Dynamic Network Analysis 00

July 19, 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('cubehelix')
from scipy.signal import *
from numpy.linalg 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_)
```

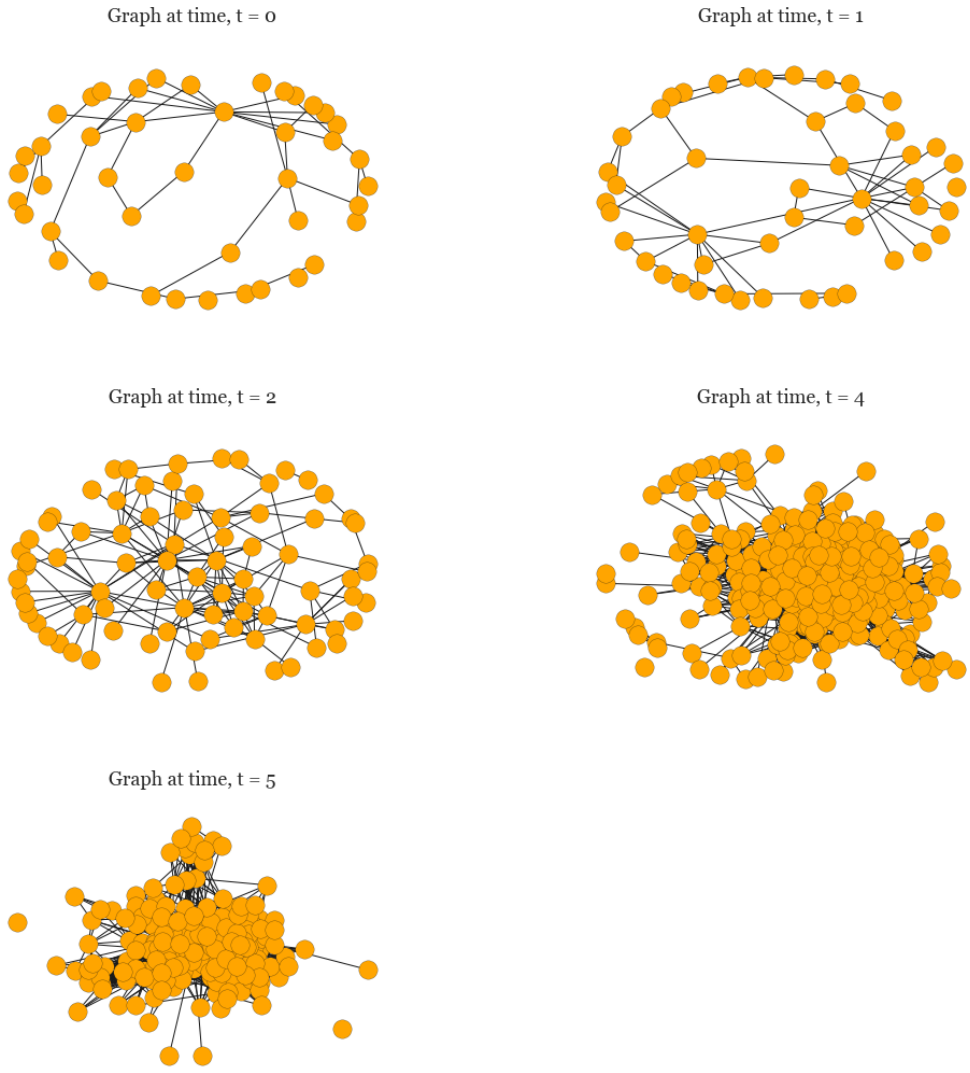
5.2.5 t = 4

```
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
```

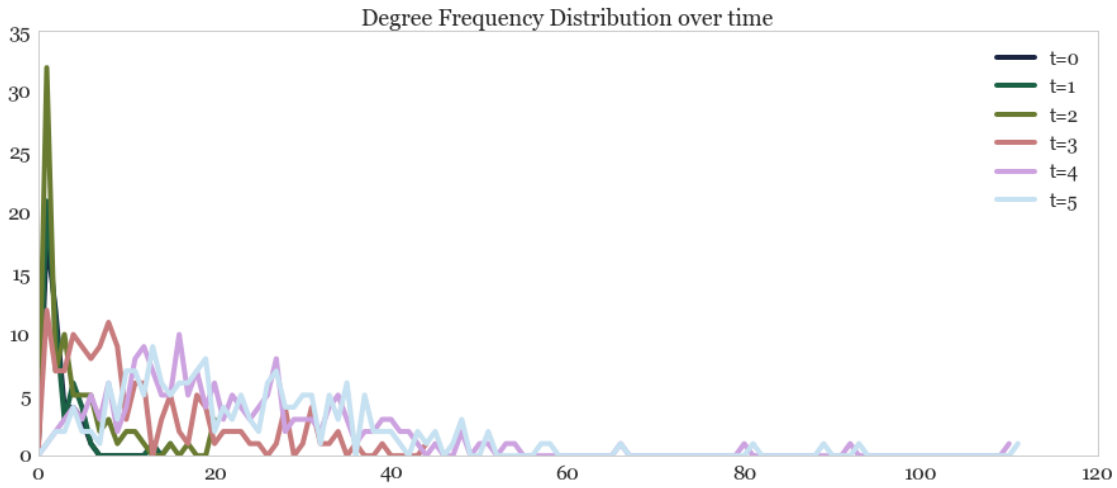
```
In [36]: def cal_stat(net):
             degC = nx.degree_centrality(net)
             cloC = nx.closeness_centrality(net)
             betC = nx.betweenness_centrality(net)
             eigC = nx.eigenvector_centrality_numpy(net)
             algC = nx.algebraic_connectivity(net)
             clustC = 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

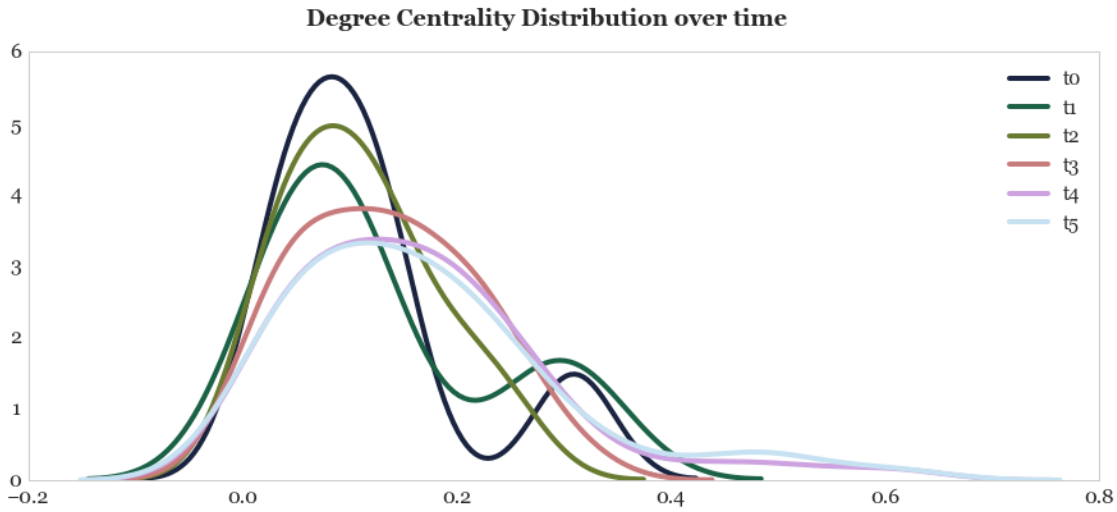
```
In [38]: plt.title("Degree Frequency Distribution over time", fontsize=18)
         plt.plot(nx.degree_histogram(Gt0), label='t=0')
         plt.plot(nx.degree_histogram(Gt1), label='t=1')
         plt.plot(nx.degree_histogram(Gt2), label='t=2')
         plt.plot(nx.degree_histogram(Gt3), label='t=3')
         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 0x2273b6cacf8>
```



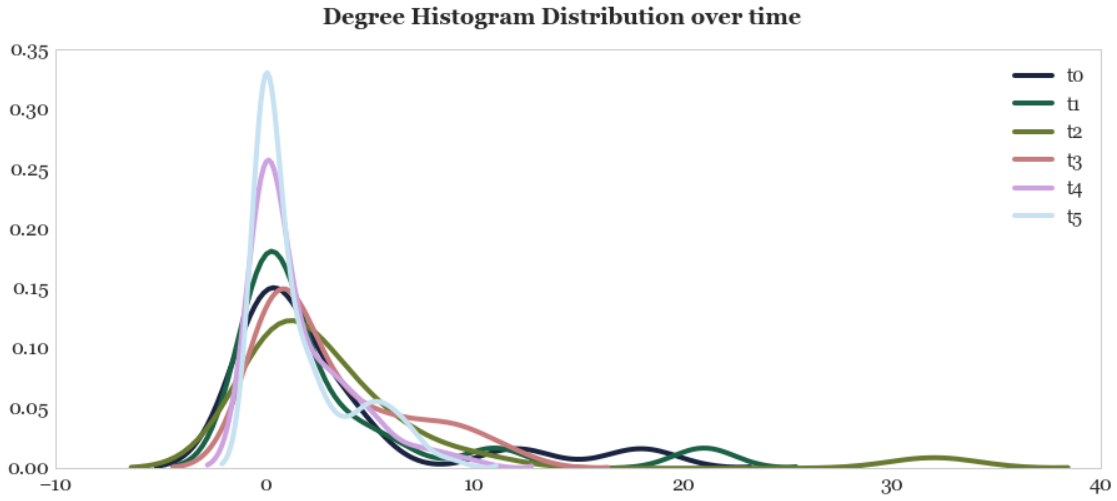

```
In [39]: plt.suptitle('Degree Centrality Distribution over time', fontsize=18)
         sns.distplot(get_val(degC0), hist=False, label='t0')
         sns.distplot(get_val(degC1), hist=False, label='t1')
         sns.distplot(get_val(degC2), hist=False, label='t2')
         sns.distplot(get_val(degC3), 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 0x2273c1df780>
```



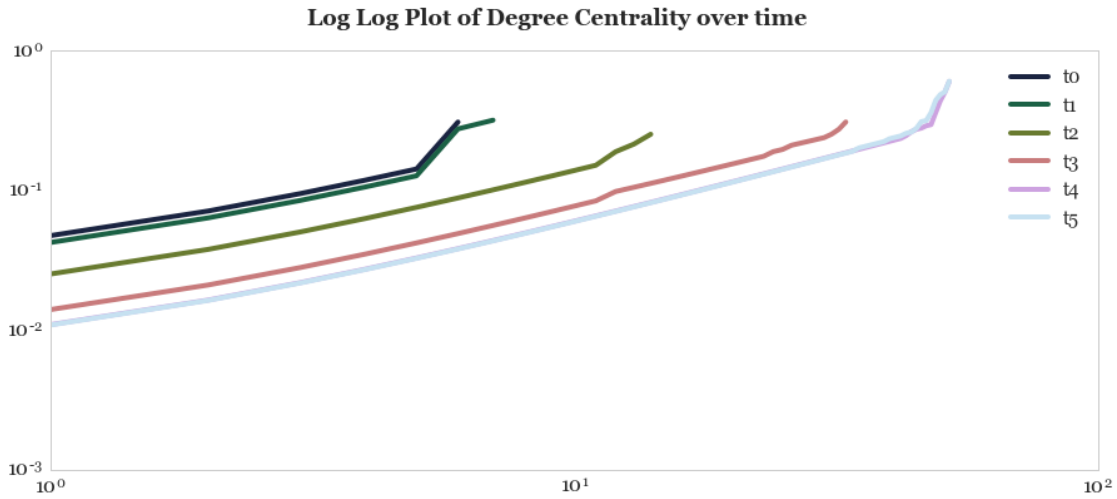
```
In [40]: plt.suptitle('Degree Histogram Distribution over time', fontsize=18)
         sns.distplot(nx.degree_histogram(Gt0), hist=False, label='t0')
         sns.distplot(nx.degree_histogram(Gt1), hist=False, label='t1')
         sns.distplot(nx.degree_histogram(Gt2), hist=False, label='t2')
         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 0x2273c228cc0>
```




```
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)
```

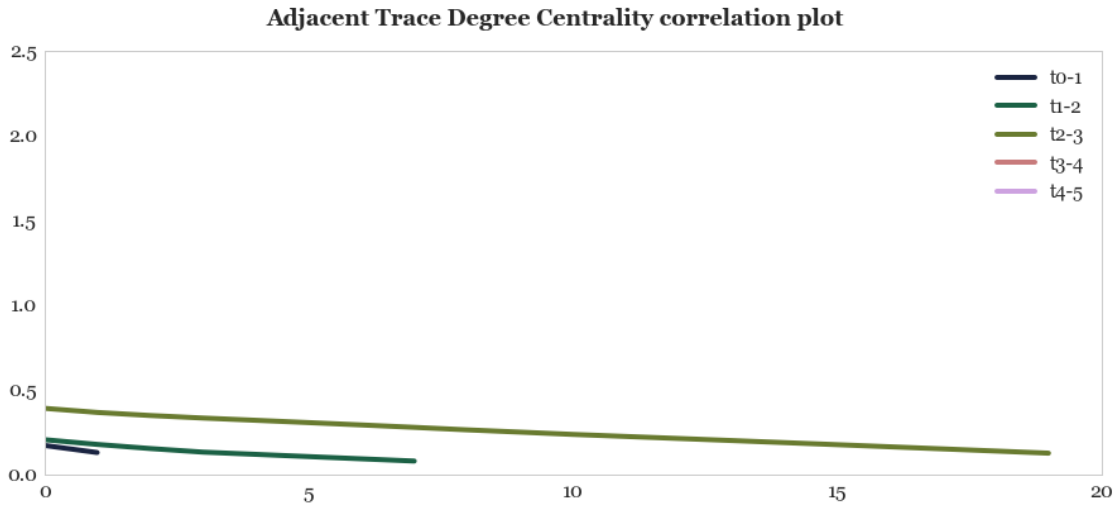
```
        plt.xticks(fontsize=16)
        plt.legend(loc=1, fontsize=15)
```

```
Out[41]: <matplotlib.legend.Legend at 0x2273b621160>
```

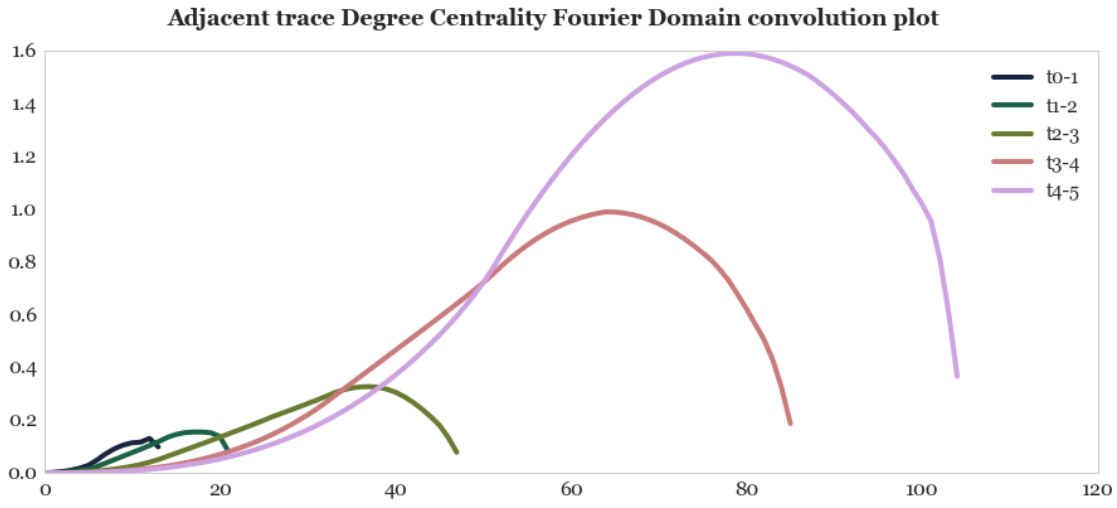



```
In [42]: plt.suptitle('Adjacent Trace Degree Centrality correlation plot', fontsize

        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 0x2273c2d2908>
```



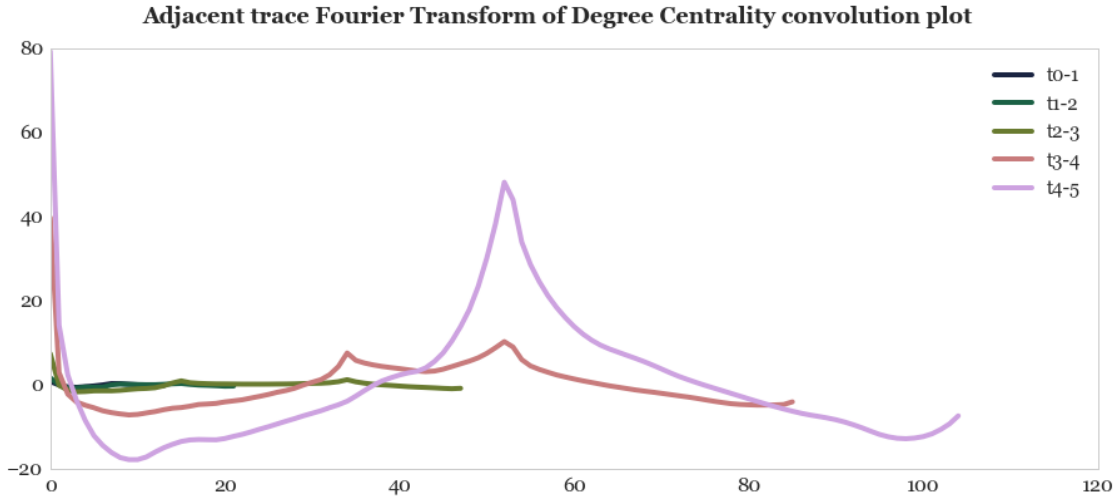

```
In [43]: plt.suptitle('Adjacent trace Degree Centrality Fourier Domain convolution

         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 0x2273bc89e10>
```



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)
```

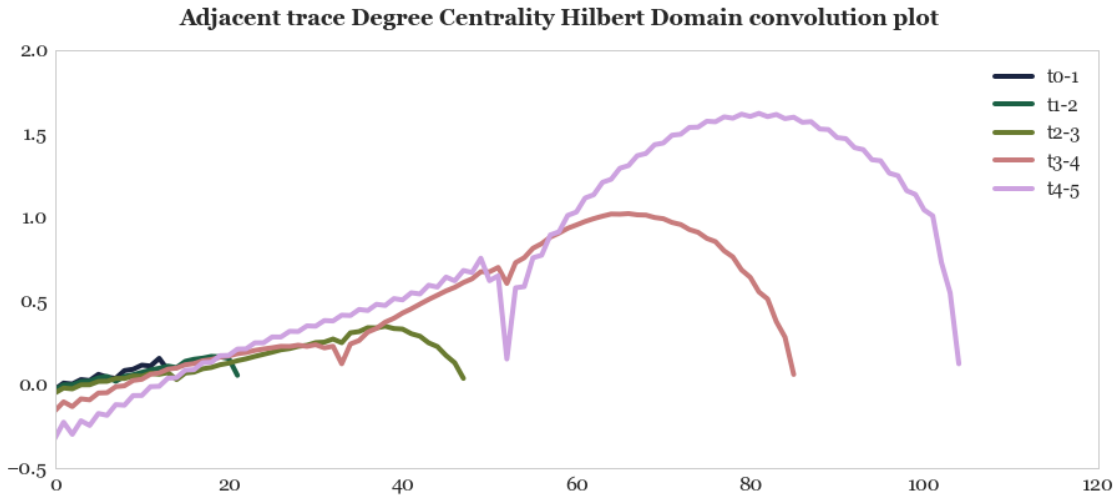
Out[44]: <matplotlib.legend.Legend at 0x2273bf07908>



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 0x2273bffbf60>
```



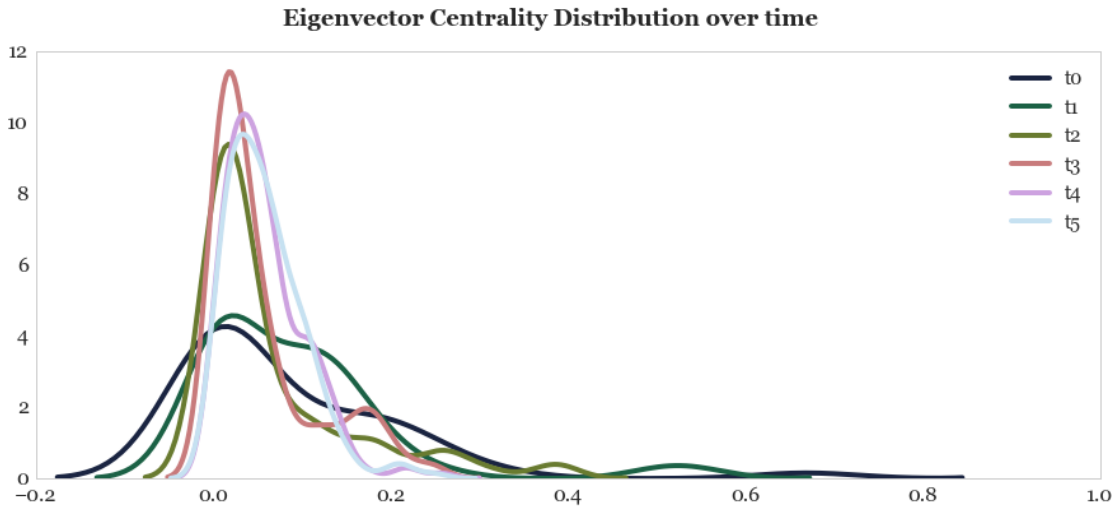
6.1.3 Eigenvector Centrality Histograms

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)
```

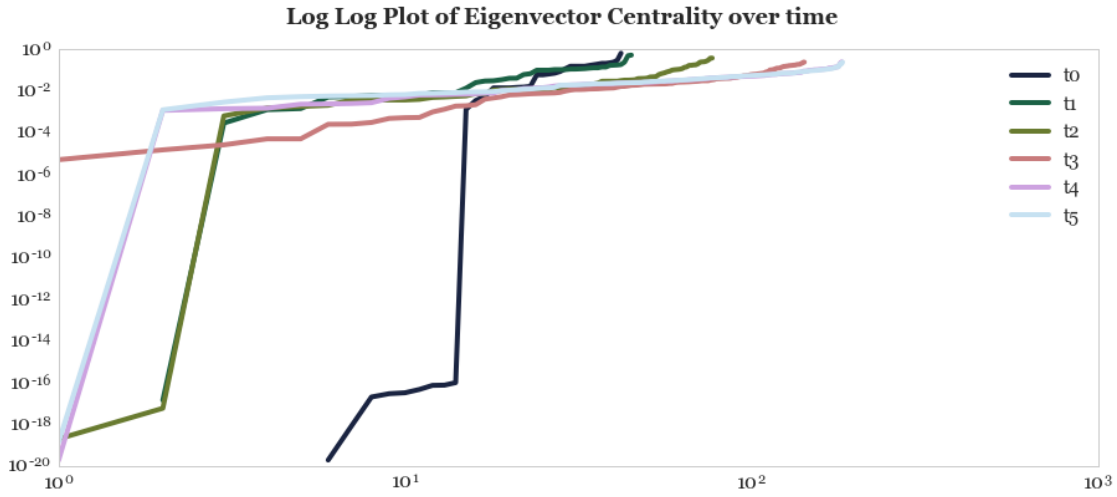
```
C:\Users\arsha_000\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

```
Out[46]: <matplotlib.legend.Legend at 0x2273bbbbcf8>
```



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)
```

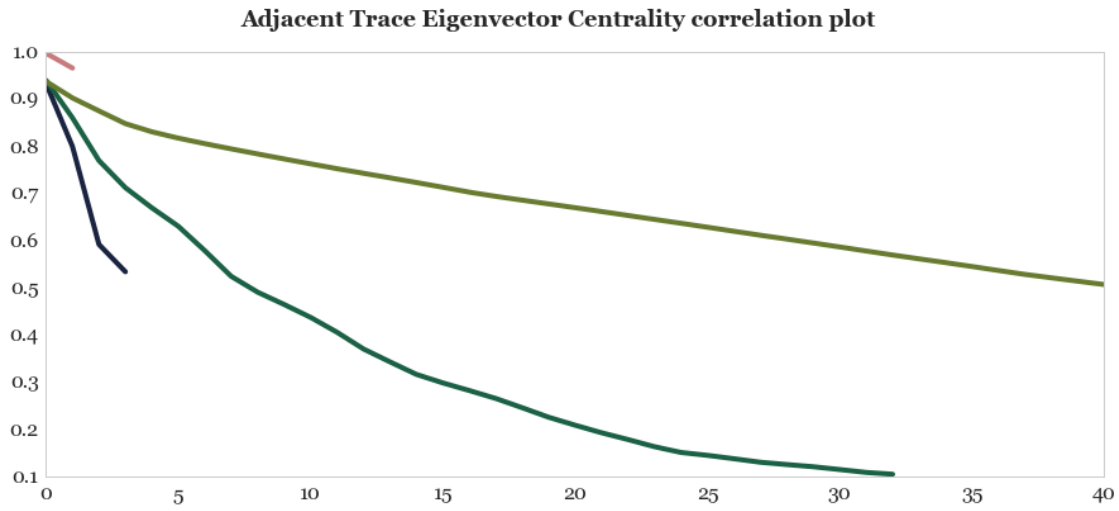
Out[47]: <matplotlib.legend.Legend at 0x2273c670a20>



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

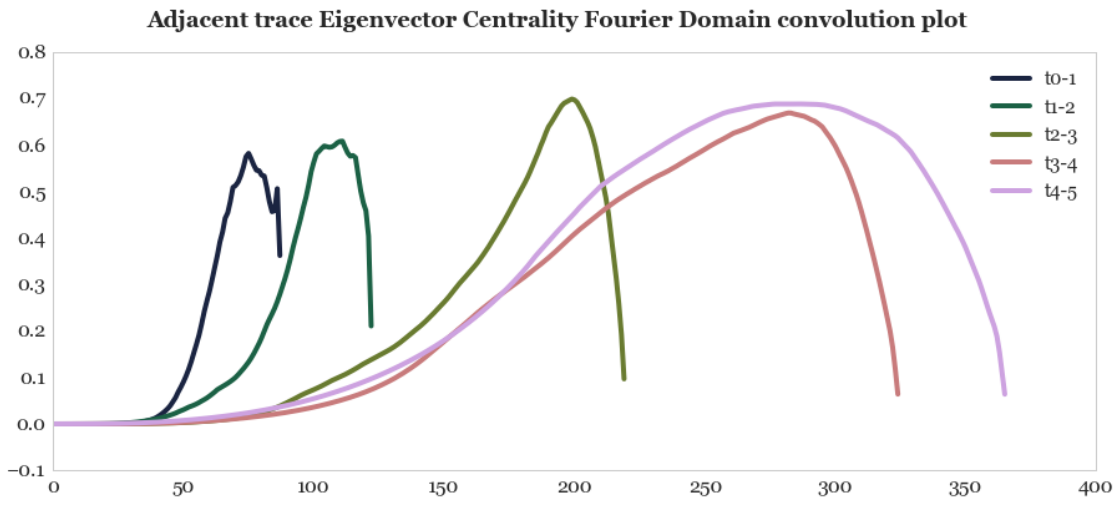
```
In [49]: plt.suptitle('Adjacent Trace Eigenvector Centrality correlation plot', fon

         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)
```

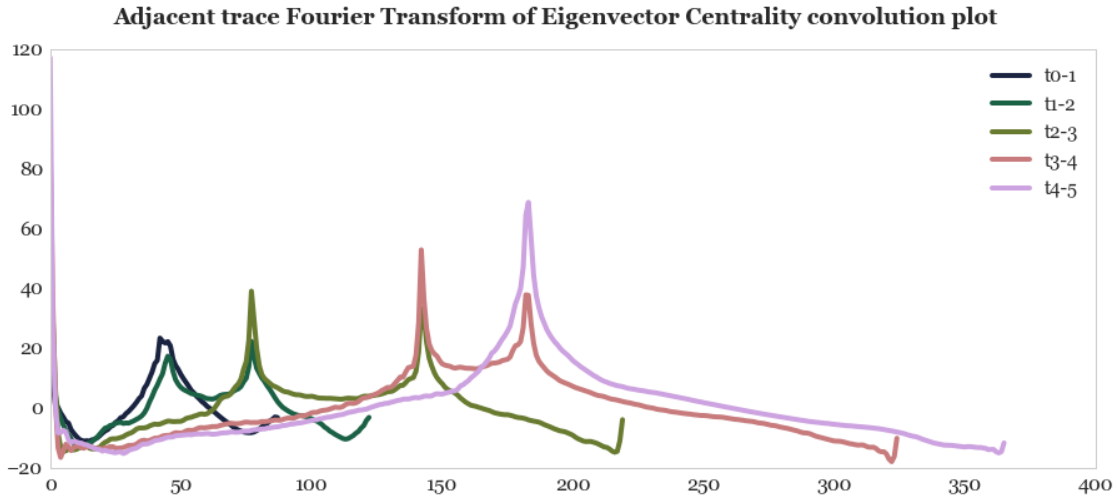
Out[50]: <matplotlib.legend.Legend at 0x2273eb25d68>



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)
```

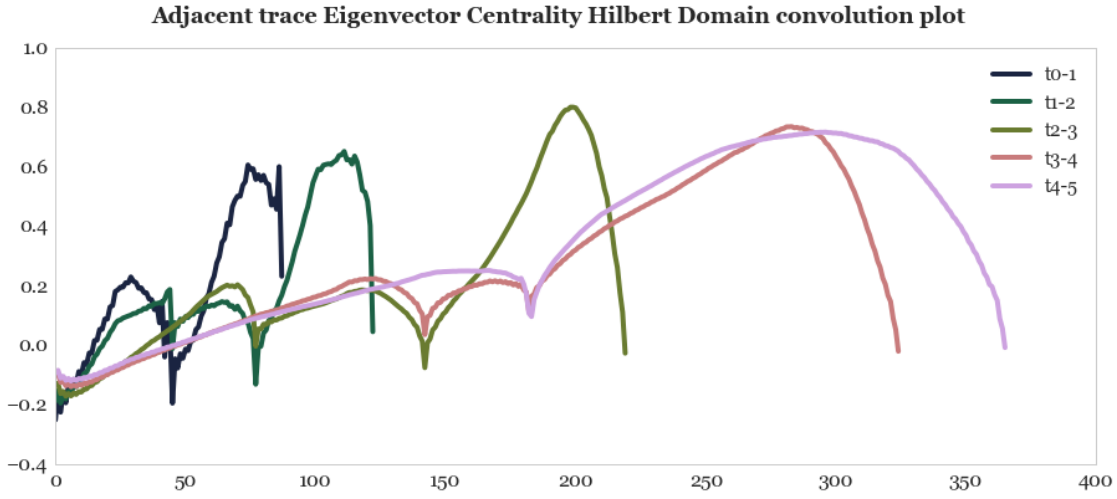
Out[51]: <matplotlib.legend.Legend at 0x2273ec1be48>




```
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 0x2273bbf7048>
```

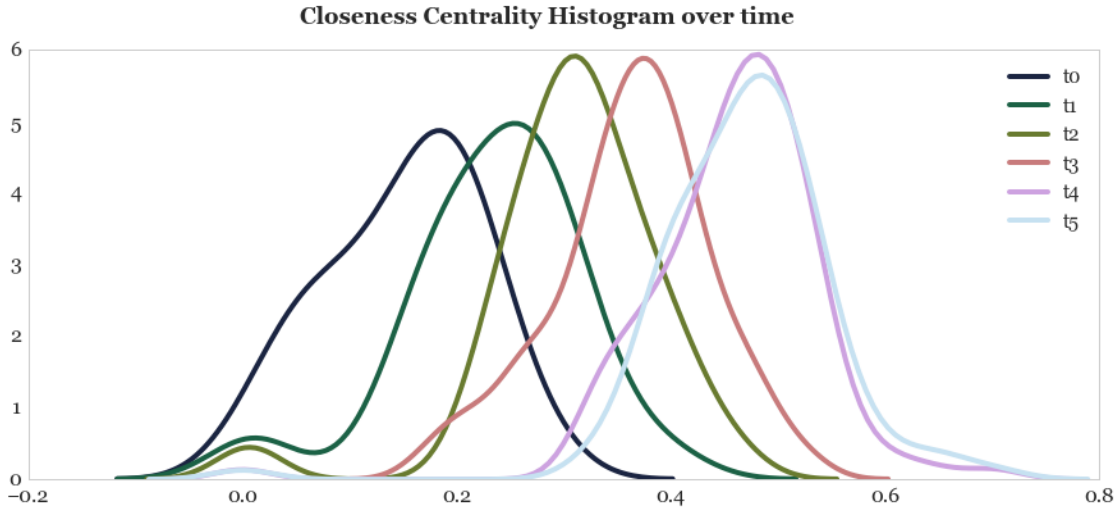


```
In [53]: print(np.correlate(eigC0_a,eigC1_a))
         print(np.correlate(eigC0_a,eigC1_a)/np.sqrt(np.correlate(eigC0_a,eigC1_a))

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

6.1.4 Closeness Centrality Histograms

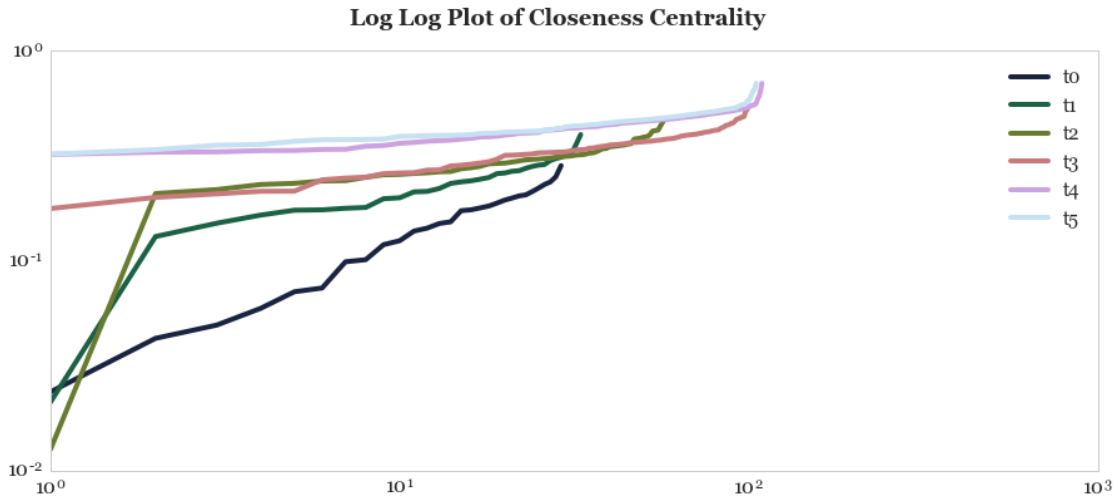
The Closeness Centrality shows a much better evolution over time than the Eigenvector Centrality Histograms

```
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 0x2273bd45f28>
```



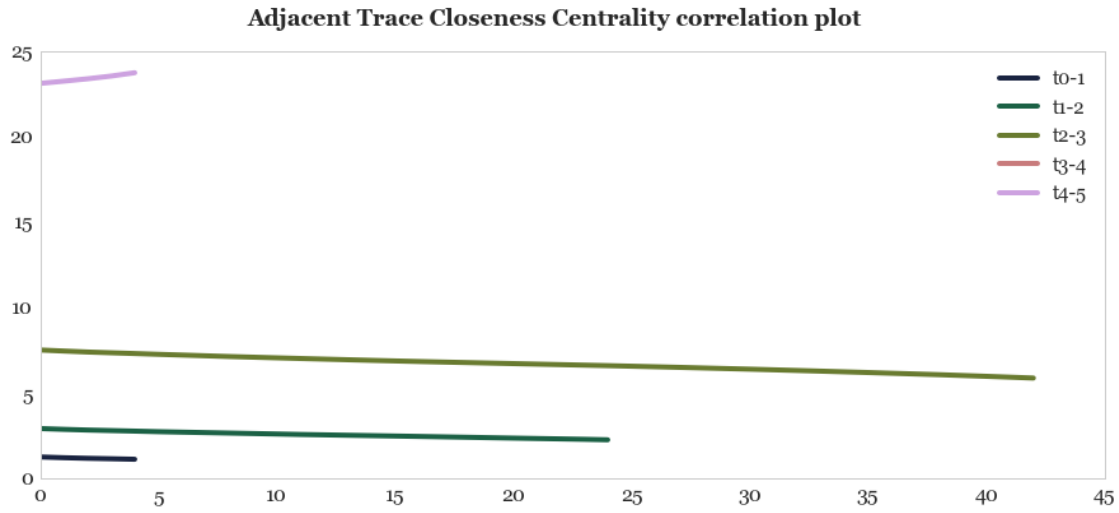
In [55]:
```
plt.suptitle('Log Log Plot of Closeness Centrality', fontsize=18)

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 0x2273da6bcc0>

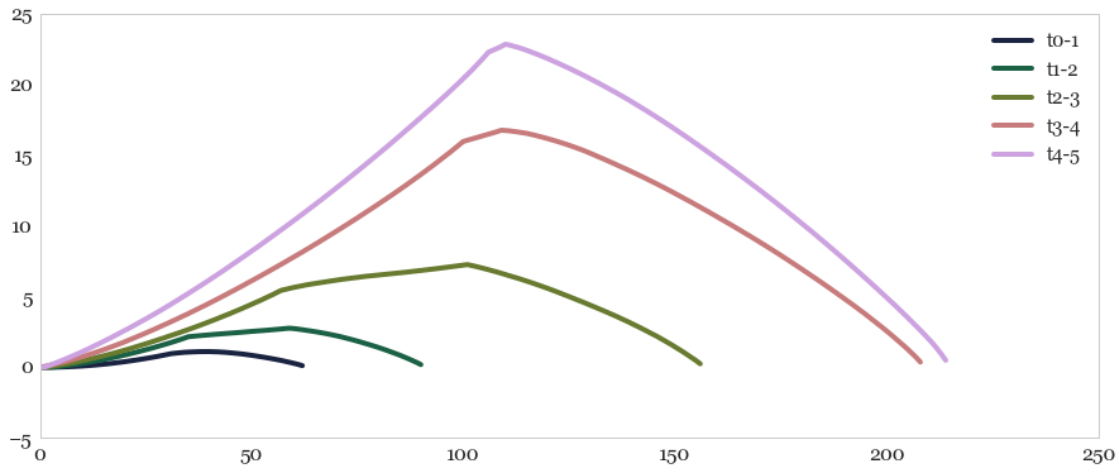


In [56]:
```
plt.suptitle('Adjacent Trace Closeness Centrality correlation plot', fonts

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 0x2273eca70f0>



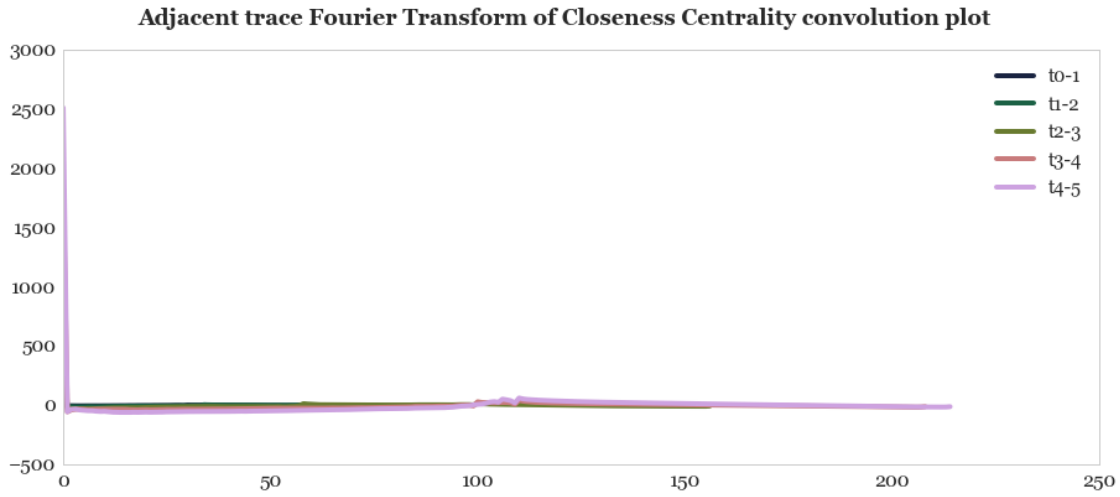

```
In [57]: plt.suptitle('Adjacent trace Closeness Centrality Fourier Domain convoluti

         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 0x22740f68550>
```



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)
```

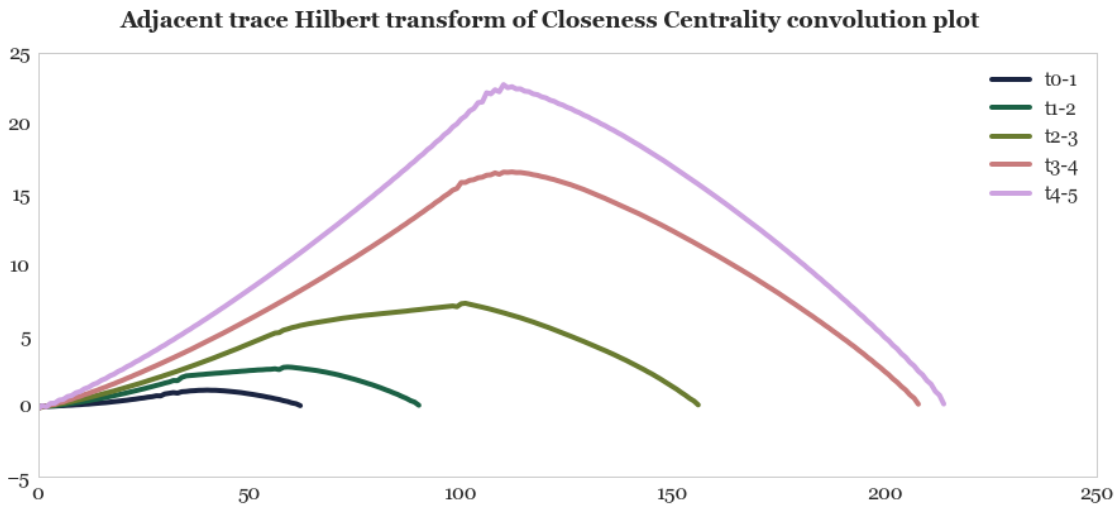
Out[58]: <matplotlib.legend.Legend at 0x22740e849e8>



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 0x22740f0ce48>

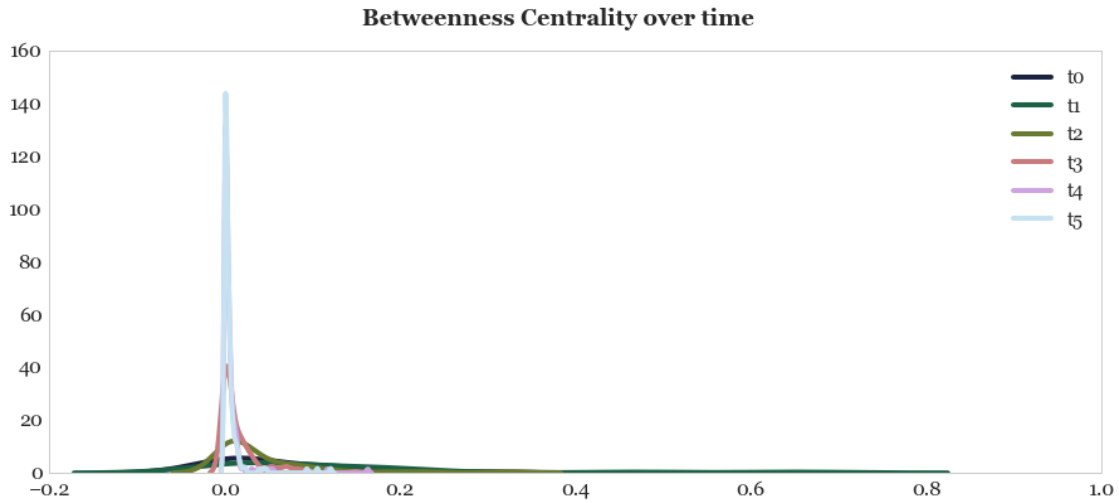


6.1.5 Betweenness Centrality Histogram

```
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)
```

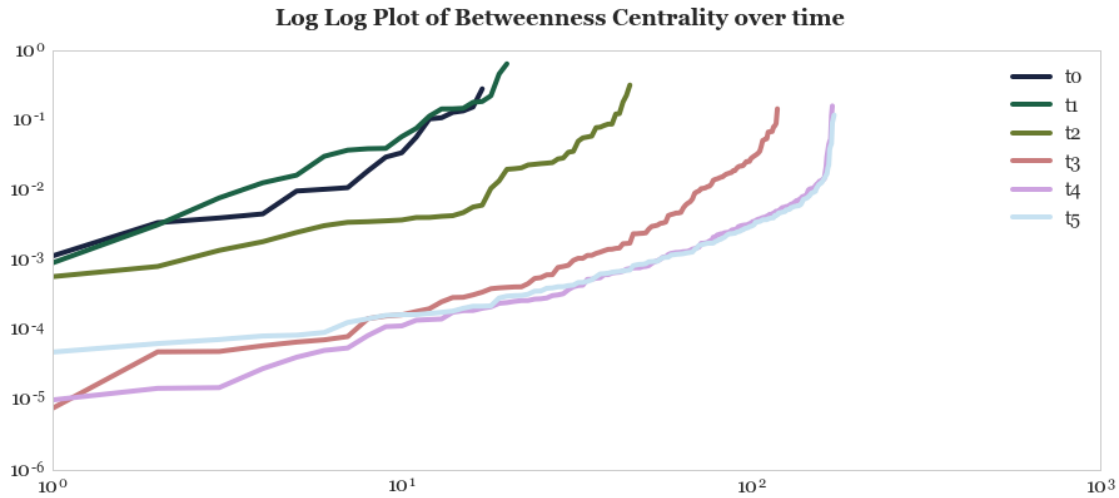
```
C:\Users\arsha_000\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

Out[60]: <matplotlib.legend.Legend at 0x2274162fd68>

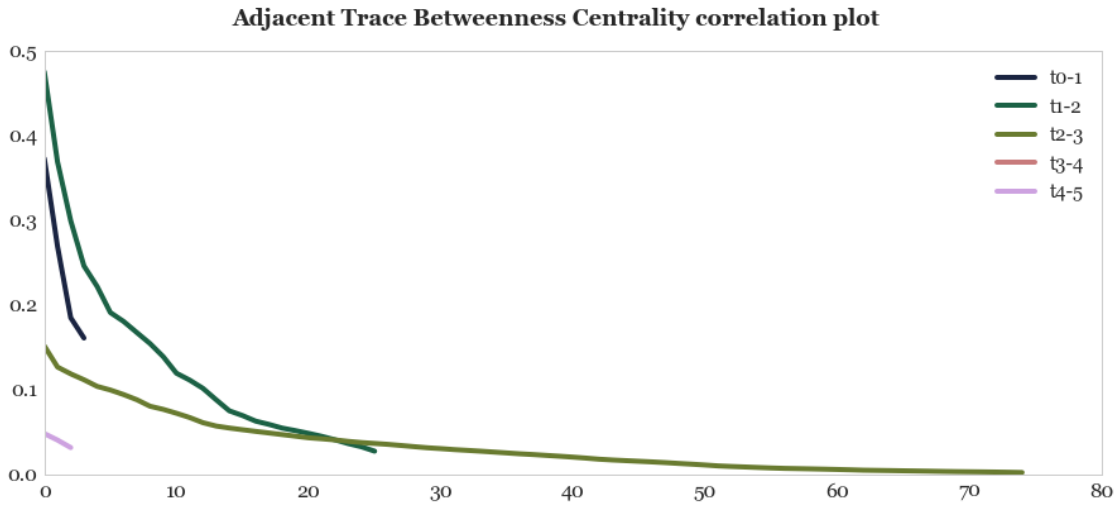


In [61]:
```
plt.suptitle('Log Log Plot of Betweenness Centrality over time', fontsize=

        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)
```

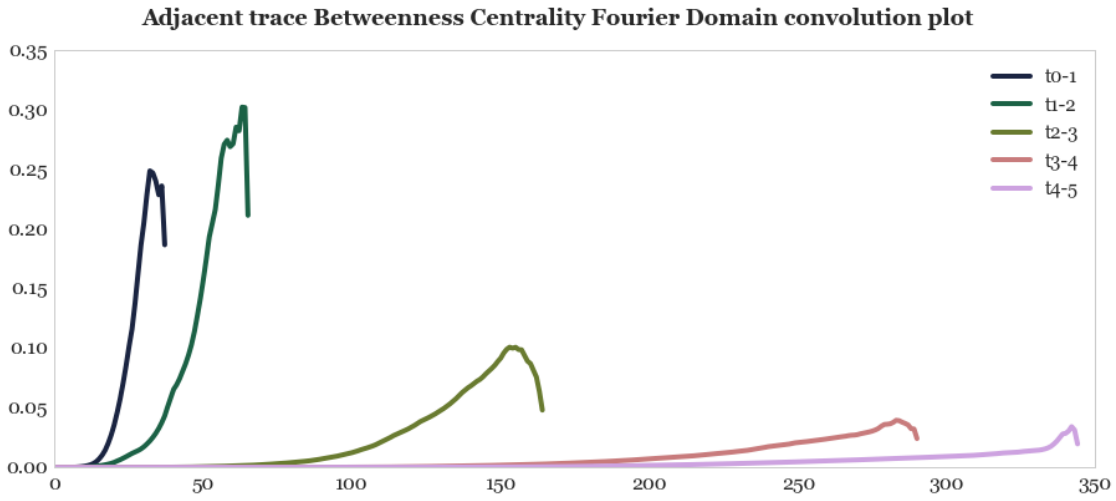
Out[61]:
```
<matplotlib.legend.Legend at 0x227418ade80>
```



```
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 0x2273bdb2a90>
```



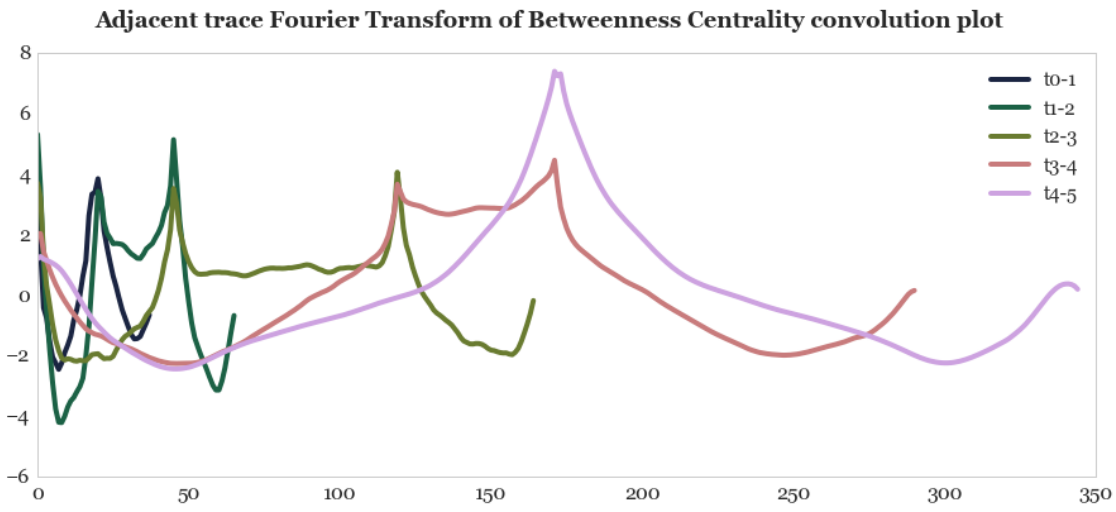

```
In [63]: plt.suptitle('Adjacent trace Betweenness Centrality Fourier Domain convolu

         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 0x22740f66588>
```



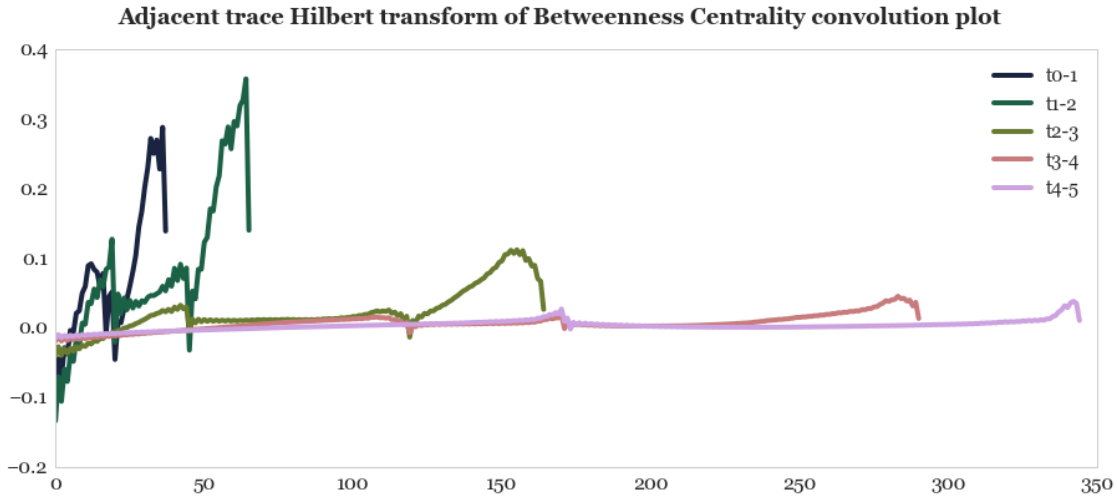
In [64]:
```
plt.suptitle('Adjacent trace Fourier Transform of Betweenness Centrality c

        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 0x22740051eb8>
```



```
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 0x2273ffa0cf8>
```



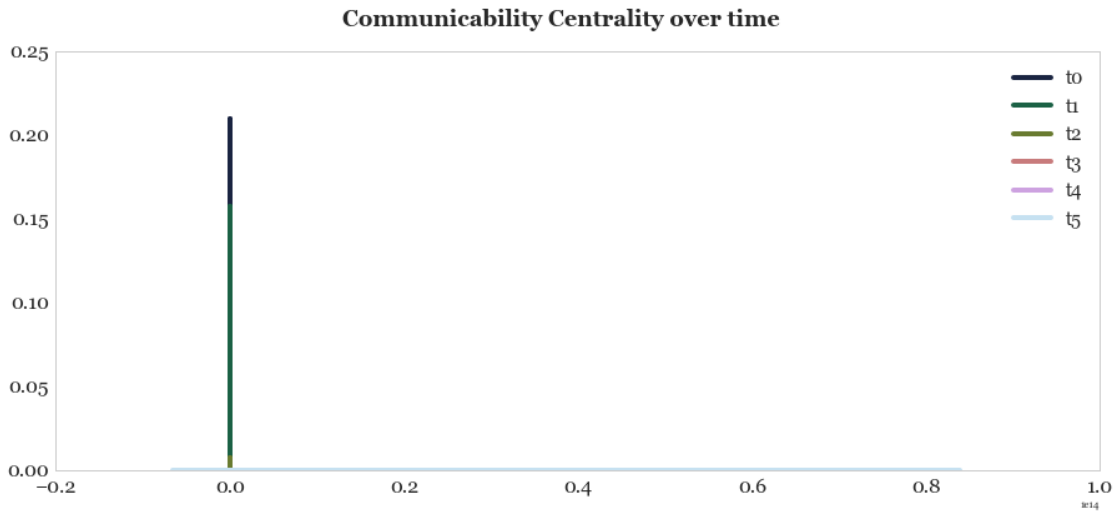
6.1.6 Communicability Centrality Histograms

```
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)
```

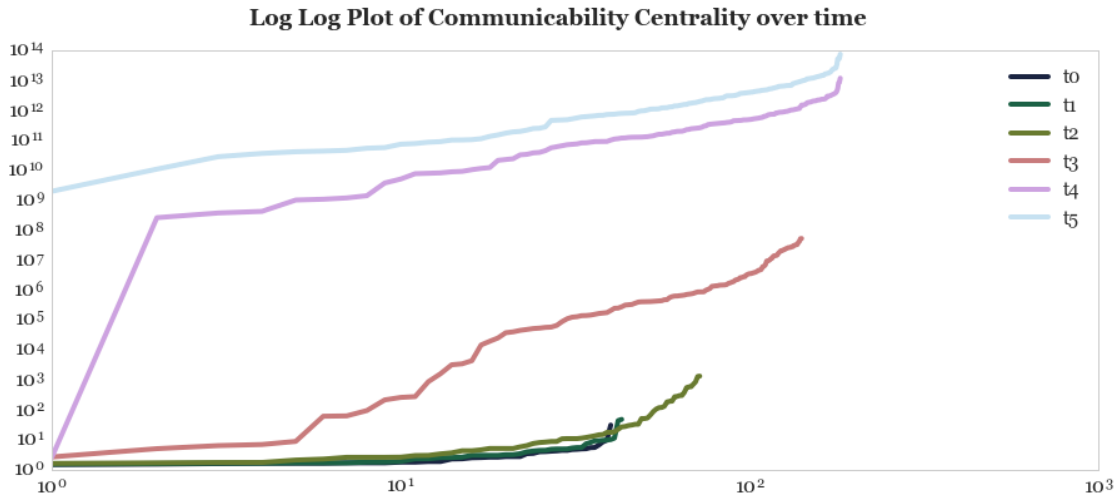
```
C:\Users\arsha_000\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

Out[66]: <matplotlib.legend.Legend at 0x2274009c5c0>

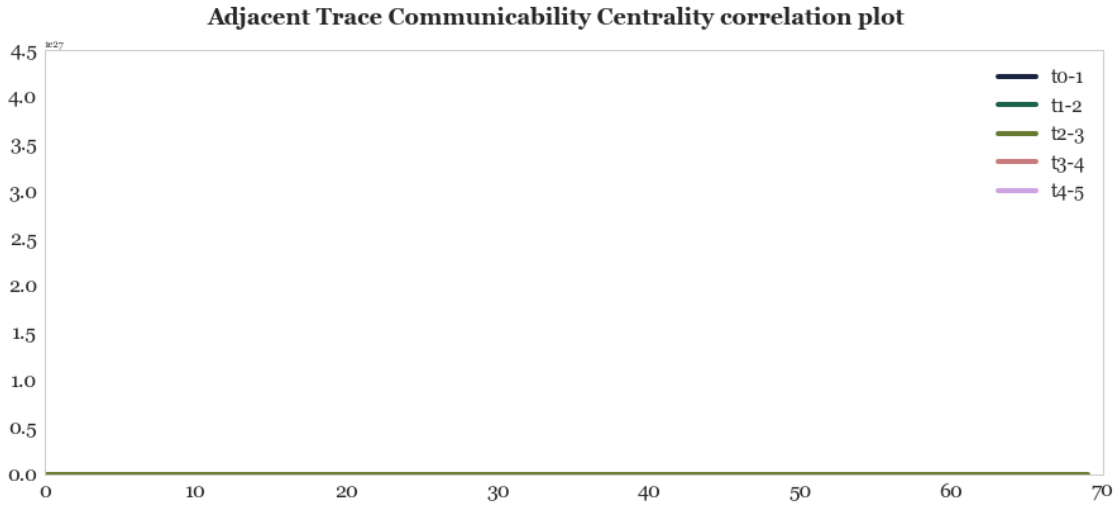


In [67]:
```
plt.suptitle('Log Log Plot of Communicability Centrality over time', fonts

        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 0x22741fcfba8>

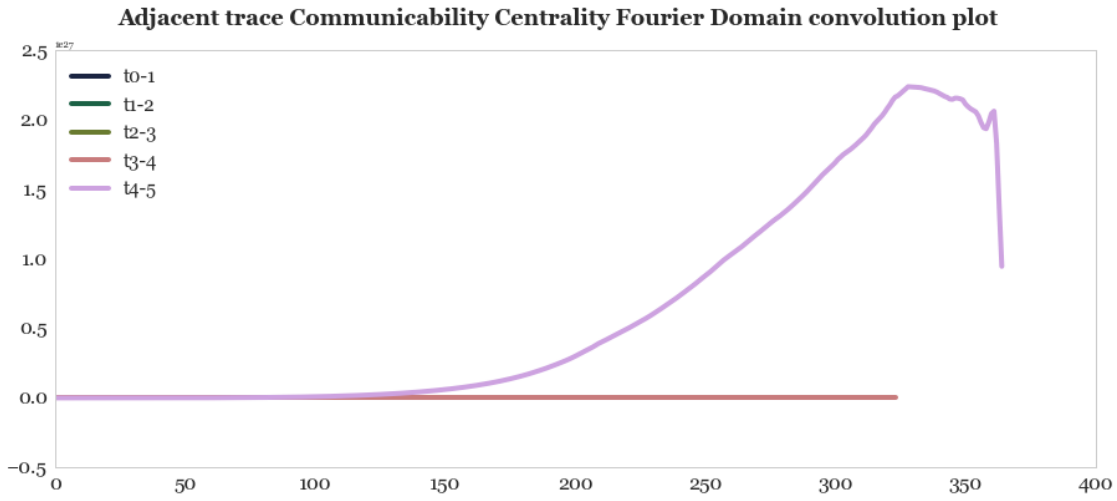



```
In [68]: plt.suptitle('Adjacent Trace Communicability Centrality correlation plot',

         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 0x22743c4dda0>
```



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=2, fontsize=15)
```

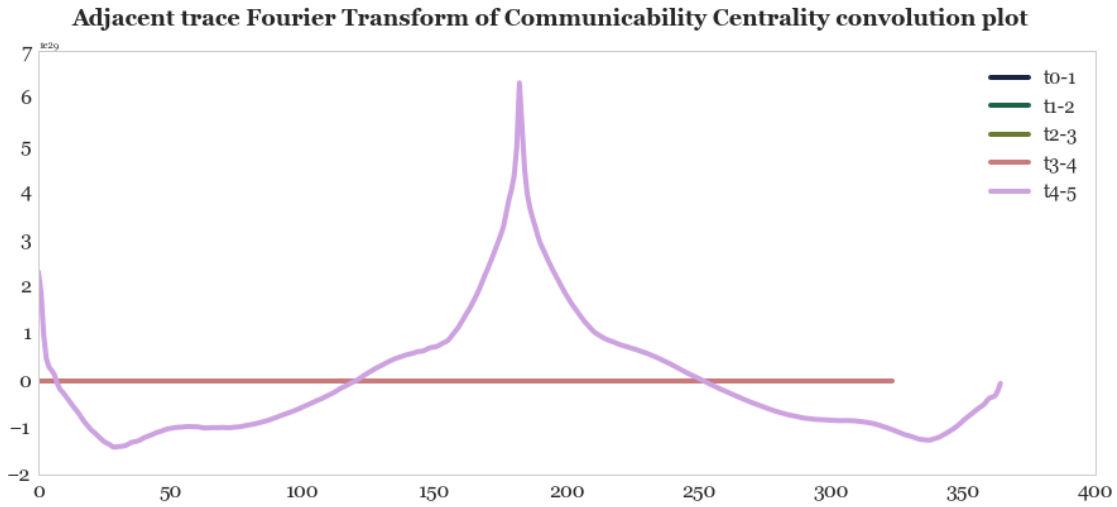
Out[69]: <matplotlib.legend.Legend at 0x22743e43e80>



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)
```

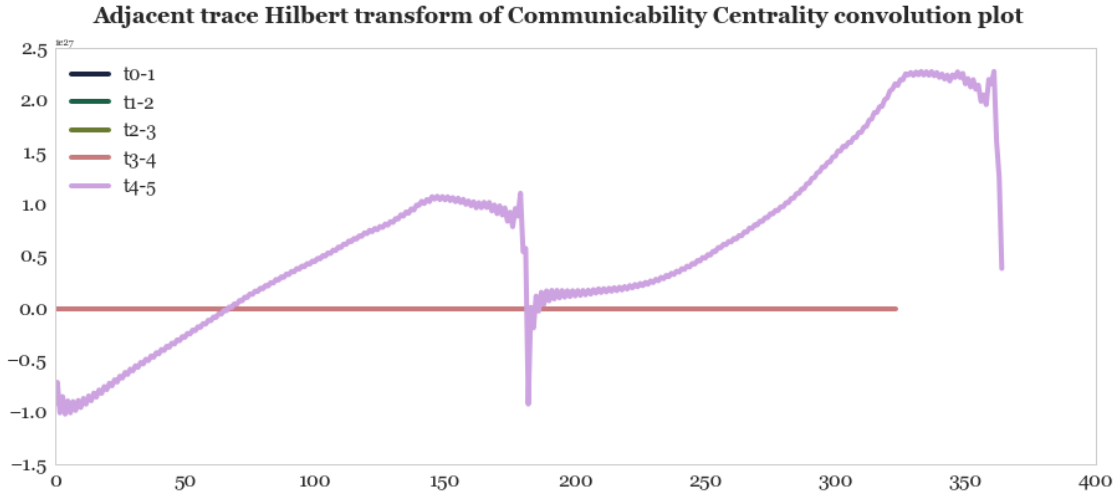
Out[70]: <matplotlib.legend.Legend at 0x2274404add8>



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=2, 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 0x22744242f28>
```

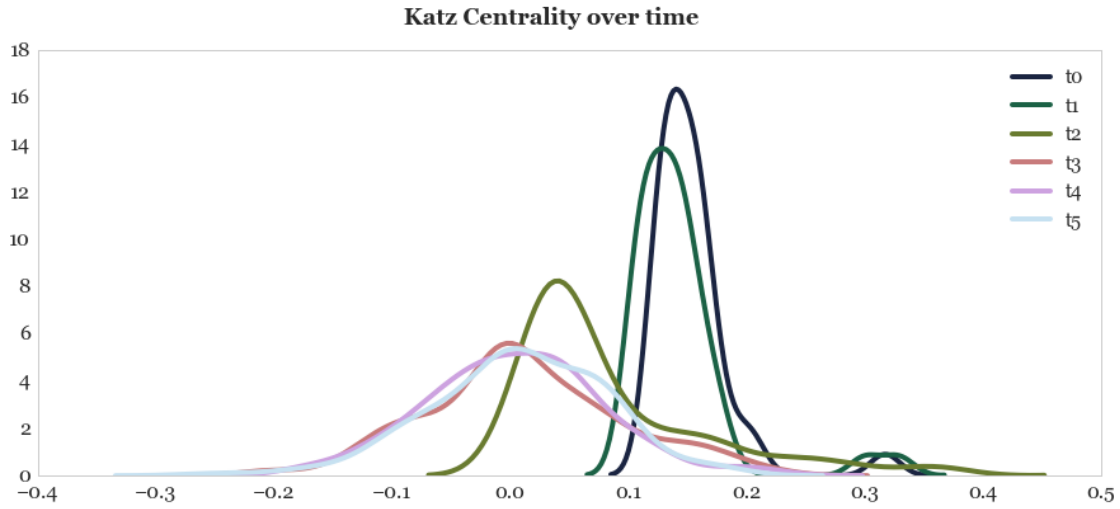


6.1.7 Katz Centrality Histograms

```
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
```

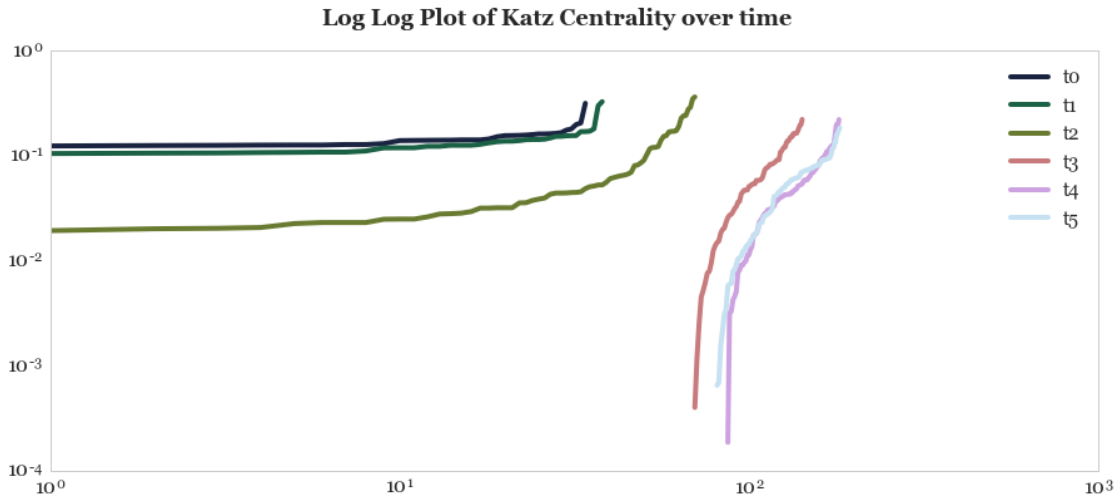
```
Out[72]: <matplotlib.legend.Legend at 0x22741d33588>
```




```
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')
```

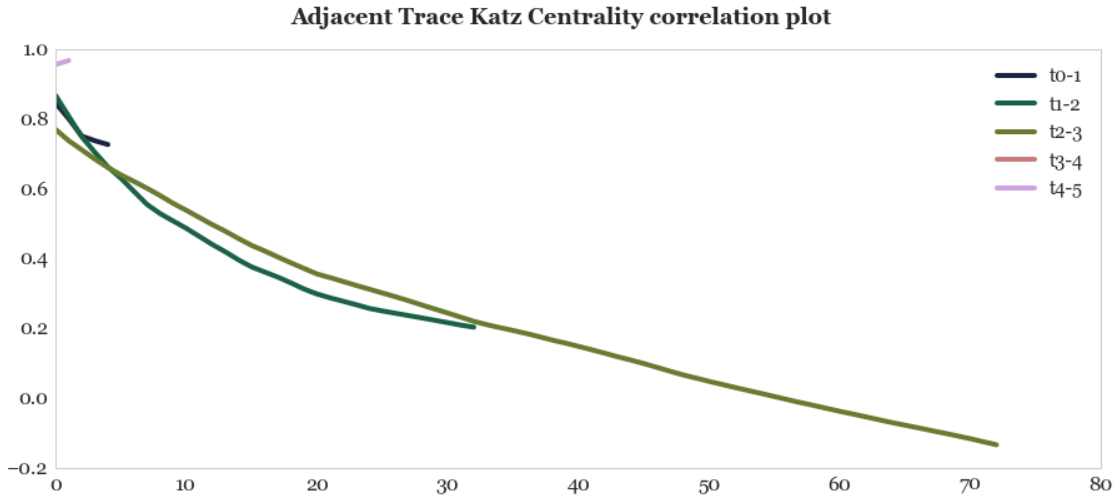
```
        plt.loglog(get_val(katzC5), label='t5')

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

```
Out[73]: <matplotlib.legend.Legend at 0x2274201cdd8>
```

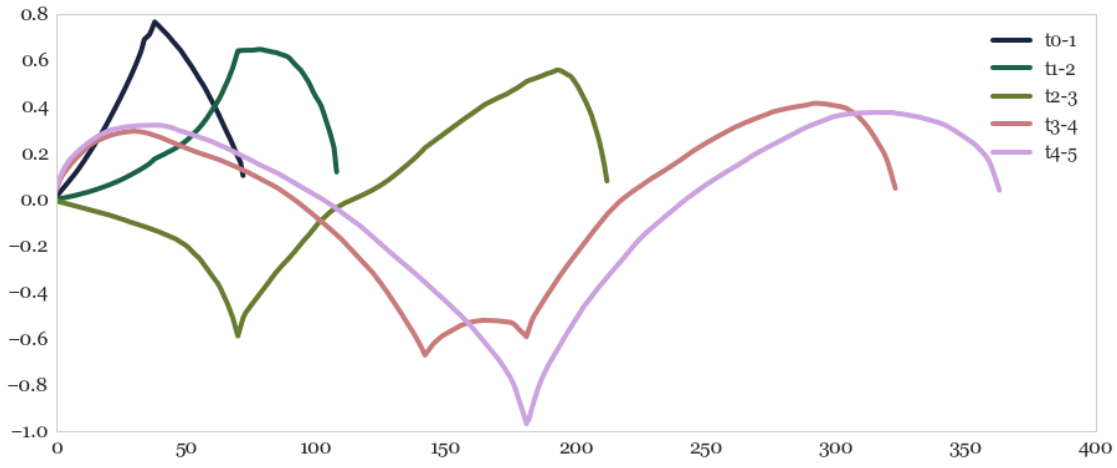



```
In [74]: plt.suptitle('Adjacent Trace Katz Centrality correlation plot', fontsize=1

        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 0x227449c2a90>
```



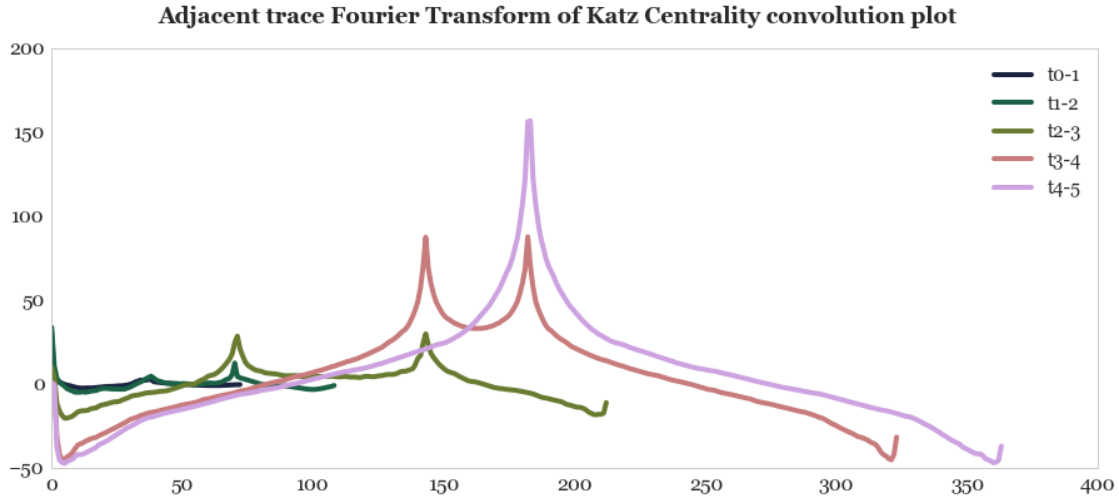

```
In [75]: plt.suptitle('Adjacent trace Katz Centrality Fourier Domain convolution pl

         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 0x22744b7a240>
```



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)
```

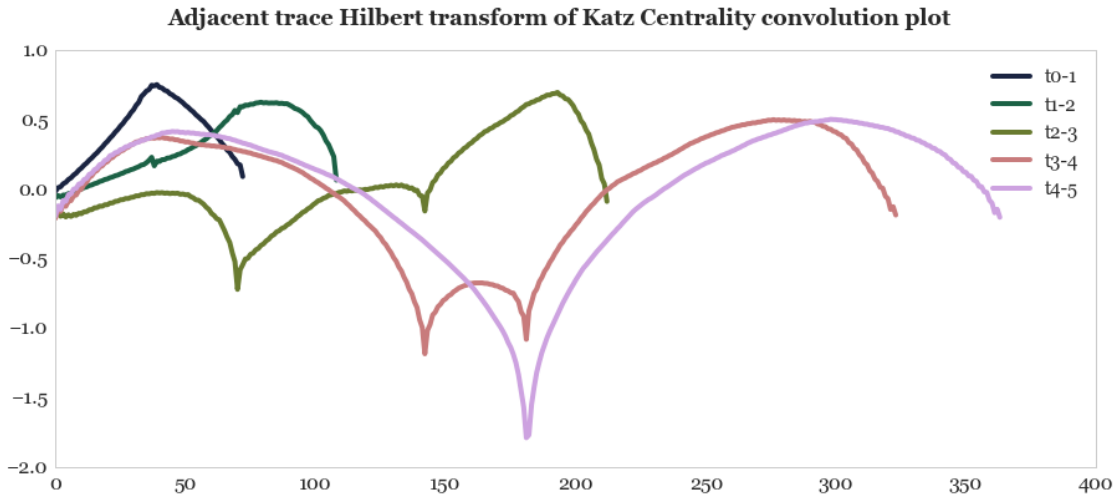
Out[76]: <matplotlib.legend.Legend at 0x22741b045c0>



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 0x22744453cc0>
```

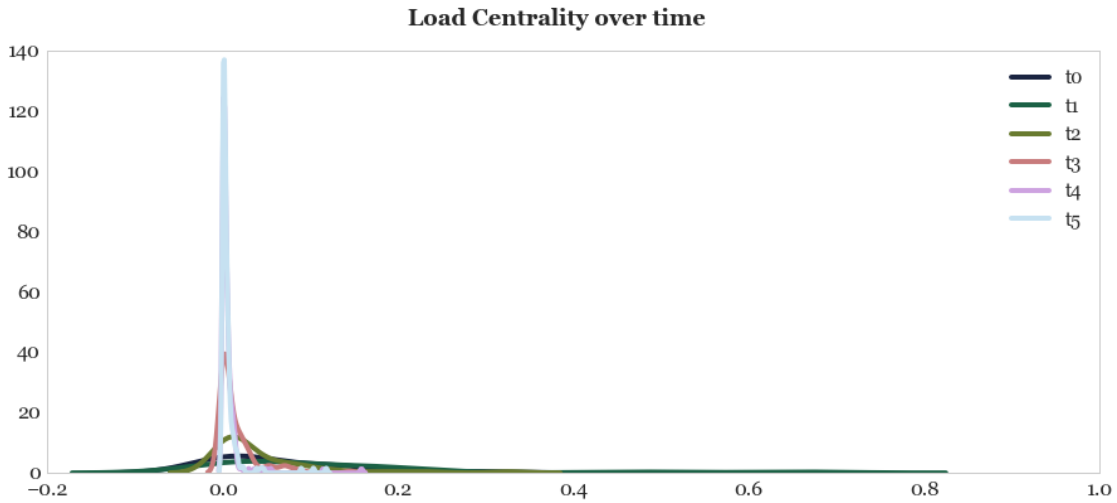


6.1.8 Load Centrality

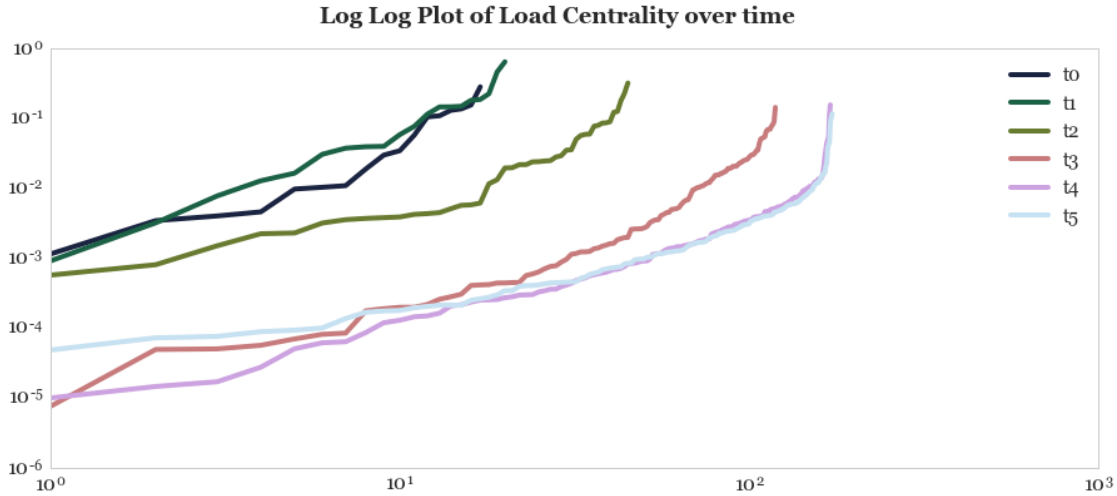
```
In [78]: plt.suptitle('Load Centrality over time', fontsize=18)

         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
```

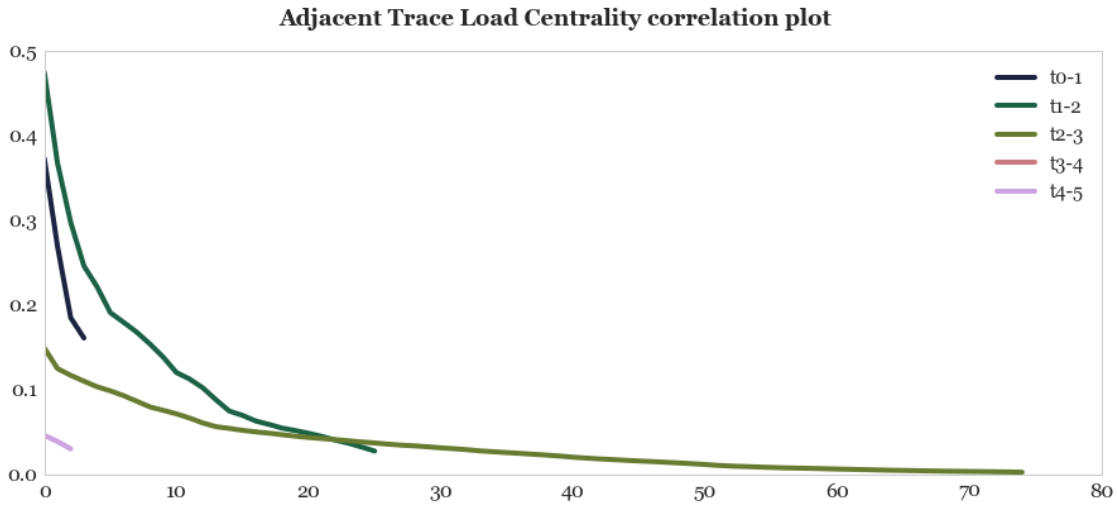
```
Out[78]: <matplotlib.legend.Legend at 0x22742b11a90>
```



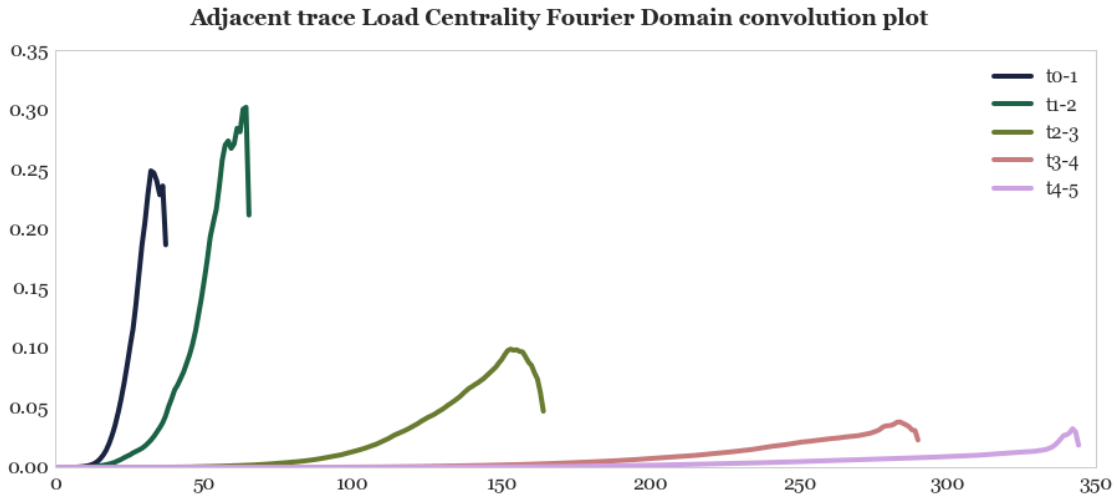

```
In [79]: plt.suptitle('Log Log Plot of Load Centrality over time', fontsize=18)

         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 0x22743b95390>
```



```
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 0x227427253c8>
```



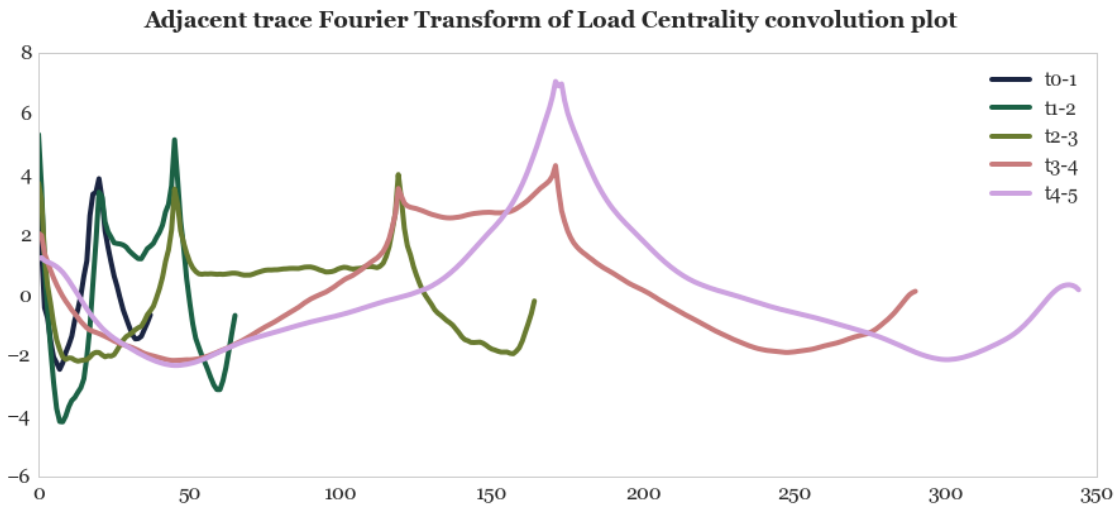

```
In [81]: plt.suptitle('Adjacent trace Load Centrality Fourier Domain convolution pl

         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 0x22744686ba8>
```



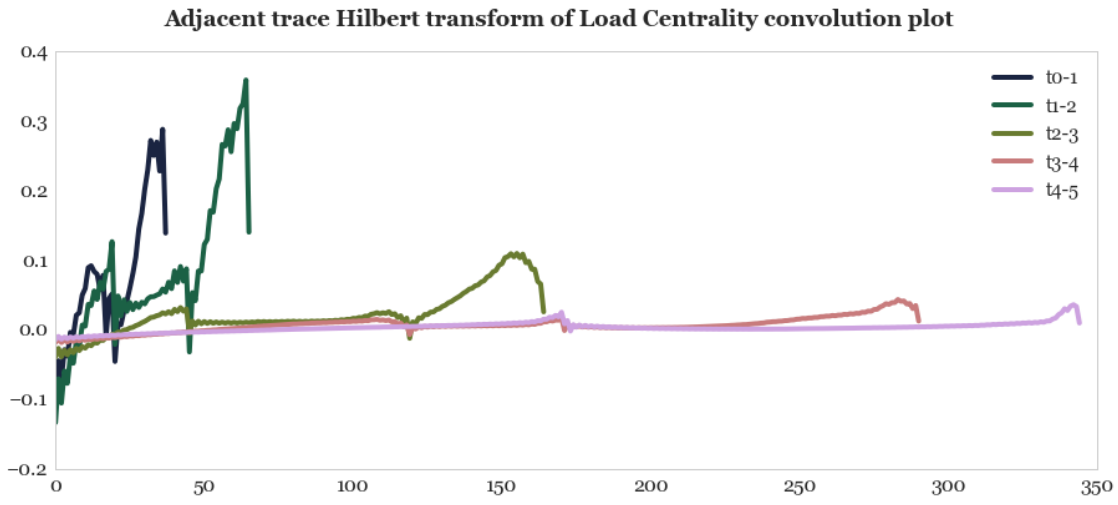
In [82]:
```
plt.suptitle('Adjacent trace Fourier Transform of Load Centrality convolut

        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 0x22744d48c88>
```



```
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 0x22745698ef0>
```

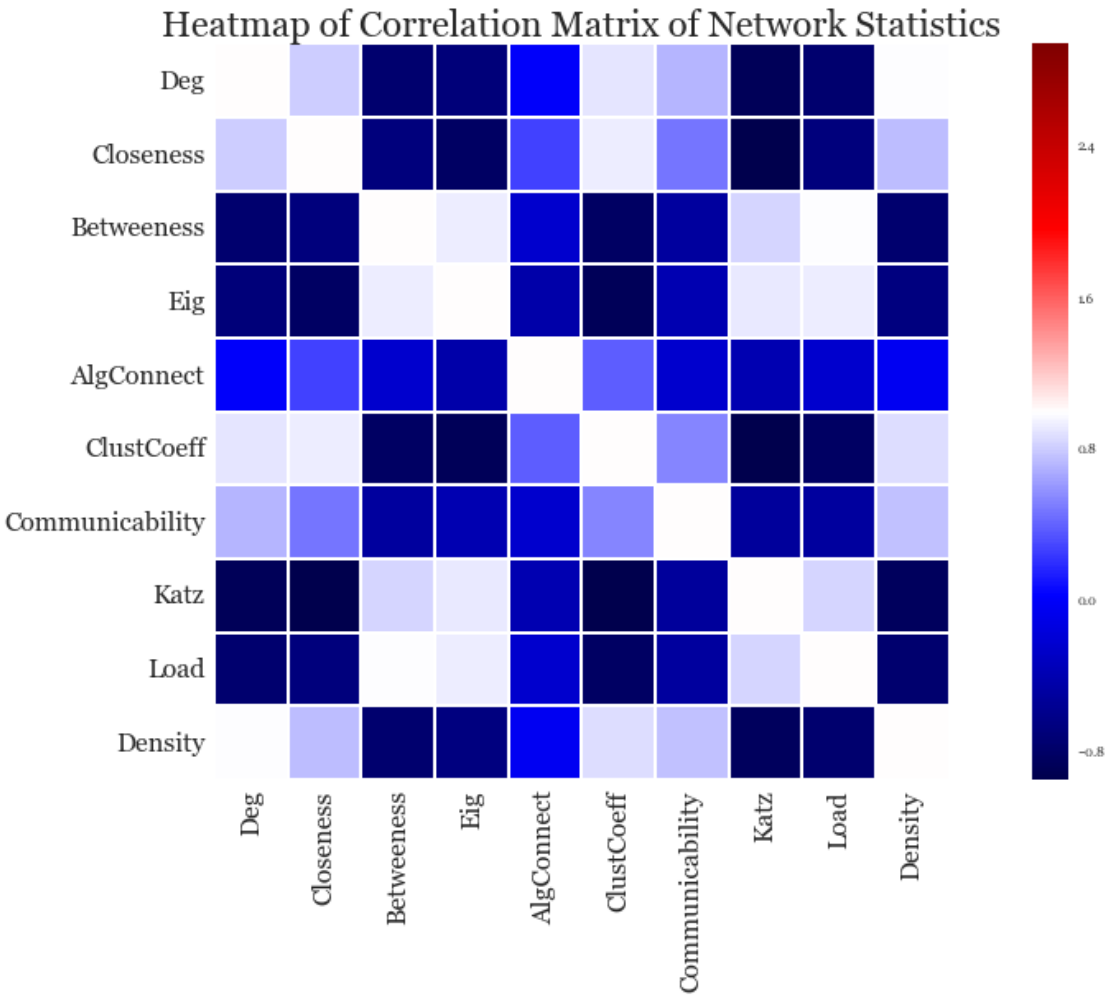


6.2 Centrality Analysis with averaging

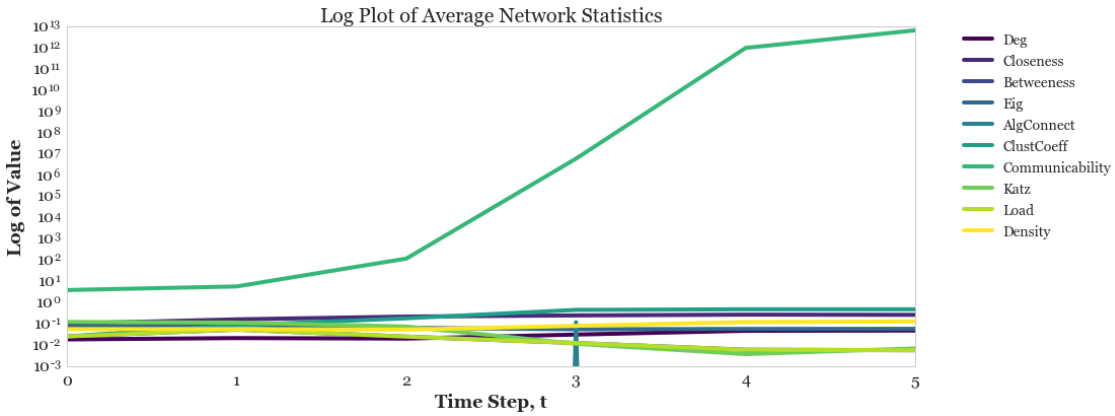
6.2.1 Calculate Centrality Statistics at different time steps

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

In [86]: #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 [87]: 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 [88]: stat_df.columns.values[0]='Deg'

In [89]: stat_df.head()

Out[89]:        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 [90]: plt.figure(figsize=(18,8))
         sns.heatmap(stat_df.corr(), cmap='seismic', center=True, robust=True, fmt=
         plt.title('Heatmap of Correlation Matrix of Network Statistics', fontsize=
         plt.xticks(fontsize=16)
         plt.yticks(fontsize=16)

Out[90]: (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 [91]: 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[91]: <matplotlib.text.Text at 0x227456b4630>
```



7 Assortativity Analysis

```
In [92]: 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 [93]: dac0 = nx.degree_assortativity_coefficient(Gt0)

In [94]: nx.degree_pearson_correlation_coefficient(Gt0)

Out[94]: -0.25126403290770427
```

7.1 Calculate Assortativity statistics for each time step

```
In [95]: 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 [96]: avg_cent(avg_neigdeg0)

Out[96]: 2.190131186642814

In [97]: assor_df = pd.DataFrame([dac0,dac1,dac2,dac3,dac4,dac5], columns=['DegAsso
         assor_df['DegPearCC'] = pd.DataFrame([dpc0,dpc1,dpc2,dpc3,dpc4,dpc5])
```

```
assor_df['Avg(AvgNeighDeg)'] = pd.DataFrame([avg_cent(avg_neigdeg0),avg_ce
                                             avg_cent(avg_neigdeg2),avg_ce
                                             avg_cent(avg_neigdeg4), avg_c

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

assor_df['AvgTriangles'] = pd.DataFrame([avg_cent(triangles0),avg_cent(tri
                                             avg_cent(triangles2),avg_cent
                                             avg_cent(triangles4), avg_cen
assor_df
```

Out[97]:

	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 [98]: stat_df2 = assor_df.join(stat_df)

In [99]: stat_df2.head()
```

Out[99]:

	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	

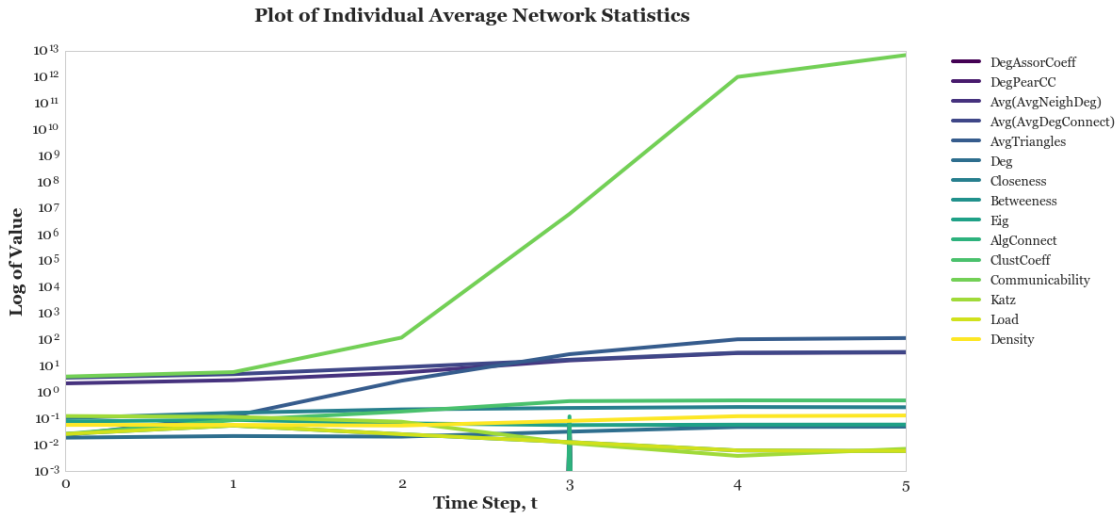
	AvgTriangles	Deg	Closeness	Betweeness	Eig	AlgConnect	\
0	0.023256	0.018826	0.104661	0.025984	0.077739	0.000000	
1	0.125000	0.021720	0.165501	0.054367	0.089506	0.000000	
2	2.762500	0.020570	0.223937	0.025495	0.066241	0.000000	
3	28.272727	0.031813	0.253585	0.012739	0.056478	0.121915	
4	104.180328	0.047889	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 [100]:
```
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[100]: <matplotlib.text.Text at 0x2274565bbe0>



In [101]:
```
stat_df2.describe()
```

Out[101]:

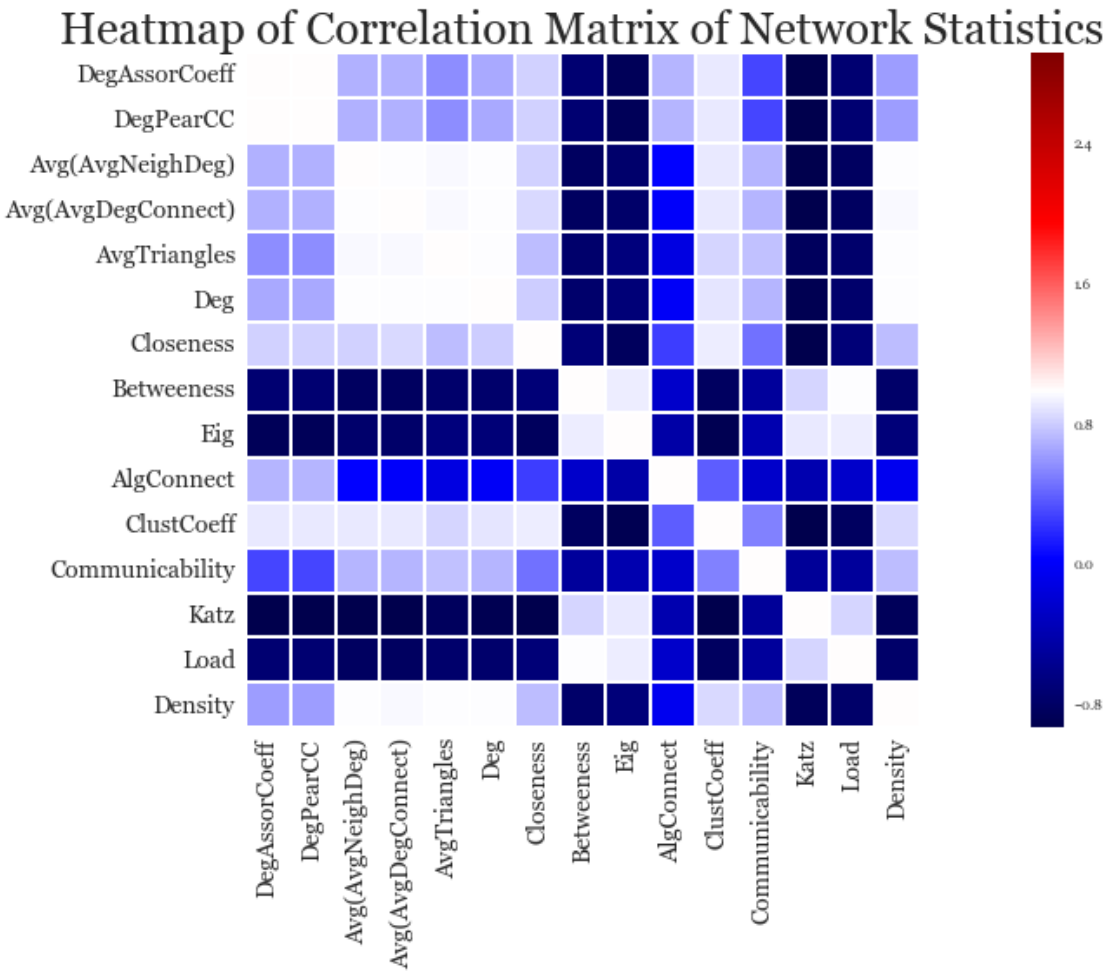
	DegAssorCoeff	DegPearCC	Avg(AvgNeighDeg)	Avg(AvgDegConnect)	\
count	6.000000	6.000000	6.000000	6.000000	
mean	-0.111398	-0.111398	15.169194	16.922168	
std	0.127366	0.127366	14.012924	13.581757	
min	-0.251264	-0.251264	2.190131	3.600092	
25%	-0.214212	-0.214212	3.563511	5.938969	
50%	-0.112558	-0.112558	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 [102]:
```
plt.figure(figsize=(18,8))
sns.heatmap(stat_df2.corr(), cmap='seismic', center=True, robust=True, fm
plt.title('Heatmap of Correlation Matrix of Network Statistics', fontsize
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.tight_layout()
```



8 Graph Spectra

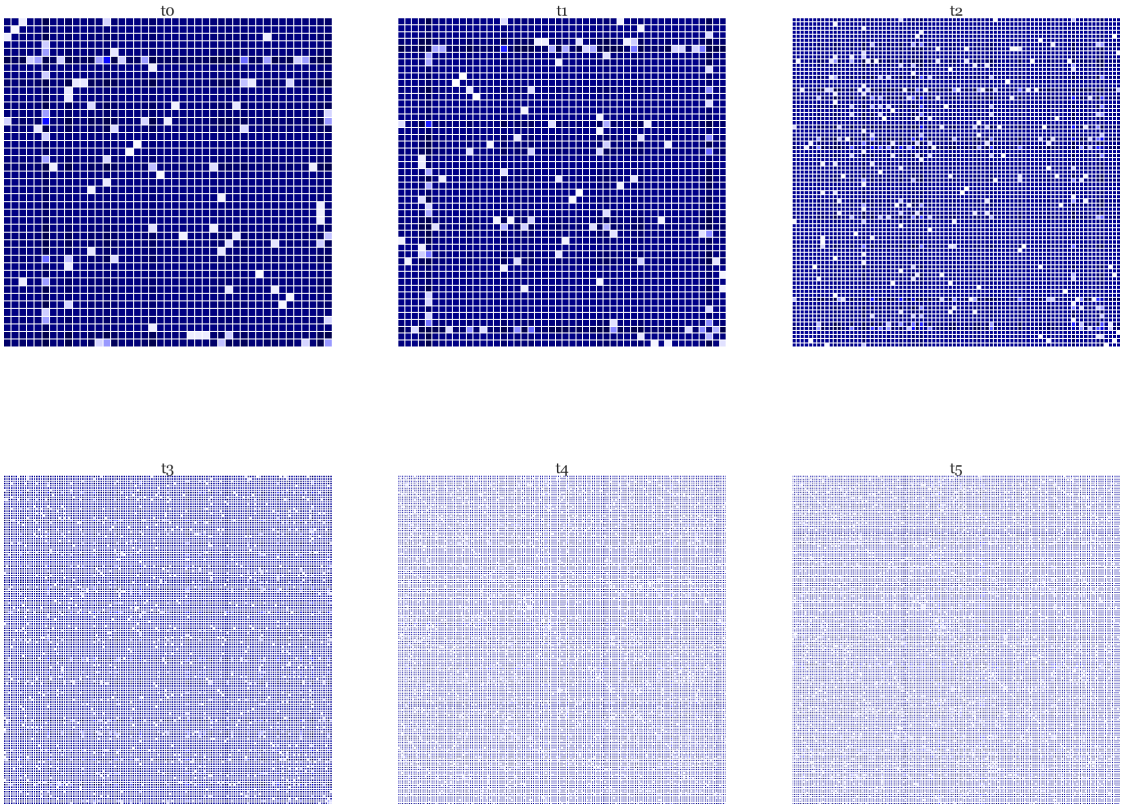
8.1 Modularity Matrix

```
In [103]: 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 [104]: plt.figure(figsize=(40, 30))

          plt.subplot(231)
          sns.heatmap(modM0,cmap='seismic', center=True, robust=True, fmt='d', line
                      yticklabels=False, xticklabels=False, cbar=False)
```

```
plt.suptitle('Heatmap of Modularity Matrix, t0-t5', fontsize=60)
plt.title('t0', fontsize=30)

plt.subplot(232)
sns.heatmap(modM1,cmap='seismic', center=True, robust=True, fmt='d', line
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t1', fontsize=30)

plt.subplot(233)
sns.heatmap(modM2,cmap='seismic', center=True, robust=True, fmt='d', line
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t2', fontsize=30)

plt.subplot(234)
sns.heatmap(modM3,cmap='seismic', center=True, robust=True, fmt='d', line
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t3', fontsize=30)

plt.subplot(235)
sns.heatmap(modM4,cmap='seismic', center=True, robust=True, fmt='d', line
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t4', fontsize=30)

plt.subplot(236)
sns.heatmap(modM5,cmap='seismic', center=True, robust=True, fmt='d', line
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t5', fontsize=30)
```

```
Out[104]: <matplotlib.text.Text at 0x22746ef63c8>
```

Heatmap of Modularity Matrix, to-t5



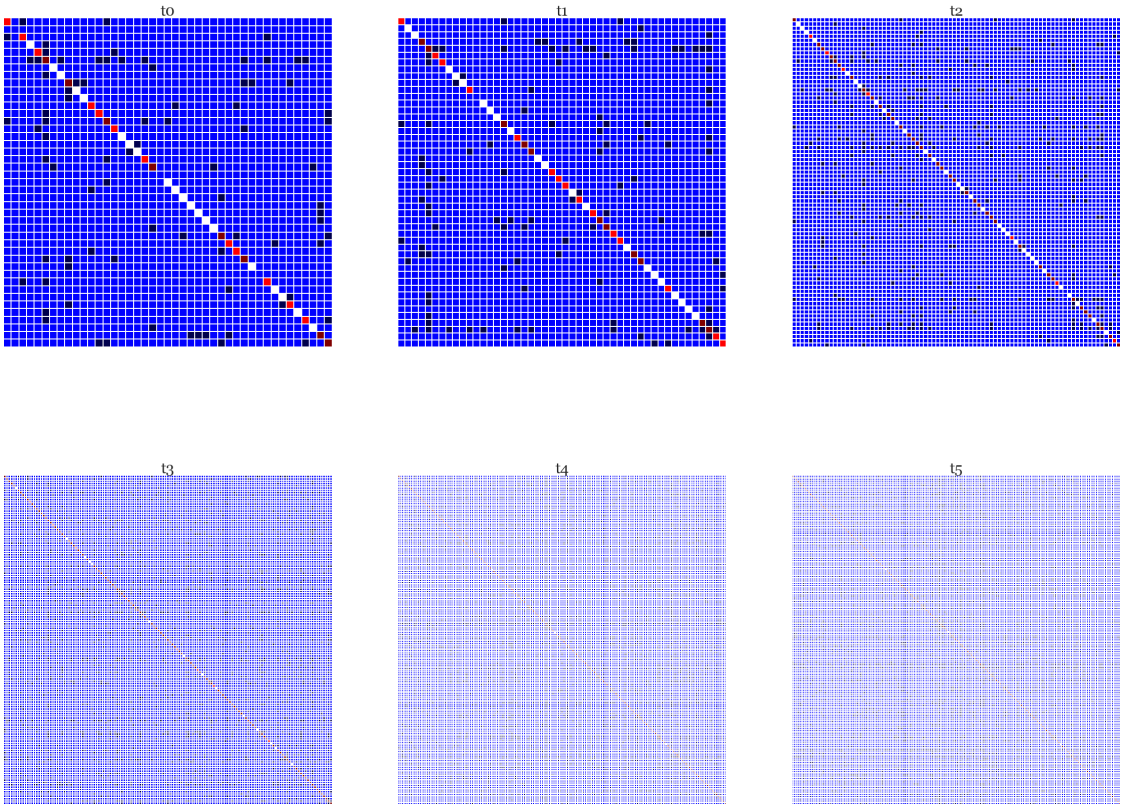
8.2 Laplacian Matrix

```
In [105]: 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 [106]: plt.figure(figsize=(40, 30))

          plt.subplot(231)
          sns.heatmap(lapM0,cmap='seismic', center=True, robust=True, fmt='d', line
                      yticklabels=False, xticklabels=False, cbar=False)

          plt.suptitle('Heatmap of Laplacian Matrix, t0-t5', fontsize=60)
          plt.title('t0', fontsize=30)

          plt.subplot(232)
```

```
            sns.heatmap(lapM1,cmap='seismic', center=True, robust=True, fmt='d', line
                        yticklabels=False, xticklabels=False, cbar=False)
            plt.title('t1', fontsize=30)

            plt.subplot(233)
            sns.heatmap(lapM2,cmap='seismic', center=True, robust=True, fmt='d', line
                        yticklabels=False, xticklabels=False, cbar=False)
            plt.title('t2', fontsize=30)

            plt.subplot(234)
            sns.heatmap(lapM3,cmap='seismic', center=True, robust=True, fmt='d', line
                        yticklabels=False, xticklabels=False, cbar=False)
            plt.title('t3', fontsize=30)

            plt.subplot(235)
            sns.heatmap(lapM4,cmap='seismic', center=True, robust=True, fmt='d', line
                        yticklabels=False, xticklabels=False, cbar=False)
            plt.title('t4', fontsize=30)

            plt.subplot(236)
            sns.heatmap(lapM5,cmap='seismic', center=True, robust=True, fmt='d', line
                        yticklabels=False, xticklabels=False, cbar=False)
            plt.title('t5', fontsize=30)
```

```
Out[106]: <matplotlib.text.Text at 0x227473b7630>
```

Heatmap of Laplacian Matrix, t0-t5



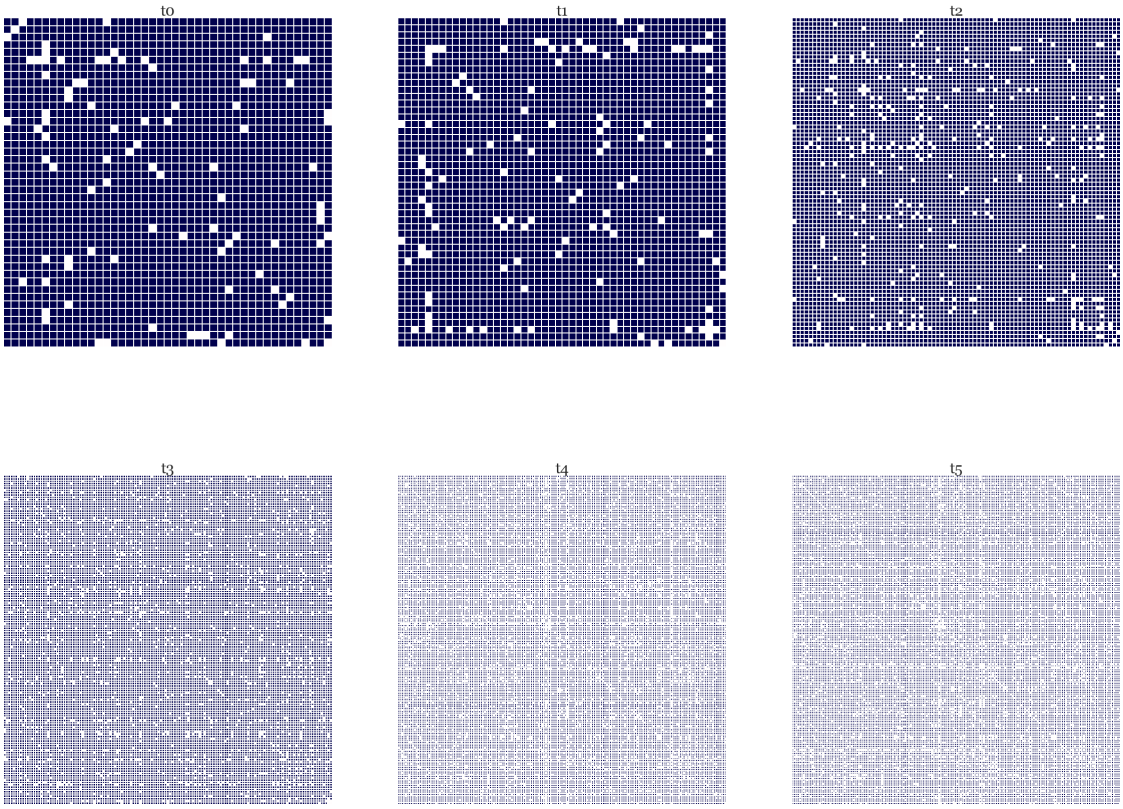
8.3 Adjacency Matrix

```
In [107]: 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 [108]: plt.figure(figsize=(40, 30))

          plt.subplot(231)
          sns.heatmap(adjM0,cmap='seismic', center=True, robust=True, fmt='d', line
                      yticklabels=False, xticklabels=False, cbar=False)

          plt.suptitle('Heatmap of Adjacency Matrix, t0-t5', fontsize=60)
          plt.title('t0', fontsize=30)

          plt.subplot(232)
```

```
            sns.heatmap(adjM1,cmap='seismic', center=True, robust=True, fmt='d', line
                        yticklabels=False, xticklabels=False, cbar=False)
            plt.title('t1', fontsize=30)

            plt.subplot(233)
            sns.heatmap(adjM2,cmap='seismic', center=True, robust=True, fmt='d', line
                        yticklabels=False, xticklabels=False, cbar=False)
            plt.title('t2', fontsize=30)

            plt.subplot(234)
            sns.heatmap(adjM3,cmap='seismic', center=True, robust=True, fmt='d', line
                        yticklabels=False, xticklabels=False, cbar=False)
            plt.title('t3', fontsize=30)

            plt.subplot(235)
            sns.heatmap(adjM4,cmap='seismic', center=True, robust=True, fmt='d', line
                        yticklabels=False, xticklabels=False, cbar=False)
            plt.title('t4', fontsize=30)

            plt.subplot(236)
            sns.heatmap(adjM5,cmap='seismic', center=True, robust=True, fmt='d', line
                        yticklabels=False, xticklabels=False, cbar=False)
            plt.title('t5', fontsize=30)
```

```
Out[108]: <matplotlib.text.Text at 0x2274acb9d30>
```

Heatmap of Adjacency Matrix, to-t5



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

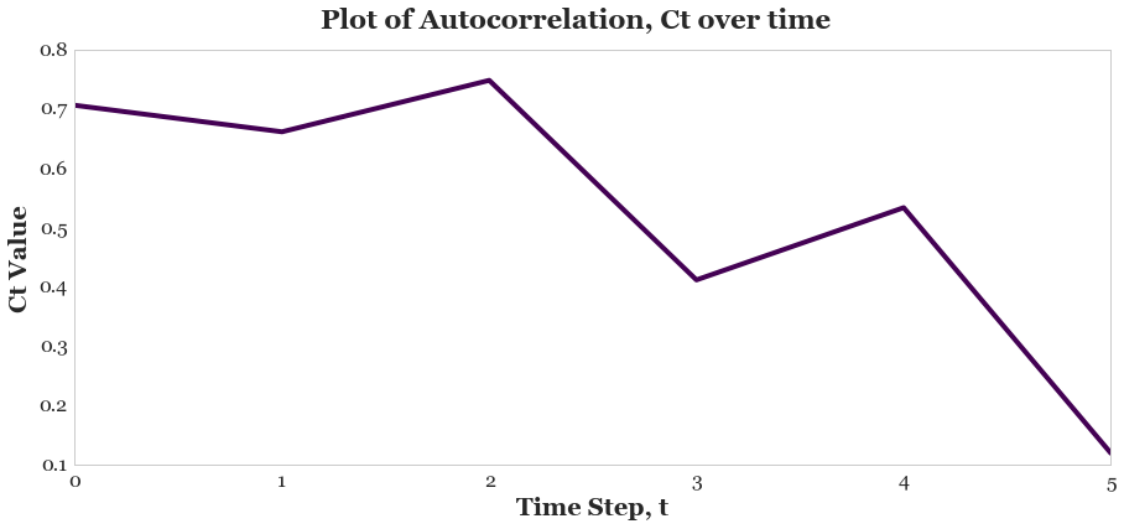
$$\zeta = \frac{\sum_{t=0}^{tmax-1} C(t, t+1)}{tmax - t0 - 1}$$

```
In [109]: def pad_shape(x,ref,offset=0):
              result = np.zeros_like(ref)
              result[0:x.shape[0]+0,0:x.shape[1]+0] = x
```

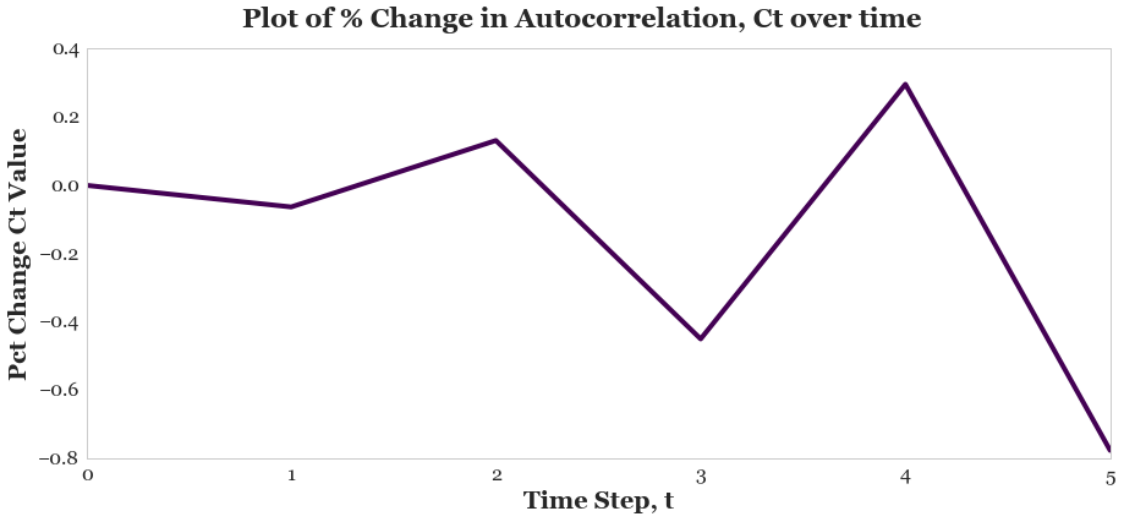
```
        return [result, nx.Graph(result)]
```

```
In [110]: #_,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[110]: 0.70710678118654757

In [111]: _,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[111]: 0.66232865352709824

In [112]: _,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[112]: 0.7492134240101288

In [113]: _,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
```

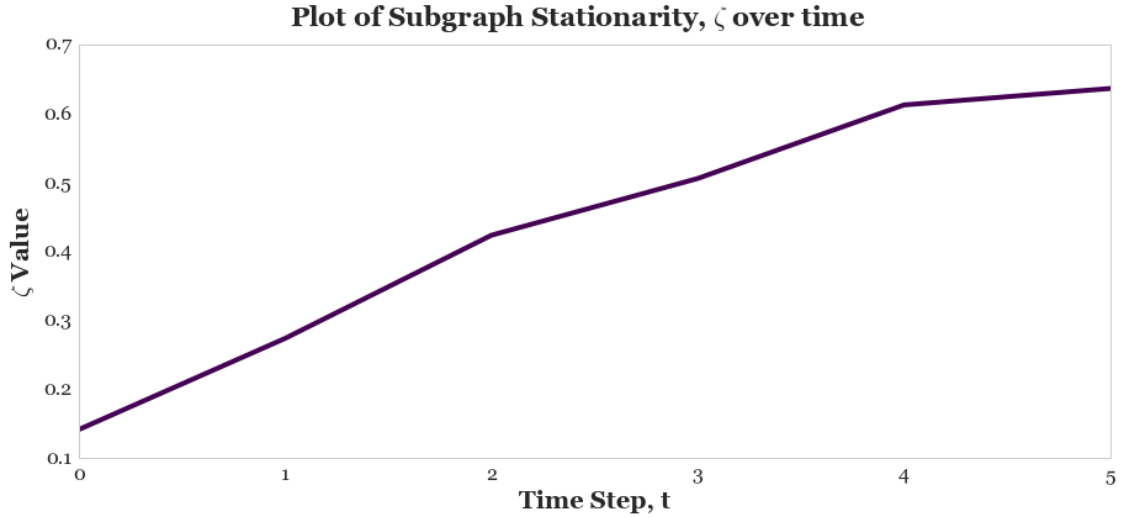
```
Out[113]: 0.41226013627128677

In [114]: _,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[114]: 0.53425969593338996

In [115]: _,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[115]: 0.11965940514762706

In [116]: 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[116]: <matplotlib.text.Text at 0x22746f4e8d0>
```



```
In [117]: 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[117]: <matplotlib.text.Text at 0x2274b0ec9e8>
```




```
In [118]: 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("$\zeta$ Value", fontsize=20)

Out[118]: <matplotlib.text.Text at 0x2274b5203c8>
```



10 Graph Stationarity Ratio

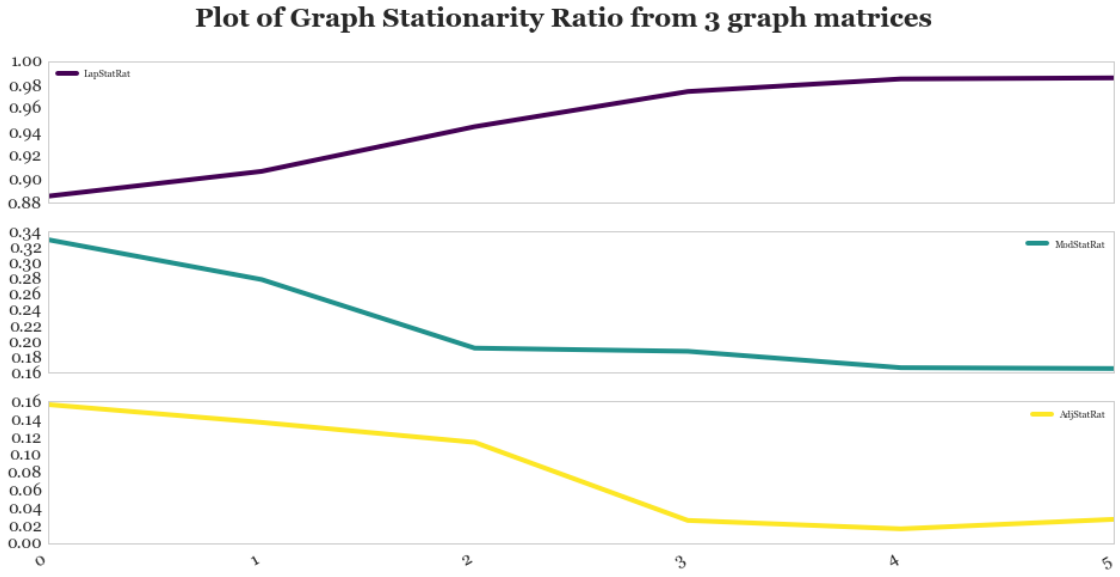
The stationarity ratio as used here is inspired by the Matlab Code based on the arXiv:1601.02522

```
In [119]: 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 [120]: 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 [121]: 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 [122]: stationarity_df.columns.values[0]='LapStatRat'
          stationarity_df

Out[122]:    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
```

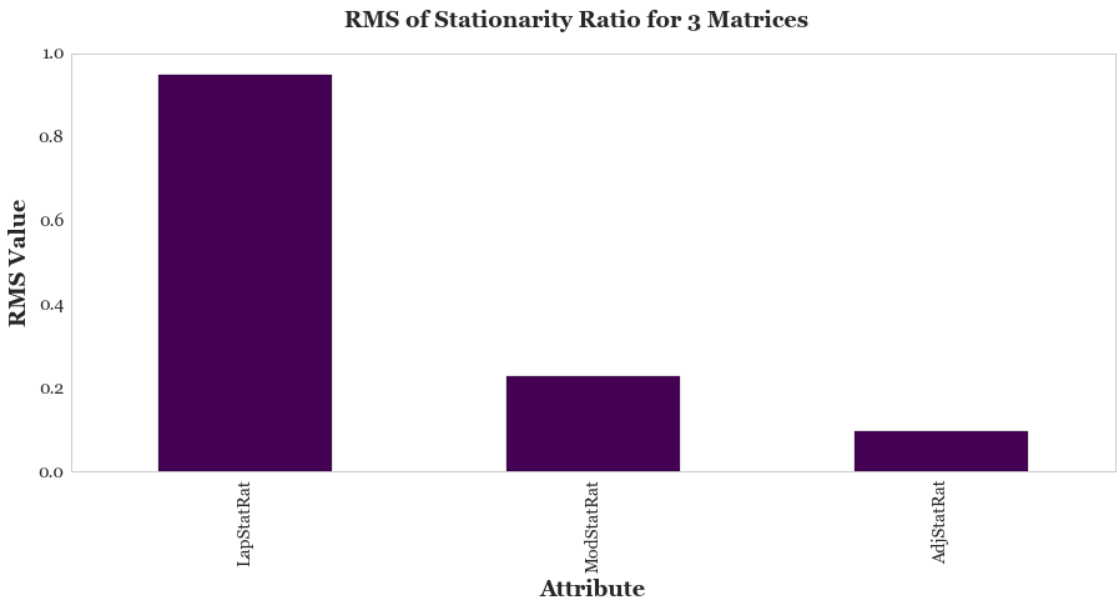
```
In [123]: stationarity_df.plot(subplots=True, fontsize=14, legend=True, sharex=True
          plt.suptitle('Plot of Graph Stationarity Ratio from 3 graph matrices', fo
```

```
Out[123]: <matplotlib.text.Text at 0x2274b522c18>
```

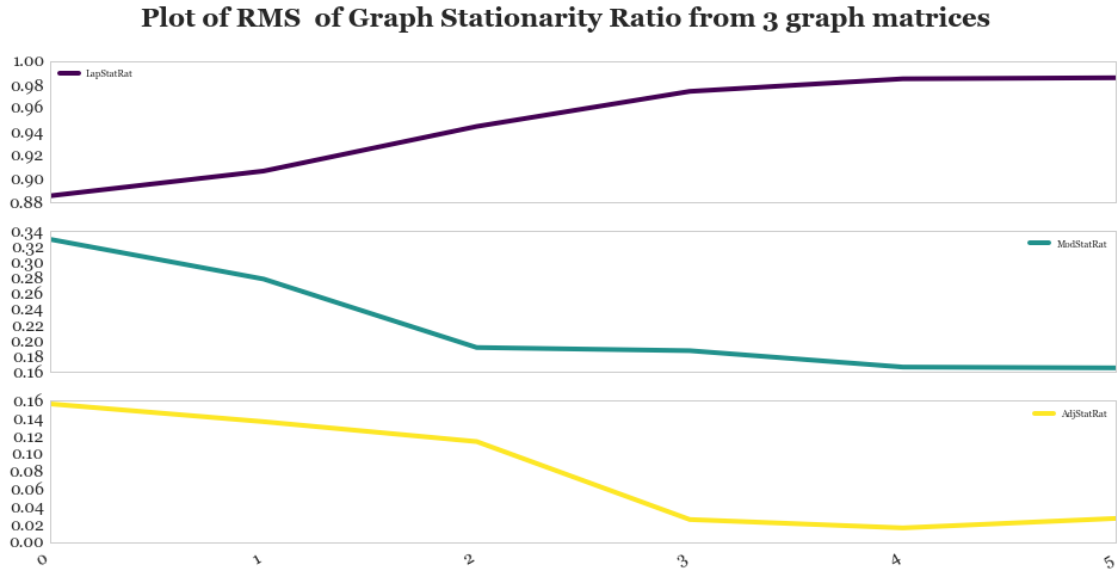



```
In [175]: stationarity_df.apply(rms).plot(kind='bar',fontsize=14,colormap='viridis'
          plt.suptitle('RMS of Stationarity Ratio for 3 Matrices', fontsize=18)
          plt.xlabel("Attribute", fontsize=18)
          plt.ylabel("RMS Value", fontsize=18)
```

```
Out[175]: <matplotlib.text.Text at 0x2274f8d1c50>
```




```
In [125]: 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[125]: <matplotlib.text.Text at 0x2274735b438>
```



11 NRMS of Stationarity

```
In [126]: stationarity_df
```

Out[126]:

	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 [127]:
```
nrms1 =[]
nrms2 =[]
nrms3 =[]
for i in range(0,5):
    x = int(i)
    y = x +1
    nrms1.append(nrms(stationarity_df.values[:,0][x],stationarity_df.valu
    nrms2.append(nrms(stationarity_df.values[:,1][x],stationarity_df.valu
    nrms3.append(nrms(stationarity_df.values[:,2][x],stationarity_df.valu
```

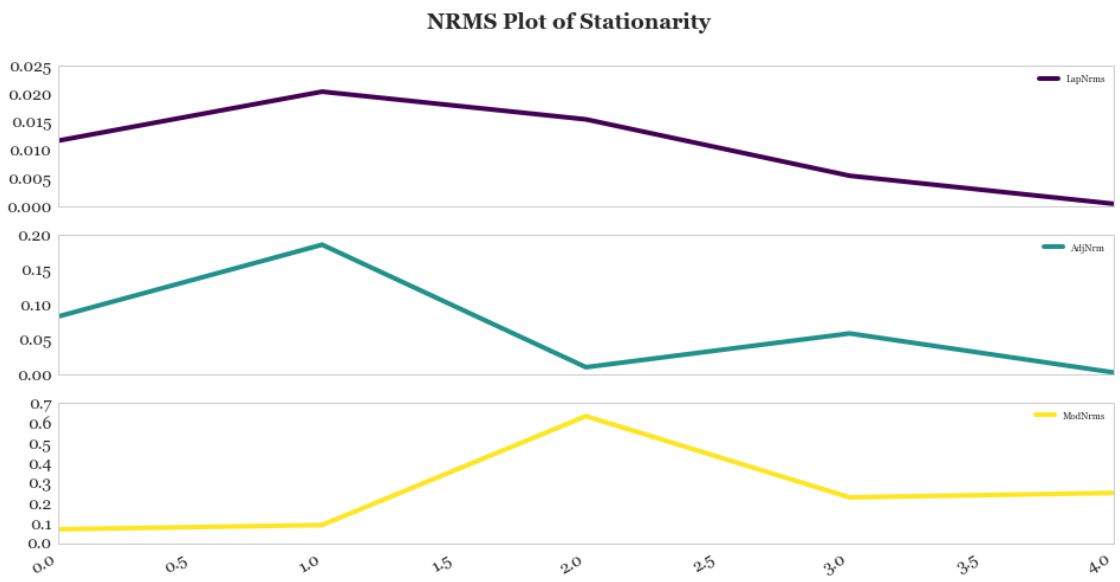
In [128]:
```
nrms1
```

Out[128]:
```
[0.011698930524145468,
 0.020427512324573554,
 0.015489170662464176,
 0.0054406173044872618,
 0.00044971666488704827]
```

In [129]:
```
Nrms_df = pd.DataFrame([nrms1,nrms2,nrms3]).T
col = ('LapNrms','AdjNrm','ModNrms')
Nrms_df.columns=col
```

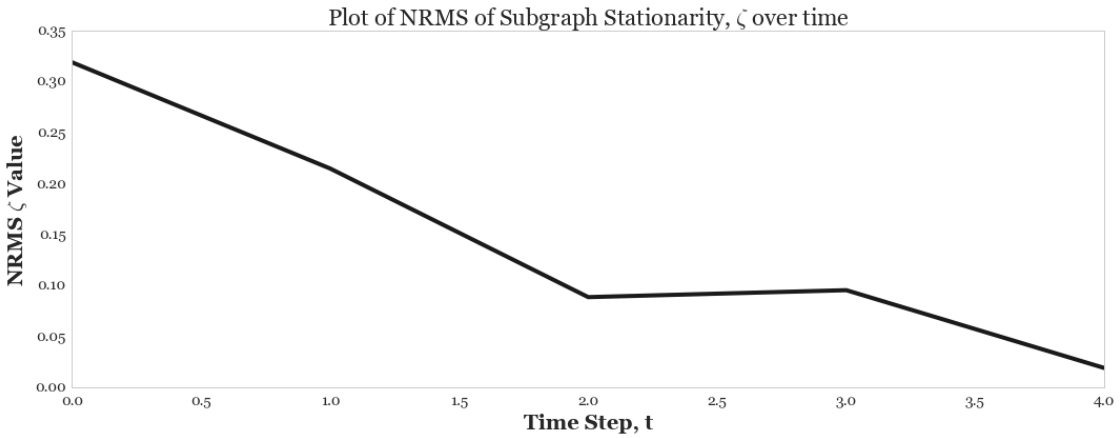
In [130]:
```
Nrms_df.plot(subplots=True,fontsize=14, legend=True, sharex=True, figsize
plt.suptitle('NRMS Plot of Stationarity', fontsize=18)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
#plt.legend(fontsize=14, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=(
```

Out[130]:
```
(array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8]),
 <a list of 9 Text yticklabel objects>)
```




```
In [131]: nrms_zeta =[]
          for i in range(0,5):
              x = int(i)
              y = x +1
              nrms_zeta.append(nrms(zeta.values[:,0][x],zeta.values[:,0][y]))
```

```
In [132]: plt.plot(nrms_zeta,'k')
          plt.title('Plot of NRMS of Subgraph Stationarity, $\zeta$ over time', for
          plt.xticks(fontsize=14)
          plt.yticks(fontsize=14)
          plt.xlabel("Time Step, t", fontsize=20)
          plt.ylabel("NRMS $\zeta$ Value", fontsize=20)
          plt.tight_layout()
```



12 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.

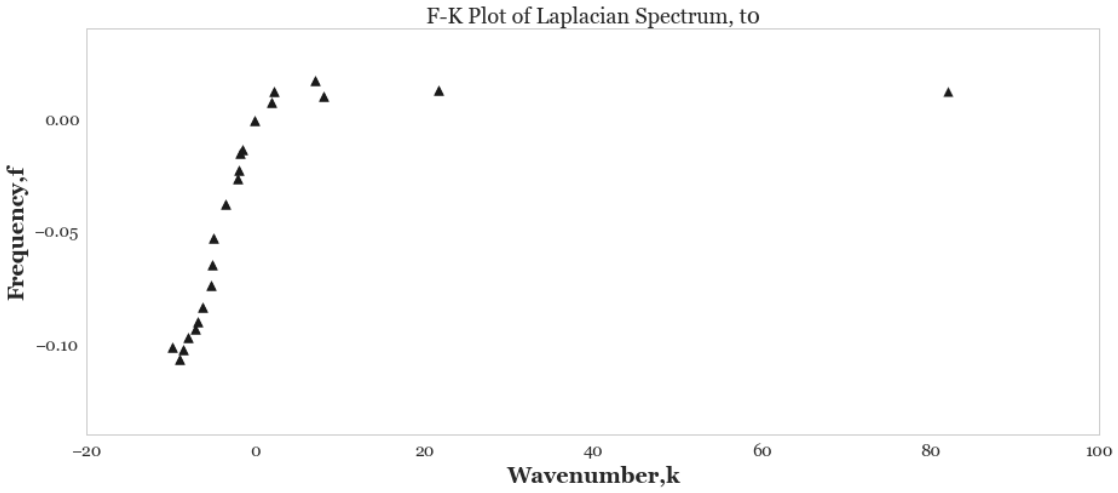
```
In [133]: lap_spec0 = nx.laplacian_spectrum(Gt0)
          mod_spec0 = nx.modularity_spectrum(Gt0)
          adj_spec0 = nx.adjacency_spectrum(Gt0)

          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)
```

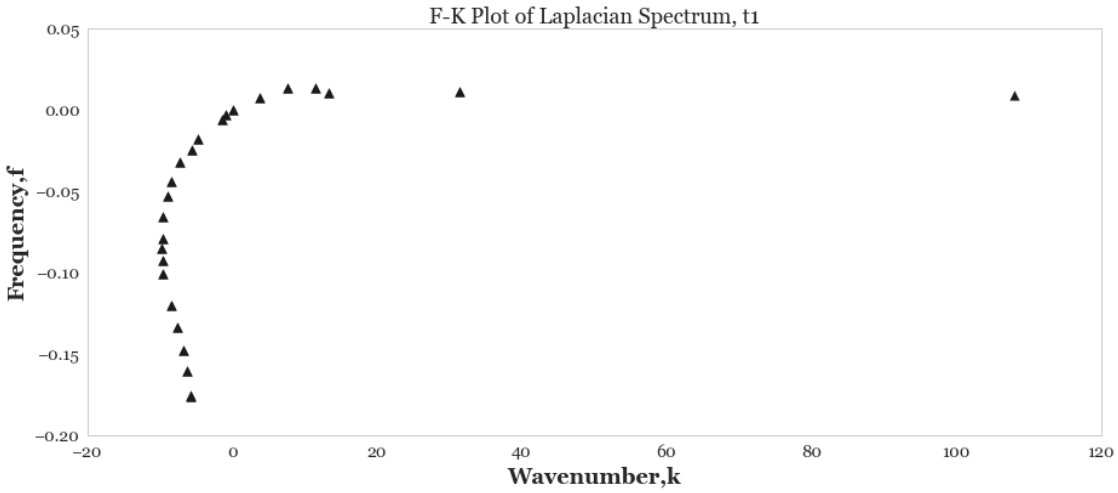
```
In [134]: def fk_plot(f,name):
              freq = sc.fft(f)
              wavnum = 1/freq

              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]
```

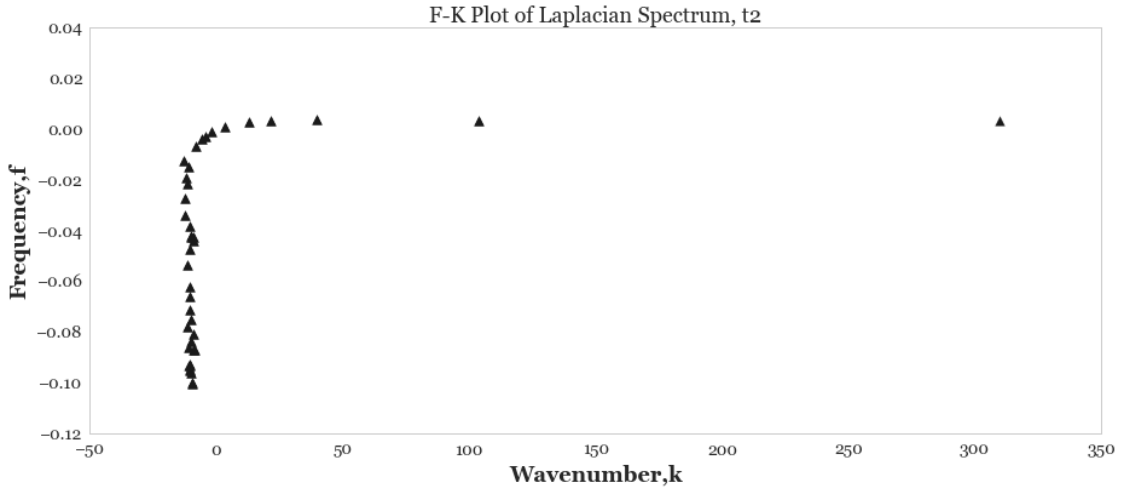
```
In [135]: freq01, k01 = fk_plot(lap_spec0,'Laplacian Spectrum, t0')
          freq1, k1 = fk_plot(lap_spec1,'Laplacian Spectrum, t1')
```

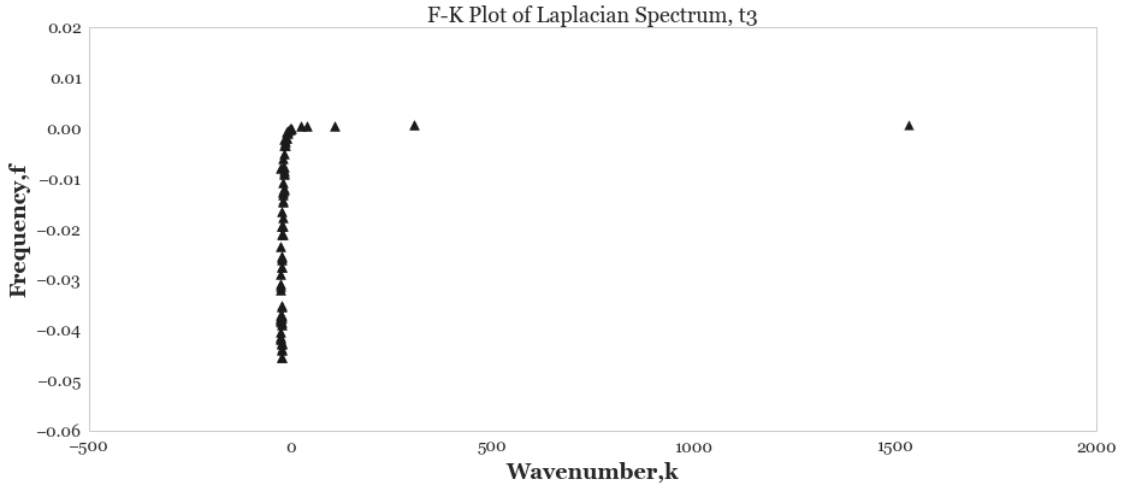
```
freq2, k2 = fk_plot(lap_spec2,'Laplacian Spectrum, t2')
freq3, k3 = fk_plot(lap_spec3,'Laplacian Spectrum, t3')
freq4, k4 = fk_plot(lap_spec4,'Laplacian Spectrum, t4')
freq5, k5 = fk_plot(lap_spec5,'Laplacian Spectrum, t5')
```

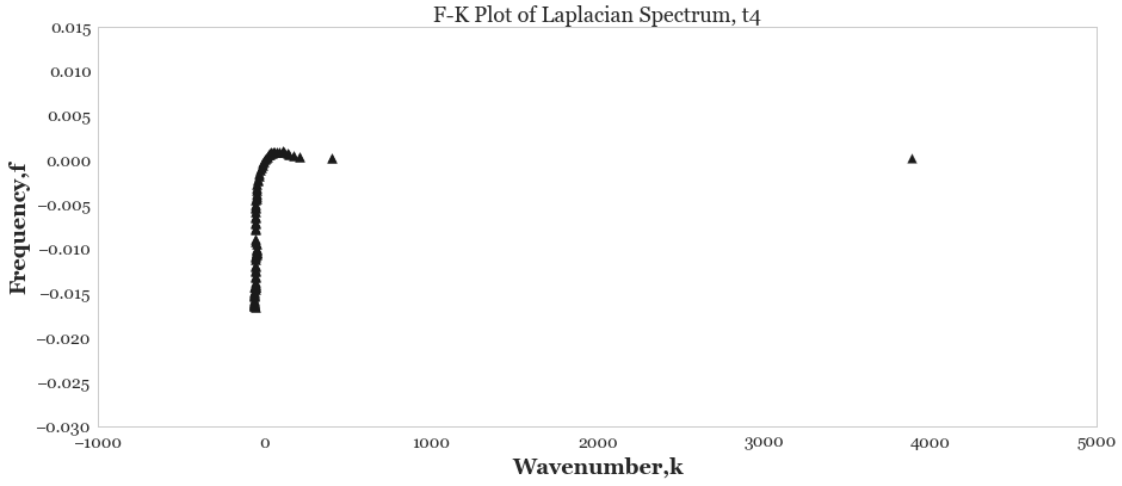
```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\core\numeric.py:533: ComplexWa
  return array(a, dtype, copy=False, order=order, subok=True)
```

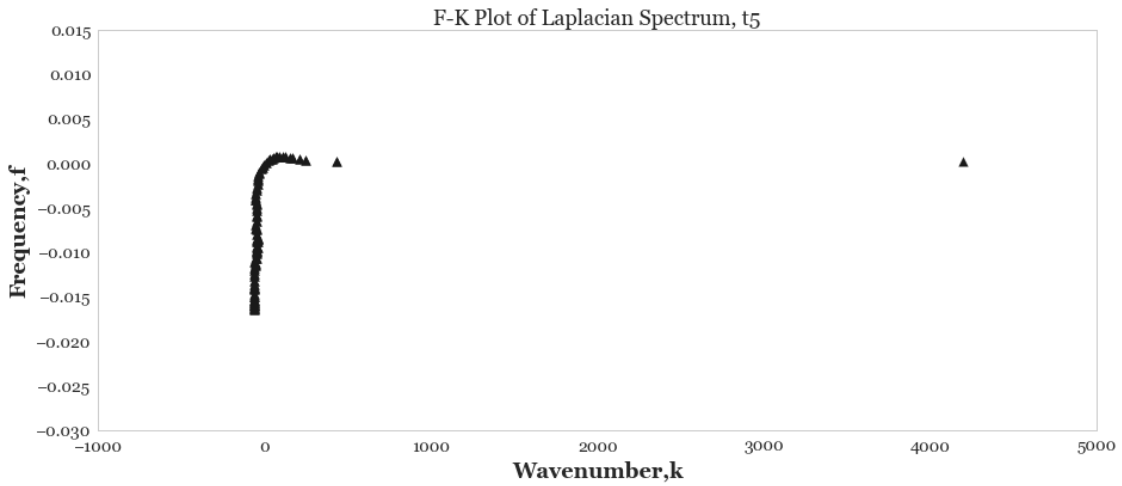




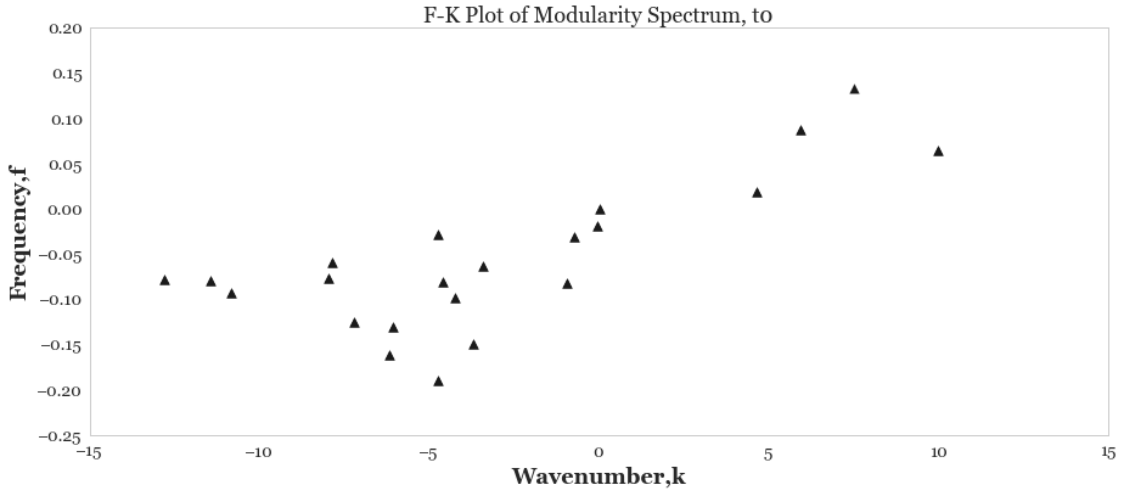

F-K Plot of Laplacian Spectrum, t2
Frequency,f
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.10
−0.12
−50
0
50
100
150
200
250
300
350
Wavenumber,k

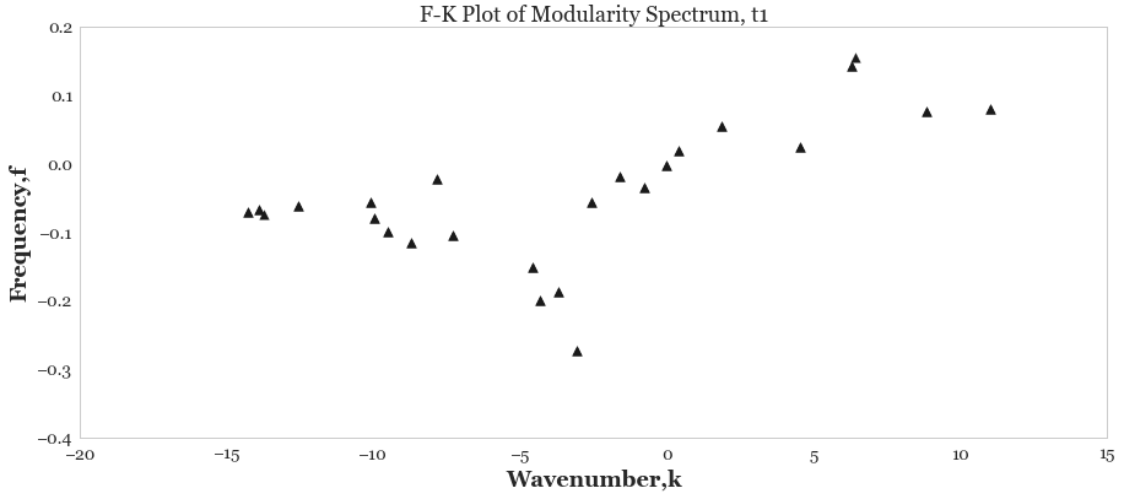

F-K Plot of Laplacian Spectrum, t3
Frequency,f
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
−500
0
500
1000
1500
2000
Wavenumber,k

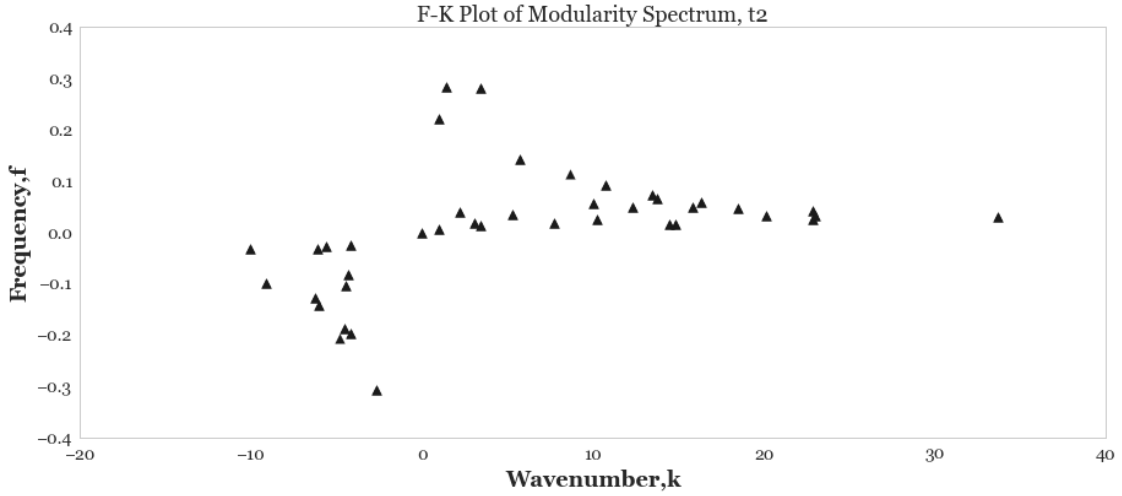

F-K Plot of Laplacian Spectrum, t4
Frequency,f
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
−0.020
−0.025
−0.030
−1000
0
1000
2000
3000
4000
5000
Wavenumber,k

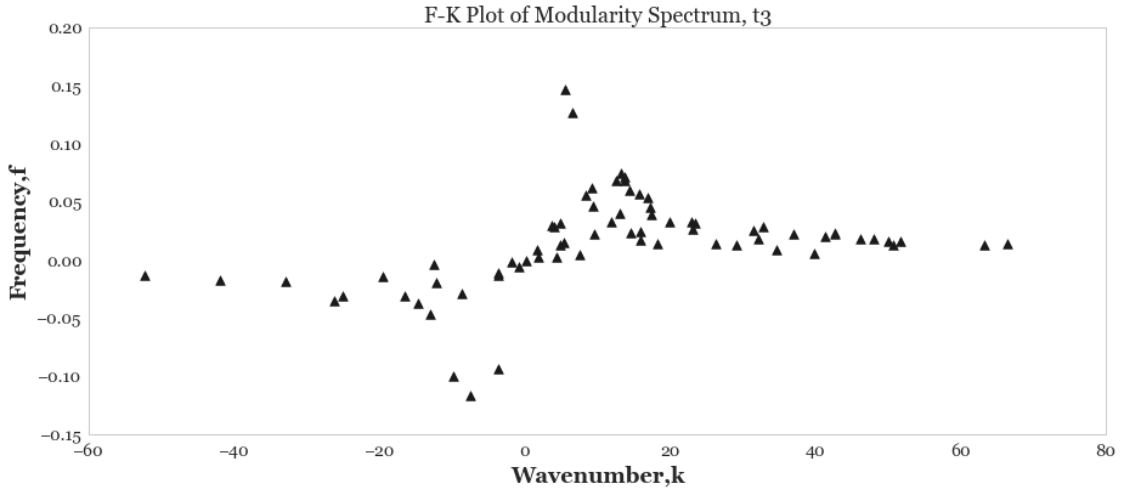


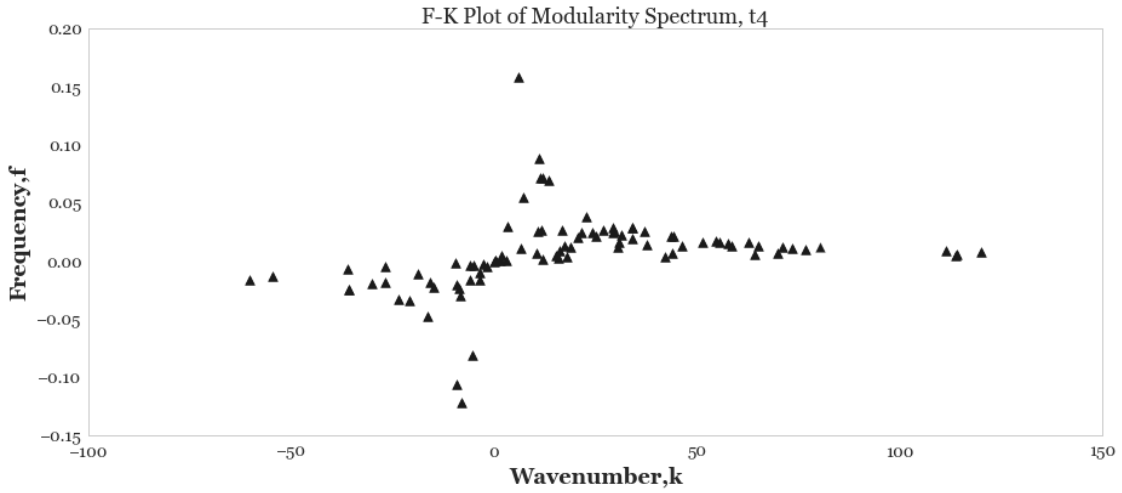

```
In [136]: freq02, k02 = fk_plot(mod_spec0,'Modularity Spectrum, t0')
          freq12, k02 = fk_plot(mod_spec1,'Modularity Spectrum, t1')
          freq22, k02 = fk_plot(mod_spec2,'Modularity Spectrum, t2')
          freq32, k02 = fk_plot(mod_spec3,'Modularity Spectrum, t3')
          freq42, k02 = 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)
```

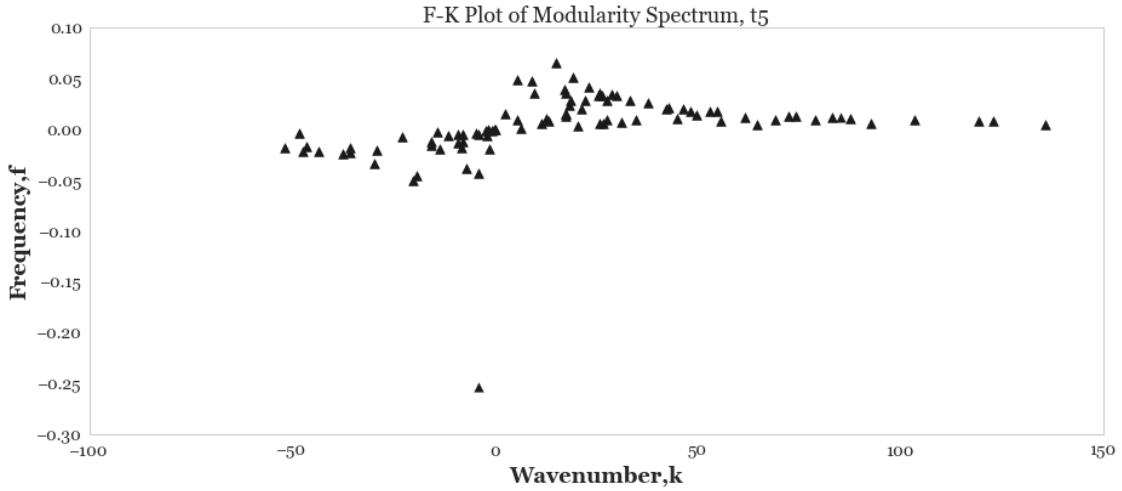



F-K Plot of Modularity Spectrum, t1
Frequency,f
Wavenumber,k
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−20
−15
−10
−5
0
5
10
15

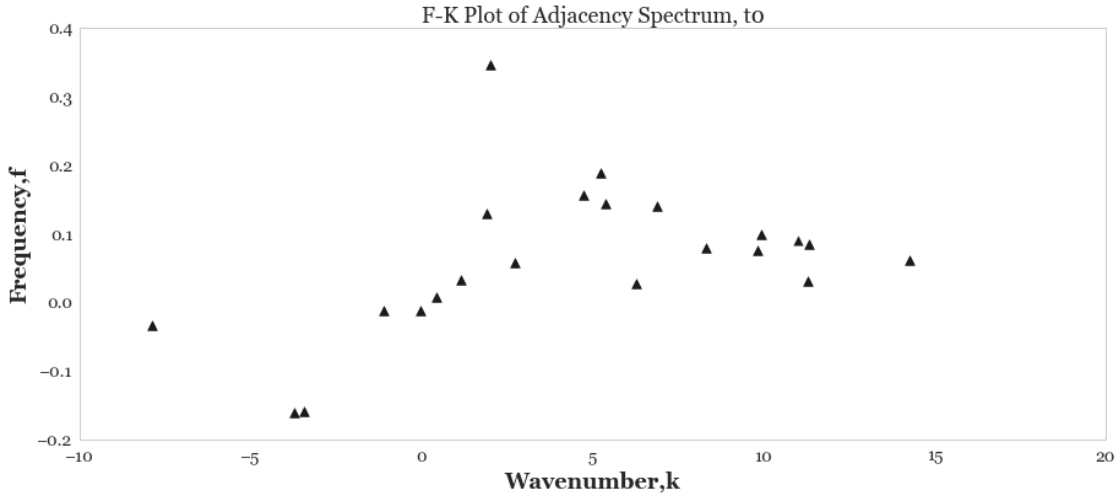

F-K Plot of Modularity Spectrum, t2
Frequency,f
Wavenumber,k
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−20
−10
0
10
20
30
40

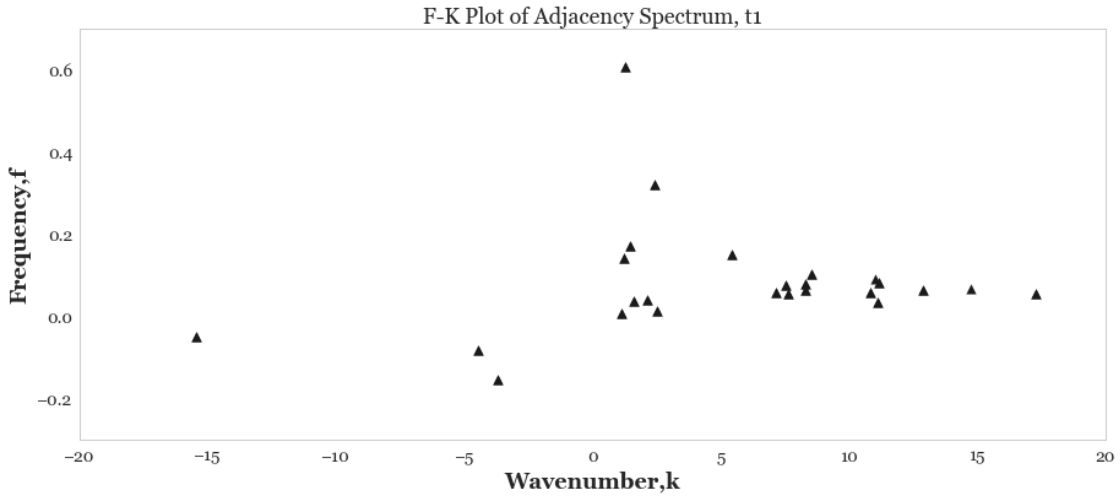

F-K Plot of Modularity Spectrum, t3
Frequency,f
Wavenumber,k
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−60
−40
−20
0
20
40
60
80

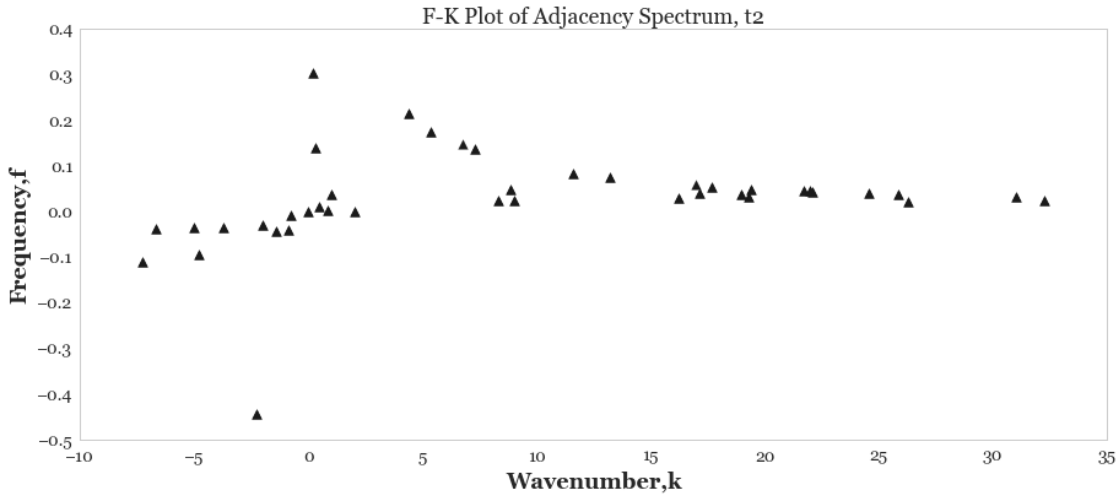


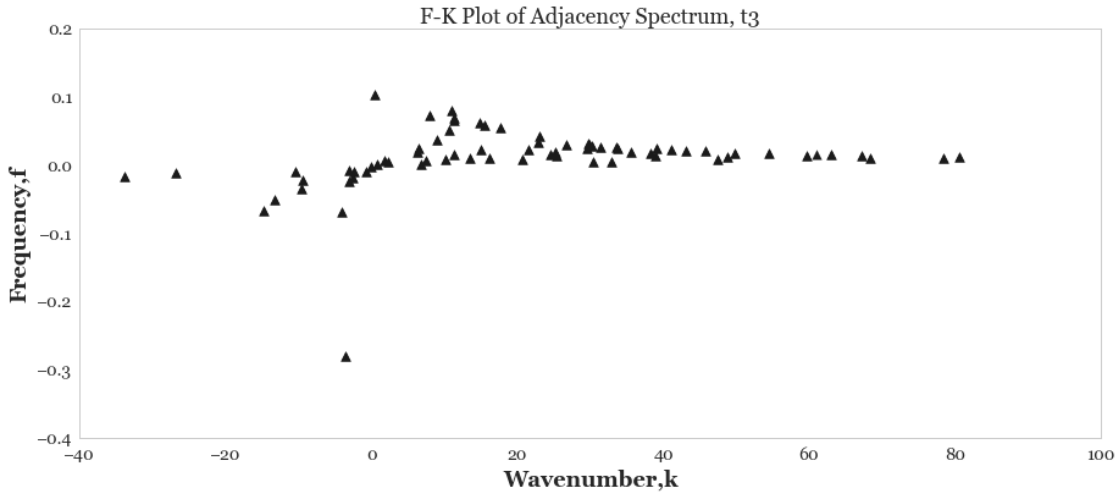


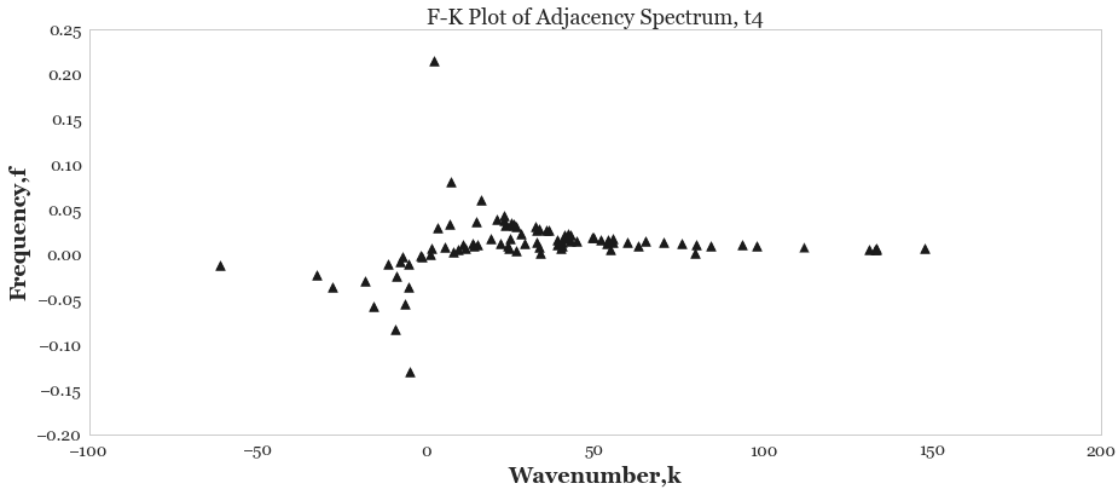

```
In [137]: freq03, k03 = fk_plot(adj_spec0,'Adjacency Spectrum, t0')
          freq13, k03 = fk_plot(adj_spec1,'Adjacency Spectrum, t1')
          freq23, k03 = fk_plot(adj_spec2,'Adjacency Spectrum, t2')
          freq33, k03 = fk_plot(adj_spec3,'Adjacency Spectrum, t3')
          freq43, k03 = fk_plot(adj_spec4,'Adjacency 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)
```

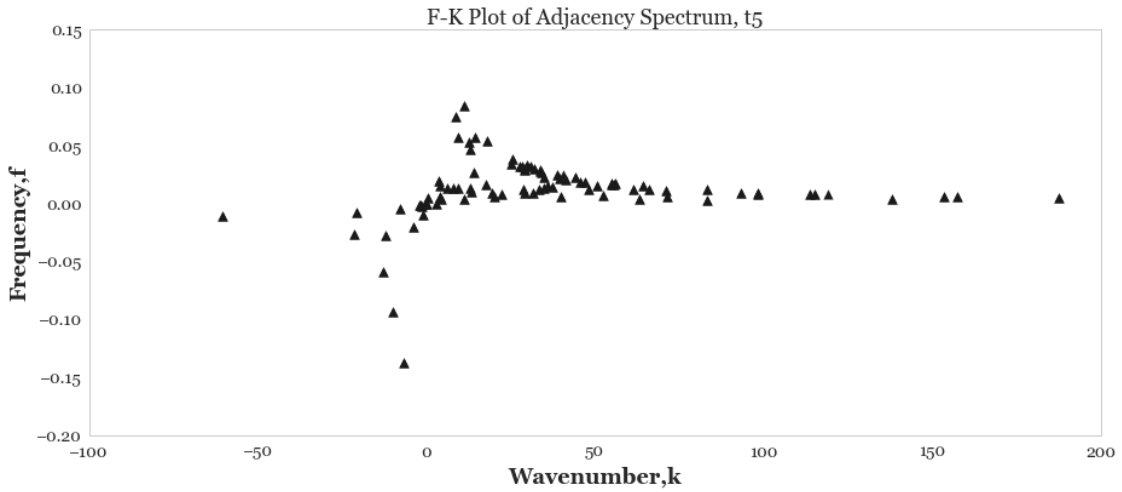

F-K Plot of Adjacency Spectrum, t0
Frequency,f
Wavenumber,k


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
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−40
−20
0
20
40
60
80
100

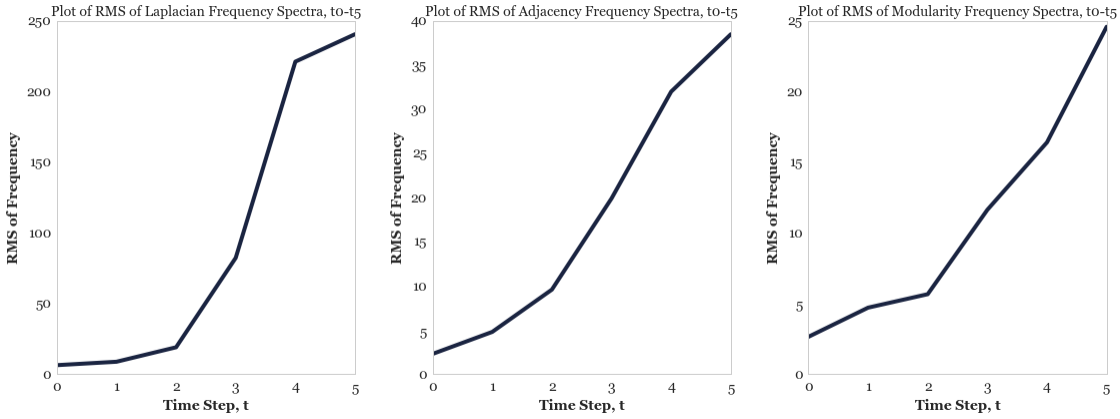

F-K Plot of Adjacency Spectrum, t4
Frequency,f
Wavenumber,k
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−100
−50
0
50
100
150
200




```
In [138]: 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 [139]: 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. "
```



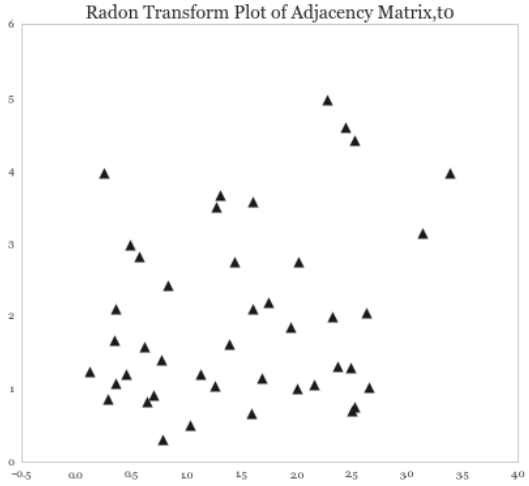
13 Radon Transform Plots

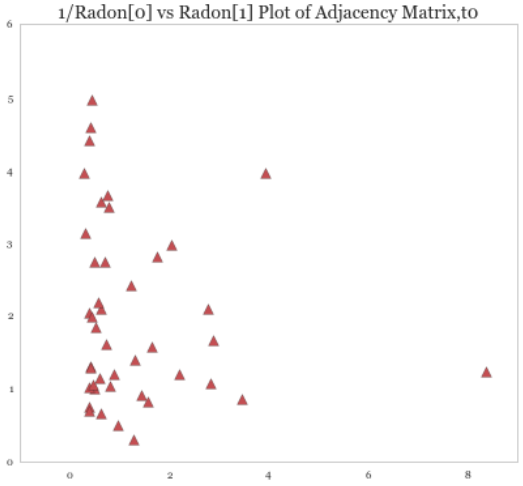
```
In [140]: def plot_radon(m, name):
              from skimage.transform import radon
              theta = np.linspace(0., 180., max(m.shape), endpoint=False)
              sinogram = radon(m, theta=theta, circle=True)

              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()
```

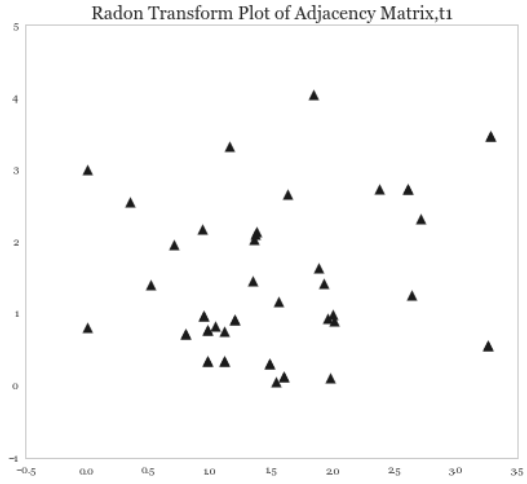
```
In [141]: plot_radon(adjM0,'Adjacency Matrix,t0')
          plot_radon(adjM1,'Adjacency Matrix,t1')
          plot_radon(adjM2,'Adjacency Matrix,t2')
```

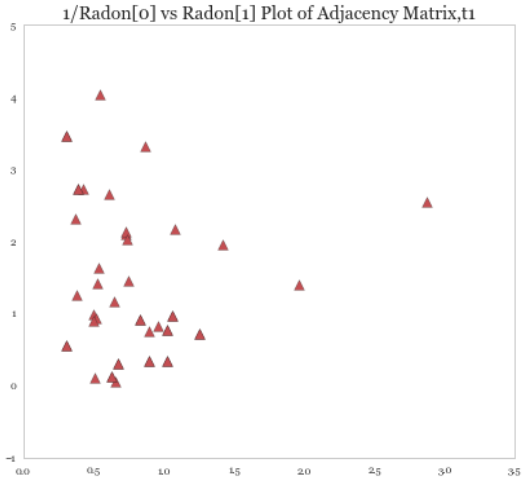
```
plot_radon(adjM3,'Adjacency Matrix,t3')
plot_radon(adjM4,'Adjacency Matrix,t4')
plot_radon(adjM5,'Adjacency Matrix,t5')
```

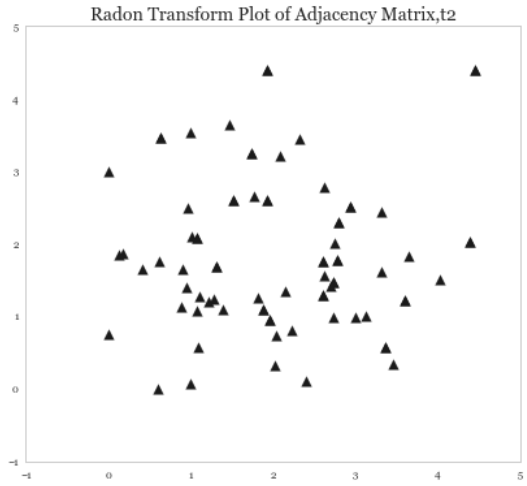
```
C:\Users\arsha_000\Anaconda3\lib\site-packages\skimage\transform\radon_transform.py
  warn('Radon transform: image must be zero outside the '
```

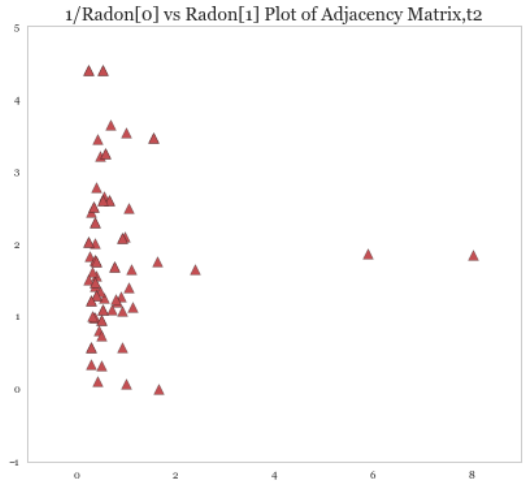


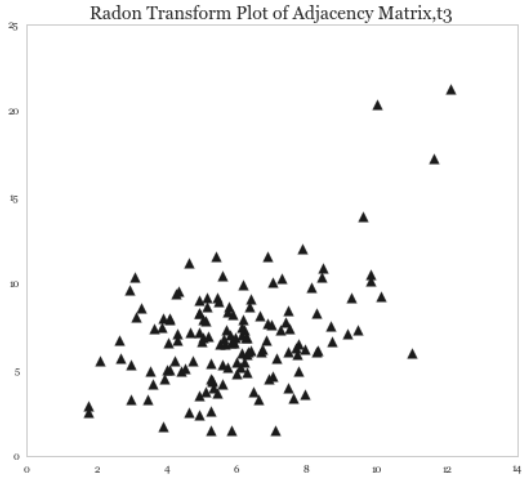


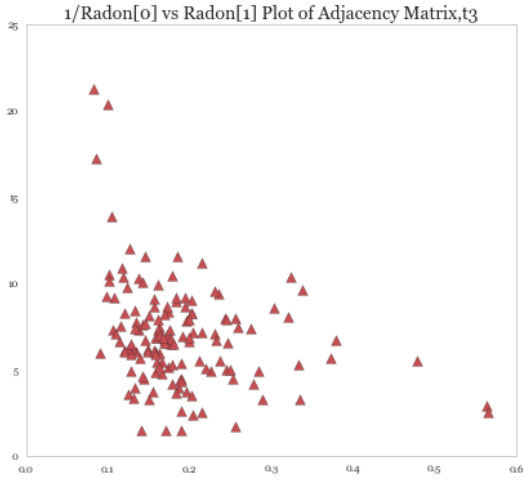


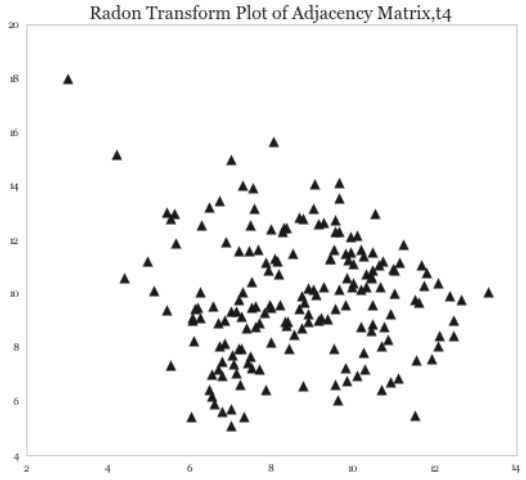


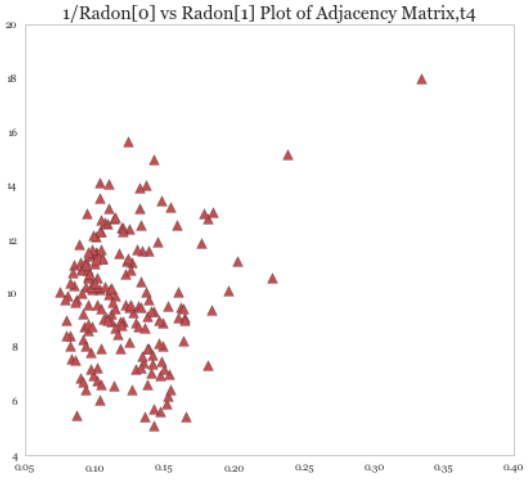

Radon Transform Plot of Adjacency Matrix,t2

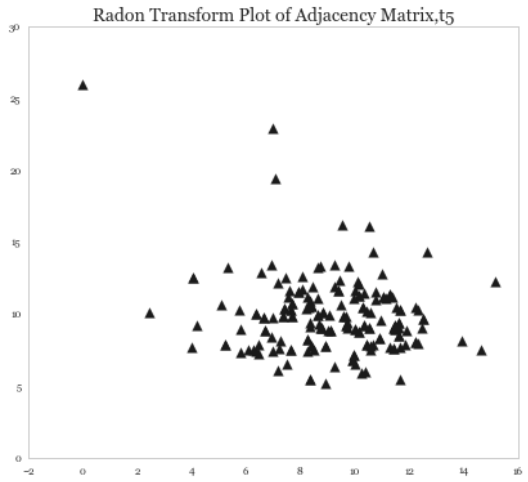

1/Radon[0] vs Radon[1] Plot of Adjacency Matrix,t2

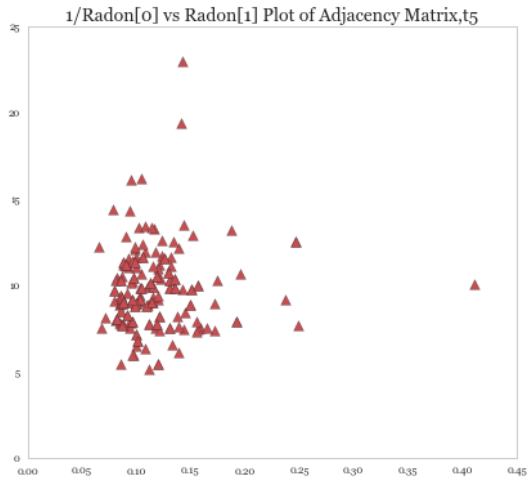

Radon Transform Plot of Adjacency Matrix,t3


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


Radon Transform Plot of Adjacency Matrix,t4

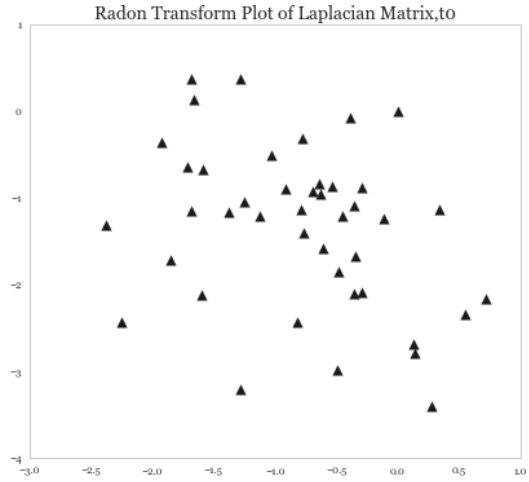

1/Radon[0] vs Radon[1] Plot of Adjacency Matrix,t4

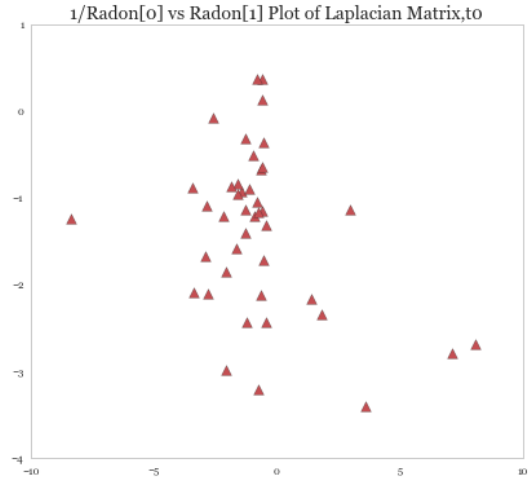


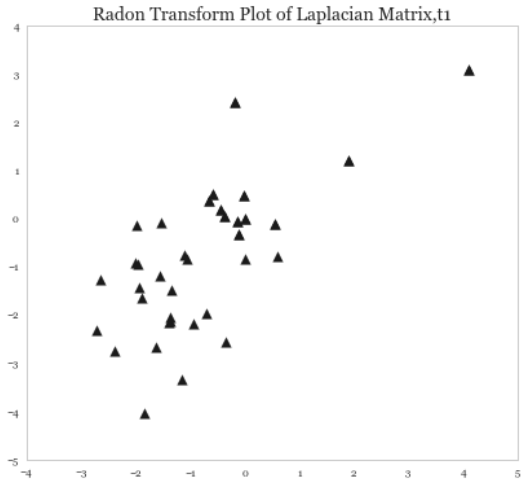


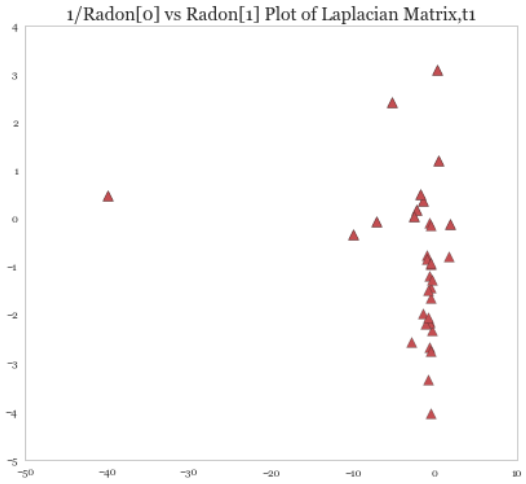

```
In [142]: plot_radon(lapM0,'Laplacian Matrix,t0')
          plot_radon(lapM1,'Laplacian Matrix,t1')
          plot_radon(lapM2,'Laplacian Matrix,t2')
          plot_radon(lapM3,'Laplacian Matrix,t3')
          plot_radon(lapM4,'Laplacian Matrix,t4')
          plot_radon(lapM5,'Laplacian Matrix,t5')
```

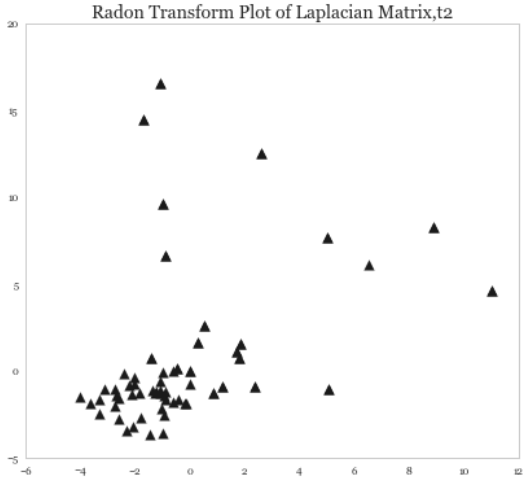
```
C:\Users\arsha_000\Anaconda3\lib\site-packages\skimage\transform\radon_transform.py
  warn('Radon transform: image must be zero outside the '
```

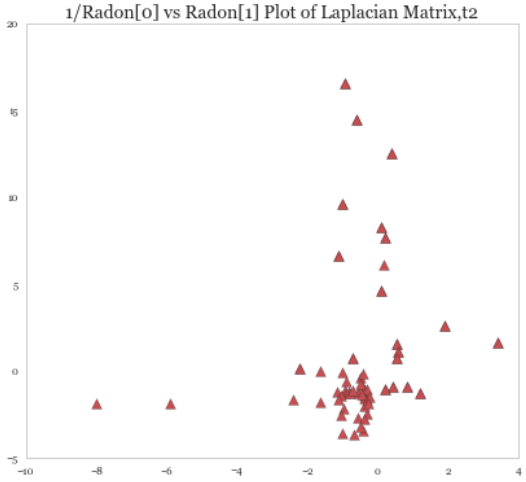


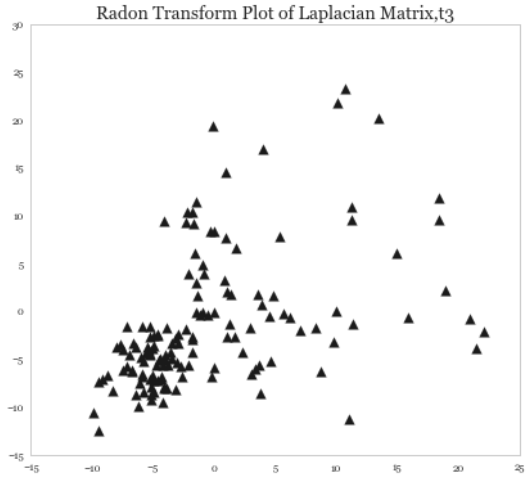


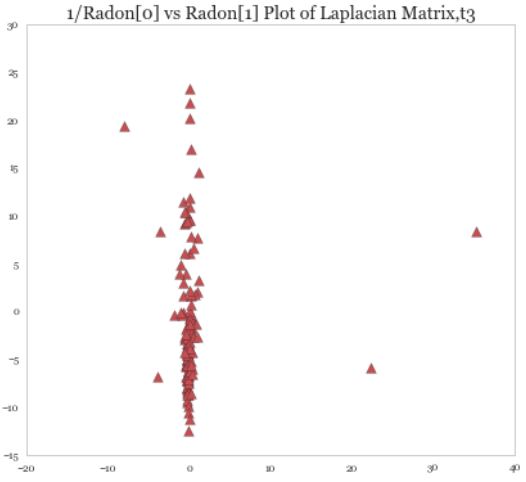

Radon Transform Plot of Laplacian Matrix,t1

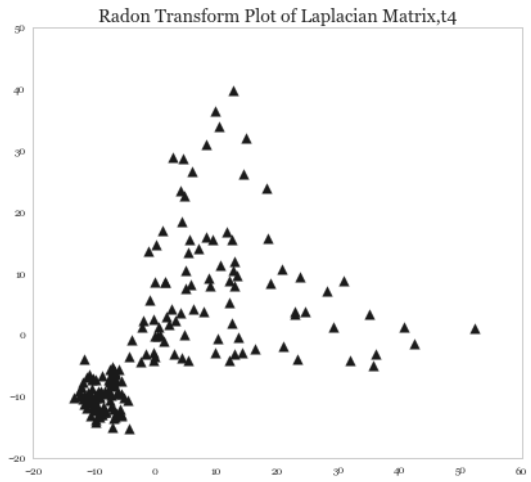

1/Radon[0] vs Radon[1] Plot of Laplacian Matrix,t1

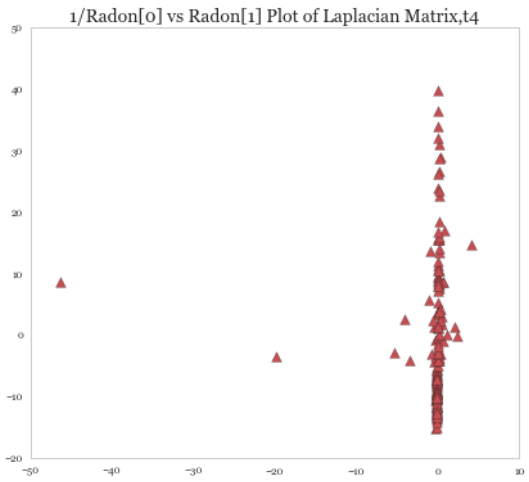

Radon Transform Plot of Laplacian Matrix,t2

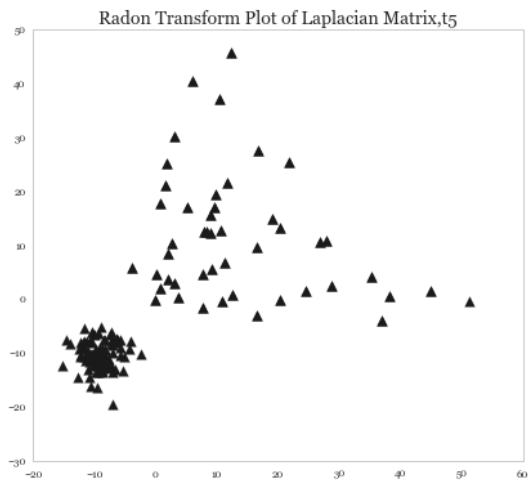

1/Radon[0] vs Radon[1] Plot of Laplacian Matrix,t2

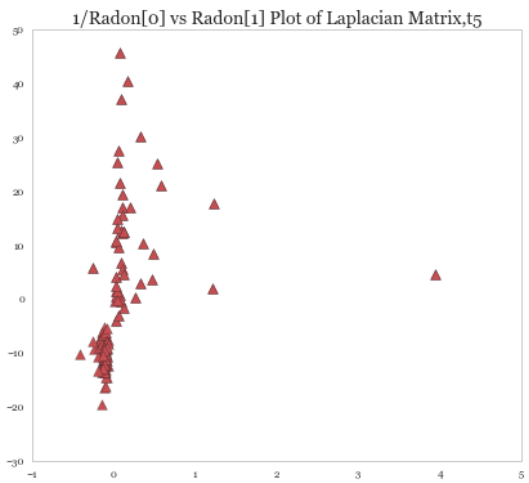

Radon Transform Plot of Laplacian Matrix,t3


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









14 Towards Attribute Analysis

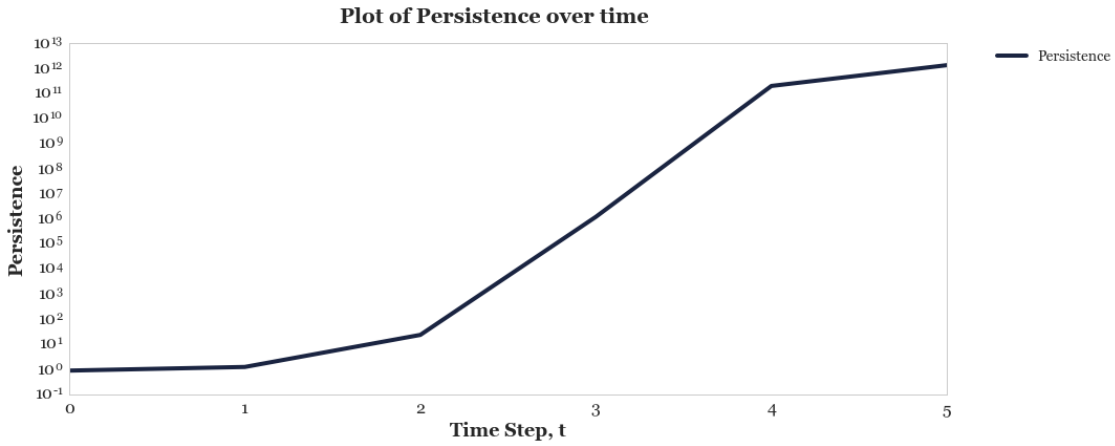
14.1 Attribute Matrices, Persistence & Emergence

The dynamic measures of Persistence and emergence are discussed here
A simplified averaging model is implemented here.

- I calculate Persistence and Emergence using the centrality measures of the network at each time step.
- I then calculate the persistence and emergence using the average of the centrality measures and the assortativity statistics calculated earlier

14.1.1 Persistence and Emergence with Time averaged centrality measures

```
In [143]: def attrmat(net):
              degC = get_val(nx.degree_centrality(net))
              cloC = get_val(nx.closeness_centrality(net))
              betC = get_val(nx.betweenness_centrality(net))
              eigC = get_val(nx.eigenvector_centrality_numpy(net))
              commC = get_val(nx.communicability_centrality(net))
              katzC = get_val(nx.katz_centrality_numpy(net))
              loadC= get_val(nx.load_centrality(net))

              mat = pd.DataFrame([degC,cloC,betC,eigC,commC,katzC, loadC]).T.fillna

              return mat

In [144]: attr0 = attrmat(Gt0)
          attr1 = attrmat(Gt1)
          attr2 = attrmat(Gt2)
          attr3 = attrmat(Gt3)
          attr4 = attrmat(Gt4)
          attr5 = attrmat(Gt5)

In [145]: persistence = pd.DataFrame([attr0.mean().sum()/5,
          attr1.mean().sum()/5,
          attr2.mean().sum()/5,
          attr3.mean().sum()/5,
          attr4.mean().sum()/5,
          attr5.mean().sum()/5], columns =['Persistence'])

In [146]: persistence.plot(fontsize=16, logy=True)
          plt.suptitle('Plot of Persistence over time', 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("Persistence", fontsize=18)

Out[146]: <matplotlib.text.Text at 0x2274f3feda0>
```

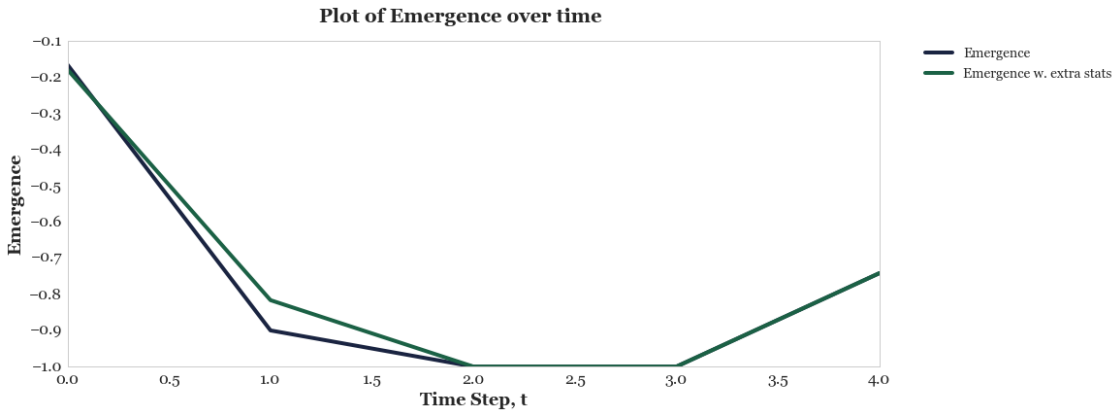



```
In [147]: def emergence(a,b):
              e = (a-b)/ (np.linalg.norm(a)+np.linalg.norm(b))
              return e

In [148]: per_arr =  persistence.values
          per_arr

Out[148]: array([[  9.14403708e-01],
                 [  1.27599030e+00],
                 [  2.42098693e+01],
                 [  1.22815824e+06],
                 [  2.01875918e+11],
                 [  1.36064749e+12]])

In [149]: emerg = []
          for i in range(0,5):
              x = int(i)
              y = x +1
              emerg.append(emergence(per_arr[x],per_arr[y]))

In [150]: emerg

Out[150]: [array([-0.16507834]),
           array([-0.8998668]),
           array([-0.99996058]),
           array([-0.99998783]),
           array([-0.7416027])]
```

14.1.2 Persistence and Emergence with averaged centrality and assortativity statistics

Since the initial values are averages so here I just take the average at each time step for all the measures of the network at that specific timestep. Therefore it is not necessary to sum the values as we have onyl one value for each time step so I just divide by $tmax - 1$.

```
In [151]: per_arr2 = stat_df2.fillna(0).mean(axis=1).apply(lambda x: x/5).values
          per_arr2

Out[151]: array([  1.30052094e-01,   1.86633762e-01,   1.84393595e+00,
                   8.18780540e+04,   1.34583945e+10,   9.07098328e+10])

In [152]: emerg2 = []
          for i in range(0,5):
              x = int(i)
              y = x +1
              emerg2.append(emergence(per_arr2[x],per_arr2[y]))

In [153]: plt.plot(emerg, label= 'Emergence')
          plt.plot(emerg2, label='Emergence w. extra stats')
          plt.suptitle('Plot of Emergence over time', 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("Emergence", fontsize=18)

Out[153]: <matplotlib.text.Text at 0x227478cadd8>
```



14.1.3 Attributes

In this section I explore the calculation of seismic attributes such as:

- Instantaneous Phase (IP)
- Instantaneous Amplitude (IA)
- Instantaneous Frequency (IA)

In addition to other attributes that might be potentially interesting.
The attribute calculation formulas are shown here
Additional attributes are discussed here

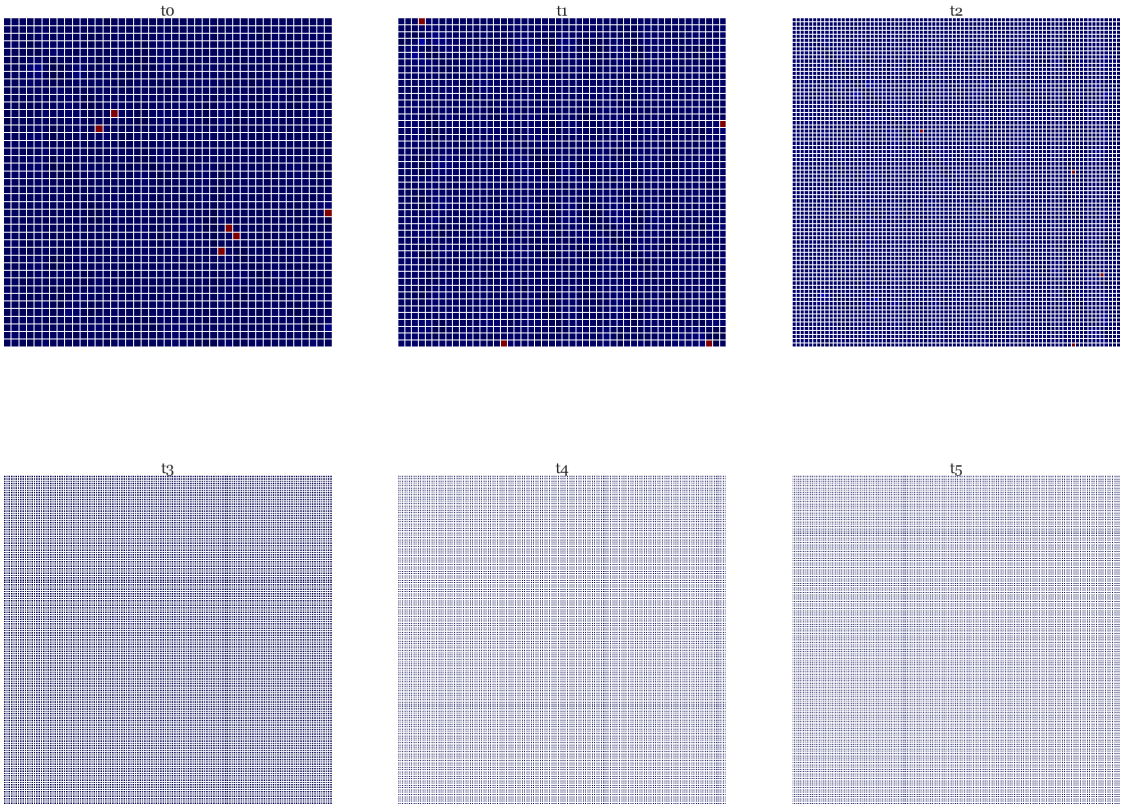
```
In [154]: def norm_mat(M):
              r=norm(np.diag(M))/norm(M)
              return r


In [155]: def calc_seisatt(M):
              Ht = hilbert(M)
              IA = np.nan_to_num(np.sqrt(np.dot(M,M)+np.dot(Ht,Ht)))
              IP = np.nan_to_num(np.arctan(Ht/M))
              IF,_ = np.nan_to_num(np.asarray(np.gradient(IP)))
              E = np.sqrt(np.dot(M,M)+np.dot(Ht,Ht))
              dE, _ = np.nan_to_num(np.asarray(np.gradient(E)))
              dEe, _ = np.nan_to_num(np.asarray(np.gradient(dE)))


              att_globalval = pd.DataFrame([norm(IA),norm(IP),norm(IF),norm(E),\
                                            norm(dE),norm(dEe)]).T

              return [IA,IP,IF,E, dE, dEe, att_globalval]
```

14.2 Curvature

```
In [156]: def curvature(M):
              from skimage.feature import hessian_matrix, hessian_matrix_det, hessi
              M = np.float64(M)
              fx, fy = np.gradient(M)
              Hxx, Hxy, Hyy = hessian_matrix(M)
              K = np.divide((np.dot(Hxx,Hxy)-np.dot(Hxy,Hxy)),\
                            (1+np.dot(fx,fx)+np.dot(fy,fy)))

              He1,He2 = hessian_matrix_eigvals(Hxx,Hxy,Hyy)
              mean_curv = np.trace(He1)
              s, a = np.linalg.slogdet(He1)
              conc = s * np.exp(a)
              Pmax = np.max(He1)
              Pmin = np.min(He1)

              return [K,mean_curv,conc]

In [157]: adjk0, adjmc0,adjc0 = curvature(adjM0)
          adjk1, adjmc1,adjc1 = curvature(adjM1)
          adjk2, adjmc2,adjc2 = curvature(adjM2)
          adjk3, adjmc3,adjc3 = curvature(adjM3)
          adjk4, adjmc4,adjc4 = curvature(adjM4)
          adjk5, adjmc5,adjc5 = curvature(adjM5)

In [158]: lapk0, lapmc0,lapc0 = curvature(lapM0)
          lapk1, lapmc1,lapc1 = curvature(lapM1)
          lapk2, lapmc2,lapc2 = curvature(lapM2)
```

```
           lapk3, lapmc3,lapc3 = curvature(lapM3)
           lapk4, lapmc4,lapc4 = curvature(lapM4)
           lapk5, lapmc5,lapc5 = curvature(lapM5)

In [159]: modk0, modmc0,modc0 = curvature(modM0)
           modk1, modmc1,modc1 = curvature(modM1)
           modk2, modmc2,modc2 = curvature(modM2)
           modk3, modmc3,modc3 = curvature(modM3)
           modk4, modmc4,modc4 = curvature(modM4)
           modk5, modmc5,modc5 = curvature(modM5)

In [160]: plt.figure(figsize=(40, 30))

           plt.subplot(231)
           sns.heatmap(adjk0,cmap='seismic', center=True, robust=True, fmt='d', line
                       yticklabels=False, xticklabels=False, cbar=False)

           plt.suptitle('Heatmap of Curvature Adjacency Matrix, t0-t5', fontsize=60)
           plt.title('t0', fontsize=30)

           plt.subplot(232)
           sns.heatmap(adjk1,cmap='seismic', center=True, robust=True, fmt='d', line
                       yticklabels=False, xticklabels=False, cbar=False)
           plt.title('t1', fontsize=30)

           plt.subplot(233)
           sns.heatmap(adjk2,cmap='seismic', center=True, robust=True, fmt='d', line
                       yticklabels=False, xticklabels=False, cbar=False)
           plt.title('t2', fontsize=30)

           plt.subplot(234)
           sns.heatmap(adjk3,cmap='seismic', center=True, robust=True, fmt='d', line
                       yticklabels=False, xticklabels=False, cbar=False)
           plt.title('t3', fontsize=30)

           plt.subplot(235)
           sns.heatmap(adjk4,cmap='seismic', center=True, robust=True, fmt='d', line
                       yticklabels=False, xticklabels=False, cbar=False)
           plt.title('t4', fontsize=30)

           plt.subplot(236)
           sns.heatmap(adjk5,cmap='seismic', center=True, robust=True, fmt='d', line
                       yticklabels=False, xticklabels=False, cbar=False)
           plt.title('t5', fontsize=30)

Out[160]: <matplotlib.text.Text at 0x2274f494160>
```

Heatmap of Curvature Adjacency Matrix, to-t5

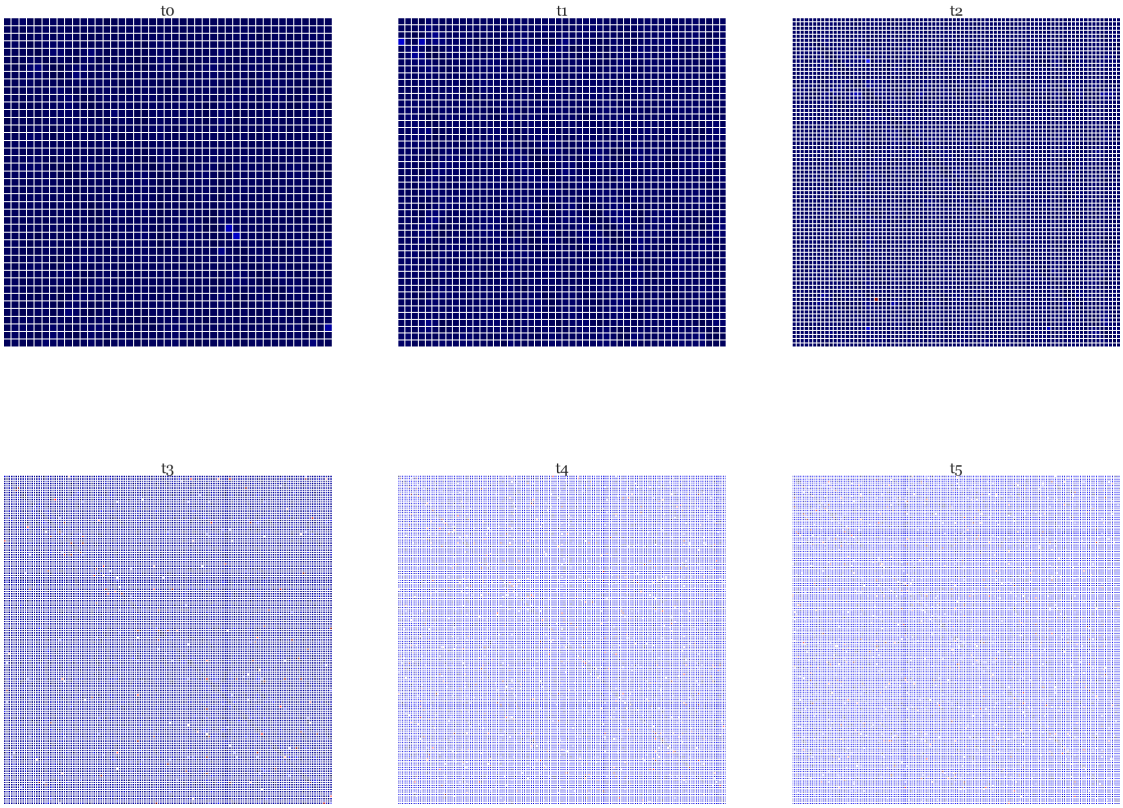



```
In [161]: plt.figure(figsize=(40, 30))

          plt.subplot(231)
          sns.heatmap(modk0,cmap='seismic', center=True, robust=True, fmt='d', line
                      yticklabels=False, xticklabels=False, cbar=False)

          plt.suptitle('Heatmap of Curvature Modularity Matrix, t0-t5', fontsize=60
          plt.title('t0', fontsize=30)

          plt.subplot(232)
          sns.heatmap(modk1,cmap='seismic', center=True, robust=True, fmt='d', line
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('t1', fontsize=30)

          plt.subplot(233)
          sns.heatmap(modk2,cmap='seismic', center=True, robust=True, fmt='d', line
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('t2', fontsize=30)
```

```
plt.subplot(234)
sns.heatmap(modk3,cmap='seismic', center=True, robust=True, fmt='d', line
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t3', fontsize=30)

plt.subplot(235)
sns.heatmap(modk4,cmap='seismic', center=True, robust=True, fmt='d', line
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t4', fontsize=30)

plt.subplot(236)
sns.heatmap(modk5,cmap='seismic', center=True, robust=True, fmt='d', line
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t5', fontsize=30)
```

```
Out[161]: <matplotlib.text.Text at 0x2274b9b9080>
```

Heatmap of Curvature Modularity Matrix, to-t5

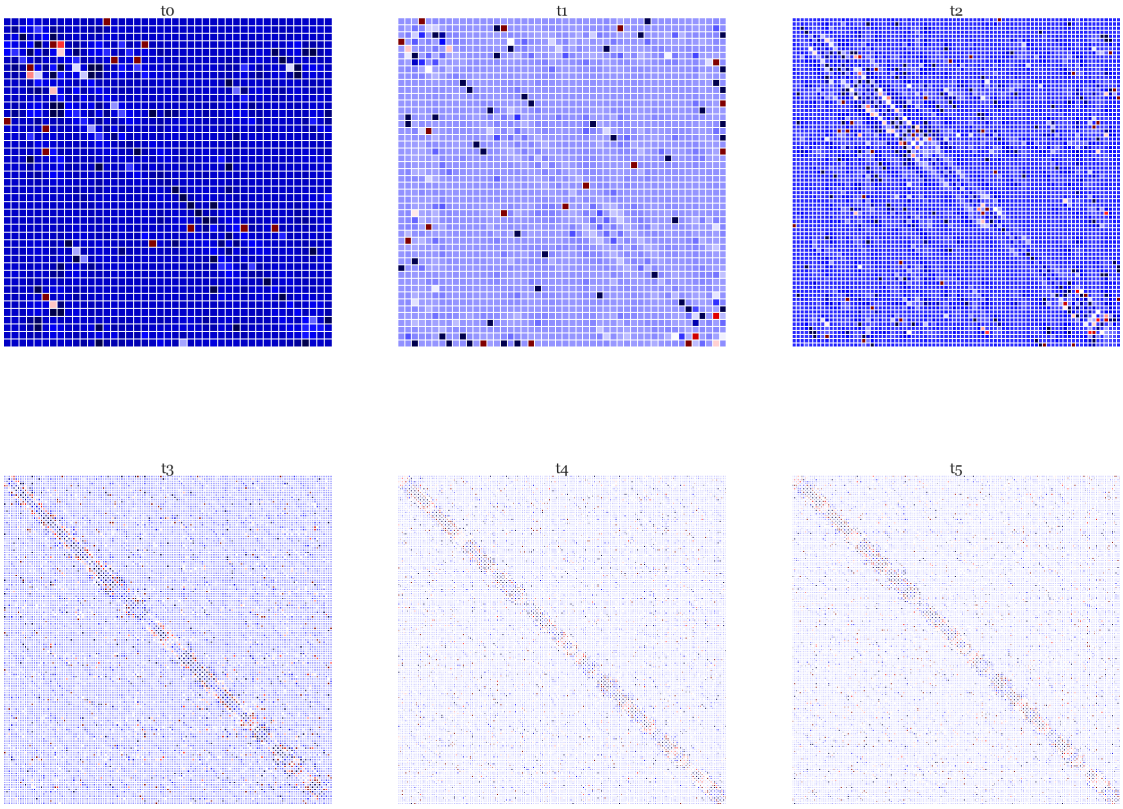



```
In [162]: plt.figure(figsize=(40, 30))
```

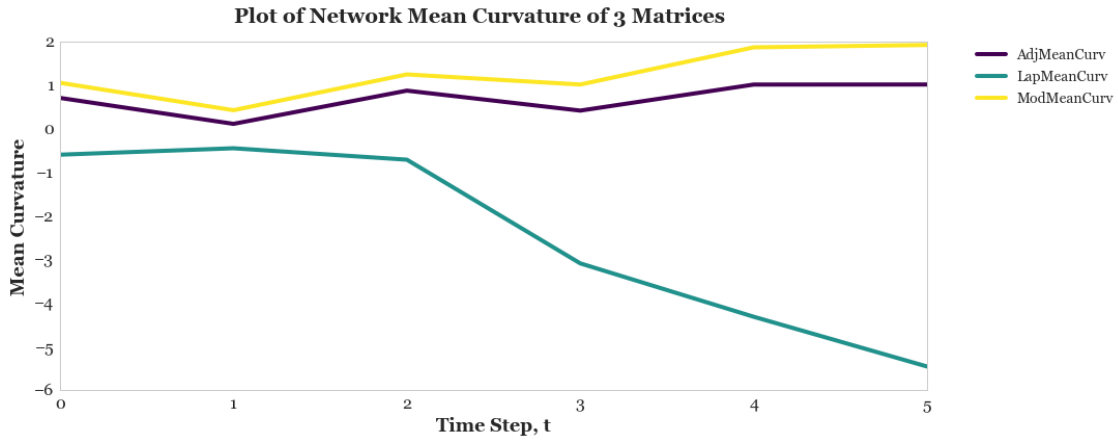
```
plt.subplot(231)
sns.heatmap(lapk0,cmap='seismic', center=True, robust=True, fmt='d', line
            yticklabels=False, xticklabels=False, cbar=False)

plt.suptitle('Heatmap of Curvature Laplacian Matrix, t0-t5', fontsize=60)
plt.title('t0', fontsize=30)

plt.subplot(232)
sns.heatmap(lapk1,cmap='seismic', center=True, robust=True, fmt='d', line
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t1', fontsize=30)

plt.subplot(233)
sns.heatmap(lapk2,cmap='seismic', center=True, robust=True, fmt='d', line
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t2', fontsize=30)

plt.subplot(234)
sns.heatmap(lapk3,cmap='seismic', center=True, robust=True, fmt='d', line
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t3', fontsize=30)

plt.subplot(235)
sns.heatmap(lapk4,cmap='seismic', center=True, robust=True, fmt='d', line
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t4', fontsize=30)

plt.subplot(236)
sns.heatmap(lapk5,cmap='seismic', center=True, robust=True, fmt='d', line
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t5', fontsize=30)
```

```
Out[162]: <matplotlib.text.Text at 0x2274f5fe978>
```

Heatmap of Curvature Laplacian Matrix, to-t5




```
In [163]: MeanCurv_df = pd.DataFrame([adjmc0,adjmc1,adjmc2,adjmc3,adjmc4,adjmc5], c
          MeanCurv_df['LapMeanCurv'] = pd.DataFrame([lapmc0,lapmc1,lapmc2,lapmc3,la
          MeanCurv_df['ModMeanCurv'] = pd.DataFrame([modmc0,modmc1,modmc2,modmc3,mo
          MeanCurv_df.plot(colormap='viridis', fontsize=16)
          plt.suptitle('Plot of Network Mean Curvature of 3 Matrices', 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("Mean Curvature", fontsize=18)

Out[163]: <matplotlib.text.Text at 0x2274ea614e0>
```



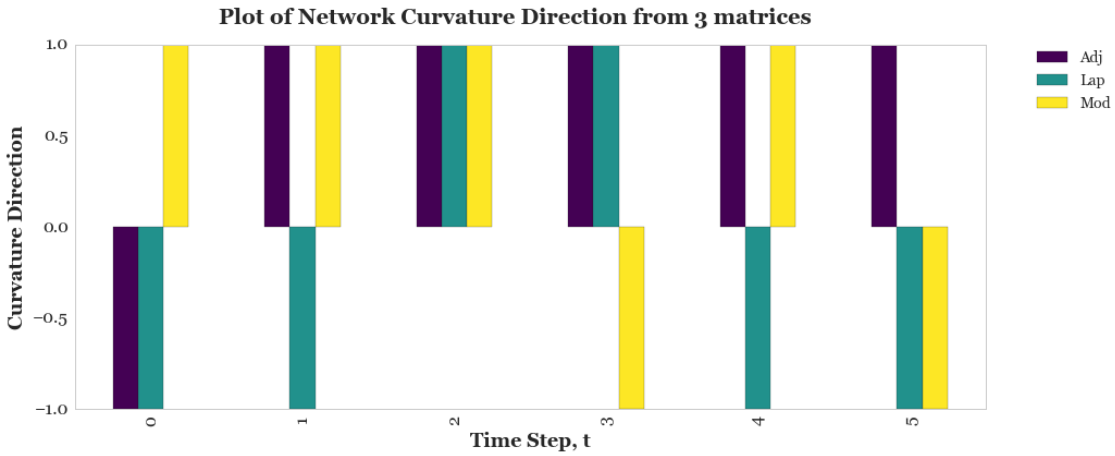

```
In [164]: Conc_df =pd.DataFrame([adjc0,adjc1,adjc2,adjc3,adjc4,adjc5],columns=['Adj
          Conc_df['Lap'] = pd.DataFrame([lapc0,lapc1,lapc2,lapc3,lapc4,lapc5])
          Conc_df['Mod'] = pd.DataFrame([modc0,modc1,modc2,modc3,modc4,modc5])
          Conc_df.head()
```

Out[164]:

	Adj	Lap	Mod
0	-2.959369e-39	-3.479674e-26	2.168546e-39
1	9.060606e-43	-9.860077e-29	1.462558e-42
2	4.566311e-64	2.533815e-29	4.889410e-65
3	1.230442e-81	7.233346e+17	-3.189491e-81
4	1.643767e-79	-9.607652e+74	5.070194e-79

```
In [165]: dir_m = np.asmatrix(np.where(Conc_df.values < 0,-1,np.where(Conc_df.value
          dir_df = pd.DataFrame(dir_m, columns=['Adj','Lap','Mod'])
          dir_df.plot(colormap='viridis', fontsize=16, kind='bar')
          plt.suptitle('Plot of Network Curvature Direction from 3 matrices', fonts
          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("Curvature Direction", fontsize=18)
```

```
Out[165]: <matplotlib.text.Text at 0x2274fac5390>
```



14.3 Attribute Analysis

```
In [166]: adjIA0, adjIP0, adjIF0, adjE0, adjdE0, adjdEe0, adjglob0 = calc_seisatt(a
          adjIA1, adjIP1, adjIF1, adjE1, adjdE1, adjdEe1, adjglob1 = calc_seisatt(a
          adjIA2, adjIP2, adjIF0, adjE2, adjdE2, adjdEe2, adjglob2 = calc_seisatt(a
          adjIA3, adjIP3, adjIF3, adjE3, adjdE3, adjdEe3, adjglob3 = calc_seisatt(a
          adjIA4, adjIP4, adjIF4, adjE4, adjdE4, adjdEe4, adjglob4 = calc_seisatt(a
          adjIA5, adjIP5, adjIF5, adjE5, adjdE5, adjdEe5, adjglob5 = calc_seisatt(a

In [167]: lapIA0, lapIP0, lapIF0, lapE0, lapdE0, lapdEe0, lapglob0 = calc_seisatt(l
          lapIA1, lapIP1, lapIF1, lapE1, lapdE1, lapdEe1, lapglob1 = calc_seisatt(l
          lapIA2, lapIP2, lapIF0, lapE2, lapdE2, lapdEe2, lapglob2 = calc_seisatt(l
          lapIA3, lapIP3, lapIF3, lapE3, lapdE3, lapdEe3, lapglob3 = calc_seisatt(l
          lapIA4, lapIP4, lapIF4, lapE4, lapdE4, lapdEe4, lapglob4 = calc_seisatt(l
          lapIA5, lapIP5, lapIF5, lapE5, lapdE5, lapdEe5, lapglob5 = calc_seisatt(l

In [168]: modIA0, modIP0, modIF0, modE0, moddE0, moddEe0, modglob0 = calc_seisatt(m
          modIA1, modIP1, modIF1, modE1, moddE1, moddEe1, modglob1 = calc_seisatt(m
          modIA2, modIP2, modIF0, modE2, moddE2, moddEe2, modglob2 = calc_seisatt(m
          modIA3, modIP3, modIF3, modE3, moddE3, moddEe3, modglob3 = calc_seisatt(m
          modIA4, modIP4, modIF4, modE4, moddE4, moddEe4, modglob4 = calc_seisatt(m
          modIA5, modIP5, modIF5, modE5, moddE5, moddEe5, modglob5 = calc_seisatt(m

In [169]: adjglob_df = adjglob0.append(adjglob1).append(adjglob2).append(adjglob3).
          adjglob_df.columns = ['InstAmp','InstPhase','InstFreq.','EnergyEnv','dEne
          adjglob_df.set_index([[0,1,2,3,4,5]], inplace=True)
          adjglob_df.head()
```

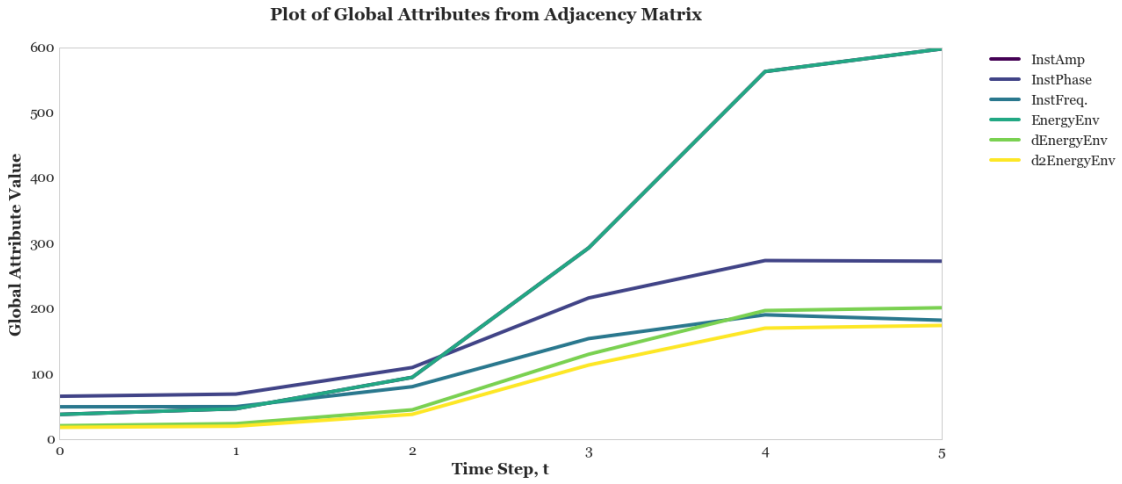
Out[169]:

	InstAmp	InstPhase	InstFreq.	EnergyEnv	dEnergyEnv	d2EnergyEn
0	38.186710	66.008877	49.881368	38.186710	20.977247	18.40432
1	46.880508	69.358658	50.101436	46.880508	24.082062	20.24474
2	94.970176	109.992126	80.671227	94.970176	45.252526	38.35810

```
3  293.054551  216.464161  154.266190  293.054551  130.265093   113.75337
4  563.036254  273.727614  190.671908  563.036254  197.190587   170.23931
```

In [170]:
```
adjglob_df.plot(colormap='viridis', fontsize=16, figsize=(18,8))
plt.suptitle('Plot of Global Attributes from Adjacency Matrix', fontsize=
plt.yticks(fontsize=16)
plt.xticks(fontsize=16)
plt.legend(fontsize=16, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
plt.xlabel("Time Step, t", fontsize=18)
plt.ylabel("Global Attribute Value", fontsize=18)
```

Out[170]:
```
<matplotlib.text.Text at 0x2274f4daf98>
```



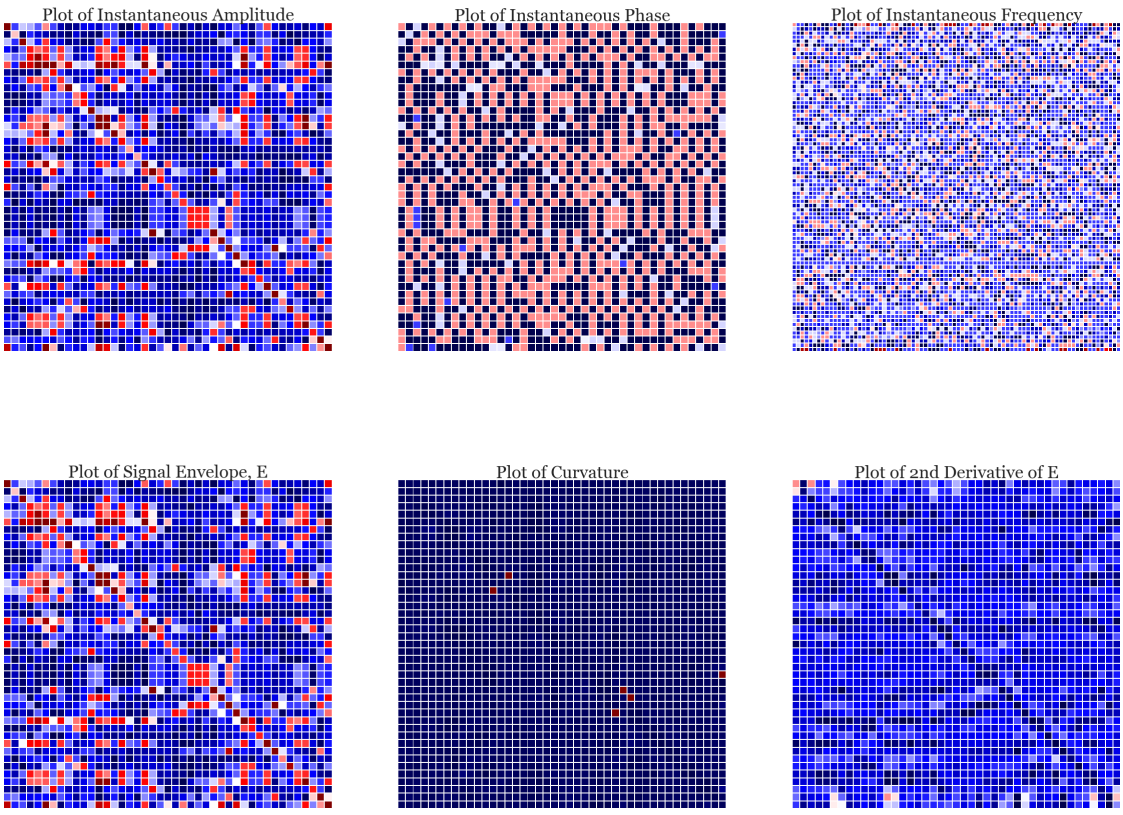
In [171]:
```
plt.figure(figsize=(40, 30))

plt.subplot(231)
sns.heatmap(adjIA0,cmap='seismic', center=True, robust=True, fmt='d', lin
            yticklabels=False, xticklabels=False, cbar=False)

plt.title('Plot of Instantaneous Amplitude', fontsize=35)
plt.suptitle('Heatmap of IA, IP, IF, E, dE, dEe of the Adjacency Matrix,

plt.subplot(232)
sns.heatmap(adjIP0, cmap='seismic', center=True, robust=True, fmt='d', li
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('Plot of Instantaneous Phase', fontsize=35)

plt.subplot(233)
sns.heatmap(adjIF0, cmap='seismic', center=True, robust=True, fmt='d', li
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('Plot of Instantaneous Frequency', fontsize=35)
```

```
plt.subplot(234)
sns.heatmap(adjE0, cmap='seismic', center=True, robust=True, fmt='d', lir
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('Plot of Signal Envelope, E', fontsize=35)

plt.subplot(235)
sns.heatmap(adjk0, cmap='seismic', center=True, robust=True, fmt='d', lir
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('Plot of Curvature', fontsize=35)

plt.subplot(236)
sns.heatmap(adjdEe0, cmap='seismic', center=True, robust=True, fmt='d', l
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('Plot of 2nd Derivative of E', fontsize=35)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\ma\core.py:3095: ComplexWarnir
  output = self._data.astype(newtype).view(type(self))
```

```
Out[171]: <matplotlib.text.Text at 0x2274f3f61d0>
```

Heatmap of IA, IP, IF, E, dE, dEe of the Adjacency Matrix, to



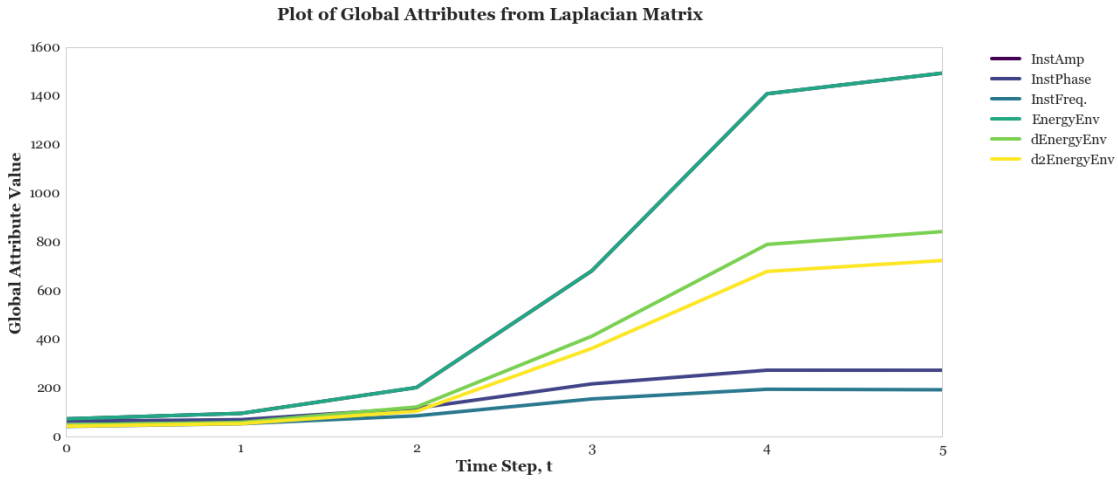
In [172]:
```
lapglob_df = lapglob0.append(lapglob1).append(lapglob2).append(lapglob3).
lapglob_df.columns = ['InstAmp','InstPhase','InstFreq.','EnergyEnv','dEne
lapglob_df.set_index([[0,1,2,3,4,5]], inplace=True)
lapglob_df.head()
```

Out[172]:

	InstAmp	InstPhase	InstFreq.	EnergyEnv	dEnergyEnv	d2Energy
0	73.014790	63.250000	41.899990	73.014790	48.708167	42.841
1	95.697029	69.973063	53.056054	95.697029	58.760915	53.666
2	201.848840	116.820044	85.648821	201.848840	121.889250	104.394
3	680.947797	216.750295	154.955039	680.947797	412.506571	362.394
4	1408.349343	273.018131	194.726052	1408.349343	789.564513	678.617

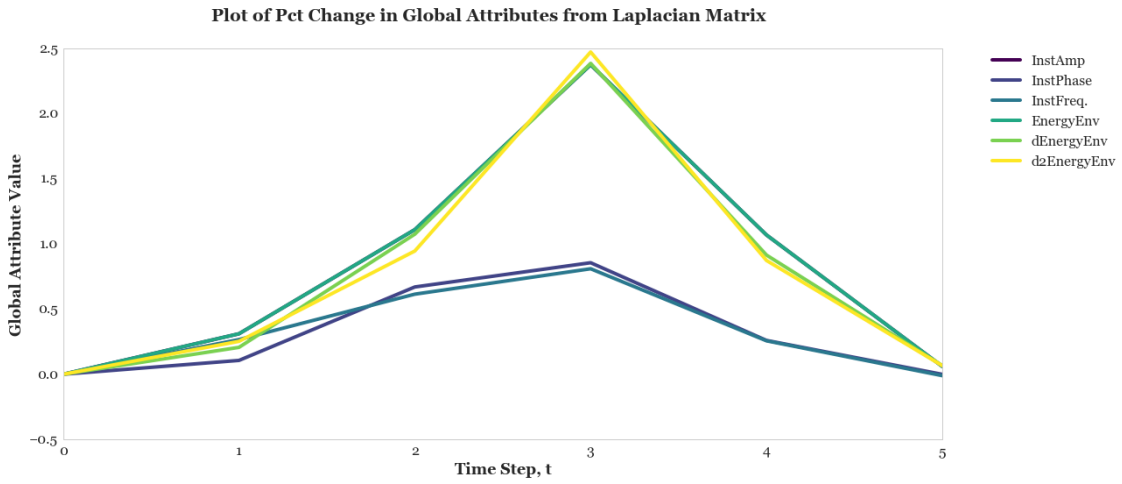
In [173]:
```
lapglob_df.plot(colormap='viridis', fontsize=16, figsize=(18,8))
plt.suptitle('Plot of Global Attributes from Laplacian Matrix', fontsize=
plt.yticks(fontsize=16)
plt.xticks(fontsize=16)
plt.xlabel("Time Step, t", fontsize=18)
plt.legend(fontsize=16, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
plt.ylabel("Global Attribute Value", fontsize=18)
```

Out[173]: <matplotlib.text.Text at 0x2274796ee10>



In [176]:
```
lapglob_df.pct_change().fillna(0).plot(colormap='viridis', fontsize=16, f
plt.suptitle('Plot of Pct Change in Global Attributes from Laplacian Matr
plt.yticks(fontsize=16)
plt.xticks(fontsize=16)
plt.xlabel("Time Step, t", fontsize=18)
plt.legend(fontsize=16, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
plt.ylabel("Global Attribute Value", fontsize=18)
```

Out[176]: <matplotlib.text.Text at 0x2274f93e160>



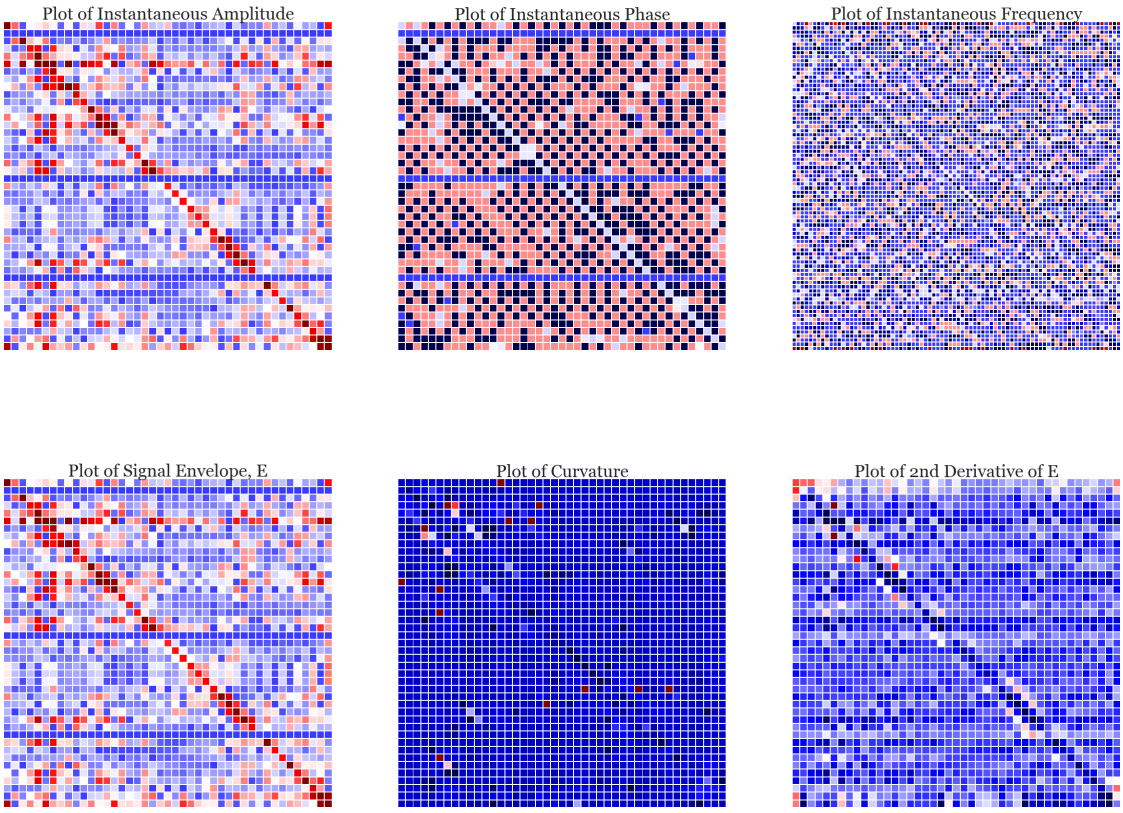

```
In [177]: plt.figure(figsize=(40, 30))

          plt.subplot(231)
          sns.heatmap(lapIA0,cmap='seismic', center=True, robust=True, fmt='d', lin
                      yticklabels=False, xticklabels=False, cbar=False)

          plt.title('Plot of Instantaneous Amplitude', fontsize=35)
          plt.suptitle('Heatmap of IA, IP, IF, E, dE, dEe of the Laplacian Matrix,

          plt.subplot(232)
          sns.heatmap(lapIP0, cmap='seismic', center=True, robust=True, fmt='d', li
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Instantaneous Phase', fontsize=35)

          plt.subplot(233)
          sns.heatmap(lapIF0, cmap='seismic', center=True, robust=True, fmt='d', li
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Instantaneous Frequency', fontsize=35)

          plt.subplot(234)
          sns.heatmap(lapE0, cmap='seismic', center=True, robust=True, fmt='d', lin
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Signal Envelope, E', fontsize=35)

          plt.subplot(235)
          sns.heatmap(lapk0, cmap='seismic', center=True, robust=True, fmt='d', lin
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Curvature', fontsize=35)

          plt.subplot(236)
          sns.heatmap(lapdEe0, cmap='seismic', center=True, robust=True, fmt='d', l
```

```
                    yticklabels=False, xticklabels=False, cbar=False)
        plt.title('Plot of 2nd Derivative of E', fontsize=35)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\ma\core.py:3095: ComplexWarnir
  output = self._data.astype(newtype).view(type(self))
```

Out[177]: <matplotlib.text.Text at 0x2275a605320>

Heatmap of IA, IP, IF, E, dE, dEe of the Laplacian Matrix, to

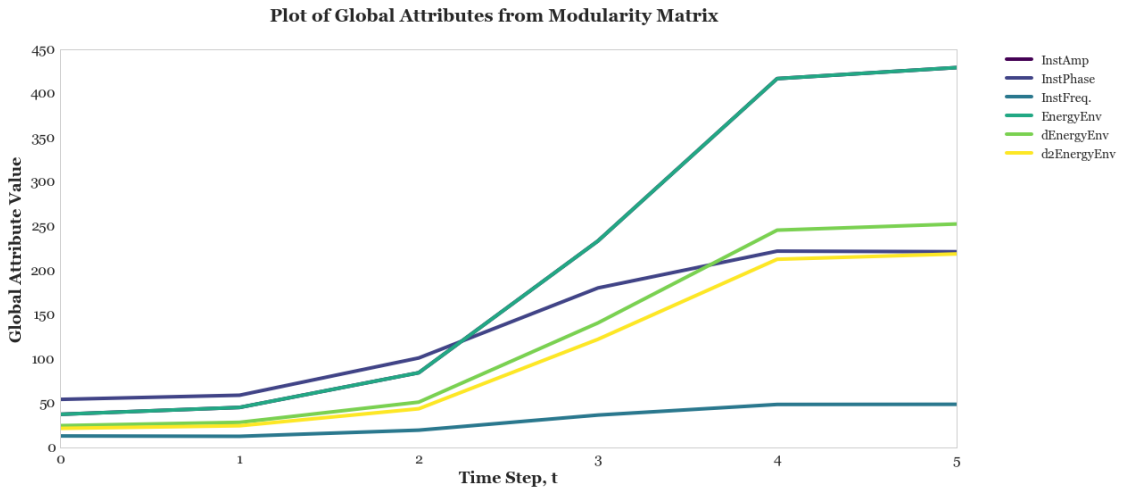


In [178]:
```
modglob_df = modglob0.append(modglob1).append(modglob2).append(modglob3).
modglob_df.columns = ['InstAmp','InstPhase','InstFreq.','EnergyEnv','dEne
modglob_df.set_index([[0,1,2,3,4,5]], inplace=True)
modglob_df.head()
```

Out[178]:

	InstAmp	InstPhase	InstFreq.	EnergyEnv	dEnergyEnv	d2EnergyEnv
0	37.664283	54.509072	13.205153	37.664283	24.876889	21.794054
1	45.417822	59.196431	12.813470	45.417822	28.570743	24.707563
2	84.616857	101.269200	19.751385	84.616857	51.404777	43.907761
3	233.408611	180.327519	36.810706	233.408611	140.862499	122.362069
4	416.720939	221.908499	48.779973	416.720939	245.597684	212.721903

```
In [179]: modglob_df.plot(colormap='viridis', fontsize=16, figsize=(18,8))
          plt.suptitle('Plot of Global Attributes from Modularity Matrix', fontsize
          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("Global Attribute Value", fontsize=18)

Out[179]: <matplotlib.text.Text at 0x2274feac668>
```



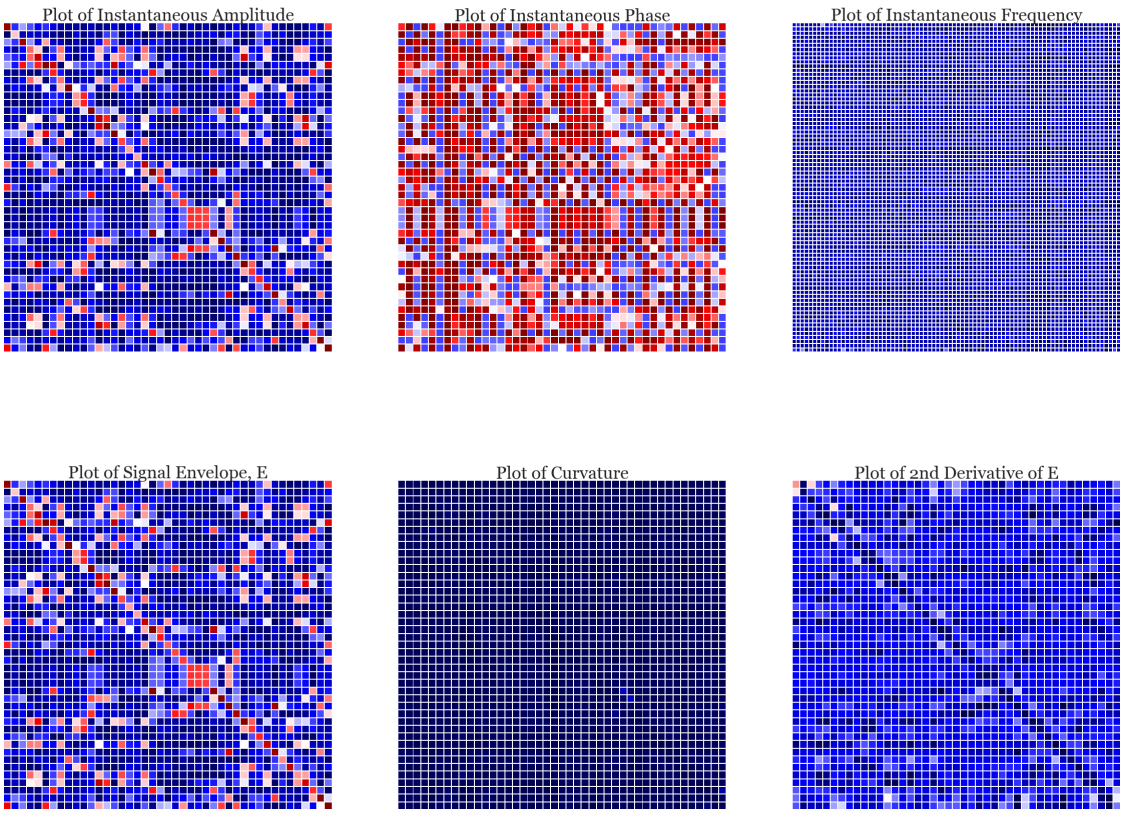

```
In [180]: plt.figure(figsize=(40, 30))

          plt.subplot(231)
          sns.heatmap(modIA0,cmap='seismic', center=True, robust=True, fmt='d', lin
                      yticklabels=False, xticklabels=False, cbar=False)

          plt.title('Plot of Instantaneous Amplitude', fontsize=35)
          plt.suptitle('Heatmap of IA, IP, IF, E, dE, dEe of the Modularity Matrix,

          plt.subplot(232)
          sns.heatmap(modIP0, cmap='seismic', center=True, robust=True, fmt='d', li
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Instantaneous Phase', fontsize=35)

          plt.subplot(233)
          sns.heatmap(modIF0, cmap='seismic', center=True, robust=True, fmt='d', li
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Instantaneous Frequency', fontsize=35)

          plt.subplot(234)
          sns.heatmap(modE0, cmap='seismic', center=True, robust=True, fmt='d', lin
```

```
                    yticklabels=False, xticklabels=False, cbar=False)
        plt.title('Plot of Signal Envelope, E', fontsize=35)

        plt.subplot(235)
        sns.heatmap(modk0, cmap='seismic', center=True, robust=True, fmt='d', lin
                    yticklabels=False, xticklabels=False, cbar=False)
        plt.title('Plot of Curvature', fontsize=35)

        plt.subplot(236)
        sns.heatmap(moddEe0, cmap='seismic', center=True, robust=True, fmt='d', l
                    yticklabels=False, xticklabels=False, cbar=False)
        plt.title('Plot of 2nd Derivative of E', fontsize=35)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\ma\core.py:3095: ComplexWarnin
  output = self._data.astype(newtype).view(type(self))
```

Out[180]: <matplotlib.text.Text at 0x22759c908d0>

Heatmap of IA, IP, IF, E, dE, dEe of the Modularity Matrix, to



```
In [181]: plt.figure(figsize=(40, 30))

          plt.subplot(231)
          sns.heatmap(adjIA5,cmap='seismic', center=True, robust=True, fmt='d', lin
                      yticklabels=False, xticklabels=False, cbar=False)

          plt.title('Plot of Instantaneous Amplitude', fontsize=35)
          plt.suptitle('Heatmap of IA, IP, IF, E, dE, dEe of the Adjacency Matrix,

          plt.subplot(232)
          sns.heatmap(adjIP5, cmap='seismic', center=True, robust=True, fmt='d', li
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Instantaneous Phase', fontsize=35)

          plt.subplot(233)
          sns.heatmap(adjIF5, cmap='seismic', center=True, robust=True, fmt='d', li
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Instantaneous Frequency', fontsize=35)

          plt.subplot(234)
          sns.heatmap(adjE5, cmap='seismic', center=True, robust=True, fmt='d', lin
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Signal Envelope, E', fontsize=35)

          plt.subplot(235)
          sns.heatmap(adjk5, cmap='seismic', center=True, robust=True, fmt='d', lin
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Curvature', fontsize=35)

          plt.subplot(236)
          sns.heatmap(adjdEe5, cmap='seismic', center=True, robust=True, fmt='d', l
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of 2nd Derivative of E', fontsize=35)

C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\ma\core.py:3095: ComplexWarnin
  output = self._data.astype(newtype).view(type(self))


Out[181]: <matplotlib.text.Text at 0x2275a2d7080>
```

Heatmap of IA, IP, IF, E, dE, dEe of the Adjacency Matrix, t5

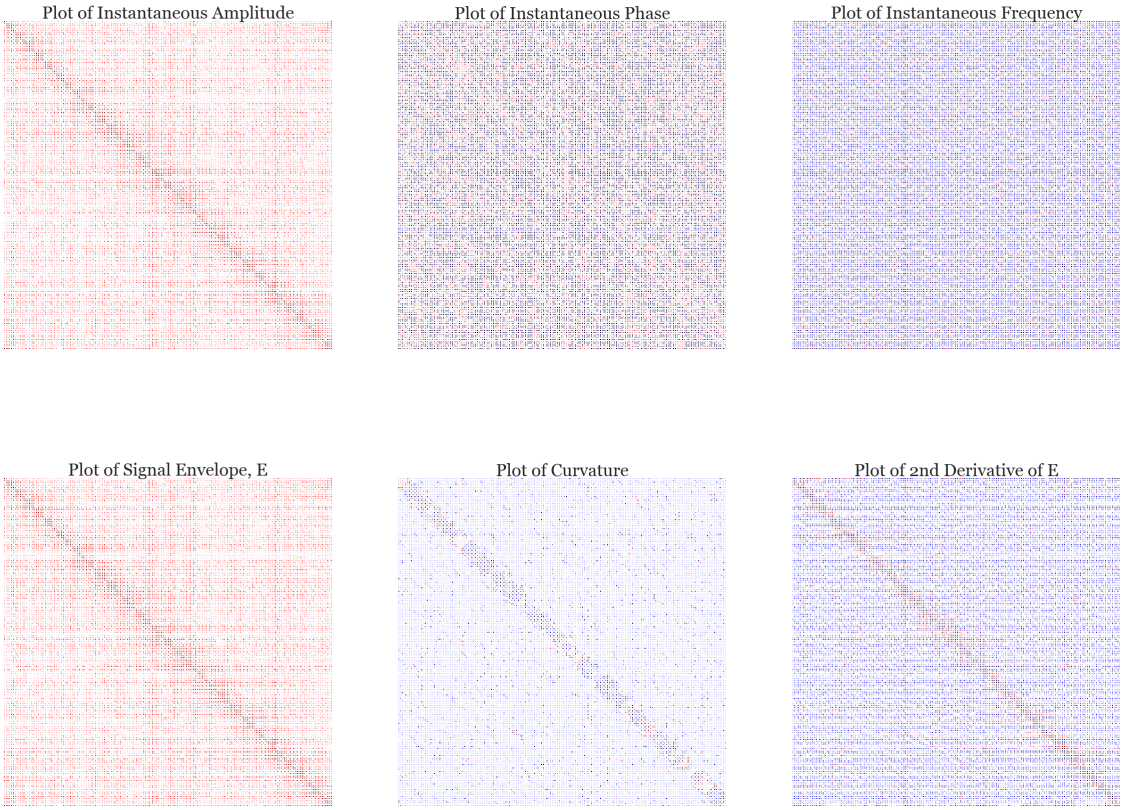

```
In [182]: plt.figure(figsize=(40, 30))

          plt.subplot(231)
          sns.heatmap(lapIA5,cmap='seismic', center=True, robust=True, fmt='d', lin
                      yticklabels=False, xticklabels=False, cbar=False)

          plt.title('Plot of Instantaneous Amplitude', fontsize=35)
          plt.suptitle('Heatmap of IA, IP, IF, E, dE, dEe of the Laplaciany Matrix,

          plt.subplot(232)
          sns.heatmap(lapIP5, cmap='seismic', center=True, robust=True, fmt='d', li
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Instantaneous Phase', fontsize=35)

          plt.subplot(233)
          sns.heatmap(lapIF5, cmap='seismic', center=True, robust=True, fmt='d', li
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Instantaneous Frequency', fontsize=35)
```

```
            plt.subplot(234)
            sns.heatmap(lapE5, cmap='seismic', center=True, robust=True, fmt='d', lin
                        yticklabels=False, xticklabels=False, cbar=False)
            plt.title('Plot of Signal Envelope, E', fontsize=35)

            plt.subplot(235)
            sns.heatmap(lapk5, cmap='seismic', center=True, robust=True, fmt='d', lin
                        yticklabels=False, xticklabels=False, cbar=False)
            plt.title('Plot of Curvature', fontsize=35)

            plt.subplot(236)
            sns.heatmap(lapdEe5, cmap='seismic', center=True, robust=True, fmt='d', l
                        yticklabels=False, xticklabels=False, cbar=False)
            plt.title('Plot of 2nd Derivative of E', fontsize=35)
```

```
C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\ma\core.py:3095: ComplexWarnin
  output = self._data.astype(newtype).view(type(self))
```

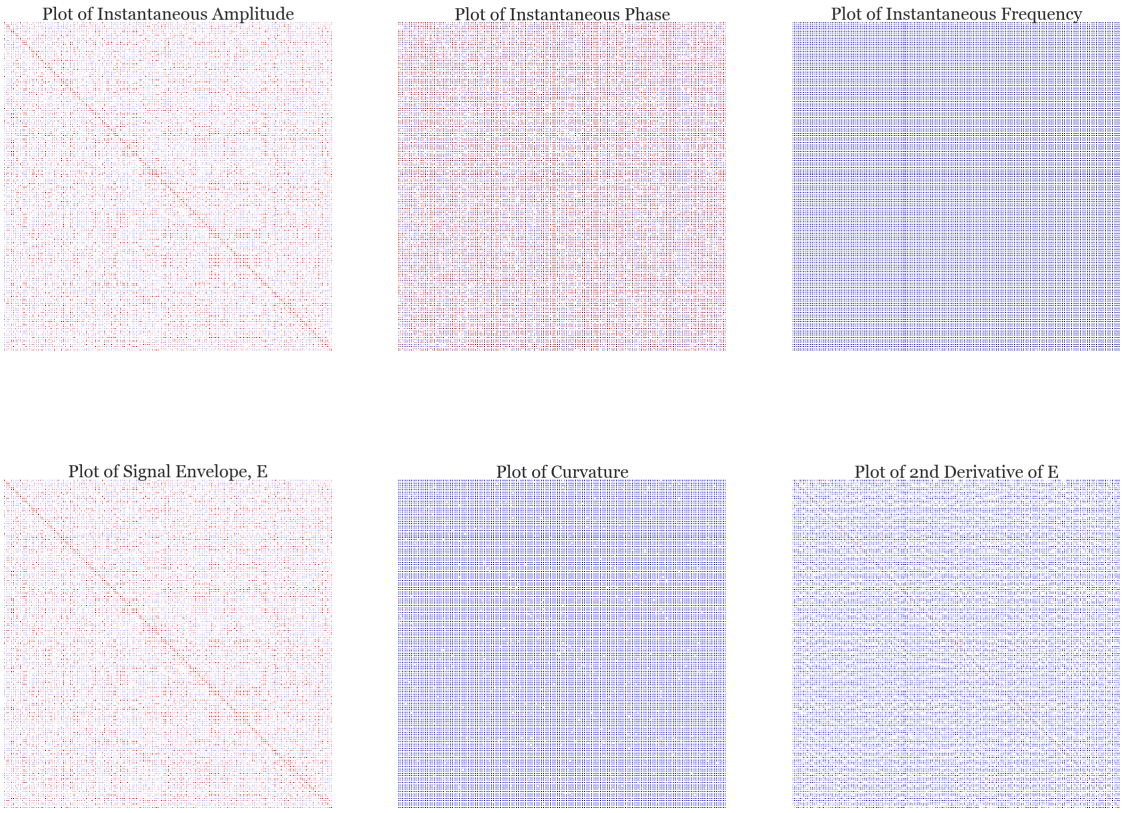
```
Out[182]: <matplotlib.text.Text at 0x2275c9f2a58>
```

Heatmap of IA, IP, IF, E, dE, dEe of the Laplaciany Matrix, t5



```
In [183]: plt.figure(figsize=(40, 30))

          plt.subplot(231)
          sns.heatmap(modIA5,cmap='seismic', center=True, robust=True, fmt='d', lin
                      yticklabels=False, xticklabels=False, cbar=False)

          plt.title('Plot of Instantaneous Amplitude', fontsize=35)
          plt.suptitle('Heatmap of IA, IP, IF, E, dE, dEe of the Adjacency Matrix,

          plt.subplot(232)
          sns.heatmap(modIP5, cmap='seismic', center=True, robust=True, fmt='d', li
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Instantaneous Phase', fontsize=35)

          plt.subplot(233)
          sns.heatmap(modIF5, cmap='seismic', center=True, robust=True, fmt='d', li
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Instantaneous Frequency', fontsize=35)

          plt.subplot(234)
          sns.heatmap(modE5, cmap='seismic', center=True, robust=True, fmt='d', lin
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Signal Envelope, E', fontsize=35)

          plt.subplot(235)
          sns.heatmap(modk5, cmap='seismic', center=True, robust=True, fmt='d', lin
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of Curvature', fontsize=35)

          plt.subplot(236)
          sns.heatmap(moddEe5, cmap='seismic', center=True, robust=True, fmt='d', l
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('Plot of 2nd Derivative of E', fontsize=35)

C:\Users\arsha_000\Anaconda3\lib\site-packages\numpy\ma\core.py:3095: ComplexWarnin
  output = self._data.astype(newtype).view(type(self))


Out[183]: <matplotlib.text.Text at 0x2275d247cc0>
```

Heatmap of IA, IP, IF, E, dE, dEe of the Adjacency Matrix, t5



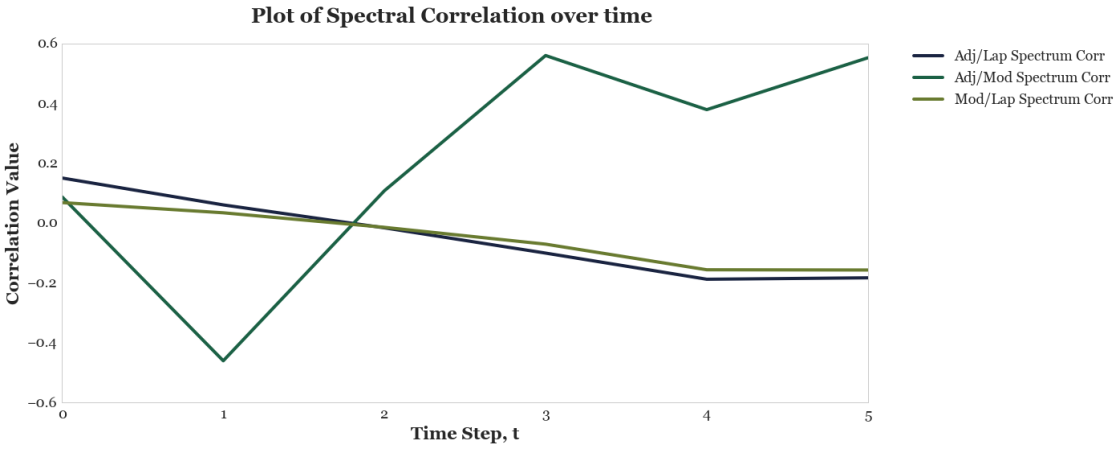
15 Spectral Correlation

```
In [184]: adj_spec_df = pd.DataFrame([adj_spec0,adj_spec1,adj_spec2,adj_spec3,adj_s
          lap_spec_df = pd.DataFrame([lap_spec0,lap_spec1,lap_spec2,lap_spec3,lap_s
          mod_spec_df = pd.DataFrame([mod_spec0,mod_spec1,mod_spec2,mod_spec3,mod_s
          adj_spec_df.columns = ['t0','t1','t2','t3','t4','t5']
          lap_spec_df.columns = ['t0','t1','t2','t3','t4','t5']
          mod_spec_df.columns = ['t0','t1','t2','t3','t4','t5']

In [185]: adj_spec_df.corrwith(lap_spec_df).values

Out[185]: array([ 0.15026144 -6.40824059e-20j,  0.06045287 -6.45250119e-20j,
                 -0.01622352 -2.27026950e-19j, -0.10064998 -3.51853695e-20j,
                 -0.18800205 +0.00000000e+00j, -0.18325464 +0.00000000e+00j])

In [186]: plt.figure(figsize=(18,8))
          plt.plot(adj_spec_df.corrwith(lap_spec_df).values, label= 'Adj/Lap Spectr
          plt.plot(adj_spec_df.corrwith(mod_spec_df).values, label= 'Adj/Mod Spectr
          plt.plot(mod_spec_df.corrwith(lap_spec_df).values, label= 'Mod/Lap Spectr
```

```
        plt.suptitle('Plot of Spectral Correlation over time', fontsize=26)
        plt.yticks(fontsize=18)
        plt.xticks(fontsize=18)
        plt.legend(fontsize=18, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
        plt.xlabel("Time Step, t", fontsize=22)
        plt.ylabel("Correlation Value", fontsize=22)
```

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

```
Out[186]: <matplotlib.text.Text at 0x2275d0910f0>
```



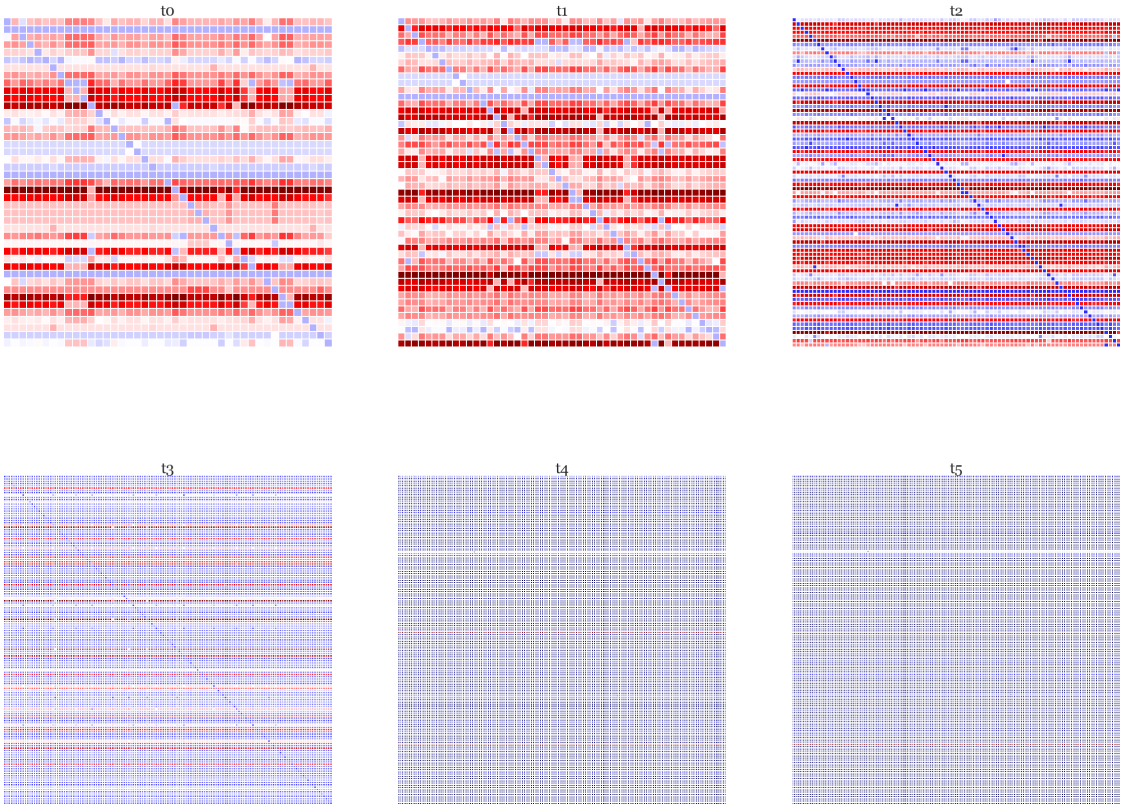
16 Resistance Distance

The resistance distance on a graph is discussed here and the code is adapted from the pygsp module here

```
In [187]: #cite:`klein1993resistance`
          def resistance_distance(M):
              pseudo = pinv(M)
              N = M.shape[0]
              d = np.diag(pseudo)
              rd  = np.kron(d,np.ones((N,1))).T+np.kron(d,np.ones((N,1))).T - pseud

              return rd
```

```
In [188]: res0 = resistance_distance(lapM0)
          res1 = resistance_distance(lapM1)
          res2 = resistance_distance(lapM2)
          res3 = resistance_distance(lapM3)
          res4 = resistance_distance(lapM4)
          res5 = resistance_distance(lapM5)

In [189]: K_res0,meanK_res0,conc_res0 = curvature(resistance_distance(lapM0))
          K_res1,meanK_res1,conc_res1 = curvature(resistance_distance(lapM1))
          K_res2,meanK_res2,conc_res2 = curvature(resistance_distance(lapM2))
          K_res3,meanK_res3,conc_res3 = curvature(resistance_distance(lapM3))
          K_res4,meanK_res4,conc_res4 = curvature(resistance_distance(lapM4))
          K_res5,meanK_res5,conc_res5 = curvature(resistance_distance(lapM5))

In [190]: plt.figure(figsize=(40, 30))

          plt.subplot(231)
          sns.heatmap(res0,cmap='seismic', center=True, robust=True, fmt='d', linew
                      yticklabels=False, xticklabels=False, cbar=False)

          plt.suptitle('Heatmap of Resistance Distance Matrix, t0-t5', fontsize=60)
          plt.title('t0', fontsize=30)

          plt.subplot(232)
          sns.heatmap(res1,cmap='seismic', center=True, robust=True, fmt='d', linew
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('t1', fontsize=30)

          plt.subplot(233)
          sns.heatmap(res2,cmap='seismic', center=True, robust=True, fmt='d', linew
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('t2', fontsize=30)

          plt.subplot(234)
          sns.heatmap(res3,cmap='seismic', center=True, robust=True, fmt='d', linew
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('t3', fontsize=30)

          plt.subplot(235)
          sns.heatmap(res4,cmap='seismic', center=True, robust=True, fmt='d', linew
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('t4', fontsize=30)

          plt.subplot(236)
          sns.heatmap(res5,cmap='seismic', center=True, robust=True, fmt='d', linew
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('t5', fontsize=30)

Out[190]: <matplotlib.text.Text at 0x22760677940>
```

Heatmap of Resistance Distance Matrix, t0-t5

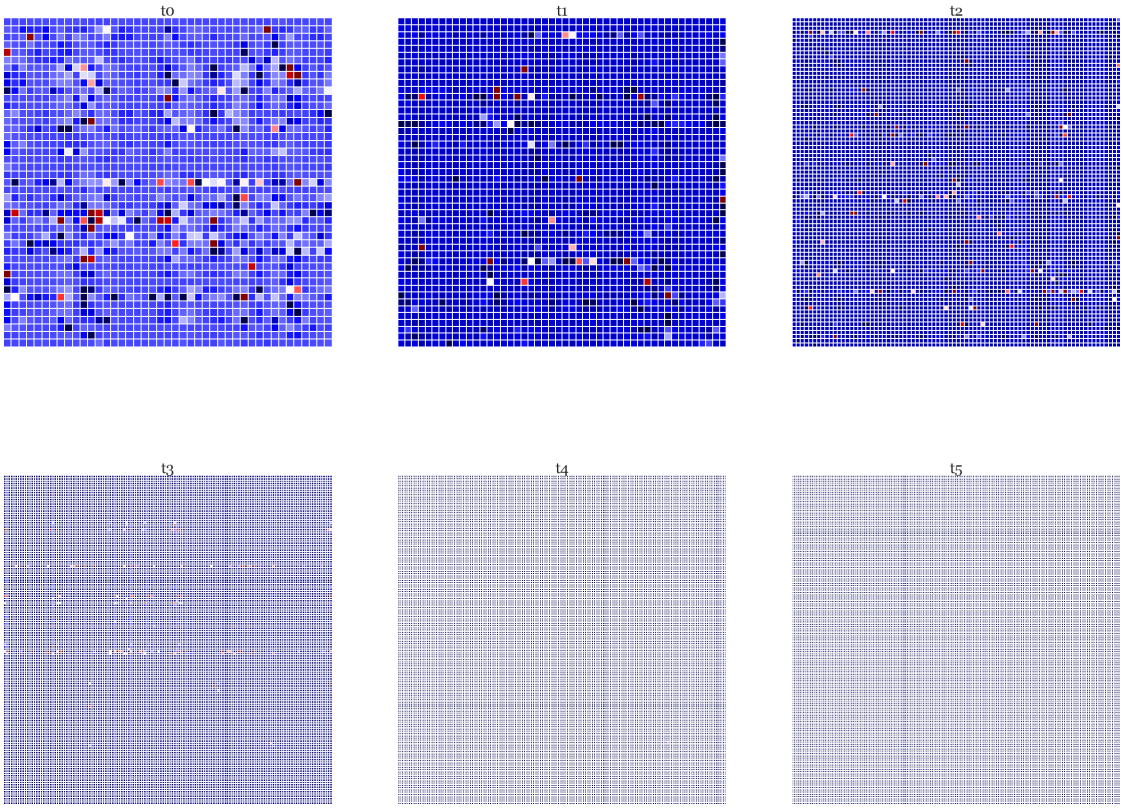



```
In [191]: plt.figure(figsize=(40, 30))

          plt.subplot(231)
          sns.heatmap(K_res0,cmap='seismic', center=True, robust=True, fmt='d', lin
                      yticklabels=False, xticklabels=False, cbar=False)

          plt.suptitle('Heatmap of Curvature of Resistance Distance Matrix, t0-t5',
          plt.title('t0', fontsize=30)

          plt.subplot(232)
          sns.heatmap(K_res1,cmap='seismic', center=True, robust=True, fmt='d', lin
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('t1', fontsize=30)

          plt.subplot(233)
          sns.heatmap(K_res2,cmap='seismic', center=True, robust=True, fmt='d', lin
                      yticklabels=False, xticklabels=False, cbar=False)
          plt.title('t2', fontsize=30)
```

```
plt.subplot(234)
sns.heatmap(K_res3,cmap='seismic', center=True, robust=True, fmt='d', lin
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t3', fontsize=30)

plt.subplot(235)
sns.heatmap(K_res4,cmap='seismic', center=True, robust=True, fmt='d', lin
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t4', fontsize=30)

plt.subplot(236)
sns.heatmap(K_res5,cmap='seismic', center=True, robust=True, fmt='d', lin
            yticklabels=False, xticklabels=False, cbar=False)
plt.title('t5', fontsize=30)
```

```
Out[191]: <matplotlib.text.Text at 0x22760a75c50>
```

Heatmap of Curvature of Resistance Distance Matrix, t0-t5

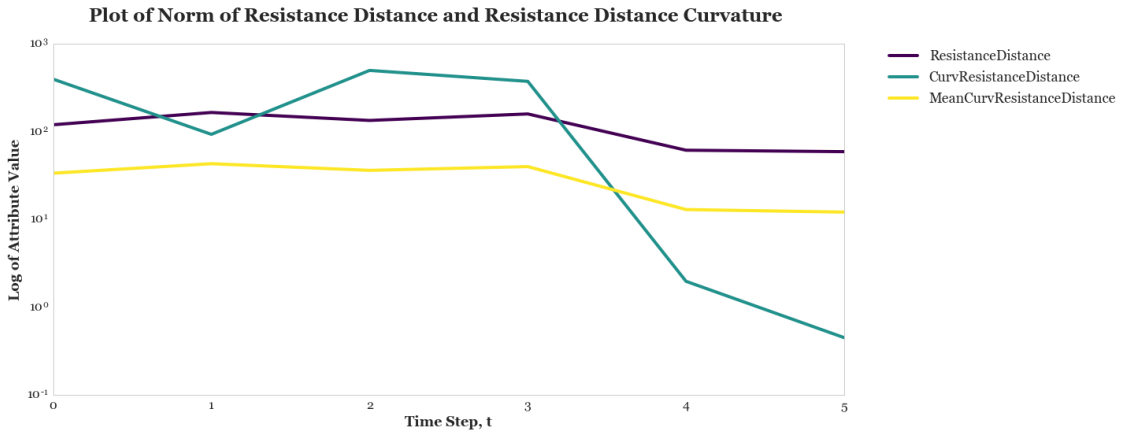



```
In [192]: res_dist_df = pd.DataFrame([norm(res0),norm(res1),norm(res2),\
                                     norm(res3),norm(res4),norm(res5)],columns=['R
```

```
        res_dist_df['CurvResistanceDistance'] = pd.DataFrame([norm(K_res0),norm(K
                                    norm(K_res3),norm(K_res4),norm(K_res5)])
        res_dist_df['MeanCurvResistanceDistance'] = pd.DataFrame([meanK_res0,mean
```

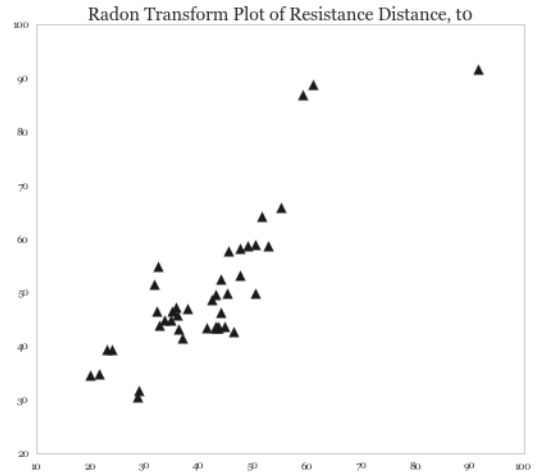
In [193]:
```
res_dist_df.plot(colormap='viridis', fontsize=16, figsize=(18,8), logy=Tr
plt.suptitle('Plot of Norm of Resistance Distance and Resistance Distance
plt.yticks(fontsize=16)
plt.xticks(fontsize=16)
plt.legend(fontsize=18, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
plt.xlabel("Time Step, t", fontsize=18)
plt.ylabel("Log of Attribute Value", fontsize=18)
```

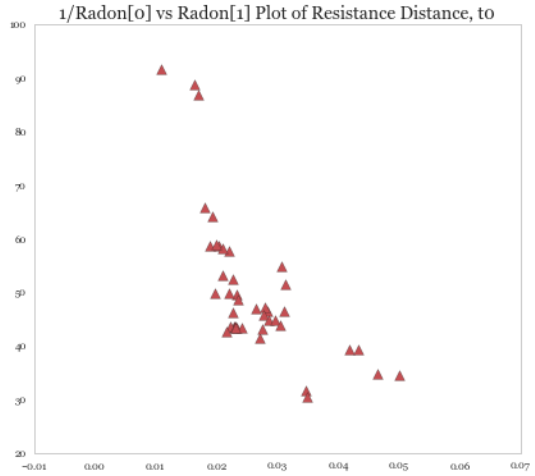
Out[193]:
```
<matplotlib.text.Text at 0x22760a2d390>
```



In [195]:
```
plot_radon(res0,'Resistance Distance, t0')
```

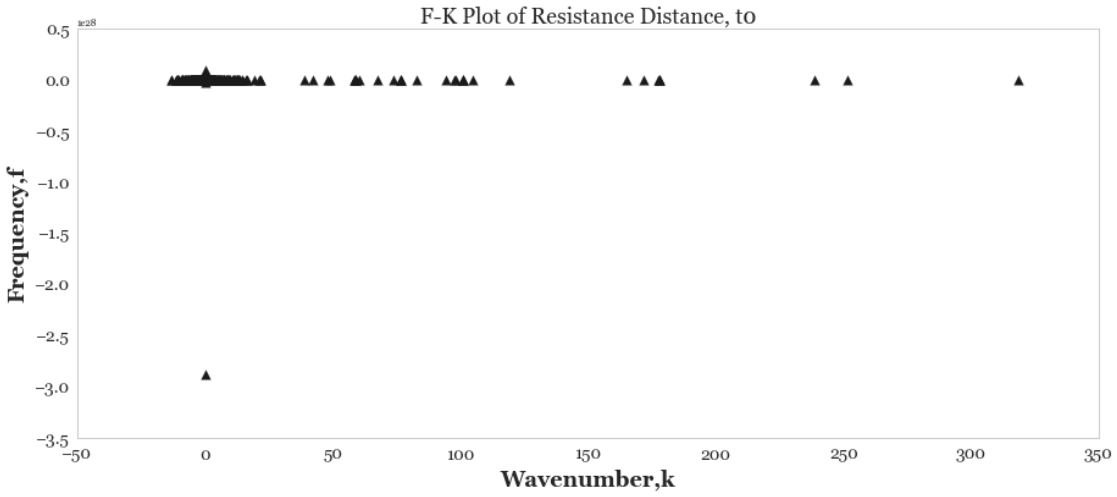
```
C:\Users\arsha_000\Anaconda3\lib\site-packages\skimage\transform\radon_transform.py
  warn('Radon transform: image must be zero outside the '
```





In [196]: _,_ =fk_plot(res0,'Resistance Distance, t0')

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



In [197]:
```
_,_,_,_,_,_,resglob0 = calc_seisatt(res0)
_,_,_,_,_,_,resglob1 = calc_seisatt(res1)
_,_,_,_,_,_,resglob2 = calc_seisatt(res2)
_,_,_,_,_,_,resglob3 = calc_seisatt(res3)
_,_,_,_,_,_,resglob4 = calc_seisatt(res4)
_,_,_,_,_,_,resglob5 = calc_seisatt(res5)
```

In [198]:
```
resglob0
```

Out[198]:

	0	1	2	3	4	5
0	849.960956	38.277122	14.135791	849.960956	267.523732	261.228304

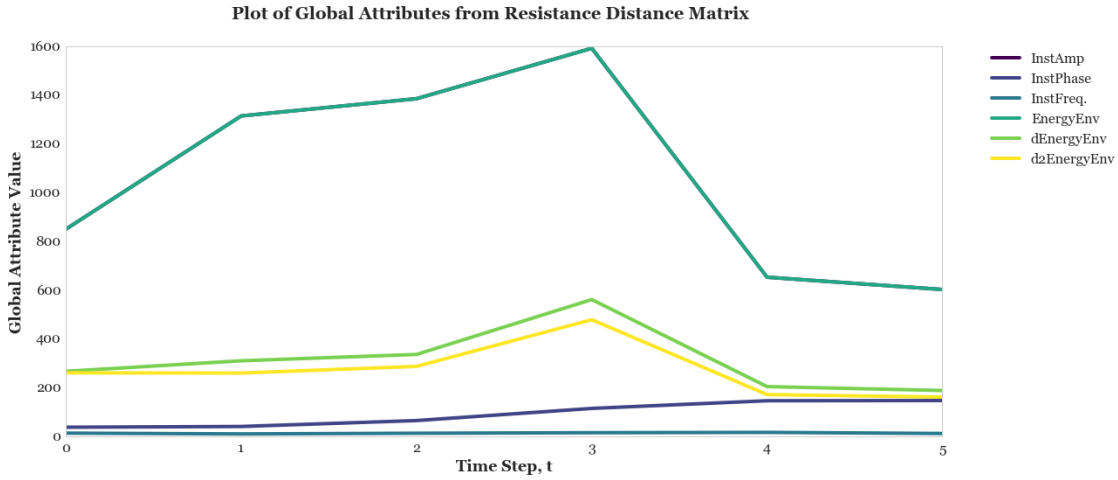
In [199]:
```
resglob_df = resglob0.append(resglob1).append(resglob2).append(resglob3).
resglob_df.columns = ['InstAmp','InstPhase','InstFreq.','EnergyEnv','dEne
#resglob_df.set_index([['t0','t1','t2','t3','t4','t5']], inplace=True)
resglob_df.set_index([[0,1,2,3,4,5]], inplace=True)
resglob_df
```

Out[199]:

	InstAmp	InstPhase	InstFreq.	EnergyEnv	dEnergyEnv	d2EnergyE
0	849.960956	38.277122	14.135791	849.960956	267.523732	261.2283
1	1314.061767	41.350631	10.841176	1314.061767	310.622418	260.3317
2	1384.761205	65.558390	13.683360	1384.761205	336.578783	287.6565
3	1591.304348	115.425732	15.792047	1591.304348	561.512110	478.9603
4	653.086537	146.862649	17.142852	653.086537	204.843474	172.1474
5	602.803361	147.705531	12.664414	602.803361	188.825256	161.9090

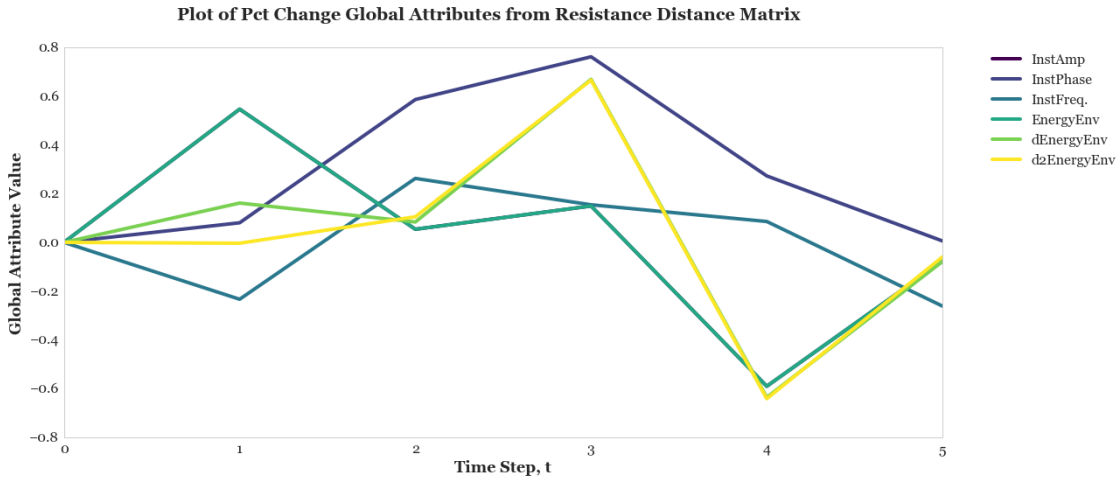
```
In [200]: resglob_df.plot(colormap='viridis', fontsize=16, figsize=(18,8))
          plt.suptitle('Plot of Global Attributes from Resistance Distance Matrix',
          plt.yticks(fontsize=16)
          plt.xticks(fontsize=16)
          plt.xlabel("Time Step, t", fontsize=18)
          plt.legend(fontsize=16, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
          plt.ylabel("Global Attribute Value", fontsize=18)
```

```
Out[200]: <matplotlib.text.Text at 0x22764c95128>
```




```
In [201]: resglob_df.pct_change().fillna(0).plot(colormap='viridis', fontsize=16, f
          plt.suptitle('Plot of Pct Change Global Attributes from Resistance Distar
          plt.yticks(fontsize=16)
          plt.xticks(fontsize=16)
          plt.xlabel("Time Step, t", fontsize=18)
          plt.legend(fontsize=16, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
          plt.ylabel("Global Attribute Value", fontsize=18)
```

```
Out[201]: <matplotlib.text.Text at 0x22764d2b588>
```



17 Towards Attribute Volumes

** In this step I merge all the attributes derived so far to create an attribute volume**

In [202]:
```
#basic network statistics
stat_df2
```

Out[202]:

	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	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	
5	116.304348	0.049240	0.270539	0.005834	0.059275	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
5	0.491793	6.803237e+12	0.007059	0.005833	0.131623

In [203]:
```
#stationarity statistics
stationarity_df['Zeta']=zeta
stationarity_df['Corr,Ct']=Ct_df
stationarity_df
```

Out[203]:

	LapStatRat	ModStatRat	AdjStatRat	Zeta	Corr,Ct
0	0.885557	0.328867	0.156174	0.141421	0.707107
1	0.906522	0.278447	0.136083	0.273887	0.662329
2	0.944330	0.191400	0.113592	0.423730	0.749213
3	0.974044	0.187360	0.025516	0.506182	0.412260
4	0.984701	0.166550	0.016042	0.613034	0.534260
5	0.985587	0.165496	0.026745	0.636966	0.119659

In [204]:
```
resglob_df
```

Out[204]:

	InstAmp	InstPhase	InstFreq.	EnergyEnv	dEnergyEnv	d2EnergyE
0	849.960956	38.277122	14.135791	849.960956	267.523732	261.2283
1	1314.061767	41.350631	10.841176	1314.061767	310.622418	260.3317
2	1384.761205	65.558390	13.683360	1384.761205	336.578783	287.6565
3	1591.304348	115.425732	15.792047	1591.304348	561.512110	478.9603
4	653.086537	146.862649	17.142852	653.086537	204.843474	172.1474
5	602.803361	147.705531	12.664414	602.803361	188.825256	161.9090

In [205]:
```
#Seismic attributes
seis_att_df = lapglob_df.join(adjglob_df,rsuffix='Adj')
seis_att_df =seis_att_df .join(modglob_df,rsuffix='Mod')
seis_att_df =seis_att_df.join(resglob_df,rsuffix='ResDist')
seis_att_df
```

Out[205]:

	InstAmp	InstPhase	InstFreq.	EnergyEnv	dEnergyEnv	d2Energy
0	73.014790	63.250000	41.899990	73.014790	48.708167	42.841
1	95.697029	69.973063	53.056054	95.697029	58.760915	53.666
2	201.848840	116.820044	85.648821	201.848840	121.889250	104.394
3	680.947797	216.750295	154.955039	680.947797	412.506571	362.394
4	1408.349343	273.018131	194.726052	1408.349343	789.564513	678.617
5	1492.873886	272.894377	192.633444	1492.873886	842.273584	723.355

	InstAmpAdj	InstPhaseAdj	InstFreq.Adj	EnergyEnvAdj	...
0	38.186710	66.008877	49.881368	38.186710	...
1	46.880508	69.358658	50.101436	46.880508	...
2	94.970176	109.992126	80.671227	94.970176	...
3	293.054551	216.464161	154.266190	293.054551	...
4	563.036254	273.727614	190.671908	563.036254	...
5	597.654978	272.750308	182.396382	597.654978	...

	InstFreq.Mod	EnergyEnvMod	dEnergyEnvMod	d2EnergyEnvMod	InstAmpResD
0	13.205153	37.664283	24.876889	21.794054	849.960
1	12.813470	45.417822	28.570743	24.707563	1314.061
2	19.751385	84.616857	51.404777	43.907761	1384.761
3	36.810706	233.408611	140.862499	122.362069	1591.304
4	48.779973	416.720939	245.597684	212.721903	653.086
5	48.944407	429.139810	252.555749	218.738497	602.803

	InstPhaseResDist	InstFreq.ResDist	EnergyEnvResDist	dEnergyEnvResDis
0	38.277122	14.135791	849.960956	267.52373
1	41.350631	10.841176	1314.061767	310.62241
2	65.558390	13.683360	1384.761205	336.57878
3	115.425732	15.792047	1591.304348	561.51211
4	146.862649	17.142852	653.086537	204.84347
5	147.705531	12.664414	602.803361	188.82525

	d2EnergyEnvResDist
0	261.228304

```
1          260.331721
2          287.656569
3          478.960323
4          172.147425
5          161.909075

[6 rows x 24 columns]
```

In [206]:
```
#curvature statistics
all_curv_df = MeanCurv_df.join(res_dist_df)
all_curv_df
```

Out[206]:

	AdjMeanCurv	LapMeanCurv	ModMeanCurv	ResistanceDistance	CurvResistanceDistance	MeanCurvResistanceDistance
0	0.716053	-0.588417	1.063719	119.102952	393.750463	33.467232
1	0.119925	-0.440728	0.433750	164.317478	92.641620	42.914155
2	0.883253	-0.703370	1.255881	133.370416	494.637269	36.047245
3	0.424695	-3.086177	1.024033	158.161756	371.517314	39.783842
4	1.023161	-4.312743	1.878105	61.134708	1.968253	12.903944
5	1.024451	-5.456495	1.932800	58.842733	0.449319	12.067692

In [207]:
```
att_vol = stat_df2.join(stationarity_df).join(all_curv_df).join(seis_att_
```

In [208]:
```
att_vol.shape
```

Out[208]:
```
(6, 50)
```

In [209]:
```
att_vol.T
```

Out[209]:

	0	1	2
DegAssorCoeff	-0.251264	-0.226108	-0.178525
DegPearCC	-0.251264	-0.226108	-0.178525
Avg(AvgNeighDeg)	2.190131	2.896955	5.563177
Avg(AvgDegConnect)	3.600092	4.906208	9.037252
AvgTriangles	0.023256	0.125000	2.762500
Deg	0.018826	0.021720	0.020570
Closeness	0.104661	0.165501	0.223937
Betweeness	0.025984	0.054367	0.025495
Eig	0.077739	0.089506	0.066241
AlgConnect	0.000000	0.000000	0.000000
ClustCoeff	0.026004	0.087087	0.185696
Communicability	3.980933	5.874870	120.611300

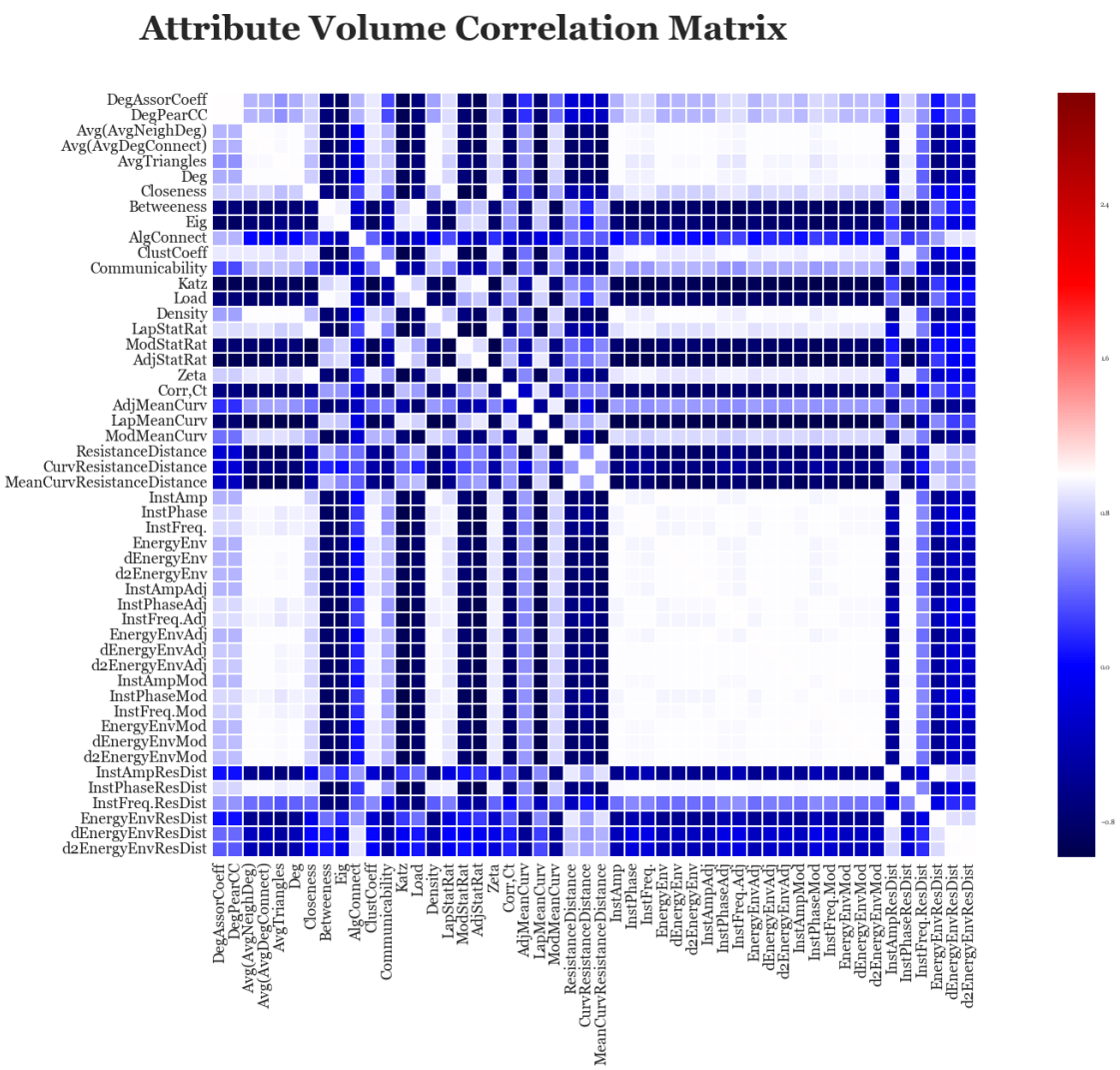
Katz	0.125240	0.116543	0.076152
Load	0.025984	0.054367	0.025495
Density	0.057586	0.057624	0.054430
LapStatRat	0.885557	0.906522	0.944330
ModStatRat	0.328867	0.278447	0.191400
AdjStatRat	0.156174	0.136083	0.113592
Zeta	0.141421	0.273887	0.423730
Corr,Ct	0.707107	0.662329	0.749213
AdjMeanCurv	0.716053	0.119925	0.883253
LapMeanCurv	-0.588417	-0.440728	-0.703370
ModMeanCurv	1.063719	0.433750	1.255881
ResistanceDistance	119.102952	164.317478	133.370416
CurvResistanceDistance	393.750463	92.641620	494.637269
MeanCurvResistanceDistance	33.467232	42.914155	36.047245
InstAmp	73.014790	95.697029	201.848840
InstPhase	63.250000	69.973063	116.820044
InstFreq.	41.899990	53.056054	85.648821
EnergyEnv	73.014790	95.697029	201.848840
dEnergyEnv	48.708167	58.760915	121.889250
d2EnergyEnv	42.841833	53.666455	104.394827
InstAmpAdj	38.186710	46.880508	94.970176
InstPhaseAdj	66.008877	69.358658	109.992126
InstFreq.Adj	49.881368	50.101436	80.671227
EnergyEnvAdj	38.186710	46.880508	94.970176
dEnergyEnvAdj	20.977247	24.082062	45.252526
d2EnergyEnvAdj	18.404320	20.244747	38.358102
InstAmpMod	37.664283	45.417822	84.616857
InstPhaseMod	54.509072	59.196431	101.269200
InstFreq.Mod	13.205153	12.813470	19.751385
EnergyEnvMod	37.664283	45.417822	84.616857
dEnergyEnvMod	24.876889	28.570743	51.404777
d2EnergyEnvMod	21.794054	24.707563	43.907761
InstAmpResDist	849.960956	1314.061767	1384.761205
InstPhaseResDist	38.277122	41.350631	65.558390
InstFreq.ResDist	14.135791	10.841176	13.683360
EnergyEnvResDist	849.960956	1314.061767	1384.761205
dEnergyEnvResDist	267.523732	310.622418	336.578783
d2EnergyEnvResDist	261.228304	260.331721	287.656569

	3	4	5
DegAssorCoeff	7.486954e-02	-4.077003e-02	-4.659182e-02
DegPearCC	7.486954e-02	-4.077003e-02	-4.659182e-02
Avg(AvgNeighDeg)	1.633236e+01	3.115903e+01	3.287352e+01
Avg(AvgDegConnect)	1.741208e+01	3.221128e+01	3.436610e+01
AvgTriangles	2.827273e+01	1.041803e+02	1.163043e+02
Deg	3.181326e-02	4.788927e-02	4.923972e-02
Closeness	2.535849e-01	2.760215e-01	2.705391e-01
Betweeness	1.273913e-02	6.151574e-03	5.833785e-03

```
Eig                              5.647762e-02  5.878076e-02  5.927522e-02
AlgConnect                       1.219154e-01  0.000000e+00  0.000000e+00
ClustCoeff                       4.625440e-01  4.937170e-01  4.917930e-01
Communicability                  6.140791e+06  1.009380e+12  6.803237e+12
Katz                             1.170323e-02  3.814843e-03  7.059498e-03
Load                             1.273913e-02  6.151059e-03  5.833311e-03
Density                          8.165074e-02  1.234012e-01  1.316227e-01
LapStatRat                       9.740444e-01  9.847012e-01  9.855872e-01
ModStatRat                       1.873597e-01  1.665504e-01  1.654964e-01
AdjStatRat                       2.551552e-02  1.604163e-02  2.674523e-02
Zeta                             5.061818e-01  6.130337e-01  6.369656e-01
Corr,Ct                          4.122601e-01  5.342597e-01  1.196594e-01
AdjMeanCurv                      4.246953e-01  1.023161e+00  1.024451e+00
LapMeanCurv                     -3.086177e+00 -4.312743e+00 -5.456495e+00
ModMeanCurv                      1.024033e+00  1.878105e+00  1.932800e+00
ResistanceDistance               1.581618e+02  6.113471e+01  5.884273e+01
CurvResistanceDistance           3.715173e+02  1.968253e+00  4.493193e-01
MeanCurvResistanceDistance       3.978384e+01  1.290394e+01  1.206769e+01
InstAmp                          6.809478e+02  1.408349e+03  1.492874e+03
InstPhase                        2.167503e+02  2.730181e+02  2.728944e+02
InstFreq.                        1.549550e+02  1.947261e+02  1.926334e+02
EnergyEnv                        6.809478e+02  1.408349e+03  1.492874e+03
dEnergyEnv                       4.125066e+02  7.895645e+02  8.422736e+02
d2EnergyEnv                      3.623941e+02  6.786173e+02  7.233551e+02
InstAmpAdj                       2.930546e+02  5.630363e+02  5.976550e+02
InstPhaseAdj                     2.164642e+02  2.737276e+02  2.727503e+02
InstFreq.Adj                     1.542662e+02  1.906719e+02  1.823964e+02
EnergyEnvAdj                     2.930546e+02  5.630363e+02  5.976550e+02
dEnergyEnvAdj                    1.302651e+02  1.971906e+02  2.014625e+02
d2EnergyEnvAdj                   1.137534e+02  1.702393e+02  1.743876e+02
InstAmpMod                       2.334086e+02  4.167209e+02  4.291398e+02
InstPhaseMod                     1.803275e+02  2.219085e+02  2.212457e+02
InstFreq.Mod                     3.681071e+01  4.877997e+01  4.894441e+01
EnergyEnvMod                     2.334086e+02  4.167209e+02  4.291398e+02
dEnergyEnvMod                    1.408625e+02  2.455977e+02  2.525557e+02
d2EnergyEnvMod                   1.223621e+02  2.127219e+02  2.187385e+02
InstAmpResDist                   1.591304e+03  6.530865e+02  6.028034e+02
InstPhaseResDist                 1.154257e+02  1.468626e+02  1.477055e+02
InstFreq.ResDist                 1.579205e+01  1.714285e+01  1.266441e+01
EnergyEnvResDist                 1.591304e+03  6.530865e+02  6.028034e+02
dEnergyEnvResDist                5.615121e+02  2.048435e+02  1.888253e+02
d2EnergyEnvResDist               4.789603e+02  1.721474e+02  1.619091e+02
```

In [210]:
```
plt.figure(figsize=(34,16))
sns.heatmap(att_vol.corr(), cmap='seismic', center=True, robust=True, fmt
plt.suptitle("Attribute Volume Correlation Matrix", fontsize=40)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
```

```
Out[210]: (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,  15.5,  16.5,  17.5,
                   18.5,  19.5,  20.5,  21.5,  22.5,  23.5,  24.5,  25.5,  26.5,
                   27.5,  28.5,  29.5,  30.5,  31.5,  32.5,  33.5,  34.5,  35.5,
                   36.5,  37.5,  38.5,  39.5,  40.5,  41.5,  42.5,  43.5,  44.5,
                   45.5,  46.5,  47.5,  48.5,  49.5]),
           <a list of 50 Text yticklabel objects>)
```



17.1 Persistence & Emergence on Attribute Volume

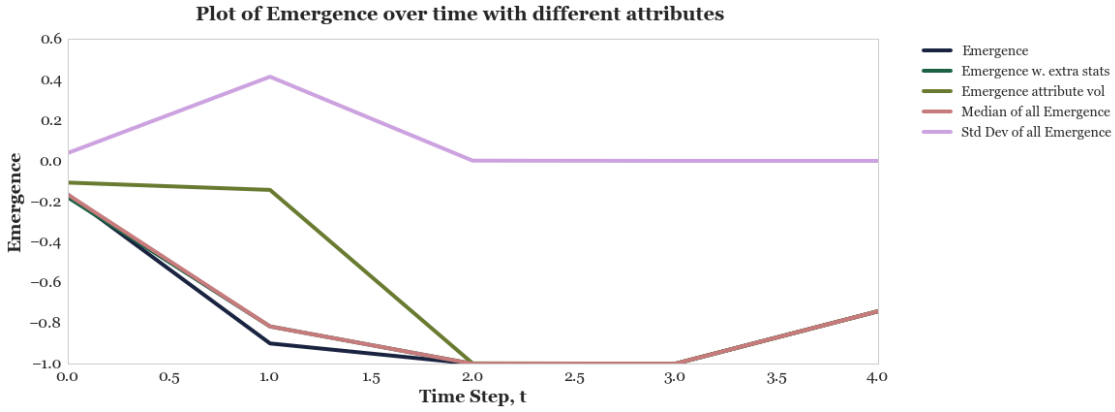
```
In [211]: persist3 = att_vol.mean(axis=1).apply(lambda x: x/5).values
```

```
In [212]: emerg3 = []
          for i in range(0,5):
              x = int(i)
```

```
            y = x +1
            emerg3.append(emergence(persist3[x],persist3[y]))
```

In [250]:
```
plt.plot(emerg, label= 'Emergence')
plt.plot(emerg2, label='Emergence w. extra stats')
plt.plot(emerg3, label='Emergence attribute vol')
plt.plot(pd.DataFrame([emerg,emerg2,emerg3]).T.median(axis=1).values,labe
plt.plot(pd.DataFrame([emerg,emerg2,emerg3]).T.std(axis=1).values,label='
plt.suptitle('Plot of Emergence over time with different attributes', for
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("Emergence", fontsize=18)
```

Out[250]: <matplotlib.text.Text at 0x227684ccb70>



In [ ]: