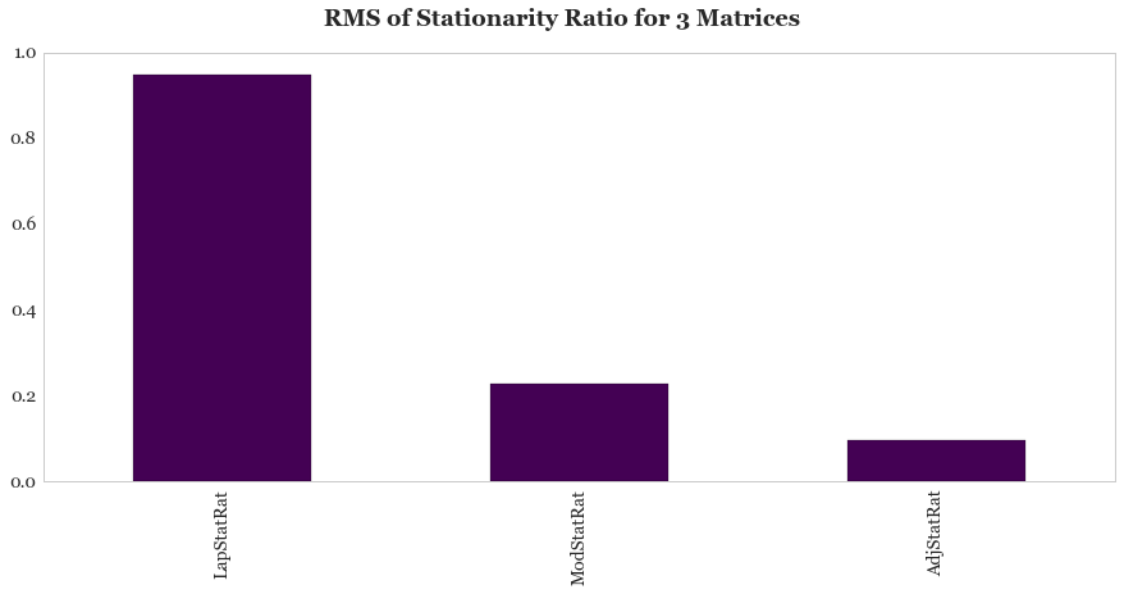
```
In [169]: stationarity_df.plot(subplots=True, fontsize=14, legend=True, sharex=True
          plt.suptitle('Plot of Graph Stationarity Ratio from 3 graph matrices', fo
```

Out[169]: <matplotlib.text.Text at 0x1adcb56d6d8>



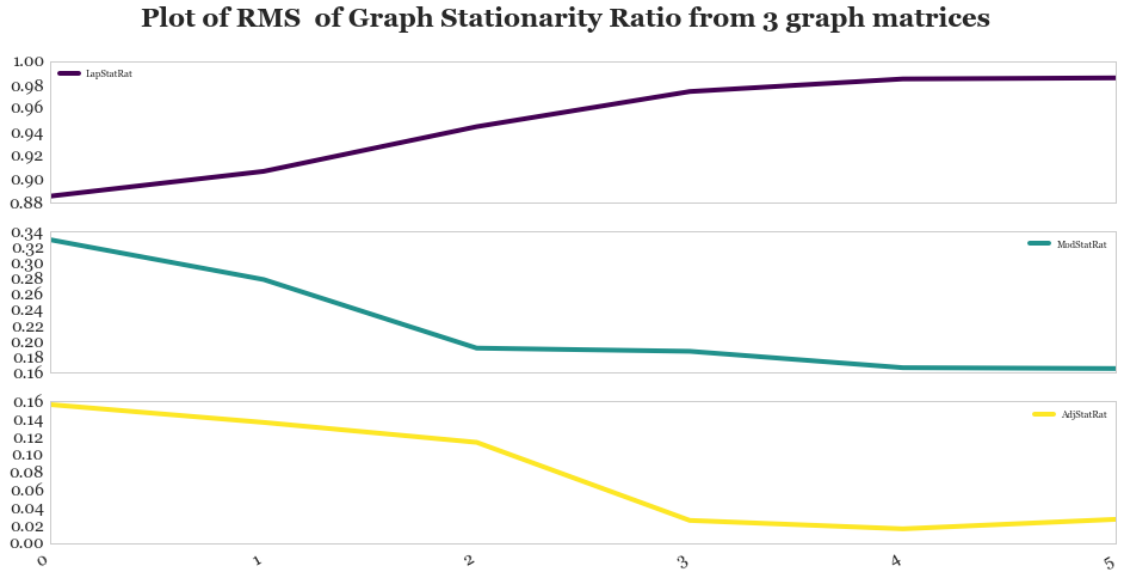

```
In [171]: stationarity_df.apply(rms).plot(kind='bar',fontsize=14,colormap='viridis'
          plt.suptitle('RMS of Stationarity Ratio for 3 Matrices', fontsize=18)
```

Out[171]: <matplotlib.text.Text at 0x1adcbdb2390>



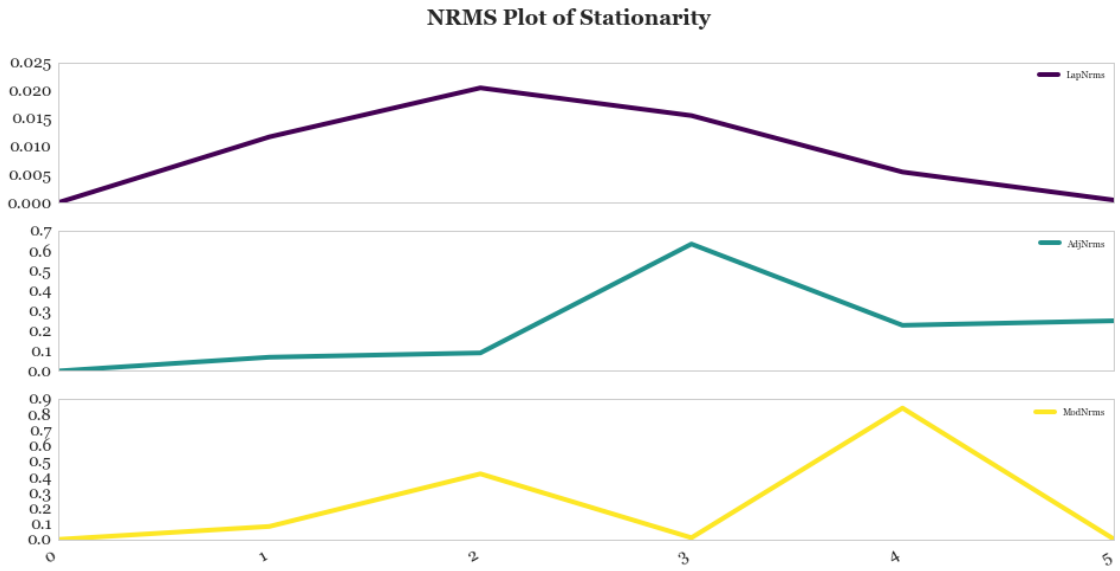

```
In [172]: stationarity_df.applymap(rms).plot(subplots=True, fontsize=14, legend=Tru
          plt.suptitle('Plot of RMS  of Graph Stationarity Ratio from 3 graph matri
```

```
Out[172]: <matplotlib.text.Text at 0x1adcb9af518>
```




```
In [139]: Nrms_df = pd.DataFrame([0,nrms(st_rat0,st_rat1), nrms(st_rat1,st_rat2), n
                      nrms(st_rat4,st_rat5)])
```

```
In [140]: Nrms_df['AdjNrms'] = pd.DataFrame([0,nrms(adj_st_rat0,adj_st_rat1), nrms
                                             nrms(adj_st_rat2,adj_st_rat3), nrms(ad
                                             nrms(adj_st_rat4,adj_st_rat5)])

In [141]: Nrms_df['ModNrms'] = pd.DataFrame([0,nrms(mod_st_rat0,mod_st_rat1), nrms
                                             nrms(mod_st_rat2,mod_st_rat3), nrms(mo
                                             nrms(mod_st_rat4,mod_st_rat5)])

In [142]: Nrms_df.columns.values[0]='LapNrms'

In [143]: Nrms_df

Out[143]:     LapNrms   AdjNrms   ModNrms
          0  0.000000  0.000000  0.000000
          1  0.011699  0.068744  0.083020
          2  0.020428  0.090079  0.420506
          3  0.015489  0.633155  0.010667
          4  0.005441  0.227972  0.842266
          5  0.000450  0.250161  0.003174

In [144]: Nrms_df.plot(subplots=True,fontsize=14, legend=True, sharex=True, figsize
          plt.suptitle('NRMS Plot of Stationarity', fontsize=18)
          plt.xticks(fontsize=14)
          plt.yticks(fontsize=14)
          #plt.legend(fontsize=14, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0

Out[144]: (array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9]),
           <a list of 10 Text yticklabel objects>)
```




```
In [145]: stat_df3= stationarity_df.join(Nrms_df)
```

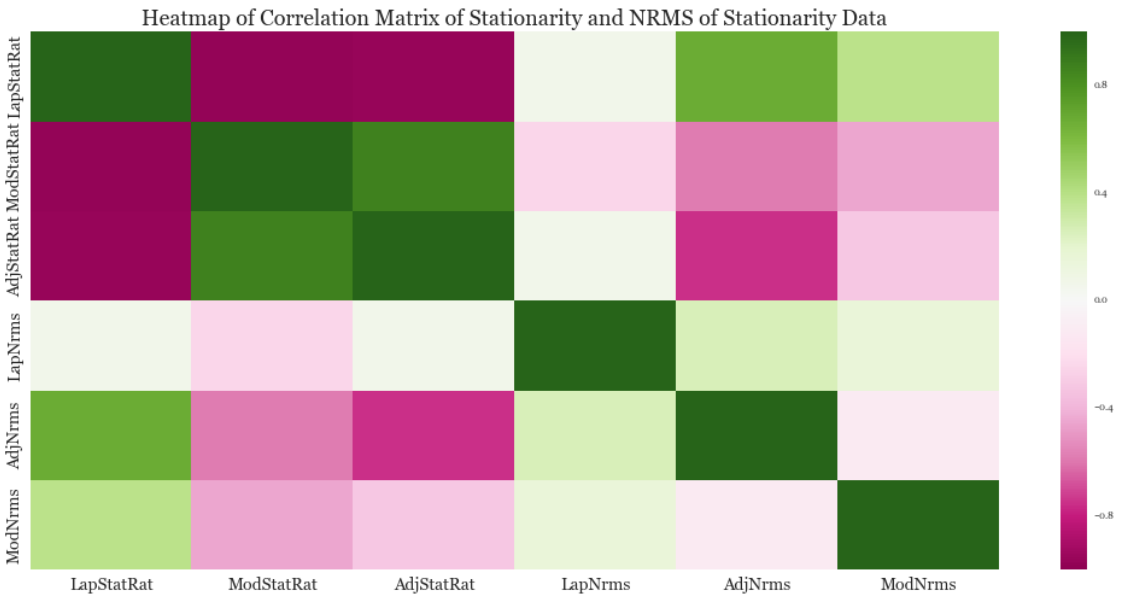
```
In [146]: stat_df3.head()
```

Out[146]:

	LapStatRat	ModStatRat	AdjStatRat	LapNrms	AdjNrms	ModNrms
0	0.885557	0.328867	0.156174	0.000000	0.000000	0.000000
1	0.906522	0.278447	0.136083	0.011699	0.068744	0.083020
2	0.944330	0.191400	0.113592	0.020428	0.090079	0.420506
3	0.974044	0.187360	0.025516	0.015489	0.633155	0.010667
4	0.984701	0.166550	0.016042	0.005441	0.227972	0.842266

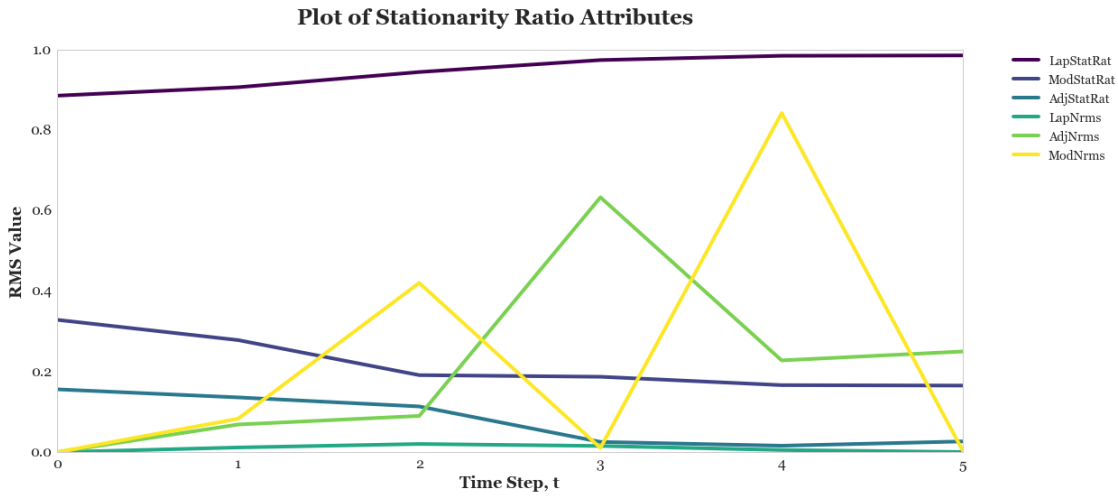
```
In [147]: plt.figure(figsize=(18,8))
          sns.heatmap(stat_df3.corr(), cmap='PiYG')
          plt.title('Heatmap of Correlation Matrix of Stationarity and NRMS of Stat
          plt.xticks(fontsize=14)
          plt.yticks(fontsize=14)
```

```
Out[147]: (array([ 0.5,  1.5,  2.5,  3.5,  4.5,  5.5]),
           <a list of 6 Text yticklabel objects>)
```




```
In [217]: stat_df3.plot(colormap='viridis', figsize=(18,8), fontsize=14)
          plt.suptitle('Plot of Stationarity Ratio Attributes', fontsize=24)
          plt.xticks(fontsize=16)
          plt.yticks(fontsize=16)
          plt.xlabel("Time Step, t", fontsize=18)
          plt.ylabel("RMS Value", fontsize=18)
          plt.legend(fontsize=14, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
```

```
Out[217]: <matplotlib.legend.Legend at 0x1addc861e10>
```



11 Frequency Wavenumber Plots, F-k

The Fourier Transform of a timeseries gives its frequency components.

The Wavenumber, k is defined as

$$\kappa = \frac{1}{frequency}$$

Essentially we expect to see the signal in a central cone and the noise distributed around it.

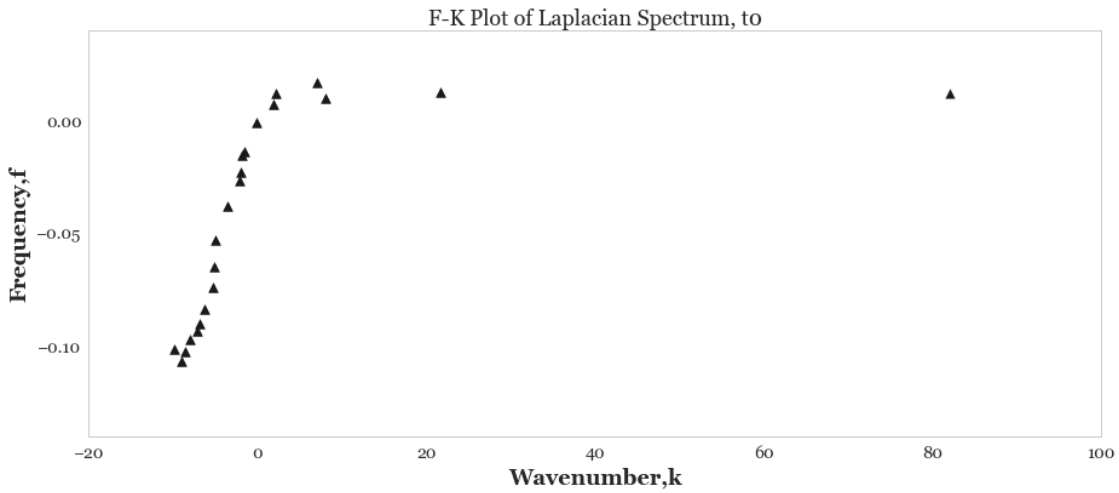
```
In [149]: lap_spec0 = nx.laplacian_spectrum(Gt0)
          mod_spec0 = nx.modularity_spectrum(Gt0)
          adj_spec0 = nx.adjacency_spectrum(Gt0)

          lap_spec1 = nx.laplacian_spectrum(Gt1)
          mod_spec1= nx.modularity_spectrum(Gt1)
          adj_spec1= nx.adjacency_spectrum(Gt1)

          lap_spec2 = nx.laplacian_spectrum(Gt2)
          mod_spec2= nx.modularity_spectrum(Gt2)
          adj_spec2= nx.adjacency_spectrum(Gt2)

          lap_spec3 = nx.laplacian_spectrum(Gt3)
          mod_spec3= nx.modularity_spectrum(Gt3)
          adj_spec3= nx.adjacency_spectrum(Gt3)

          lap_spec4 = nx.laplacian_spectrum(Gt4)
          mod_spec4= nx.modularity_spectrum(Gt4)
          adj_spec4= nx.adjacency_spectrum(Gt4)

          lap_spec5 = nx.laplacian_spectrum(Gt5)
```

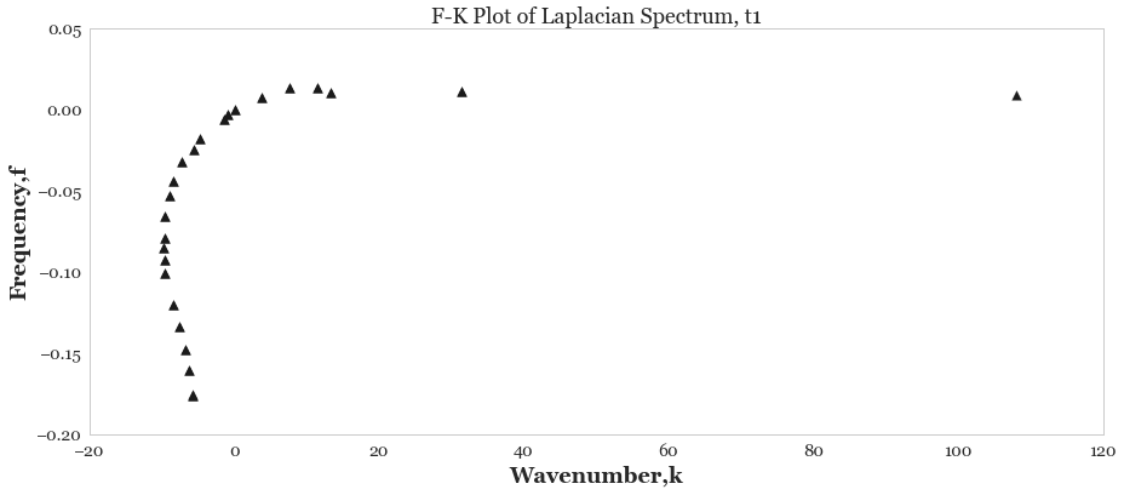
```
mod_spec5= nx.modularity_spectrum(Gt5)
adj_spec5= nx.adjacency_spectrum(Gt5)
```

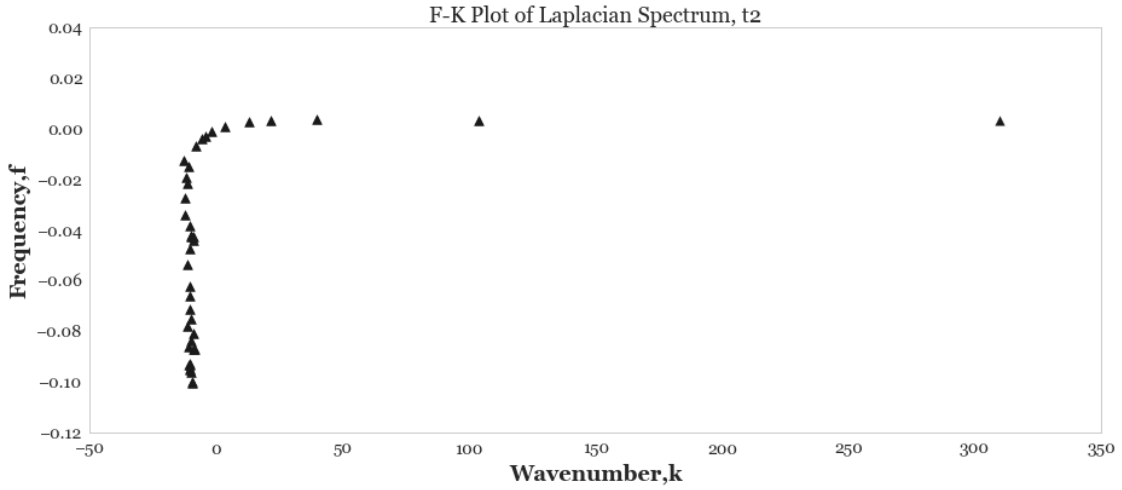
```
In [150]: def fk_plot(f,name):
              freq = sc.fft(f)
              wavnum = 1/freq

              plt.scatter(freq, wavnum, s=60, marker='^', c='k')
              plt.title("F-K Plot of "+str(name), fontsize=18)
              plt.xticks(fontsize=14)
              plt.yticks(fontsize=14)
              plt.xlabel("Wavenumber,k", fontsize=18)
              plt.ylabel("Frequency,f", fontsize=18)
              plt.show()

              return [freq,wavnum]
```

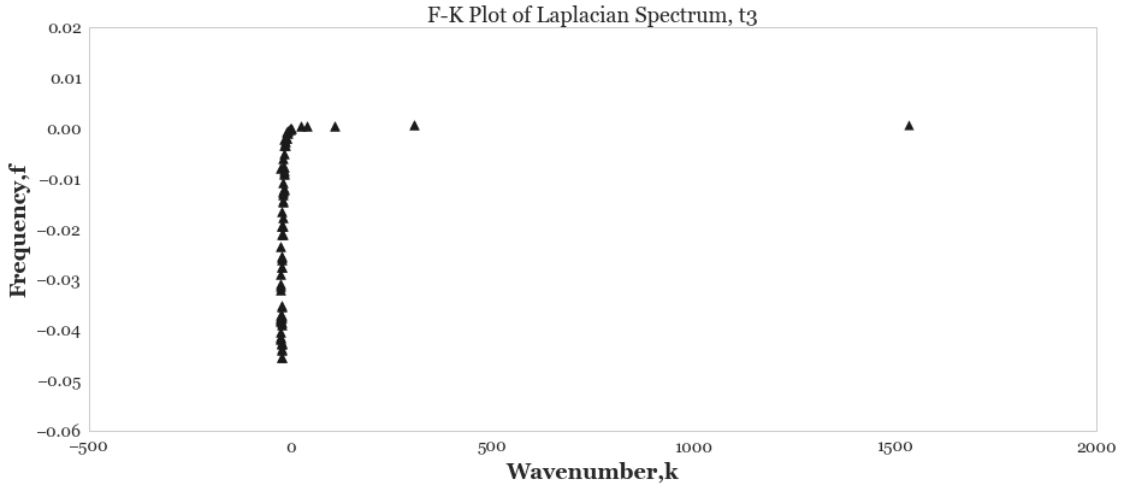
```
In [151]: freq01, k01 = fk_plot(lap_spec0,'Laplacian Spectrum, t0')
          freq1, k1 = fk_plot(lap_spec1,'Laplacian Spectrum, t1')
          freq2, k2 = fk_plot(lap_spec2,'Laplacian Spectrum, t2')
          freq3, k3 = fk_plot(lap_spec3,'Laplacian Spectrum, t3')
          freq4, k4 = fk_plot(lap_spec4,'Laplacian Spectrum, t4')
          freq5, k5 = fk_plot(lap_spec5,'Laplacian Spectrum, t5')
```

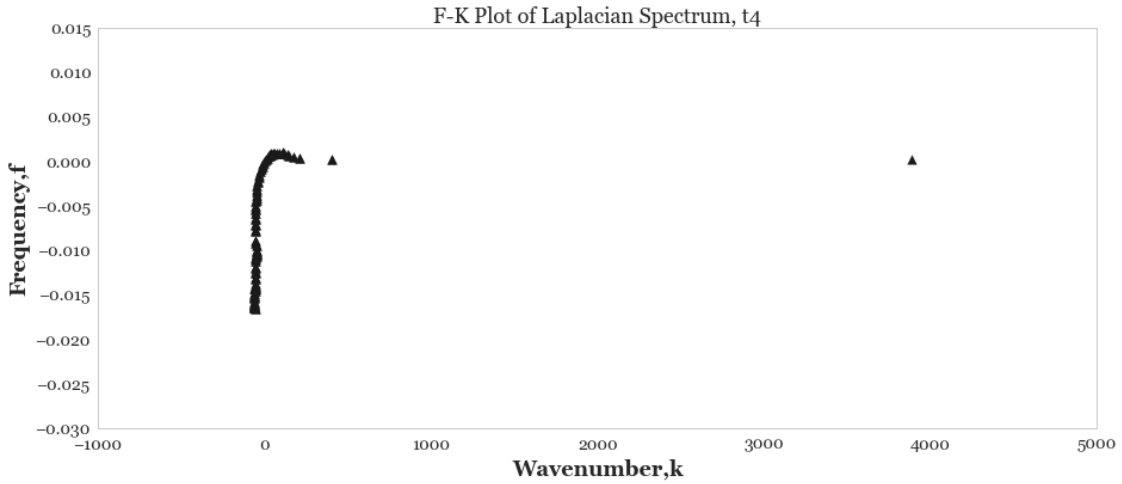
```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:533: ComplexWa
  return array(a, dtype, copy=False, order=order, subok=True)
```

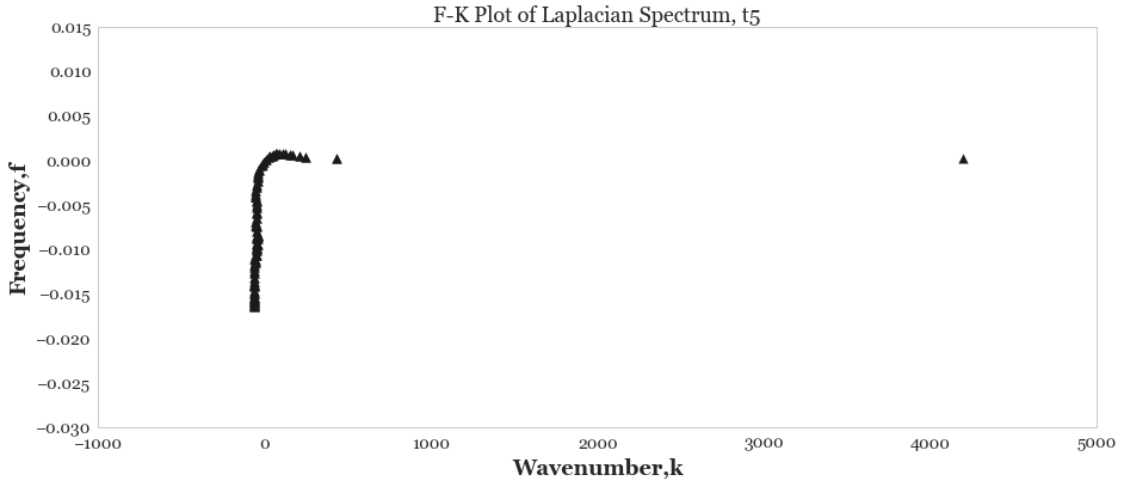



F-K Plot of Laplacian Spectrum, t1
Frequency,f
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−20
0
20
40
60
80
100
120
Wavenumber,k

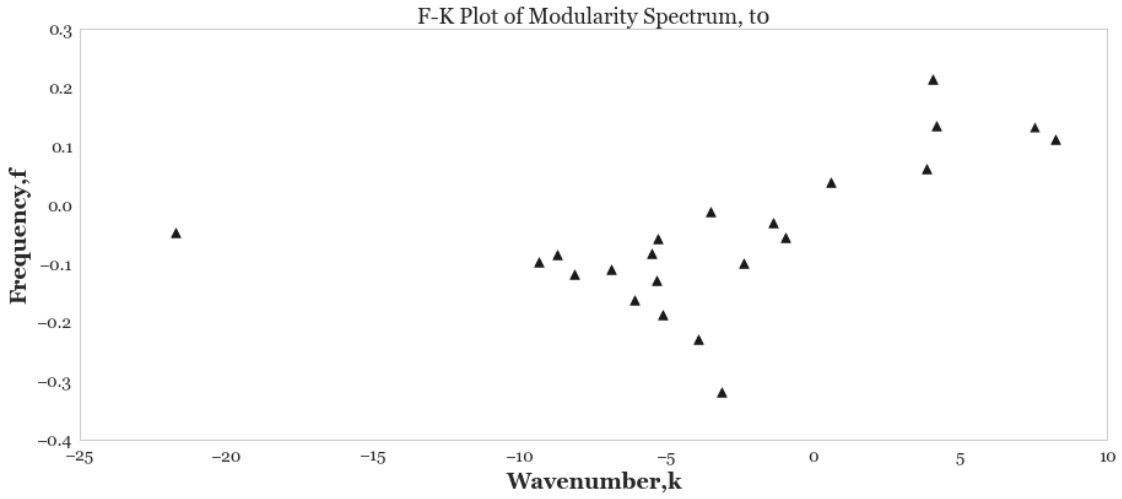

F-K Plot of Laplacian Spectrum, t2
Frequency,f
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.10
−0.12
−50
0
50
100
150
200
250
300
350
Wavenumber,k

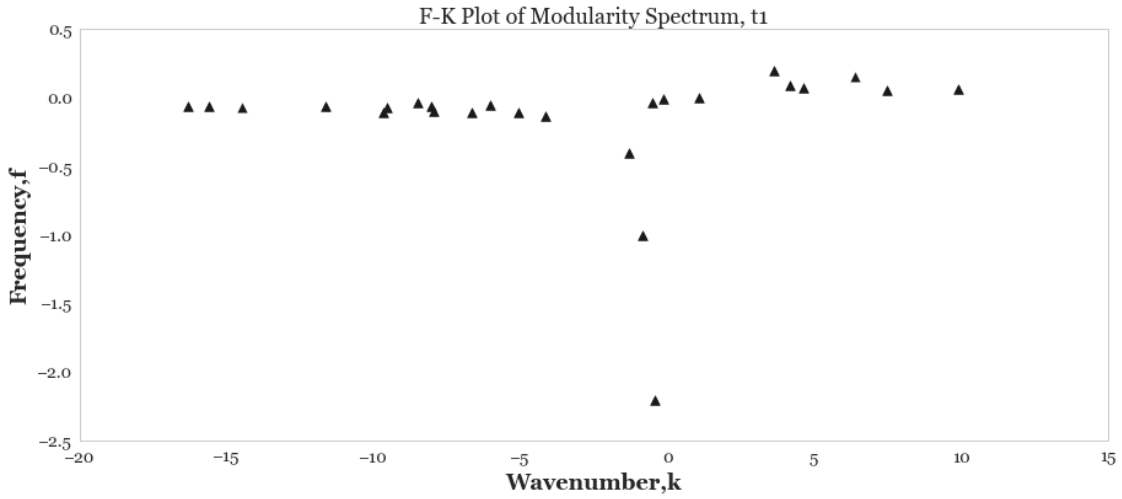

F-K Plot of Laplacian Spectrum, t3
Frequency,f
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−500
0
500
1000
1500
2000
Wavenumber,k

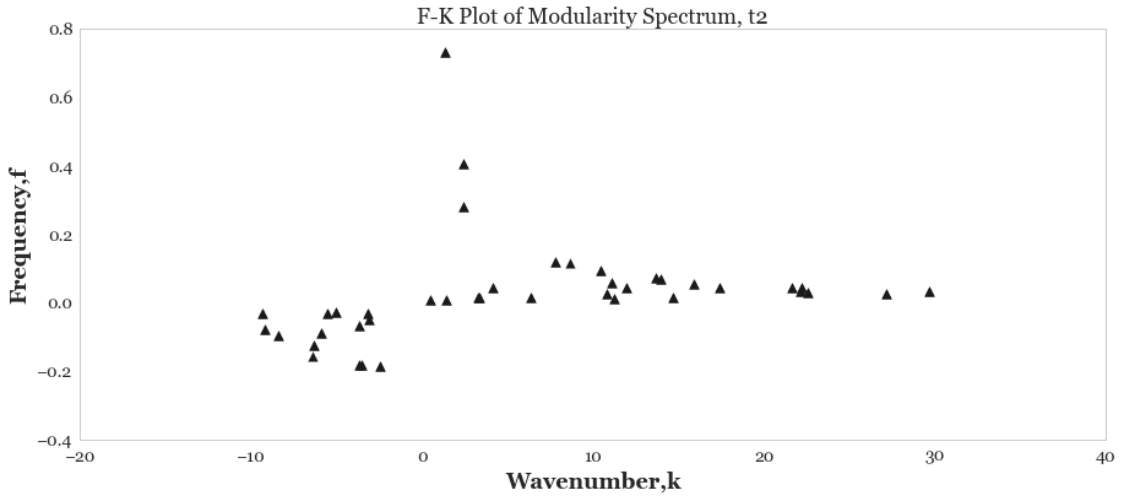


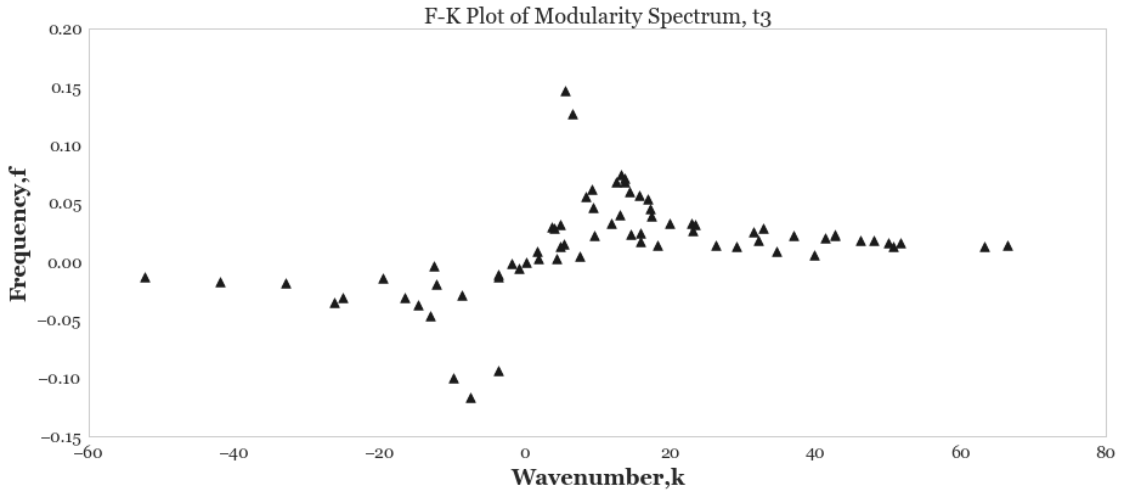


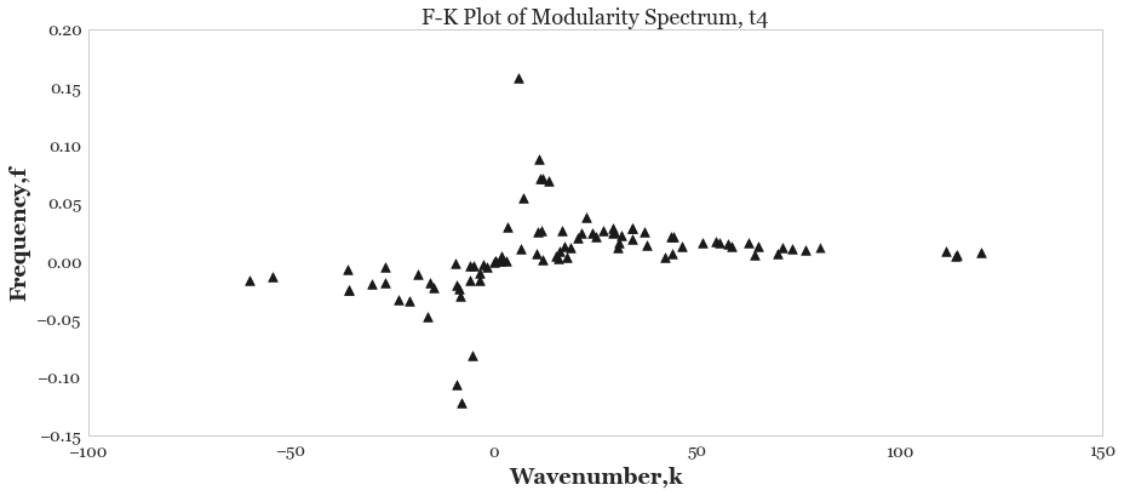

```
In [152]: freq02, k02 = fk_plot(mod_spec0,'Modularity Spectrum, t0')
          freq12, k02 = fk_plot(mod_spec1,'Modularity Spectrum, t1')
          freq22, k02 = fk_plot(mod_spec2,'Modularity Spectrum, t2')
          freq32, k02 = fk_plot(mod_spec3,'Modularity Spectrum, t3')
          freq42, k02 = fk_plot(mod_spec4,'Modularity Spectrum, t4')
          freq52, k02 = fk_plot(mod_spec5,'Modularity Spectrum, t5')

C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:533: ComplexWa
  return array(a, dtype, copy=False, order=order, subok=True)
```

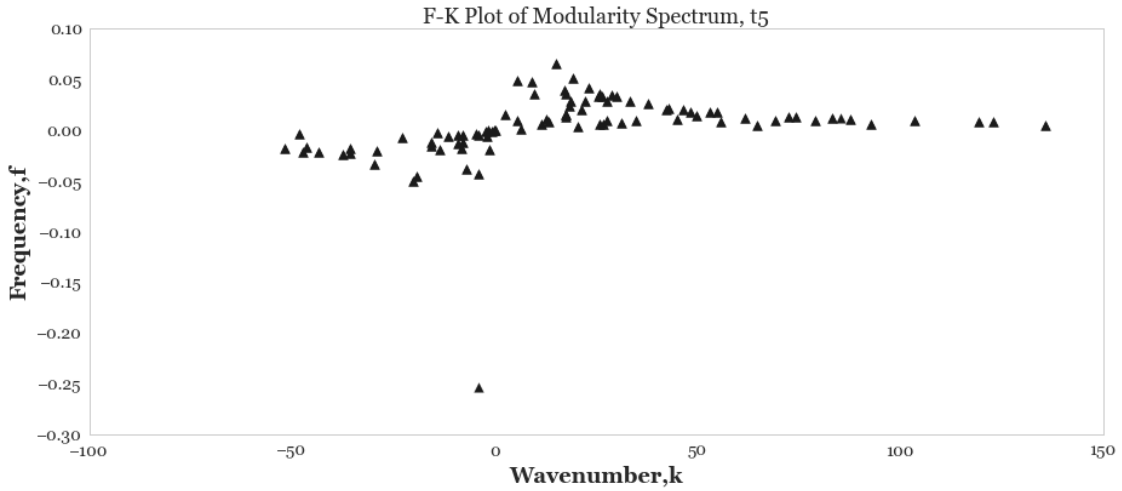

F-K Plot of Modularity Spectrum, t0
Frequency,f
Wavenumber,k


F-K Plot of Modularity Spectrum, t1
Frequency,f
Wavenumber,k


F-K Plot of Modularity Spectrum, t2
Frequency,f
Wavenumber,k

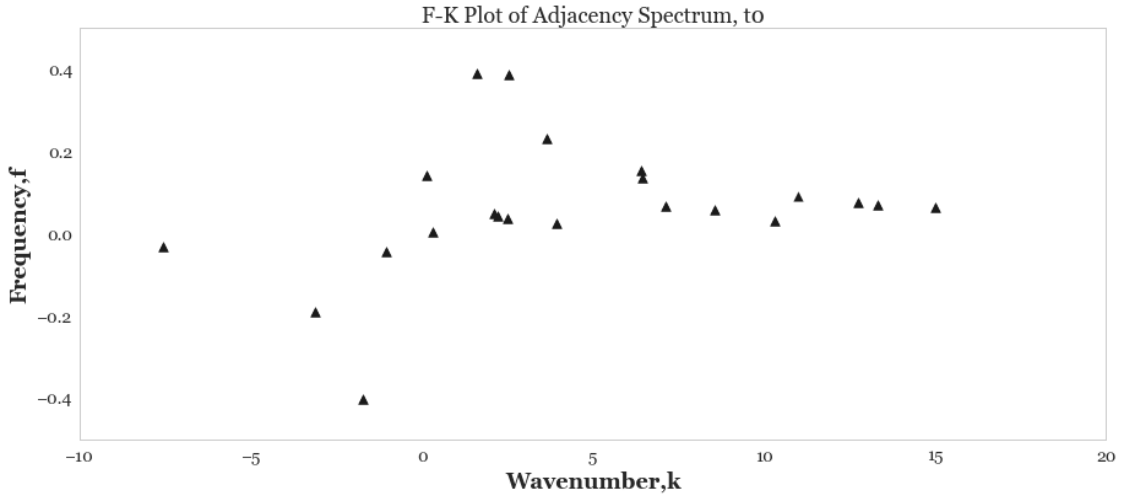

F-K Plot of Modularity Spectrum, t3
Frequency,f
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−60
−40
−20
0
20
40
60
80
Wavenumber,k

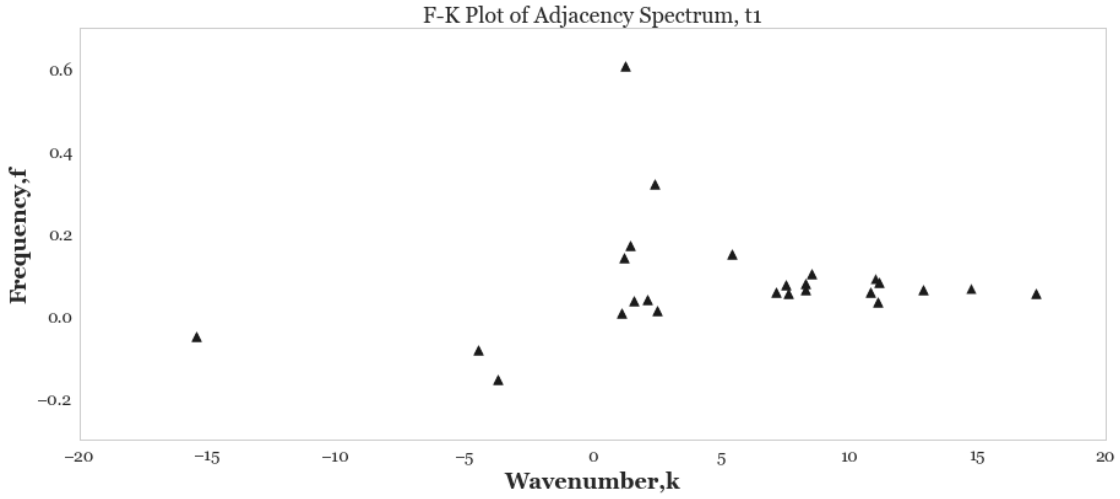

F-K Plot of Modularity Spectrum, t4
Frequency,f
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−100
−50
0
50
100
150
Wavenumber,k

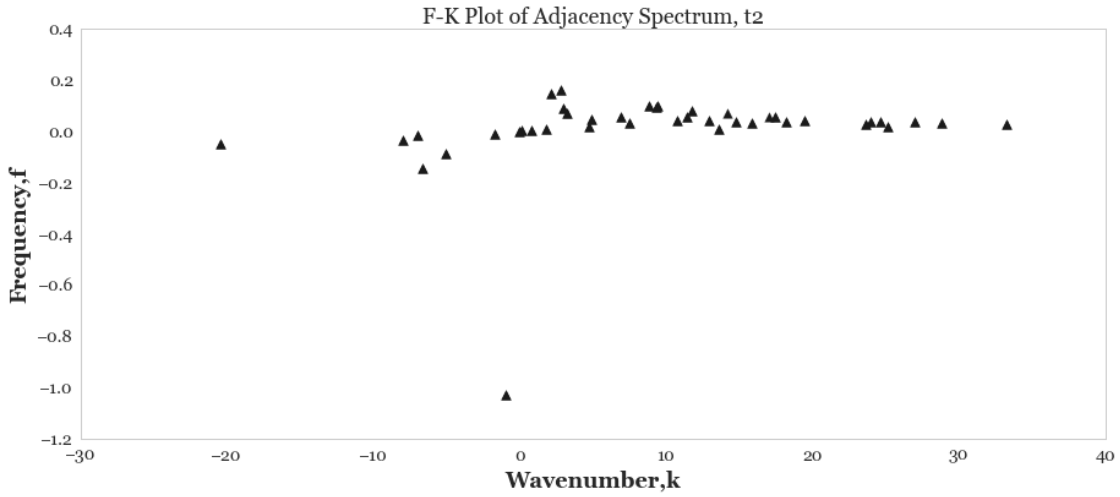


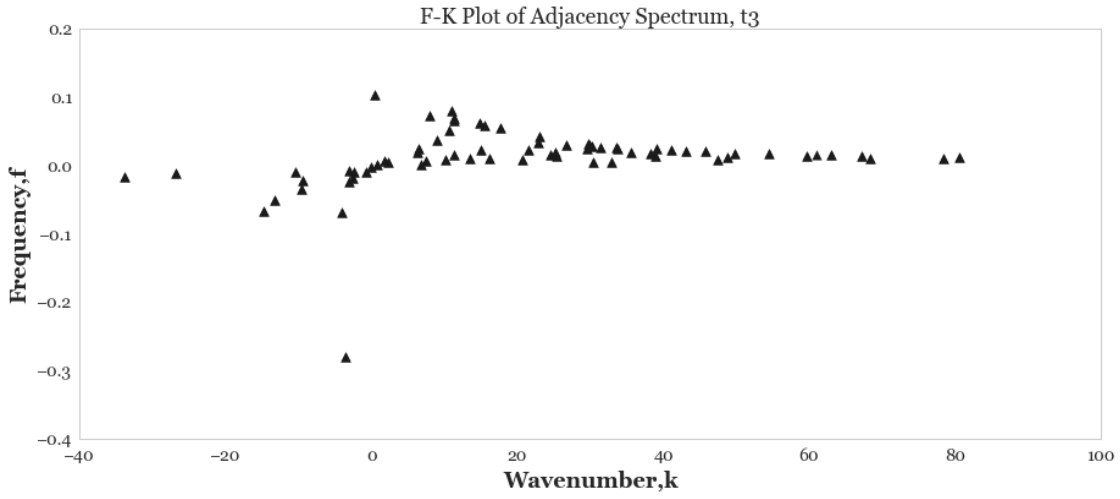

```
In [153]: freq03, k03 = fk_plot(adj_spec0,'Adjacency Spectrum, t0')
          freq13, k03 = fk_plot(adj_spec1,'Adjacency Spectrum, t1')
          freq23, k03 = fk_plot(adj_spec2,'Adjacency Spectrum, t2')
          freq33, k03 = fk_plot(adj_spec3,'Adjacency Spectrum, t3')
          freq43, k03 = fk_plot(adj_spec4,'Adjacency Spectrum, t4')
          freq53, k03 = fk_plot(adj_spec5,'Adjacency Spectrum, t5')
```

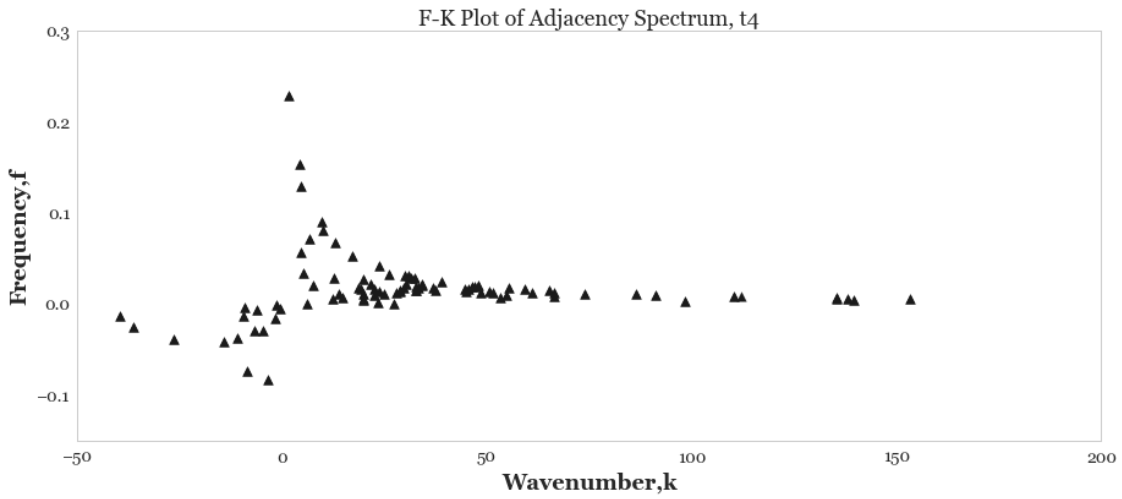
```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:533: ComplexWa
  return array(a, dtype, copy=False, order=order, subok=True)
```

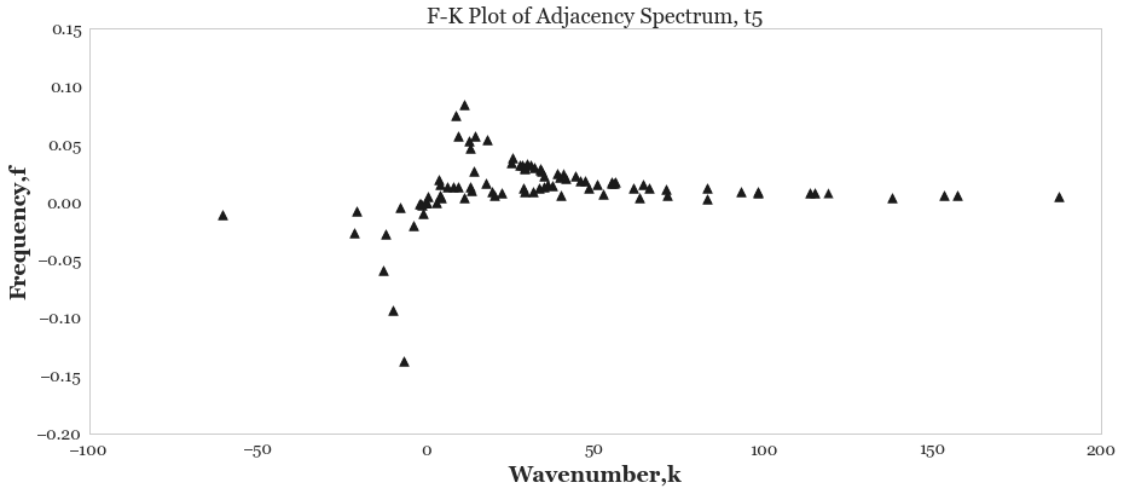



F-K Plot of Adjacency Spectrum, t1
Frequency,f
Wavenumber,k


F-K Plot of Adjacency Spectrum, t2
Frequency,f
Wavenumber,k


F-K Plot of Adjacency Spectrum, t3
Frequency,f
Wavenumber,k

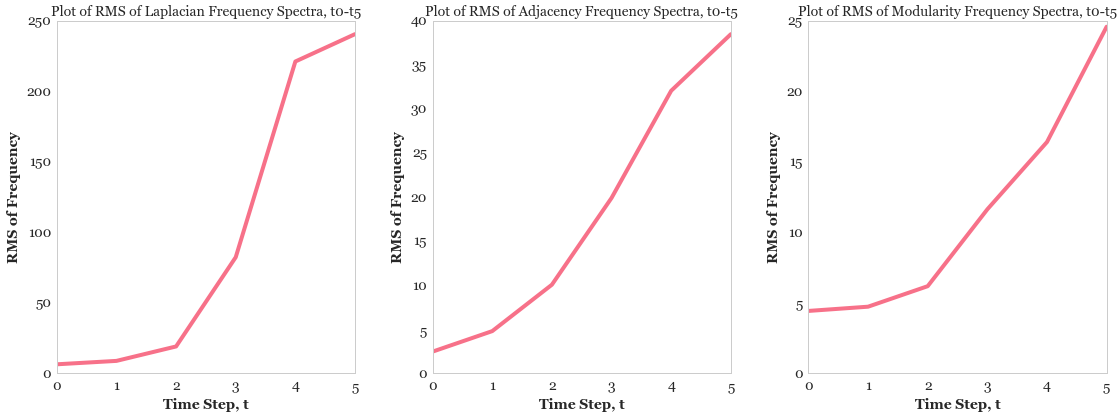





```
In [154]: freq_lap = pd.DataFrame([freq01,freq1,freq2,freq3,freq4,freq5]).T.apply(r
          freq_adj = pd.DataFrame([freq03,freq13,freq23,freq33,freq43,freq53]).T.ap
          freq_mod = pd.DataFrame([freq02,freq12,freq22,freq32,freq42,freq52]).T.ap

In [155]: plt.figure(figsize=(16,6))

          plt.subplot(131)
          plt.plot(freq_lap)
          plt.title('Plot of RMS of Laplacian Frequency Spectra, t0-t5', fontsize=1
          plt.xticks(fontsize=14)
```

```
plt.yticks(fontsize=14)
plt.xlabel("Time Step, t", fontsize=14)
plt.ylabel("RMS of Frequency", fontsize=14)

plt.subplot(132)
plt.plot(freq_adj)
plt.title('Plot of RMS of Adjacency Frequency Spectra, t0-t5', fontsize=1
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.legend(fontsize=16, loc=2)
plt.xlabel("Time Step, t", fontsize=14)
plt.ylabel("RMS of Frequency", fontsize=14)

plt.subplot(133)
plt.plot(freq_mod)
plt.title('Plot of RMS of Modularity Frequency Spectra, t0-t5', fontsize=
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.legend(fontsize=16, loc=2)
plt.xlabel("Time Step, t", fontsize=14)
plt.ylabel("RMS of Frequency", fontsize=14)

plt.tight_layout()
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)
C:\Users\arsha_000\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:519: UserWa
  warnings.warn("No labelled objects found. "
```

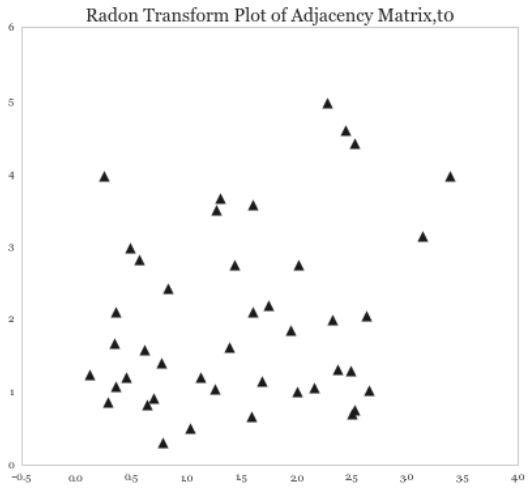


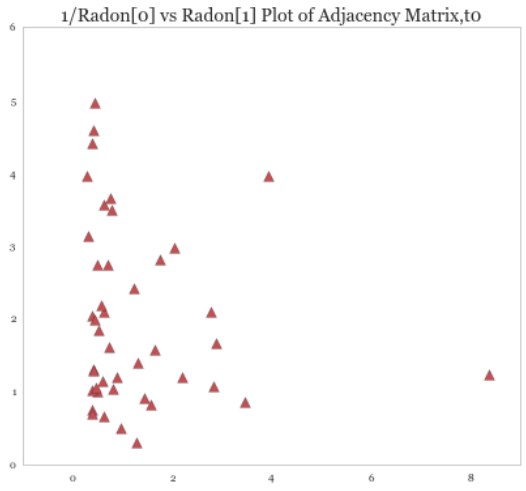
12 Radon Transform Plots

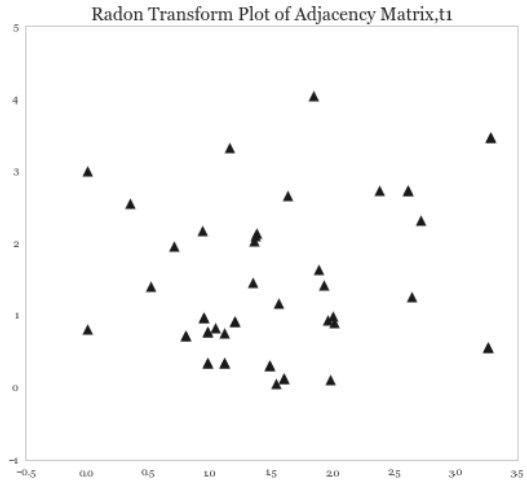
```
In [156]: def plot_radon(m, name):
              from skimage.transform import radon
              theta = np.linspace(0., 180., max(m.shape), endpoint=False)
              sinogram = radon(m, theta=theta, circle=True)

              fig, ax = plt.subplots(1,2)

              ax[0].scatter(sinogram[0], sinogram[1], s=60, marker='^', c='k')
              ax[0].set_title("Radon Transform Plot of "+str(name), fontsize=14)


              ax[1].scatter(1/sinogram[0], sinogram[1], s=60, marker='^', c='r')
              ax[1].set_title("1/Radon[0] vs Radon[1] Plot of "+str(name), fontsize


              plt.show()
```

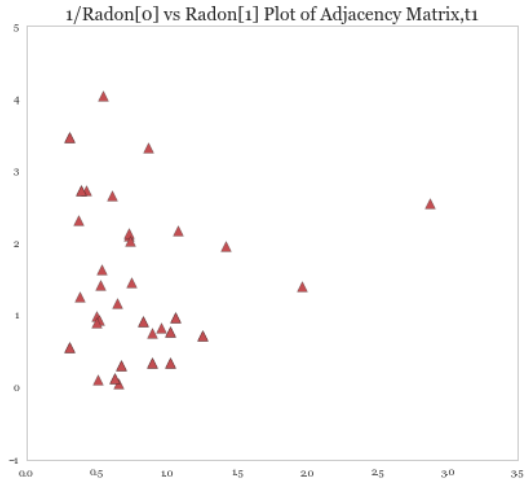
```
In [157]: plot_radon(adjM0,'Adjacency Matrix,t0')
          plot_radon(adjM1,'Adjacency Matrix,t1')
          plot_radon(adjM2,'Adjacency Matrix,t2')
          plot_radon(adjM3,'Adjacency Matrix,t3')
          plot_radon(adjM4,'Adjacency Matrix,t4')
          plot_radon(adjM5,'Adjacency Matrix,t5')
```

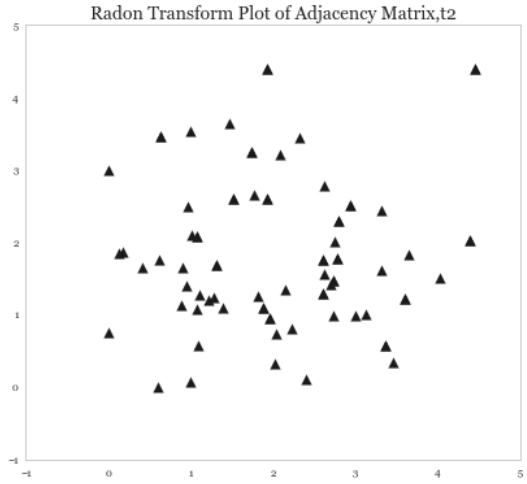
```
C:\Users\arsha_000\Anaconda3\lib\site-packages\skimage\transform\radon_transform.py
  warn('Radon transform: image must be zero outside the '
```

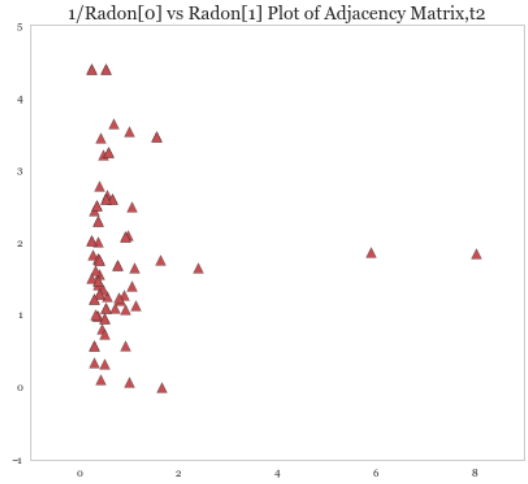


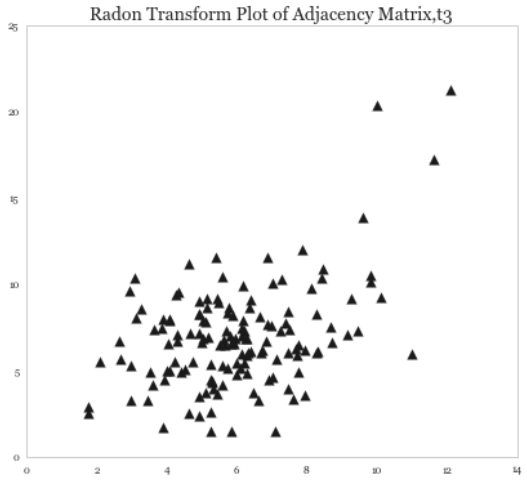


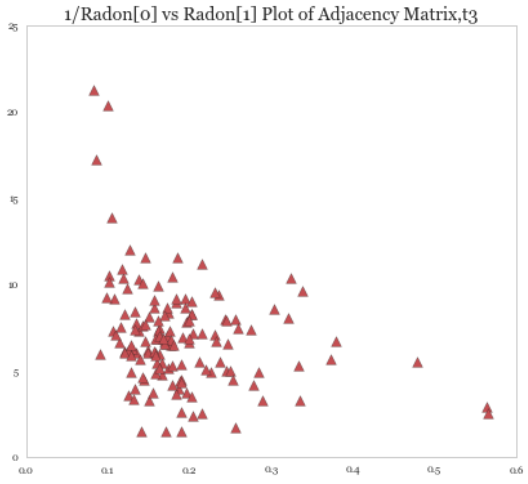

Radon Transform Plot of Adjacency Matrix,t1

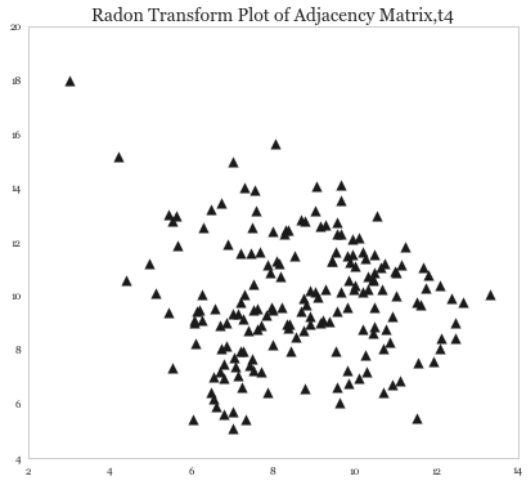

1/Radon[0] vs Radon[1] Plot of Adjacency Matrix,t1

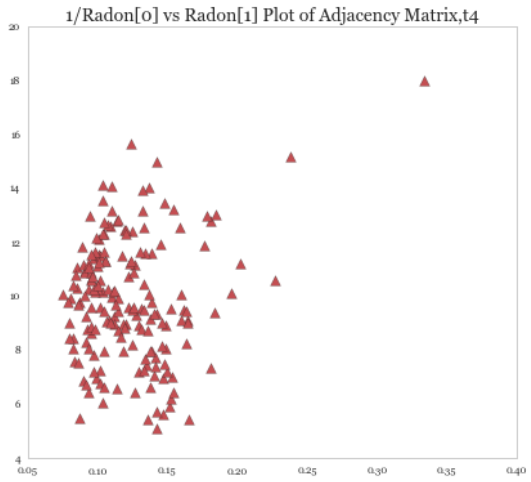

Radon Transform Plot of Adjacency Matrix,t2

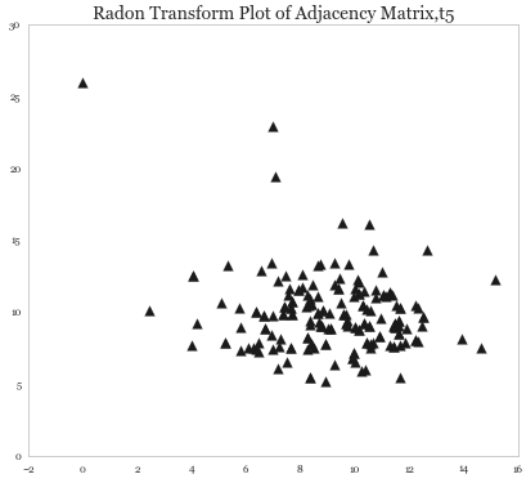

1/Radon[0] vs Radon[1] Plot of Adjacency Matrix,t2

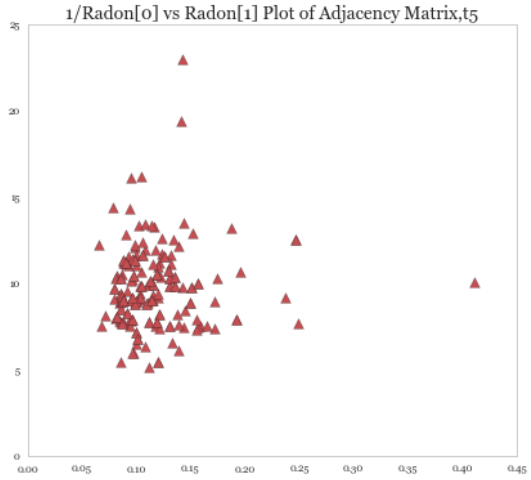

Radon Transform Plot of Adjacency Matrix,t3


1/Radon[0] vs Radon[1] Plot of Adjacency Matrix,t3

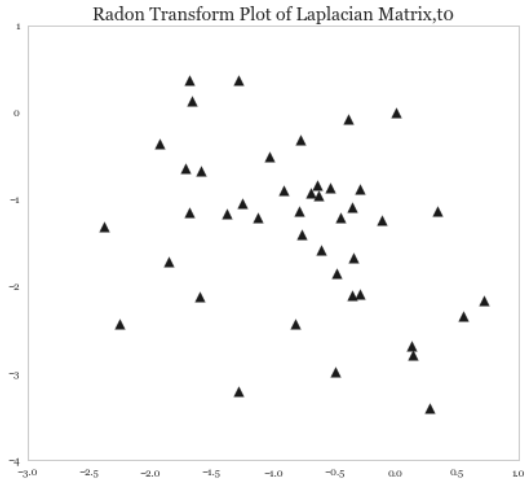


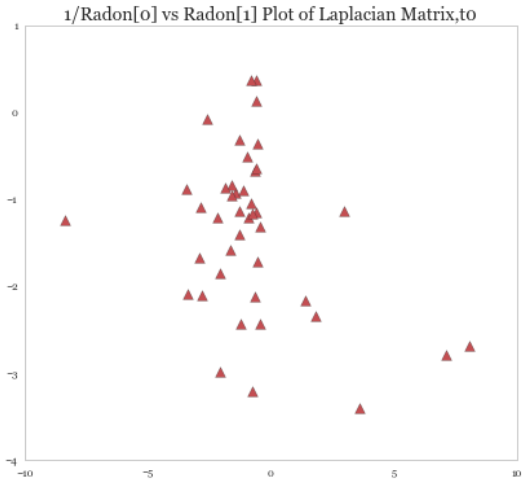


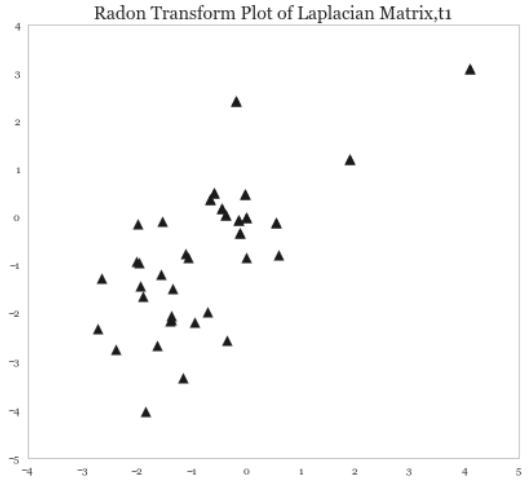


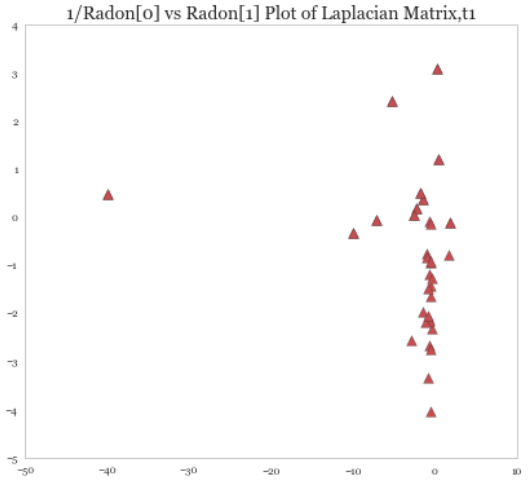


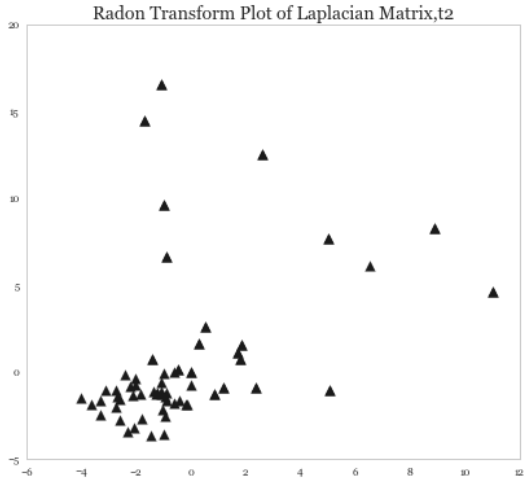

```
In [158]: plot_radon(lapM0,'Laplacian Matrix,t0')
          plot_radon(lapM1,'Laplacian Matrix,t1')
          plot_radon(lapM2,'Laplacian Matrix,t2')
          plot_radon(lapM3,'Laplacian Matrix,t3')
          plot_radon(lapM4,'Laplacian Matrix,t4')
          plot_radon(lapM5,'Laplacian Matrix,t5')

C:\Users\arsha_000\Anaconda3\lib\site-packages\skimage\transform\radon_transform.py
  warn('Radon transform: image must be zero outside the '
```

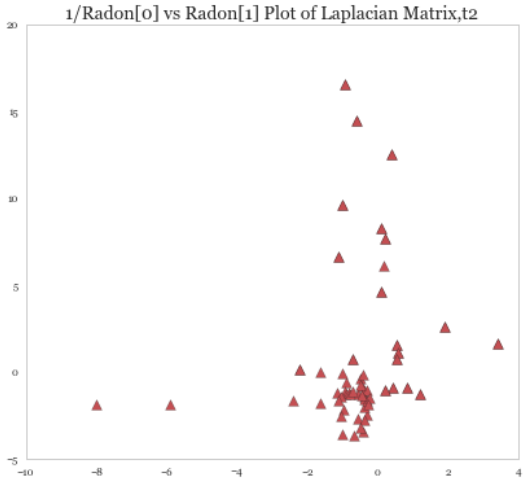

Radon Transform Plot of Laplacian Matrix,t0

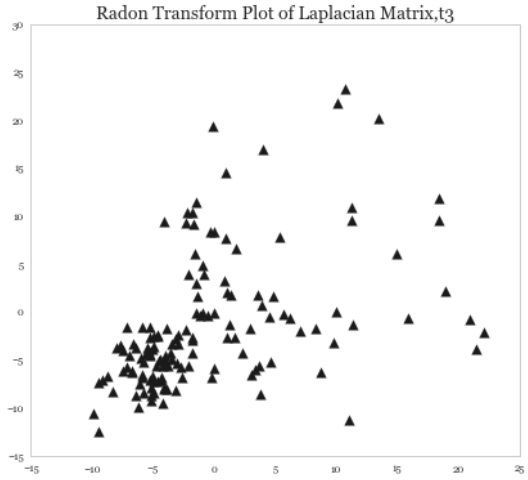

1/Radon[0] vs Radon[1] Plot of Laplacian Matrix,t0

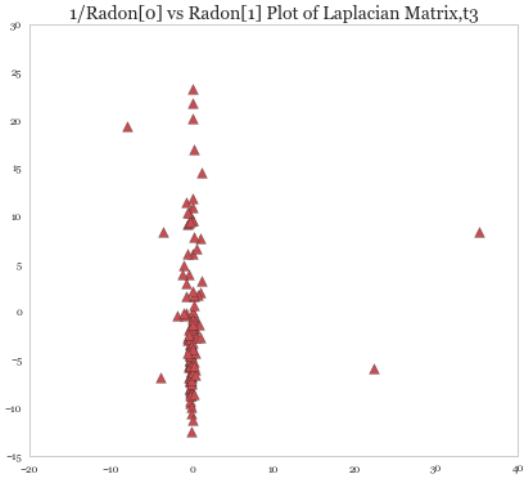

Radon Transform Plot of Laplacian Matrix,t1

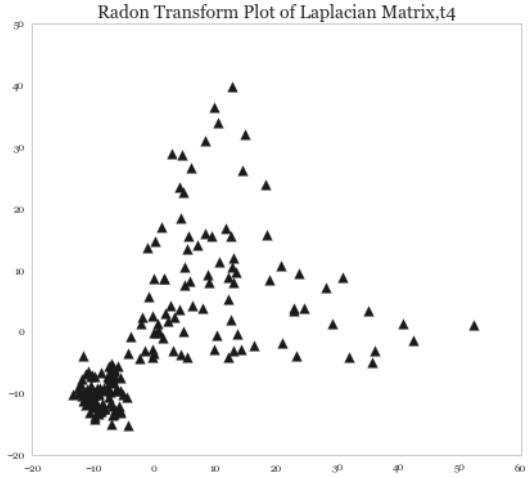

1/Radon[0] vs Radon[1] Plot of Laplacian Matrix,t1

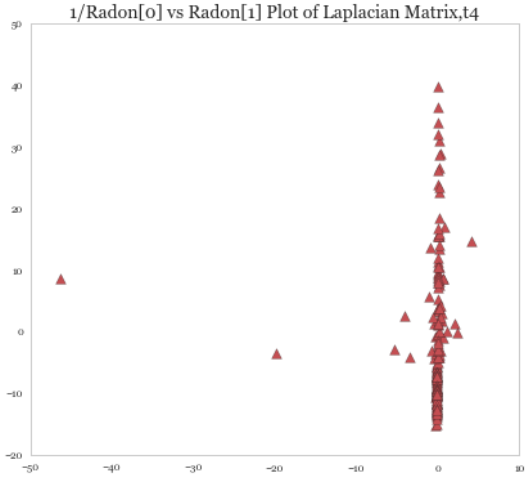

Radon Transform Plot of Laplacian Matrix,t2

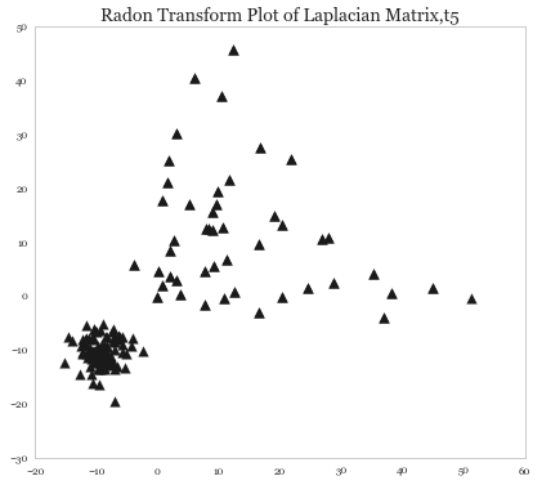

1/Radon[0] vs Radon[1] Plot of Laplacian Matrix,t2

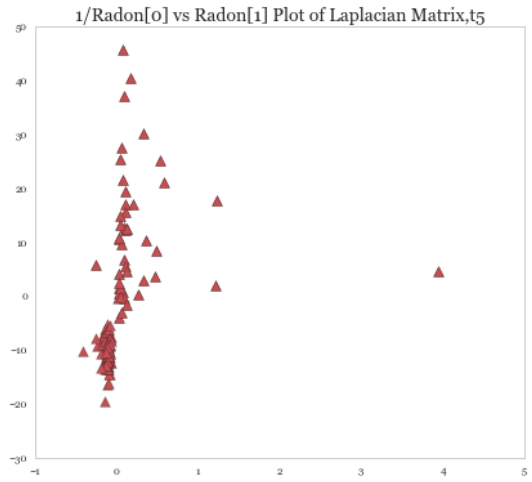

Radon Transform Plot of Laplacian Matrix,t3


1/Radon[0] vs Radon[1] Plot of Laplacian Matrix,t3


Radon Transform Plot of Laplacian Matrix,t4

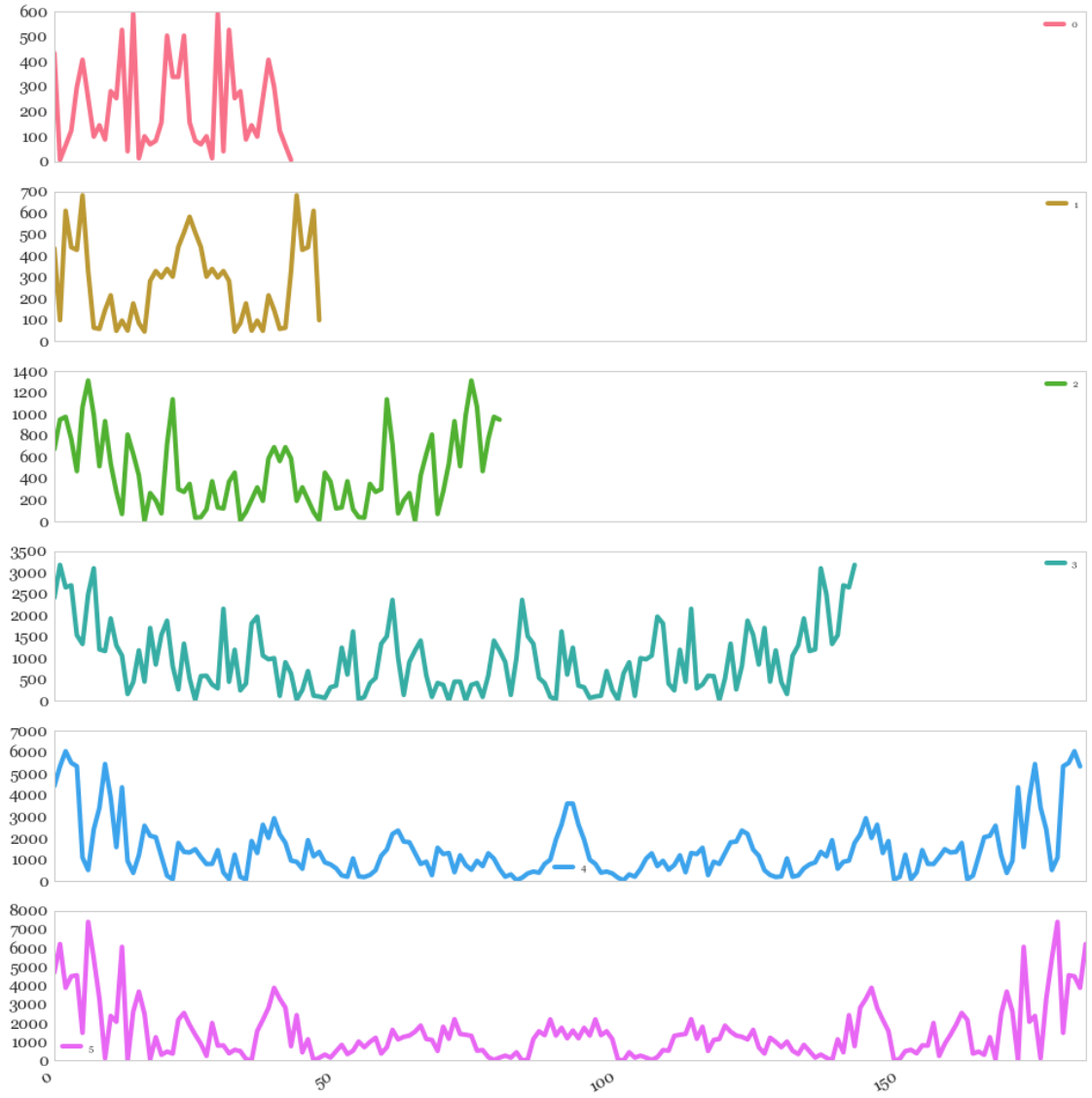

1/Radon[0] vs Radon[1] Plot of Laplacian Matrix,t4





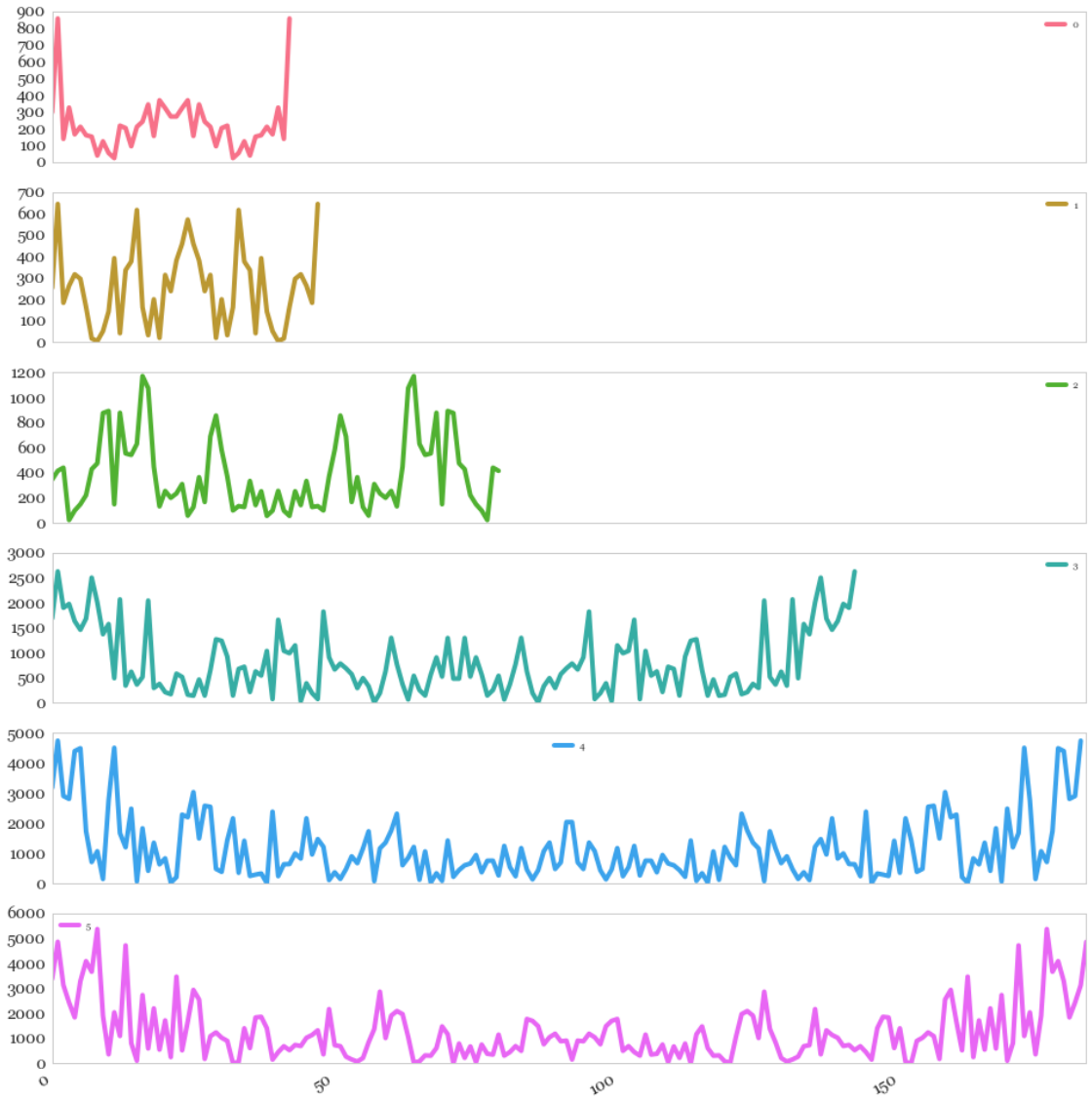
13 Instantaneous Frequency, ω

```
In [190]: ifreq_lap = pd.DataFrame([freq01,freq1,freq2,freq3,freq4,freq5]).T
          ifreq_adj = pd.DataFrame([freq03,freq13,freq23,freq33,freq43,freq53]).T
          ifreq_mod = pd.DataFrame([freq02,freq12,freq22,freq32,freq42,freq52]).T

In [191]: ifreq_adj= iphase_adj.applymap(lambda x: np.real(x)*2*np.pi)
          iifreq_lap = iphase_lap.applymap(lambda x: np.real(x)*2*np.pi)
          ifreq_mod = iphase_mod.applymap(lambda x: np.real(x)*2*np.pi)

In [211]: ifreq_adj.applymap(rms).plot(subplots=True, figsize=(16,18), fontsize=14)
          plt.suptitle('RMS Instantaneous Frequecy of Adjacency Matrix', fontsize=2

Out[211]: <matplotlib.text.Text at 0x1addb49fa58>
```

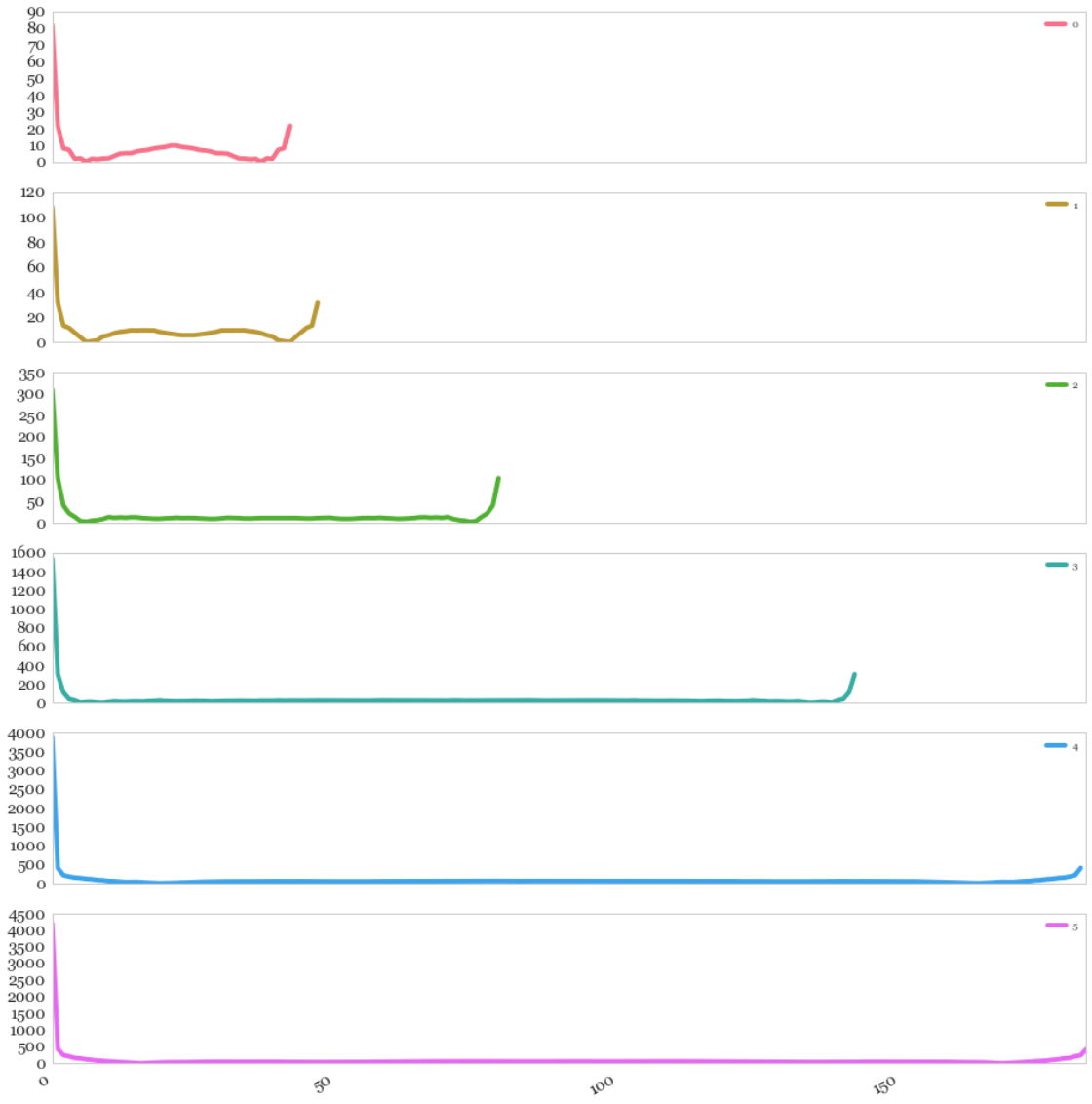



```
In [208]: ifreq_mod.applymap(rms).plot(subplots=True, figsize=(16,18), fontsize=14)
          plt.suptitle('RMS Instantaneous Frequecy of Modularity Matrix', fontsize=
```

```
Out[208]: <matplotlib.text.Text at 0x1add9abf780>
```




```
In [207]: ifreq_lap.applymap(rms).plot(subplots=True, figsize=(16,18), fontsize=14)
          plt.suptitle('RMS Instantaneous Frequecy of Laplacian Matrix', fontsize=2

C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\ma\core.py:2720: ComplexWarnin
  order=order, subok=True, ndmin=ndmin)
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:482: ComplexWa
  return array(a, dtype, copy=False, order=order)


Out[207]: <matplotlib.text.Text at 0x1add8cf1780>
```


RMS Instantaneous Frequecy of Laplacian Matrix
0
1
2
3
4
5
0
50
100
150

In [ ]: