DNA_06

September 4, 2016

1 Table of Contents

2 Introduction

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.

Here I consolidate earlier analysis to make presentation of the final results easier and more readable.

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 the required level of granularity

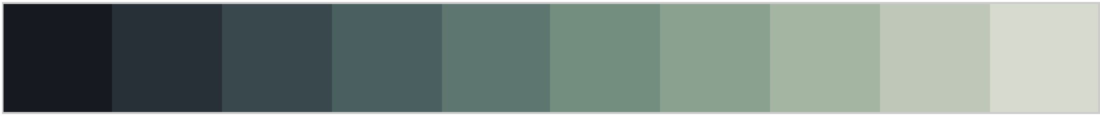
```
In [1]: import pandas as pd
        import numpy as np
        import networkx as nx
        import seaborn as sns
        import scipy as sc
        import random
        from scipy.signal import *
        from numpy.linalg import *
        from sklearn.decomposition import *
        from sklearn.preprocessing import *
        import librosa
        import abel
        #plotting parameters
        %matplotlib inline
        sns.set(style="whitegrid", color_codes=True, context='paper')
```

```
In [221]: from matplotlib import rcParams
          rcParams['font.family'] = 'serif'
          rcParams['font.sans-serif'] = ['CMU Serif']
```

```
In [3]: import matplotlib.pyplot as plt
```

```
In [222]: plt.rc('axes', grid=False, titlesize='large', labelsize='medium',labelwei
          plt.rc('lines', linewidth=4)
          plt.rc('figure', figsize = (8,6),titlesize='large',titleweight='black')
          plt.rc('font', weight='medium', size=9)
          plt.rc('grid',linewidth=3)
```

```
In [5]: sns.palplot(sns.cubehelix_palette(10,hue=0.3, reverse=True, rot=-0.55, dark
```



```
In [6]: sns.set_palette(sns.cubehelix_palette(10, hue=0.3, reverse=True, rot=-0.55,
```

```
In [7]: data = pd.read_excel("../Data/data 03.2.xlsx")
```

```
In [8]: years = sorted(set(data.year))
        years = years[1:]
        years
```

```
Out[8]: [1998, 1999, 2000, 2001, 2002]
```

```
In [9]: months = ['Nov98', 'Dec98', 'jan_99', 'feb_99', 'mar_99', 'apr_99', 'may_9
                  'nov_99', 'dec_99', 'jan_2k', 'feb_2k', 'mar_2k', 'apr_2k', 'may_
                  'nov_2k', 'dec_2k', 'jan_2k1', 'feb_2k1', 'mar_2k1', 'apr_2k1', '
                  'oct_2k1', 'nov_2k1', 'dec_2k1', 'jan_2k2', 'feb_2k2', 'mar_2k2',
```

```
In [10]: df_98 = data[data.year==years[0]]
         df_99 = data[data.year==years[1]]
         df_2k = data[data.year==years[2]]
         df_2k1 = data[data.year==years[3]]
         df_2k2 = data[data.year==years[4]]
```

```
In [11]: def create_graph(df):
             tmp = df.values[:,1:3]
             G= nx.Graph()
             G = nx.from_edgelist(tmp)

             return G
```

```
In [12]: Gt0 = create_graph(df_98)
         Gt1 = create_graph(df_99)
         Gt2 = create_graph(df_2k)
         Gt3 = create_graph(df_2k1)
         Gt4 = create_graph(df_2k2)
```

```
In [13]: nov_98 = df_98[df_98.month==11]
         dec_98= df_98[df_98.month==12]

         G_nov98 = create_graph(nov_98)
         G_dec98 = create_graph(dec_98)
```

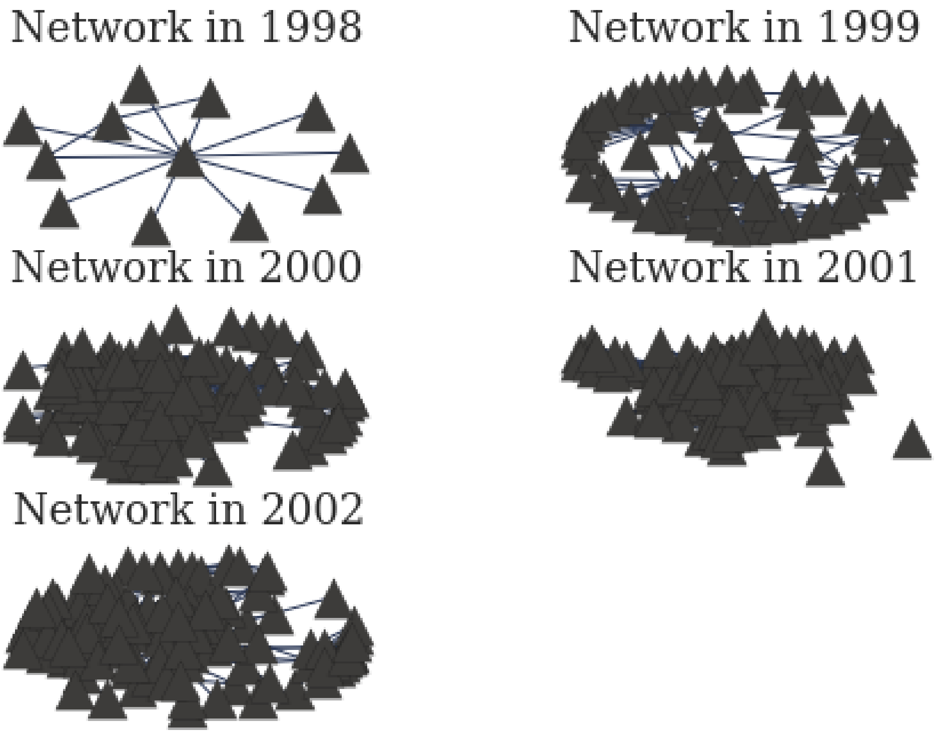
```
In [14]: jan_99=df_99[df_99.month==1]
         feb_99=df_99[df_99.month==2]
         mar_99=df_99[df_99.month==3]
         apr_99=df_99[df_99.month==4]
         may_99=df_99[df_99.month==5]
         jun_99=df_99[df_99.month==6]
         jul_99=df_99[df_99.month==7]
         aug_99=df_99[df_99.month==8]
         sep_99=df_99[df_99.month==9]
         oct_99=df_99[df_99.month==10]
         nov_99=df_99[df_99.month==11]
         dec_99=df_99[df_99.month==12]


         G_jan_99=create_graph(jan_99)
         G_feb_99=create_graph(feb_99)
         G_mar_99=create_graph(mar_99)
         G_apr_99=create_graph(apr_99)
         G_may_99=create_graph(may_99)
         G_jun_99=create_graph(jun_99)
         G_jul_99=create_graph(jul_99)
         G_aug_99=create_graph(aug_99)
         G_sep_99=create_graph(sep_99)
         G_oct_99=create_graph(oct_99)
         G_nov_99=create_graph(nov_99)
         G_dec_99=create_graph(dec_99)

In [15]: jan_2k=df_2k[df_2k.month==1]
         feb_2k=df_2k[df_2k.month==2]
         mar_2k=df_2k[df_2k.month==3]
         apr_2k=df_2k[df_2k.month==4]
         may_2k=df_2k[df_2k.month==5]
         jun_2k=df_2k[df_2k.month==6]
         jul_2k=df_2k[df_2k.month==7]
         aug_2k=df_2k[df_2k.month==8]
         sep_2k=df_2k[df_2k.month==9]
         oct_2k=df_2k[df_2k.month==10]
         nov_2k=df_2k[df_2k.month==11]
         dec_2k=df_2k[df_2k.month==12]


         G_jan_2k=create_graph(jan_2k)
         G_feb_2k=create_graph(feb_2k)
         G_mar_2k=create_graph(mar_2k)
         G_apr_2k=create_graph(apr_2k)
         G_may_2k=create_graph(may_2k)
         G_jun_2k=create_graph(jun_2k)
         G_jul_2k=create_graph(jul_2k)
         G_aug_2k=create_graph(aug_2k)
         G_sep_2k=create_graph(sep_2k)
         G_oct_2k=create_graph(oct_2k)
         G_nov_2k=create_graph(nov_2k)
         G_dec_2k=create_graph(dec_2k)

In [16]: jan_2k1=df_2k1[df_2k1.month==1]
         feb_2k1=df_2k1[df_2k1.month==2]
         mar_2k1=df_2k1[df_2k1.month==3]
         apr_2k1=df_2k1[df_2k1.month==4]
         may_2k1=df_2k1[df_2k1.month==5]
         jun_2k1=df_2k1[df_2k1.month==6]
         jul_2k1=df_2k1[df_2k1.month==7]
         aug_2k1=df_2k1[df_2k1.month==8]
         sep_2k1=df_2k1[df_2k1.month==9]
         oct_2k1=df_2k1[df_2k1.month==10]
         nov_2k1=df_2k1[df_2k1.month==11]
         dec_2k1=df_2k1[df_2k1.month==12]
         G_jan_2k1=create_graph(jan_2k1)
         G_feb_2k1=create_graph(feb_2k1)
         G_mar_2k1=create_graph(mar_2k1)
         G_apr_2k1=create_graph(apr_2k1)
         G_may_2k1=create_graph(may_2k1)
         G_jun_2k1=create_graph(jun_2k1)
         G_jul_2k1=create_graph(jul_2k1)
         G_aug_2k1=create_graph(aug_2k1)
         G_sep_2k1=create_graph(sep_2k1)
         G_oct_2k1=create_graph(oct_2k1)
         G_nov_2k1=create_graph(nov_2k1)
         G_dec_2k1=create_graph(dec_2k1)

In [17]: jan_2k2=df_2k2[df_2k2.month==1]
         feb_2k2=df_2k2[df_2k2.month==2]
         mar_2k2=df_2k2[df_2k2.month==3]
         apr_2k2=df_2k2[df_2k2.month==4]
         may_2k2=df_2k2[df_2k2.month==5]
         jun_2k2=df_2k2[df_2k2.month==6]
         jul_2k2=df_2k2[df_2k2.month==7]
         aug_2k2=df_2k2[df_2k2.month==8]
         sep_2k2=df_2k2[df_2k2.month==9]
         oct_2k2=df_2k2[df_2k2.month==10]
         nov_2k2=df_2k2[df_2k2.month==11]
         dec_2k2=df_2k2[df_2k2.month==12]


         G_jan_2k2=create_graph(jan_2k2)
         G_feb_2k2=create_graph(feb_2k2)
         G_mar_2k2=create_graph(mar_2k2)
         G_apr_2k2=create_graph(apr_2k2)
         G_may_2k2=create_graph(may_2k2)
         G_jun_2k2=create_graph(jun_2k2)
         G_jul_2k2=create_graph(jul_2k2)
         G_aug_2k2=create_graph(aug_2k2)
         G_sep_2k2=create_graph(sep_2k2)
         G_oct_2k2=create_graph(oct_2k2)
         G_nov_2k2=create_graph(nov_2k2)
         G_dec_2k2=create_graph(dec_2k2)

In [18]: all_year_G  =tuple([Gt0,Gt1,Gt2,Gt3,Gt4])

In [19]: all_month_G = tuple([G_nov98,G_dec98,G_jan_99,G_feb_99,G_mar_99,G_apr_99,
                       G_nov_99,G_dec_99,G_jan_2k,G_feb_2k,G_mar_2k,G_apr_2k,G_may_
                       G_oct_2k,G_nov_2k,G_dec_2k,G_jan_2k1,G_feb_2k1,G_mar_2k1,G_a
                       G_aug_2k1,G_sep_2k1,G_oct_2k1,G_nov_2k1,G_dec_2k1,G_jan_2k2,
```

3 Network Visualisation

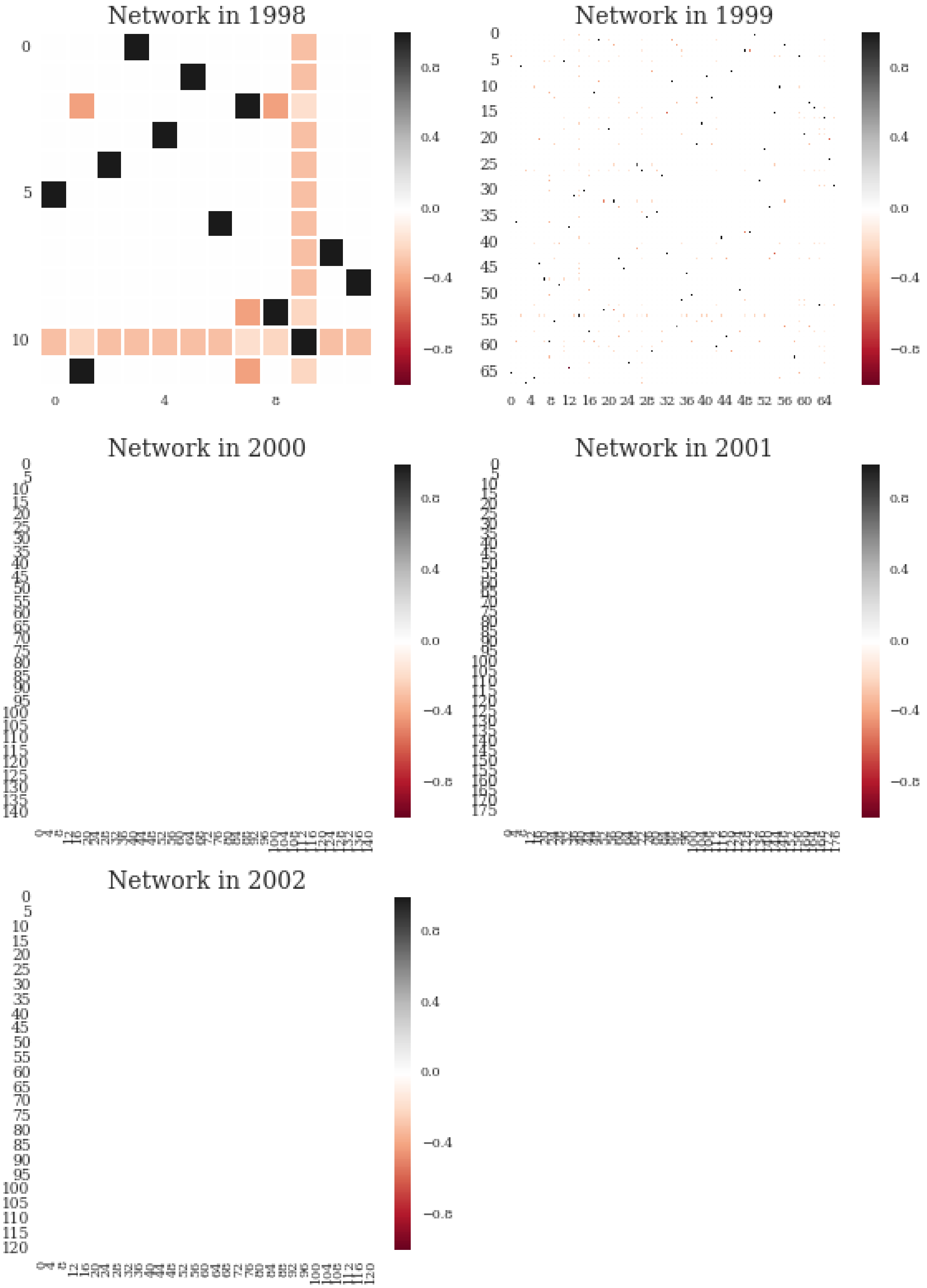
- Node Link Diagram
- Audio Waveform
- Matrix Visualisation

3.1 Yearly Networks

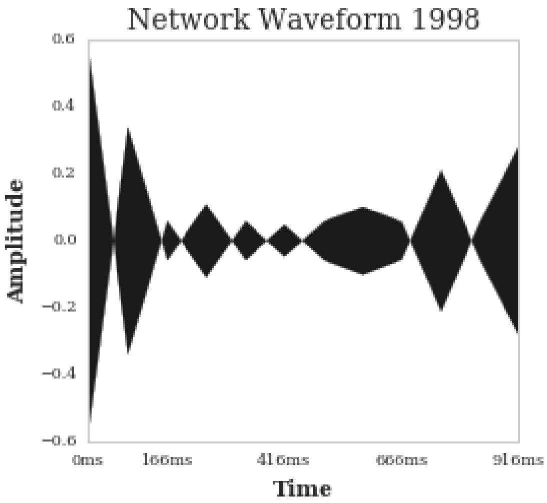
```
In [111]: plt.figure(figsize=(8,11))
          for i in range(len(all_year_G)):
              plt.subplot(6,2,i+1)
              nx.draw_spring(all_year_G[i],  node_color='#3D3C3A',node_shape='^',ed
              plt.title("Network in "  + str(years[i]), fontsize=14)
          plt.savefig('images/yearly_net.png')
```

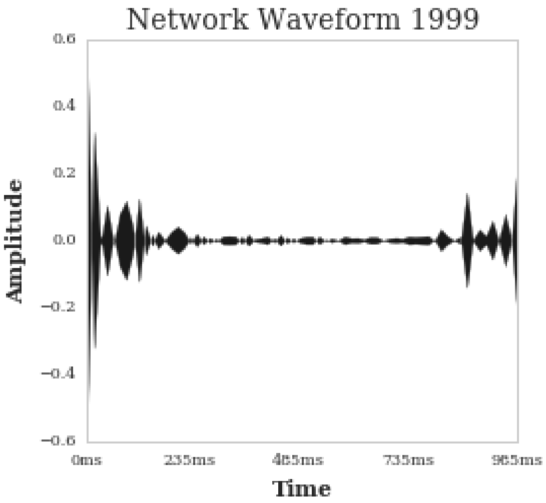


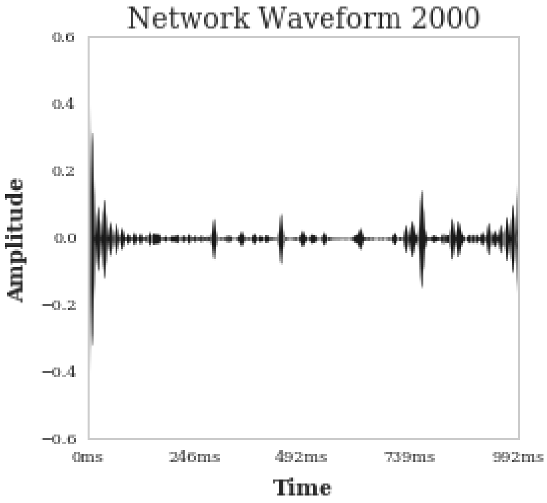
```
In [112]: plt.figure(figsize=(8,11))
          for i in range(len(all_year_G)):
              m = nx.normalized_laplacian_matrix(all_year_G[i]).todense()
              g = sns.clustermap(m)
              plt.close()
              ind = g.dendrogram_row.reordered_ind
              plt.subplot(3,2,i+1)
              sns.heatmap(m[ind][ind], cmap='RdGy', linewidths=1, xticklabels=4,yti
              plt.title("Network in "  + str(years[i]), fontsize=14)
              plt.yticks(fontsize=9, rotation=360)
              plt.tight_layout()
          plt.savefig('images/yearly_net_mat.png')
```

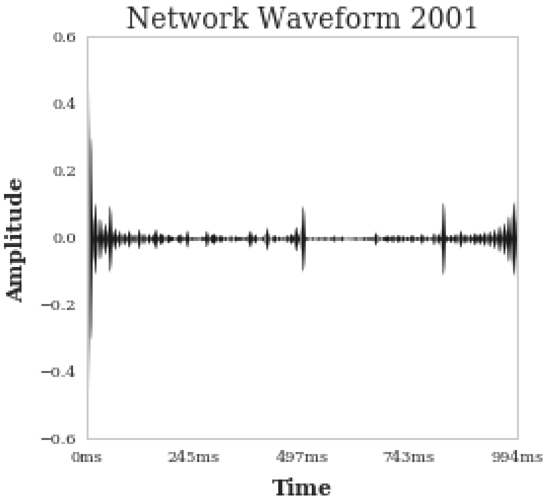


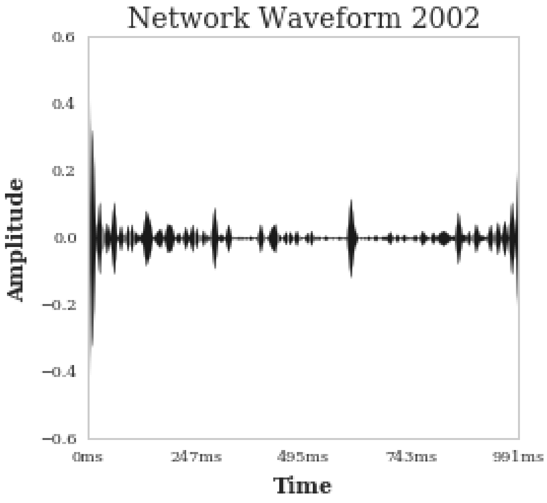
```
In [122]: plt.figure(figsize=(8,11))
          for i in range(len(all_year_G)):
              f = sc.fftpack.rfft(nx.normalized_laplacian_matrix(all_year_G[i]).too
              fs = f.shape[0]

              plt.subplot(3,2,i+1)
              librosa.display.waveplot(f, fs,color='k')
              plt.title("Network Waveform " + str(years[i]), fontsize=14)
              plt.xlabel("Time", fontsize=11)
              plt.ylabel("Amplitude", fontsize=11)
              plt.grid(False)
              plt.tight_layout()
          plt.savefig('images/yearly_net_audio.png')
```


Network Waveform 1998


Network Waveform 1999

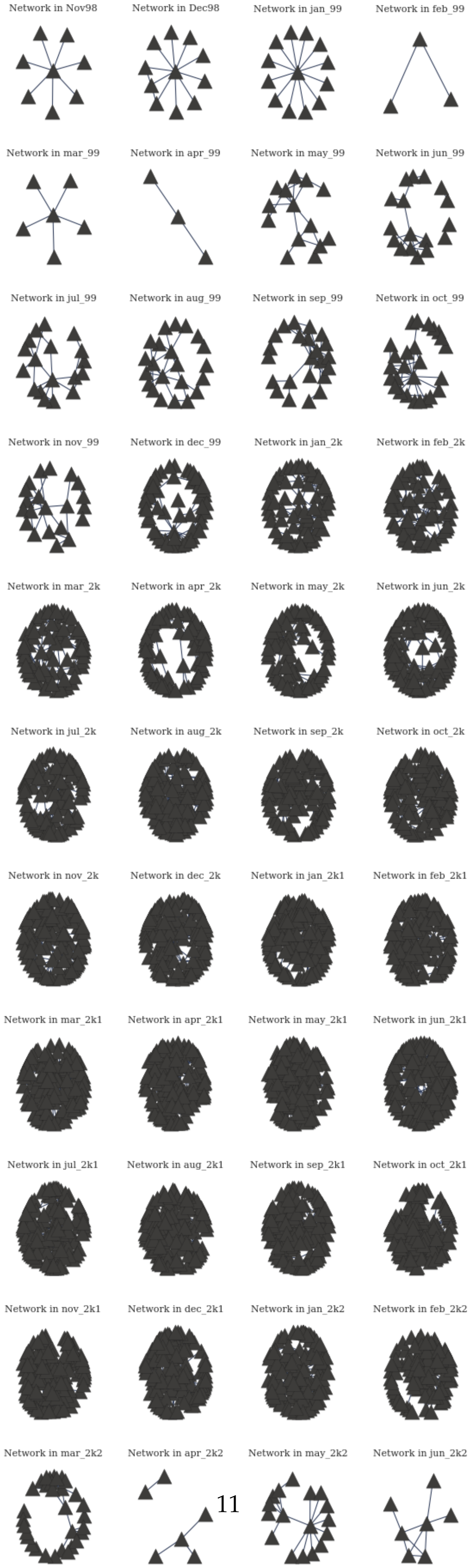

Network Waveform 2000


Network Waveform 2001

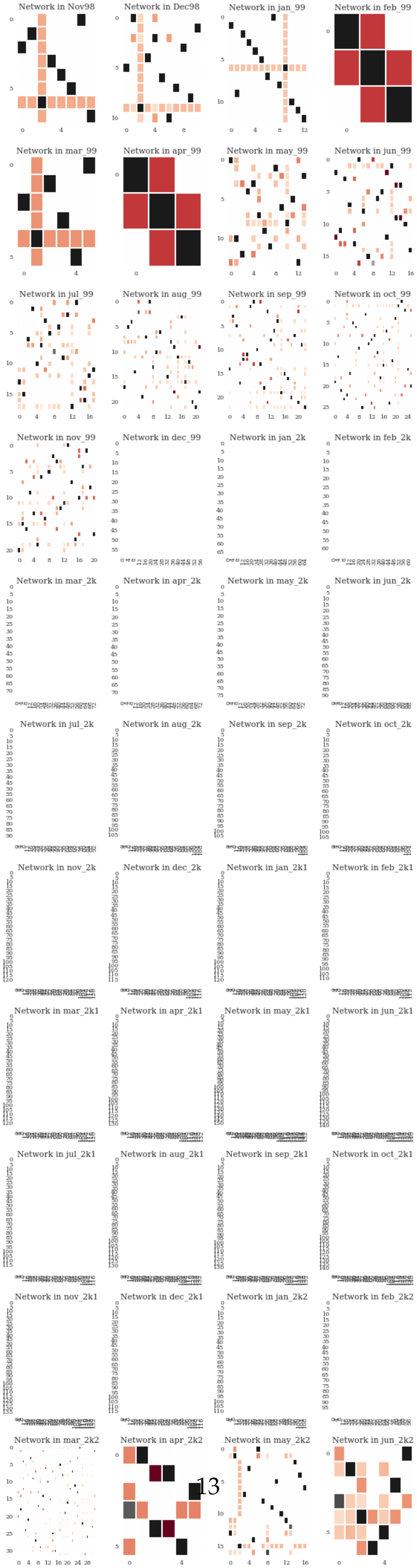

Network Waveform 2002

3.2 Monthly networks

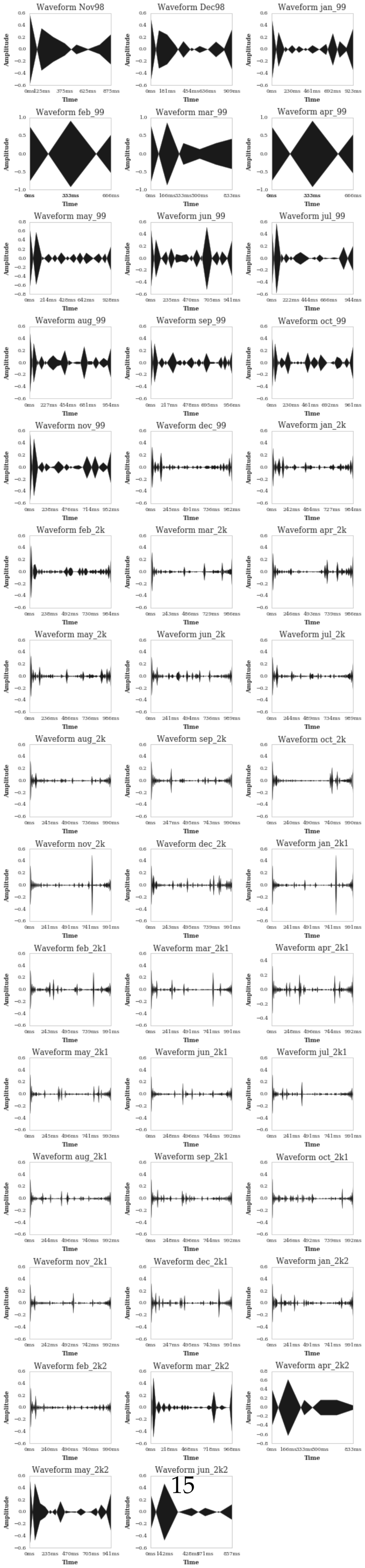
```
In [121]: plt.figure(figsize=(8,28))
          for i in range(len(all_month_G)):
              plt.subplot(12,4, i+1)
              nx.draw_spring(all_month_G[i],  node_color='#3D3C3A',node_shape='^',e
              plt.title("Network in "  + str(months[i]), fontsize=11)
              plt.tight_layout()
          plt.savefig('images/mth_net_nodelink.png')
```


Network in Nov98
Network in Dec98
Network in jan_99
Network in feb_99
Network in mar_99
Network in apr_99
Network in may_99
Network in jun_99
Network in jul_99
Network in aug_99
Network in sep_99
Network in oct_99
Network in nov_99
Network in dec_99
Network in jan_2k
Network in feb_2k
Network in mar_2k
Network in apr_2k
Network in may_2k
Network in jun_2k
Network in jul_2k
Network in aug_2k
Network in sep_2k
Network in oct_2k
Network in nov_2k
Network in dec_2k
Network in jan_2k1
Network in feb_2k1
Network in mar_2k1
Network in apr_2k1
Network in may_2k1
Network in jun_2k1
Network in jul_2k1
Network in aug_2k1
Network in sep_2k1
Network in oct_2k1
Network in nov_2k1
Network in dec_2k1
Network in jan_2k2
Network in feb_2k2
Network in mar_2k2
Network in apr_2k2
Network in may_2k2
Network in jun_2k2

```
In [124]: plt.figure(figsize=(8,32))
          for i in range(len(all_month_G)):
              m = nx.normalized_laplacian_matrix(all_month_G[i]).todense()
              g = sns.clustermap(m)
              plt.close()
              ind = g.dendrogram_row.reordered_ind
              plt.subplot(12, 4, i+1)
              sns.heatmap(m[ind][ind], cmap='RdGy', linewidths=1, xticklabels=4,yti
              plt.title("Network in "  + str(months[i]), fontsize=11)
              plt.yticks(rotation=360)
              plt.tight_layout()
          plt.savefig('images/mth_net_mat.png')
```


Network in Nov98
Network in Dec98
Network in jan_99
Network in feb_99
Network in mar_99
Network in apr_99
Network in may_99
Network in jun_99
Network in jul_99
Network in aug_99
Network in sep_99
Network in oct_99
Network in nov_99
Network in dec_99
Network in jan_2k
Network in feb_2k
Network in mar_2k
Network in apr_2k
Network in may_2k
Network in jun_2k
Network in jul_2k
Network in aug_2k
Network in sep_2k
Network in oct_2k
Network in nov_2k
Network in dec_2k
Network in jan_2k1
Network in feb_2k1
Network in mar_2k1
Network in apr_2k1
Network in may_2k1
Network in jun_2k1
Network in jul_2k1
Network in aug_2k1
Network in sep_2k1
Network in oct_2k1
Network in nov_2k1
Network in dec_2k1
Network in jan_2k2
Network in feb_2k2
Network in mar_2k2
Network in apr_2k2
Network in may_2k2
Network in jun_2k2

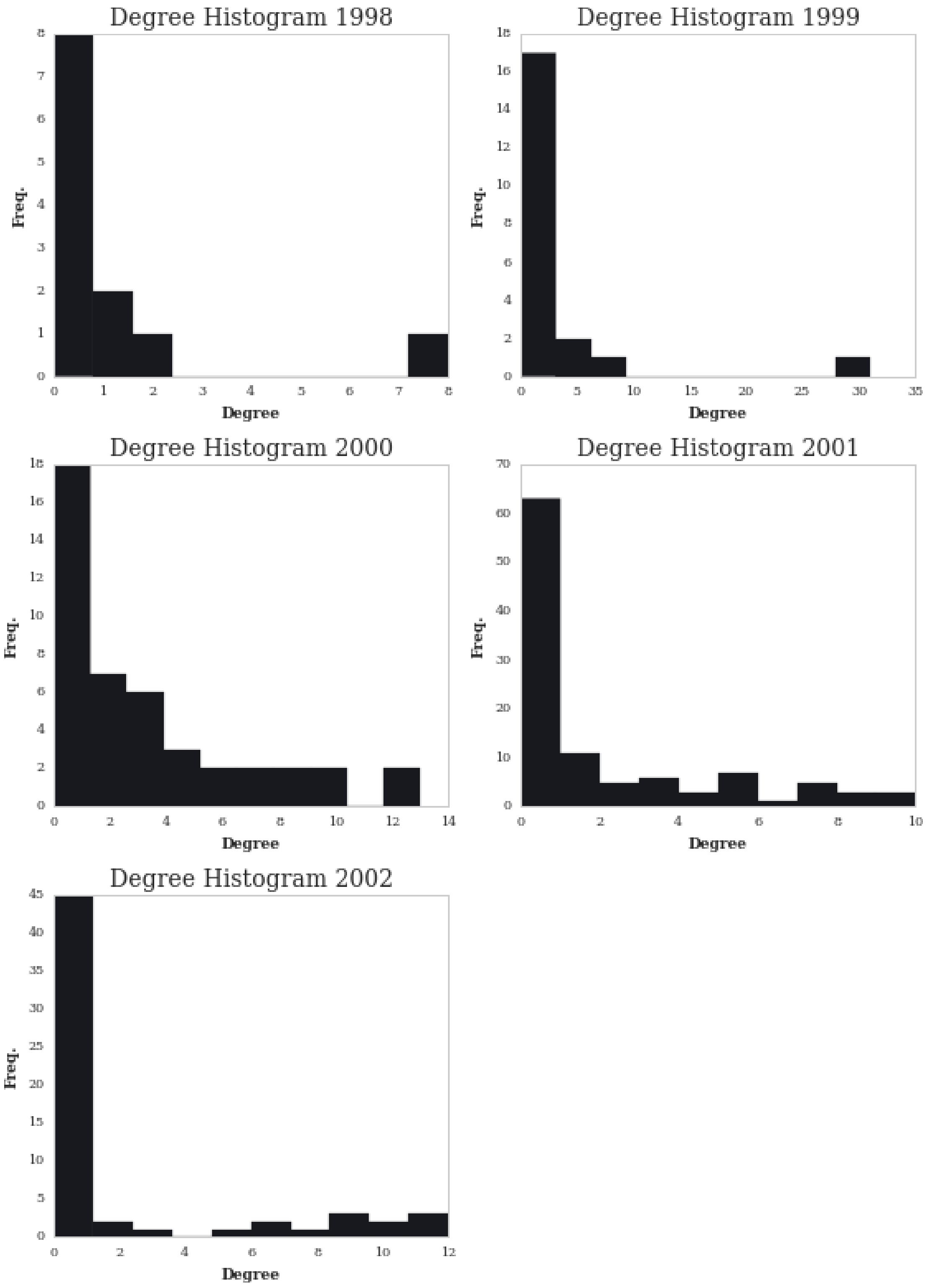
```
In [135]: plt.figure(figsize=(8,36))

          for i in range(len(all_month_G)):
              f = sc.fftpack.rfft(nx.normalized_laplacian_matrix(all_month_G[i]).to
              fs = f.shape[0]

              plt.subplot(16, 3, i+1)
              librosa.display.waveplot(f, fs,color='k')
              plt.title("Waveform " + str(months[i]), fontsize=12)

              plt.xlabel("Time")
              plt.ylabel("Amplitude")
              plt.grid(False)
              plt.tight_layout()
          plt.savefig('images/mth_net_audio.png')
```


Waveform Nov98
Waveform Dec98
Waveform jan_99
Waveform feb_99
Waveform mar_99
Waveform apr_99
Waveform may_99
Waveform jun_99
Waveform jul_99
Waveform aug_99
Waveform sep_99
Waveform oct_99
Waveform nov_99
Waveform dec_99
Waveform jan_2k
Waveform feb_2k
Waveform mar_2k
Waveform apr_2k
Waveform may_2k
Waveform jun_2k
Waveform jul_2k
Waveform aug_2k
Waveform sep_2k
Waveform oct_2k
Waveform nov_2k
Waveform dec_2k
Waveform jan_2k1
Waveform feb_2k1
Waveform mar_2k1
Waveform apr_2k1
Waveform may_2k1
Waveform jun_2k1
Waveform jul_2k1
Waveform aug_2k1
Waveform sep_2k1
Waveform oct_2k1
Waveform nov_2k1
Waveform dec_2k1
Waveform jan_2k2
Waveform feb_2k2
Waveform mar_2k2
Waveform apr_2k2
Waveform may_2k2
Waveform jun_2k2
Amplitude
Time

4 Exploratory Analysis

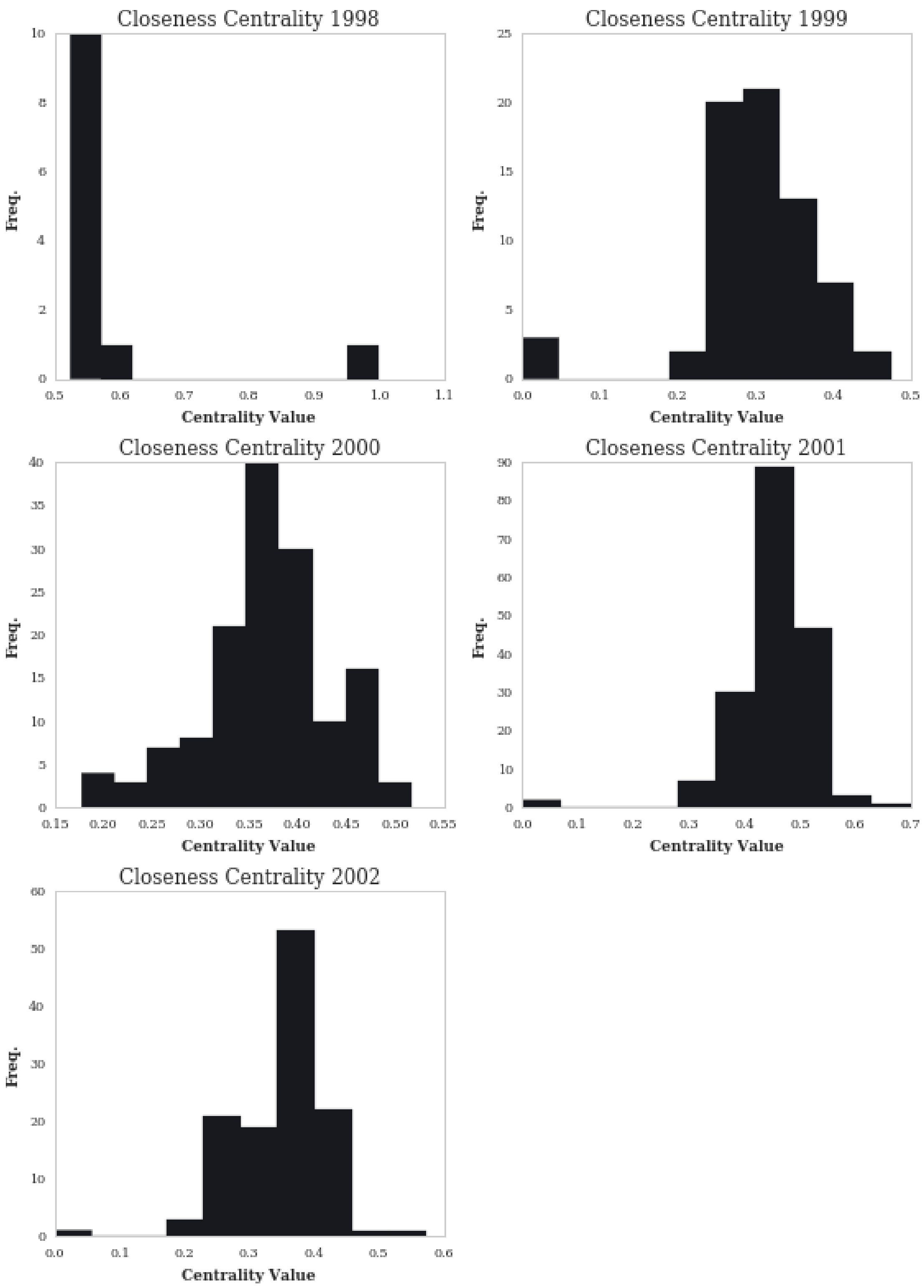
4.1 Yearly Networks

```
In [126]: plt.figure(figsize=(8,11))
          for i in range(len(all_year_G)):
              deg = nx.degree_histogram(all_year_G[i])
              plt.subplot(3, 2, i+1)
              plt.hist(deg)
              plt.title("Degree Histogram " + str(years[i]), fontsize=14)
              plt.xlabel("Degree")
              plt.ylabel("Freq.")
              plt.grid(False)
              plt.tight_layout()
          plt.savefig('images/year_deghist.png')
```



```
In [136]: plt.figure(figsize=(8,11))
          for i in range(len(all_year_G)):
```

```
    deg = nx.closeness_centrality(all_year_G[i]).values()
    deg = sorted(deg)
    plt.subplot(3, 2, i+1)
    plt.hist(deg)
    plt.title("Closeness Centrality " + str(years[i]), fontsize=12)
    plt.xlabel("Centrality Value")
    plt.ylabel("Freq.")
    plt.grid(False)
    plt.tight_layout()
plt.savefig('images/year_clohist.png')
```


Closeness Centrality 1998

Closeness Centrality 1999

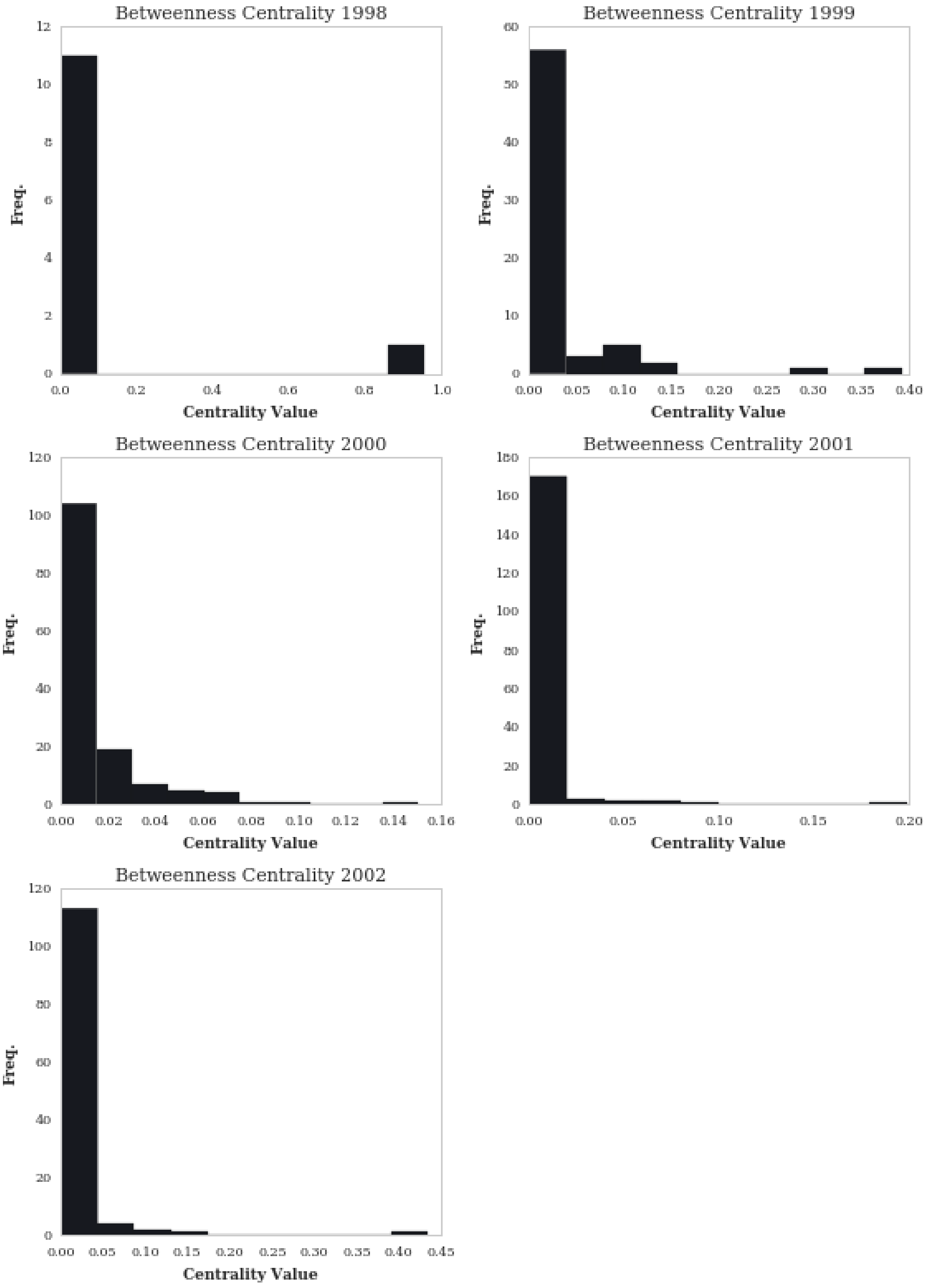
Closeness Centrality 2000

Closeness Centrality 2001

Closeness Centrality 2002

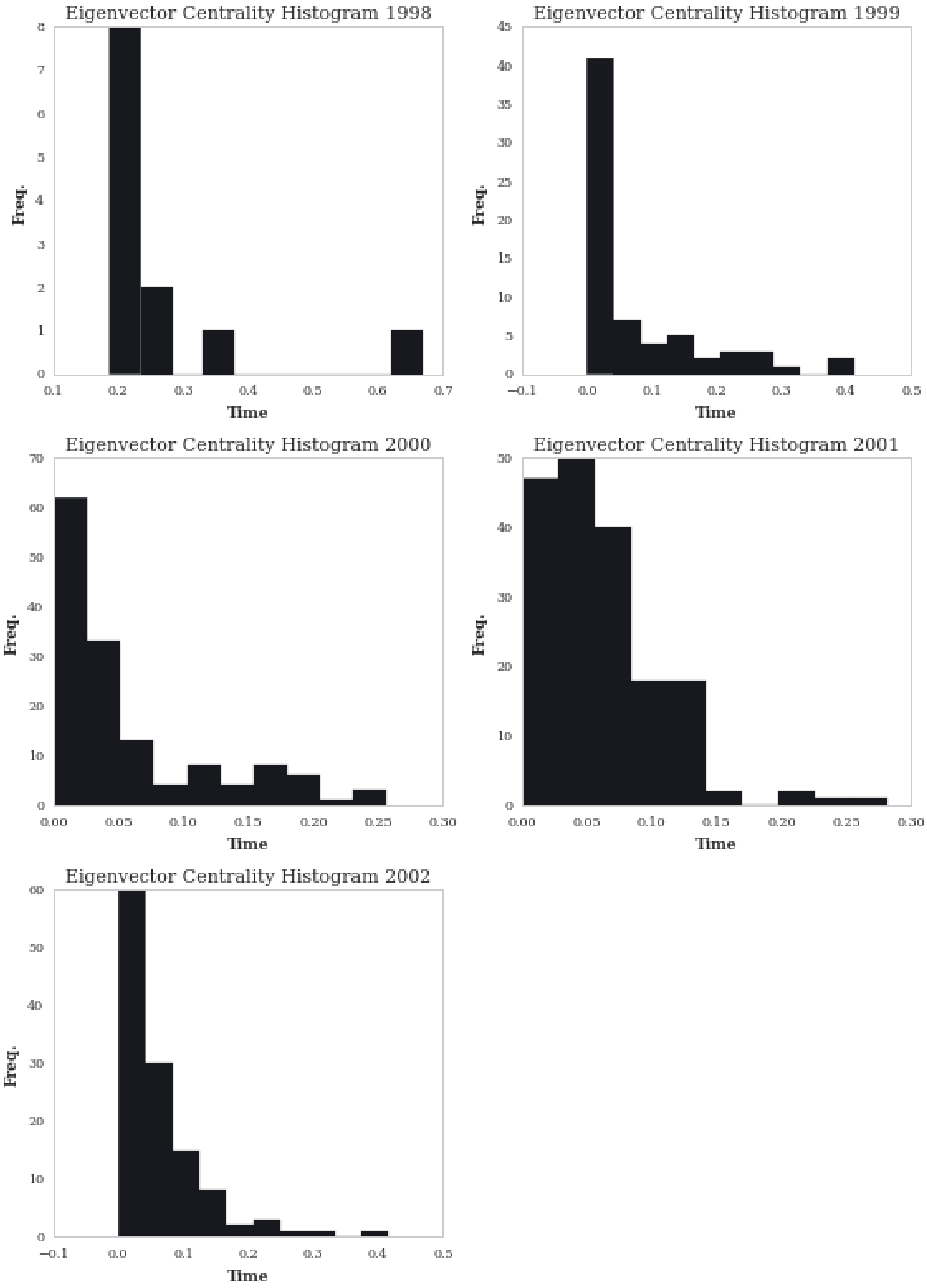
```
In [137]: plt.figure(figsize=(8,11))
          for i in range(len(all_year_G)):
```

```
        deg = nx.betweenness_centrality(all_year_G[i]).values()
        deg = sorted(deg)
        plt.subplot(3, 2, i+1)
        plt.hist(deg)
        plt.title("Betweenness Centrality " + str(years[i]), fontsize=11)
        plt.xlabel("Centrality Value")
        plt.ylabel("Freq.")
        plt.grid(False)
        plt.tight_layout()
    plt.savefig('images/year_bethist.png')
```



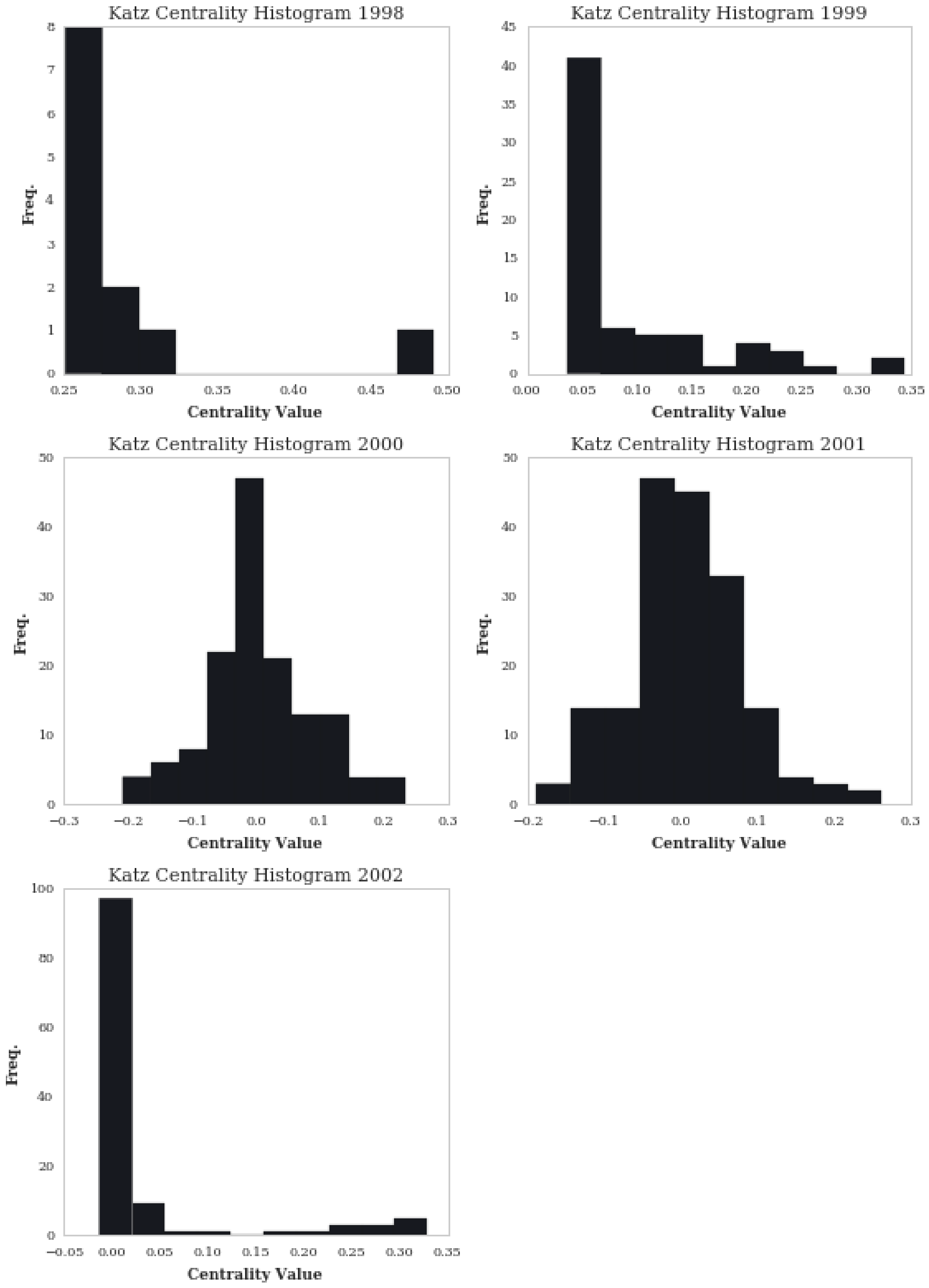
```
In [138]: plt.figure(figsize=(8,11))
          for i in range(len(all_year_G)):
```

```
    deg = nx.eigenvector_centrality_numpy(all_year_G[i]).values()
    deg = sorted(deg)
    plt.subplot(3, 2, i+1)
    plt.hist(deg)
    plt.title("Eigenvector Centrality Histogram " + str(years[i]), fontsi
    plt.xlabel("Time")
    plt.ylabel("Freq.")
    plt.grid(False)
    plt.tight_layout()
plt.savefig('images/year_eighist.png')
```



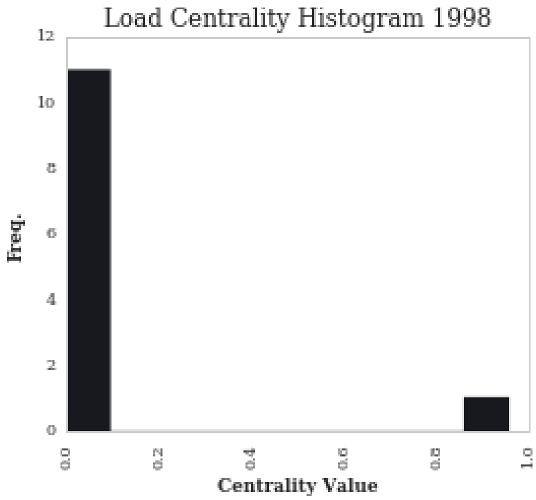
```
In [139]: plt.figure(figsize=(8,11))
          for i in range(len(all_year_G)):
```

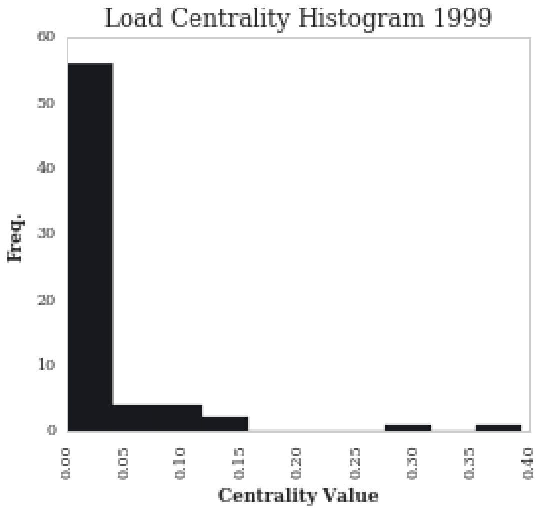
```
    deg = nx.katz_centrality_numpy(all_year_G[i]).values()
    deg = sorted(deg)
    plt.subplot(3, 2, i+1)
    plt.hist(deg)
    plt.title("Katz Centrality Histogram " + str(years[i]), fontsize=11)
    plt.xlabel("Centrality Value")
    plt.ylabel("Freq.")
    plt.grid(False)
    plt.tight_layout()
plt.savefig('images/year_katzhist.png')
```

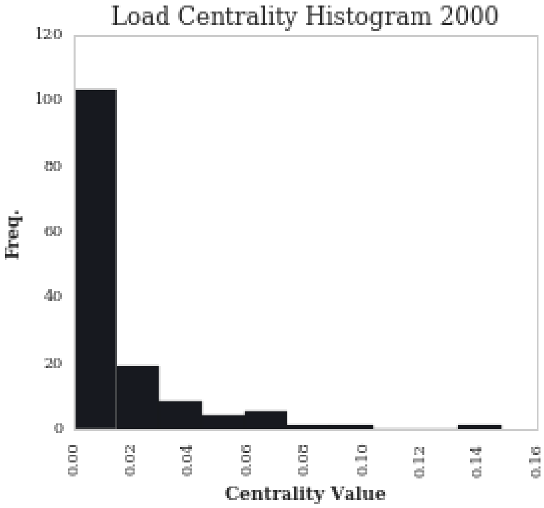


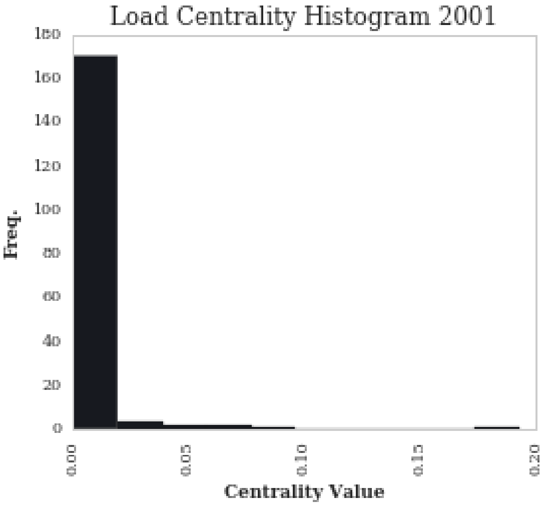
```
In [140]: plt.figure(figsize=(8,11))
          for i in range(len(all_year_G)):
```

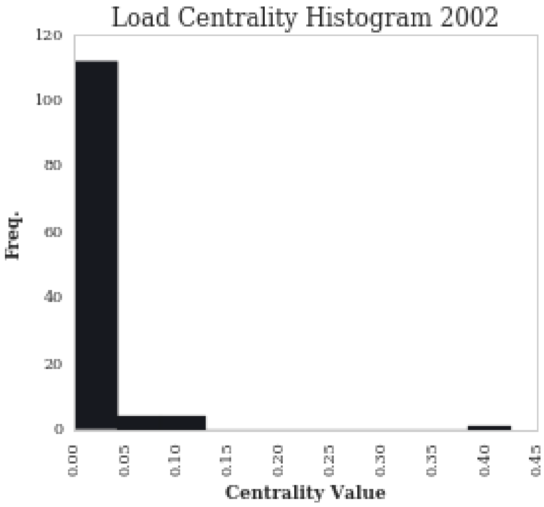
```
    deg = nx.load_centrality(all_year_G[i]).values()
    deg = sorted(deg)
    plt.subplot(3, 2, i+1)
    plt.hist(deg)
    plt.title("Load Centrality Histogram " + str(years[i]), fontsize=12)
    plt.xticks(rotation=90)
    plt.xlabel("Centrality Value")
    plt.ylabel("Freq.")
    plt.grid(False)
    plt.tight_layout()
plt.savefig('images/year_loadhist.png')
```


Load Centrality Histogram 1998
Freq.
0
2
4
6
8
10
12
0.0
0.2
0.4
0.6
0.8
1.0
Centrality Value


Load Centrality Histogram 1999
Freq.
0
10
20
30
40
50
60
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Centrality Value

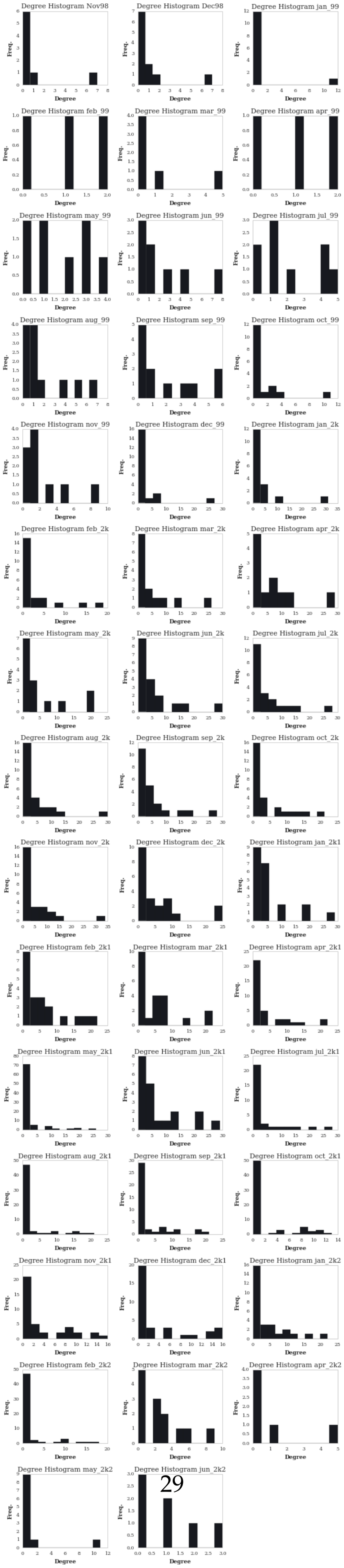

Load Centrality Histogram 2000
Freq.
0
20
40
60
80
100
120
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Centrality Value


Load Centrality Histogram 2001
Freq.
0
20
40
60
80
100
120
140
160
180
0.00
0.05
0.10
0.15
0.20
Centrality Value

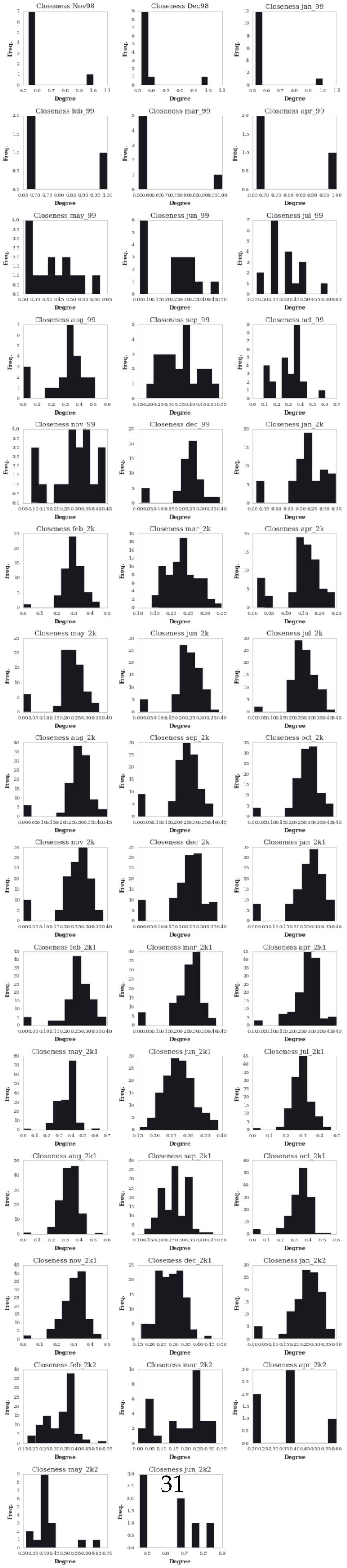

Load Centrality Histogram 2002
Freq.
0
20
40
60
80
100
120
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
Centrality Value

4.2 Monthly Networks

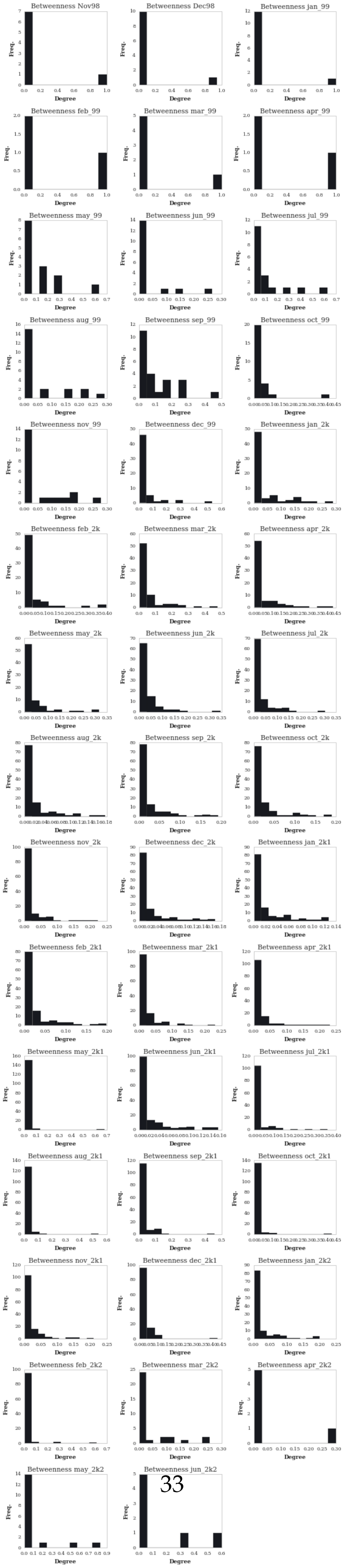
```
In [141]: plt.figure(figsize=(8,38))
          for i in range(len(all_month_G)):
              deg = nx.degree_histogram(all_month_G[i])
              plt.subplot(16, 3, i+1)
              plt.hist(deg)
              plt.title("Degree Histogram " + str(months[i]), fontsize=11)
              plt.xlabel("Degree")
              plt.ylabel("Freq.")
              plt.grid(False)
              plt.tight_layout()
          plt.savefig('images/mth_deghist.png')
```


Degree Histogram Nov98
Degree Histogram Dec98
Degree Histogram jan_99
Degree Histogram feb_99
Degree Histogram mar_99
Degree Histogram apr_99
Degree Histogram may_99
Degree Histogram jun_99
Degree Histogram jul_99
Degree Histogram aug_99
Degree Histogram sep_99
Degree Histogram oct_99
Degree Histogram nov_99
Degree Histogram dec_99
Degree Histogram jan_2k
Degree Histogram feb_2k
Degree Histogram mar_2k
Degree Histogram apr_2k
Degree Histogram may_2k
Degree Histogram jun_2k
Degree Histogram jul_2k
Degree Histogram aug_2k
Degree Histogram sep_2k
Degree Histogram oct_2k
Degree Histogram nov_2k
Degree Histogram dec_2k
Degree Histogram jan_2k1
Degree Histogram feb_2k1
Degree Histogram mar_2k1
Degree Histogram apr_2k1
Degree Histogram may_2k1
Degree Histogram jun_2k1
Degree Histogram jul_2k1
Degree Histogram aug_2k1
Degree Histogram sep_2k1
Degree Histogram oct_2k1
Degree Histogram nov_2k1
Degree Histogram dec_2k1
Degree Histogram jan_2k2
Degree Histogram feb_2k2
Degree Histogram mar_2k2
Degree Histogram apr_2k2
Degree Histogram may_2k2
Degree Histogram jun_2k2
Freq.
Degree

```
In [142]: plt.figure(figsize=(8,38))
          for i in range(len(all_month_G)):
              deg = nx.closeness_centrality(all_month_G[i]).values()
              deg = sorted(deg)
              plt.subplot(16, 3, i+1)
              plt.hist(deg)
              plt.title("Closeness " + str(months[i]), fontsize=11)
              plt.xlabel("Degree")
              plt.ylabel("Freq.")
              plt.grid(False)
              plt.tight_layout()
          plt.savefig('images/mth_clohist.png')
```

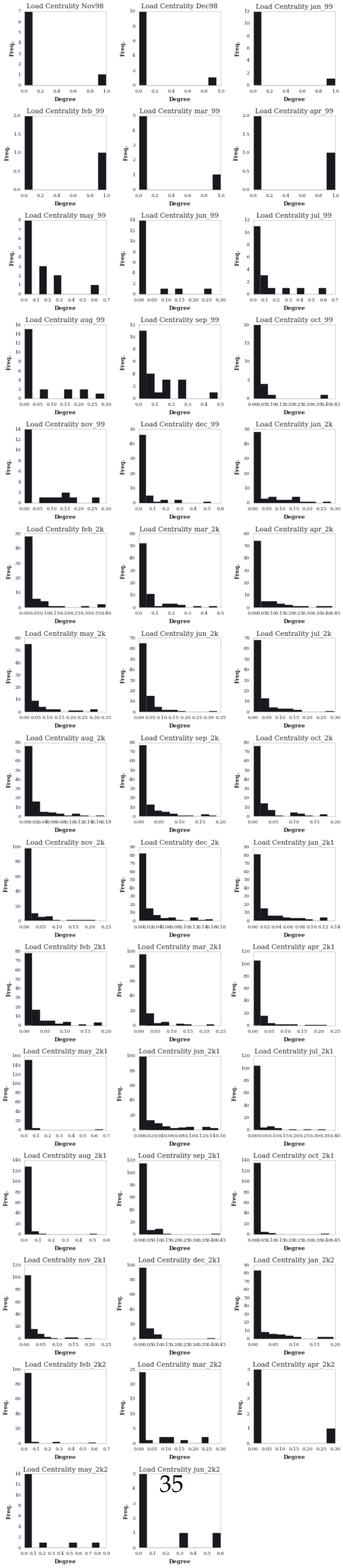

Closeness Nov98
Closeness Dec98
Closeness jan_99
Closeness feb_99
Closeness mar_99
Closeness apr_99
Closeness may_99
Closeness jun_99
Closeness jul_99
Closeness aug_99
Closeness sep_99
Closeness oct_99
Closeness nov_99
Closeness dec_99
Closeness jan_2k
Closeness feb_2k
Closeness mar_2k
Closeness apr_2k
Closeness may_2k
Closeness jun_2k
Closeness jul_2k
Closeness aug_2k
Closeness sep_2k
Closeness oct_2k
Closeness nov_2k
Closeness dec_2k
Closeness jan_2k1
Closeness feb_2k1
Closeness mar_2k1
Closeness apr_2k1
Closeness may_2k1
Closeness jun_2k1
Closeness jul_2k1
Closeness aug_2k1
Closeness sep_2k1
Closeness oct_2k1
Closeness nov_2k1
Closeness dec_2k1
Closeness jan_2k2
Closeness feb_2k2
Closeness mar_2k2
Closeness apr_2k2
Closeness may_2k2
Closeness jun_2k2
Freq.
Degree

```
In [143]: plt.figure(figsize=(8,38))
          for i in range(len(all_month_G)):
              deg = nx.betweenness_centrality(all_month_G[i]).values()
              deg = sorted(deg)
              plt.subplot(16, 3, i+1)
              plt.hist(deg)
              plt.title("Betweenness " + str(months[i]), fontsize=11)
              plt.xlabel("Degree")
              plt.ylabel("Freq.")
              plt.grid(False)
              plt.tight_layout()
          plt.savefig('images/mth_bethist.png')
```


Betweenness Nov98
Betweenness Dec98
Betweenness jan_99
Betweenness feb_99
Betweenness mar_99
Betweenness apr_99
Betweenness may_99
Betweenness jun_99
Betweenness jul_99
Betweenness aug_99
Betweenness sep_99
Betweenness oct_99
Betweenness nov_99
Betweenness dec_99
Betweenness jan_2k
Betweenness feb_2k
Betweenness mar_2k
Betweenness apr_2k
Betweenness may_2k
Betweenness jun_2k
Betweenness jul_2k
Betweenness aug_2k
Betweenness sep_2k
Betweenness oct_2k
Betweenness nov_2k
Betweenness dec_2k
Betweenness jan_2k1
Betweenness feb_2k1
Betweenness mar_2k1
Betweenness apr_2k1
Betweenness may_2k1
Betweenness jun_2k1
Betweenness jul_2k1
Betweenness aug_2k1
Betweenness sep_2k1
Betweenness oct_2k1
Betweenness nov_2k1
Betweenness dec_2k1
Betweenness jan_2k2
Betweenness feb_2k2
Betweenness mar_2k2
Betweenness apr_2k2
Betweenness may_2k2
Betweenness jun_2k2
Freq.
Degree

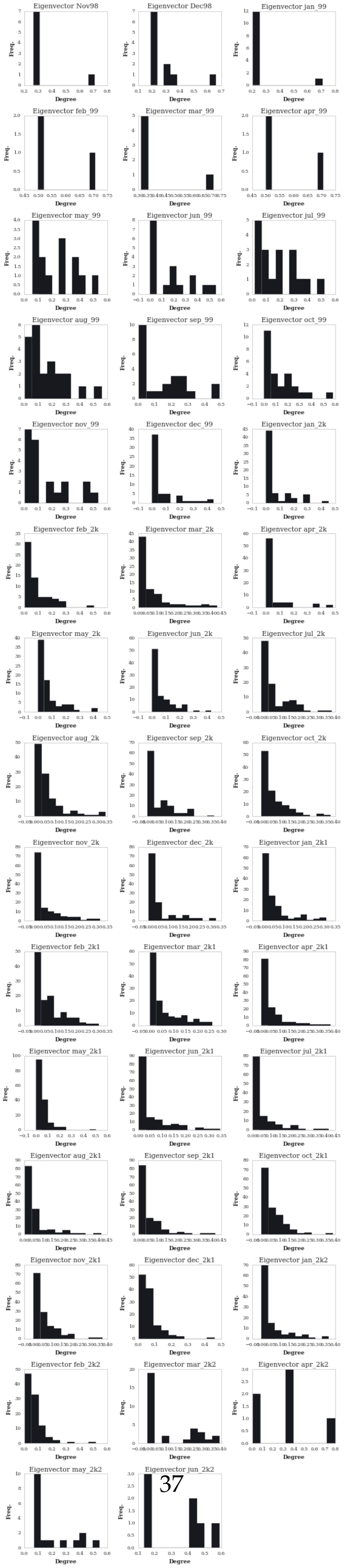
```
In [229]: plt.figure(figsize=(8,38))
          for i in range(len(all_month_G)):
              deg = nx.load_centrality(all_month_G[i]).values()
              deg = sorted(deg)
              plt.subplot(16, 3, i+1)
              plt.hist(deg)
              plt.title("Load Centrality " + str(months[i]), fontsize=11)
              plt.xlabel("Degree")
              plt.ylabel("Freq.")
              plt.grid(False)
              plt.tight_layout()
          plt.savefig('images/mth_loadhist.png')
```

```
/home/arshad/anaconda3/lib/python3.5/site-packages/matplotlib/figure.py:1742: UserW
  warnings.warn("This figure includes Axes that are not "
```

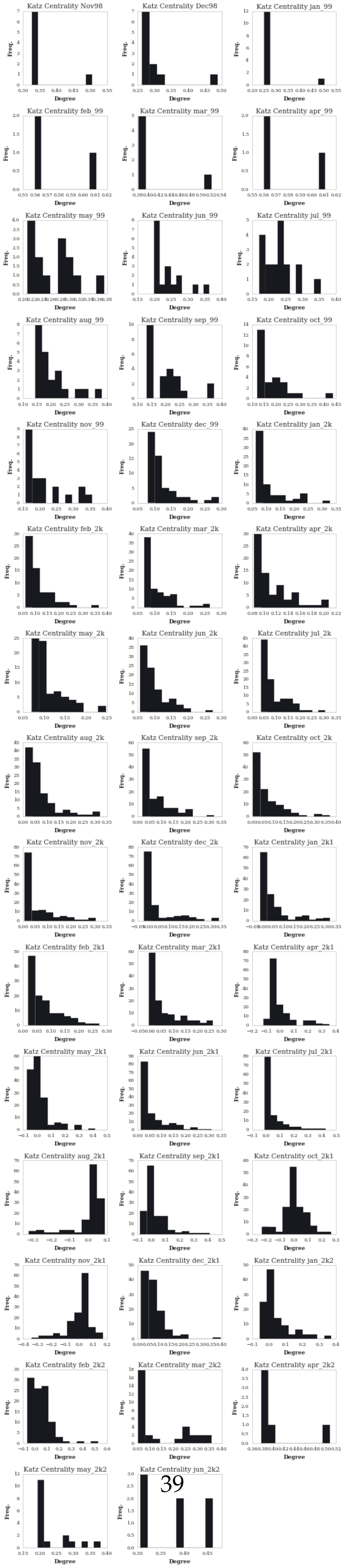

Load Centrality Nov98
Load Centrality Dec98
Load Centrality jan_99
Load Centrality feb_99
Load Centrality mar_99
Load Centrality apr_99
Load Centrality may_99
Load Centrality jun_99
Load Centrality jul_99
Load Centrality aug_99
Load Centrality sep_99
Load Centrality oct_99
Load Centrality nov_99
Load Centrality dec_99
Load Centrality jan_2k
Load Centrality feb_2k
Load Centrality mar_2k
Load Centrality apr_2k
Load Centrality may_2k
Load Centrality jun_2k
Load Centrality jul_2k
Load Centrality aug_2k
Load Centrality sep_2k
Load Centrality oct_2k
Load Centrality nov_2k
Load Centrality dec_2k
Load Centrality jan_2k1
Load Centrality feb_2k1
Load Centrality mar_2k1
Load Centrality apr_2k1
Load Centrality may_2k1
Load Centrality jun_2k1
Load Centrality jul_2k1
Load Centrality aug_2k1
Load Centrality sep_2k1
Load Centrality oct_2k1
Load Centrality nov_2k1
Load Centrality dec_2k1
Load Centrality jan_2k2
Load Centrality feb_2k2
Load Centrality mar_2k2
Load Centrality apr_2k2
Load Centrality may_2k2
Load Centrality jun_2k2
Freq.
Degree

```
In [228]: plt.figure(figsize=(8,38))
          for i in range(len(all_month_G)):
              deg = nx.eigenvector_centrality_numpy(all_month_G[i]).values()
              deg = sorted(deg)
              plt.subplot(16,3, i+1)
              plt.hist(deg)
              plt.title("Eigenvector " + str(months[i]), fontsize=11)
              plt.xlabel("Degree")
              plt.ylabel("Freq.")
              plt.grid(False)
              plt.tight_layout()
          plt.savefig('images/mth_eighist.png')
```

```
/home/arshad/anaconda3/lib/python3.5/site-packages/matplotlib/figure.py:1742: UserW
  warnings.warn("This figure includes Axes that are not "
```


Eigenvector Nov98
Eigenvector Dec98
Eigenvector jan_99
Eigenvector feb_99
Eigenvector mar_99
Eigenvector apr_99
Eigenvector may_99
Eigenvector jun_99
Eigenvector jul_99
Eigenvector aug_99
Eigenvector sep_99
Eigenvector oct_99
Eigenvector nov_99
Eigenvector dec_99
Eigenvector jan_2k
Eigenvector feb_2k
Eigenvector mar_2k
Eigenvector apr_2k
Eigenvector may_2k
Eigenvector jun_2k
Eigenvector jul_2k
Eigenvector aug_2k
Eigenvector sep_2k
Eigenvector oct_2k
Eigenvector nov_2k
Eigenvector dec_2k
Eigenvector jan_2k1
Eigenvector feb_2k1
Eigenvector mar_2k1
Eigenvector apr_2k1
Eigenvector may_2k1
Eigenvector jun_2k1
Eigenvector jul_2k1
Eigenvector aug_2k1
Eigenvector sep_2k1
Eigenvector oct_2k1
Eigenvector nov_2k1
Eigenvector dec_2k1
Eigenvector jan_2k2
Eigenvector feb_2k2
Eigenvector mar_2k2
Eigenvector apr_2k2
Eigenvector may_2k2
Eigenvector jun_2k2
Freq.
Degree

```
In [145]: plt.figure(figsize=(8,38))
          for i in range(len(all_month_G)):
              deg = nx.katz_centrality_numpy(all_month_G[i]).values()
              deg = sorted(deg)
              plt.subplot(16,3, i+1)
              plt.hist(deg)
              plt.title("Katz Centrality " + str(months[i]), fontsize=11)
              plt.xlabel("Degree")
              plt.ylabel("Freq.")
              plt.grid(False)
              plt.tight_layout()
          plt.savefig('images/mth_katzhist.png')
```


Katz Centrality Nov98
Katz Centrality Dec98
Katz Centrality jan_99
Katz Centrality feb_99
Katz Centrality mar_99
Katz Centrality apr_99
Katz Centrality may_99
Katz Centrality jun_99
Katz Centrality jul_99
Katz Centrality aug_99
Katz Centrality sep_99
Katz Centrality oct_99
Katz Centrality nov_99
Katz Centrality dec_99
Katz Centrality jan_2k
Katz Centrality feb_2k
Katz Centrality mar_2k
Katz Centrality apr_2k
Katz Centrality may_2k
Katz Centrality jun_2k
Katz Centrality jul_2k
Katz Centrality aug_2k
Katz Centrality sep_2k
Katz Centrality oct_2k
Katz Centrality nov_2k
Katz Centrality dec_2k
Katz Centrality jan_2k1
Katz Centrality feb_2k1
Katz Centrality mar_2k1
Katz Centrality apr_2k1
Katz Centrality may_2k1
Katz Centrality jun_2k1
Katz Centrality jul_2k1
Katz Centrality aug_2k1
Katz Centrality sep_2k1
Katz Centrality oct_2k1
Katz Centrality nov_2k1
Katz Centrality dec_2k1
Katz Centrality jan_2k2
Katz Centrality feb_2k2
Katz Centrality mar_2k2
Katz Centrality apr_2k2
Katz Centrality may_2k2
Katz Centrality jun_2k2
Freq.
Degree

5 Attribute Analysis

5.1 Traditional Measures

5.1.1 Centrality

- Degree
- Betweeness
- Closeness
- Katz
- Load

5.1.2 Assortativity & Linear Algebra

- Density
- Average Clustering Coefficient
- Algebraic Connectivity

5.2 Complex Trace Attributes

- Instantaneous Amplitude
- Instantaneous Frequency
- Instantaneous Phase
- Derivative of Amplitude
- Second Derivative of Amplitude
- Power
- Cosine of Instantaneous Phase
- Amplitude weighted Frequency
- Amplitude weighted Phase
- Power Spectral Density

5.3 Matrix

- Resistance Distance
- Stationarity Ratio
- Subgraph Stationarity
- Curvature
- Norm of Abel Transform

5.4 Matrix Decomposition

- KLPCA Ratio Change
- Norm NMF Ratio Change

5.5 Music Attributes

- Zero Crossing Rate
- Spectral Centroid

5.6 Aggregation Measures

- Persistence
- Emergence
- NRMS of Attributes

```
In [38]: def calc_seisatt(net):
             M = nx.normalized_laplacian_matrix(net).todense()
             Ht = hilbert(M)
             rHt = np.real(Ht)
             iHt = np.imag(Ht)

             #Basic attributes IA, IP, IF
             IA = np.real(np.nan_to_num(np.sqrt(np.dot(rHt,rHt)+np.dot(iHt,iHt))))
             IP = np.real(np.nan_to_num(np.arctan(iHt/rHt)))
             IF,_ = np.real(np.nan_to_num(np.asarray(np.gradient(IP))))
             P = np.square(IA)

             #Derivatives
             dIA,_ = np.nan_to_num(np.asarray(np.gradient(IA)))
             d2IA,_ = np.nan_to_num(np.asarray(np.gradient(dIA)))
             IAcc,_ =np.nan_to_num(np.asarray(np.gradient(IF)))

             #Derived Attributes
             cosIP = np.cos(IP)
             IA_weit_IF = IA * IF
             IA_weit_IP = IA*IP

             return [IA,IP,IF,P, dIA,d2IA, IAcc,cosIP,IA_weit_IF,IA_weit_IP]
```

```
In [39]: def stationarity_ratio(G):
             #stationarity ratio with laplian
             L = nx.normalized_laplacian_matrix(G).todense()
             U = eigvals(L)
             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)

             return r
```

```
In [40]: #cite:`klein1993resistance`
         def resistance_distance(net):
             M = nx.normalized_laplacian_matrix(net).todense()
             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 - pseudo

             return rd.mean()
```

```
In [41]: def curvature(net):
             from skimage.feature import hessian_matrix, hessian_matrix_det, hessia
             M = nx.normalized_laplacian_matrix(net).todense()
             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,_ = hessian_matrix_eigvals(Hxx,Hxy,Hyy)
             mean_curv = np.trace(He1)

             return mean_curv
```

```
In [42]: def kpca_ratio(net):
             from sklearn.decomposition import KernelPCA
             kpca = KernelPCA(n_components=3,kernel='rbf')
             m = nx.normalized_laplacian_matrix(net).todense()
             X_kpca = kpca.fit_transform(m)
             pc1 = X_kpca[:,0]
             pc2 = X_kpca[:,1]
             pc3 = X_kpca[:,2]
             kpca_rat = norm(pc1-pc3/(pc1-pc2))
             return kpca_rat
```

```
In [43]: def kpca_att(net):
             kpca_chg = []
             for i in range(len(net)-1):
                 x = int(i)
                 y = 1+x
                 kpcar1= kpca_ratio(net[x])
                 kpcar2= kpca_ratio(net[y])
                 kpca_chg.append(np.divide(kpcar2,kpcar1))
             kpca_chgpad = np.zeros(len(kpca_chg)+1);
             kpca_chgpad[1:] = kpca_chg

             return kpca_chgpad
```

```
In [44]: def nmf_att(net):
             nmf_chg = []

             nmf = NMF(init='nndsvda',solver='cd', random_state=0, l1_ratio=1)
             for i in range(len(net)-1):
                 x = int(i)
                 y = 1+x
                 m1= abs(nx.normalized_laplacian_matrix(net[x]).todense())
                 m2= abs(nx.normalized_laplacian_matrix(net[y]).todense())
                 nmf1 = norm(nmf.fit_transform(m1))
                 nmf2 = norm(nmf.fit_transform(m2))
                 nmf_chg.append(np.divide(nmf2,nmf1))
             nmf_chgpad = np.zeros(len(nmf_chg)+1);
             nmf_chgpad[1:] = nmf_chg

             return nmf_chgpad
```

```
In [45]: 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

         def subgraph_stat(net1,net2):
             net1_int_net2 = net1.copy()
             net1_int_net2.remove_nodes_from(n for n in net1 if n not in net2)
             net1_u_net2 = nx.disjoint_union(net1, net2)
             int_adjmat = nx.normalized_laplacian_matrix(net1_int_net2).todense()
             uni_adjmat = nx.normalized_laplacian_matrix(net1_u_net2).todense()
             int_adjmat_pad = pad_shape(int_adjmat,uni_adjmat)

             Ct = np.divide(norm(int_adjmat_pad),norm(uni_adjmat))

             return Ct

         def zeta(n):

             Ct_m = []
             for i in range(len(n)-1):
                 x = int(i)
                 y = x +1
                 Ct_m.append(subgraph_stat(n[x],n[y]))
             Ct_m_pad = np.zeros(len(Ct_m)+1);
             Ct_m_pad[1:] = Ct_m
             return Ct_m_pad
```

```
In [46]: def music_att(n):
             #music attributes
             f = sc.fftpack.rfft(nx.normalized_laplacian_matrix(n).todense()).mean
             zcr = librosa.feature.zero_crossing_rate(f)[0,0]
             spc = librosa.feature.spectral_centroid(f)[0,0]

             return [zcr,spc]
```

```
In [47]: def norm_fabel(x):
```

```
             import abel
             fabel_att =[]
             for n in x:
                 tmp = nx.normalized_laplacian_matrix(n).todense()
                 tmp2 = sc.fftpack.fft2(tmp)
                 fabel2 = abel.Transform(tmp2, direction='forward', method='direct'
                 mag = np.sqrt(np.square(np.real(fabel2)) + np.square(np.imag(fabel
                 fabel_att.append(norm(mag))

             return np.log10(fabel_att)

In [48]: def cal_avg_atts(x):

             #define attributes placeholders here
             deg_y = []
             bet_y = []
             clo_y = []
             load_y = []
             eig_y = []
             katz_y = []
             den_y = []
             algc_y = []
             cluscof_y = []
             IA_y = []
             IP_y = []
             IF_y = []
             P_y = []
             dIA_y = []
             d2IA_y =[]
             IAcc_y = []
             cosIP_y = []
             IA_weit_IF_y = []
             IA_weit_IP_y = []
             psd_y = []
             rd_y = []
             zcr_y = []
             spc_y = []
             statr_y = []
             meank_y = []

             #matrix decompostion attributes
             zeta_y = zeta(x)
             nmf_ratio_y = nmf_att(x)
             kpca_chg_y= kpca_att(x)
             prop_members_chg_1_zeta = 1-zeta_y
             fabel = norm_fabel(x)

             for n in x:
```

```
    deg = np.mean(sorted(set(nx.degree_centrality(n).values())))
    bet = np.mean(sorted(set(nx.betweenness_centrality(n).values())))
    clo = np.mean(sorted(set(nx.closeness_centrality(n).values())))
    katz = np.mean(sorted(set(nx.katz_centrality_numpy(n).values())))
    eig = np.mean(sorted(set(nx.eigenvector_centrality_numpy(n).values
    load = np.mean(sorted(set(nx.degree_centrality(n).values())))
    den = nx.density(n)
    algc = nx.algebraic_connectivity(n)
    clustcof = nx.average_clustering(n)

    #all network metrics
    deg_y.append(deg), bet_y.append(bet), clo_y.append(clo), load_y.ap
    den_y.append(den),  algc_y.append(algc), cluscof_y.append(clustcof

    #complex trace attributes
    IA,IP,IF,P, dIA,d2IA, IAcc,cosIP,IA_weit_IF,IA_weit_IP = calc_seis
    IA_y.append(IA.mean())
    IP_y.append(IP.mean())
    IF_y.append(IF.mean())
    P_y.append(P.mean())
    dIA_y.append(dIA.mean())
    d2IA_y.append(d2IA.mean())
    IAcc_y.append(IAcc.mean())
    cosIP_y.append(cosIP.mean())
    IA_weit_IF_y.append(IA_weit_IF.mean())
    IA_weit_IP_y.append(IA_weit_IP.mean())

    psd, _ = plt.psd(nx.laplacian_matrix(n).todense());
    plt.close()
    psd_y.append(psd.mean())

    zcr, spc = music_att(n)
    zcr_y.append(zcr)
    spc_y.append(np.log10(spc))

    #matrix attributes
    rdm = resistance_distance(n)
    rd_y.append(rdm)

    statrat = stationarity_ratio(n)
    statr_y.append(statrat)

    meank = curvature(n)
    meank_y.append(meank)



colnames = ['AvgDeg','AvgBet','AvgClo','AvgLoad','AvgKatz','AvgDensity
```

```
                'AvgEig' ,'InstAmp','InstPhase','InstFreq','Power','dInstAm
                'A_wt_IF','A_wt_IP','PowerSpecDen','ResDist','ZeroCrossRate

    attvol_y = pd.DataFrame([deg_y,bet_y, clo_y ,load_y ,katz_y ,den_y, al
                 P_y ,dIA_y ,d2IA_y,IAcc_y ,cosIP_y ,IA_weit_IF_y ,IA_weit
                 statr_y , meank_y]).T

    attvol_y.columns = colnames
    attvol_y['SubgraphStat']=zeta_y
    attvol_y['1-Zeta'] = prop_members_chg_1_zeta
    attvol_y['LogKPCARatioChg'] = np.log10(kpca_chg_y)
    attvol_y.LogKPCARatioChg[0]=0
    attvol_y['NormNMFRatioChg']= nmf_ratio_y
    attvol_y['NormFAbel'] = fabel
    attvol_y_sc = attvol_y.apply(lambda x: minmax_scale(x, feature_range=

    return attvol_y_sc
```

```
In [49]: attvol_y = cal_avg_atts(all_year_G)
```

```
/home/arshad/anaconda3/lib/python3.5/site-packages/abel/transform.py:341: ComplexWa
  self.IM = self.IM.astype('float64')
```

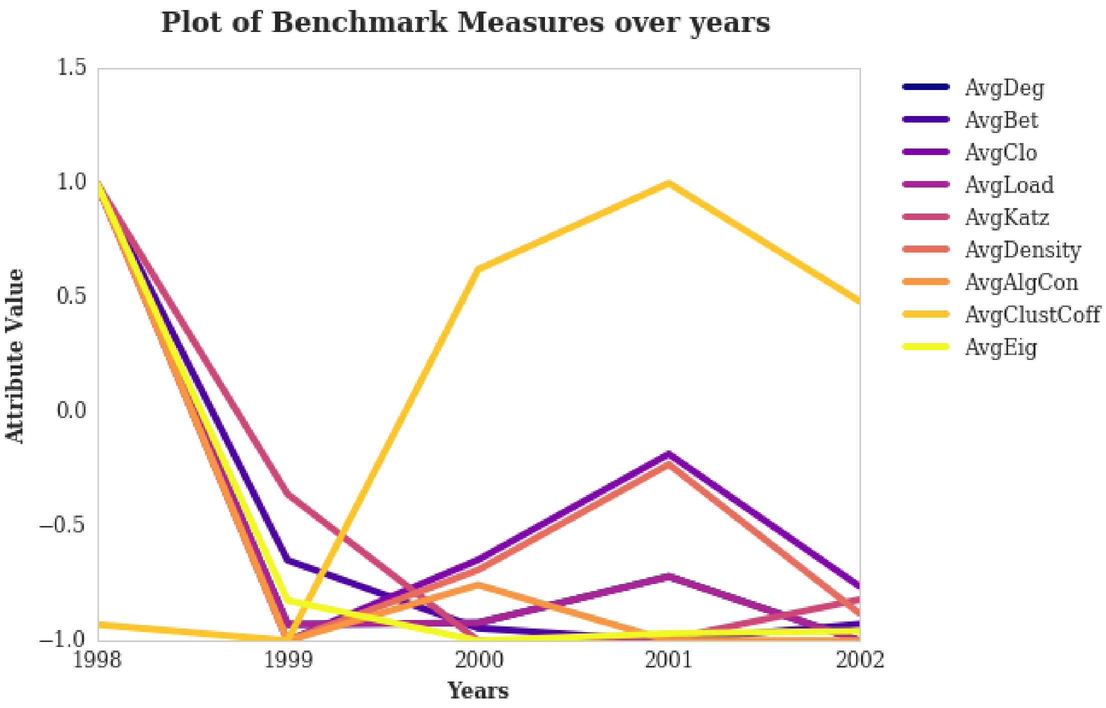
```
In [50]: attvol_m = cal_avg_atts(all_month_G)
```

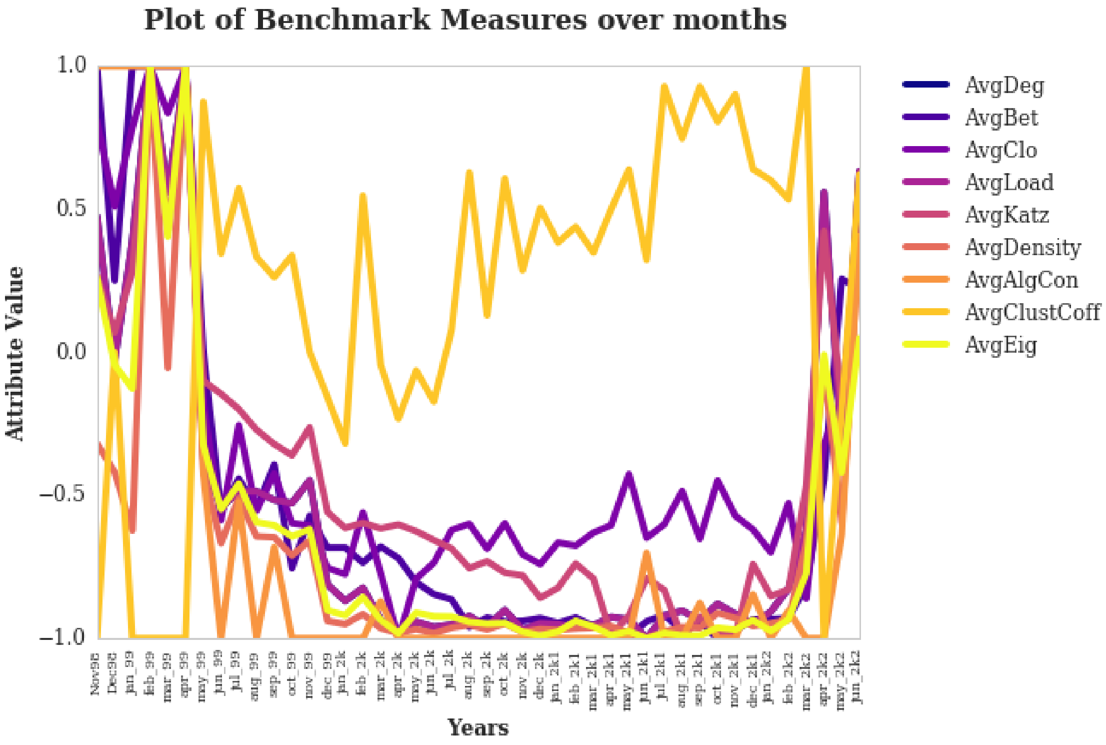
```
/home/arshad/anaconda3/lib/python3.5/site-packages/abel/transform.py:341: ComplexWa
  self.IM = self.IM.astype('float64')
```

6 Overview Plots

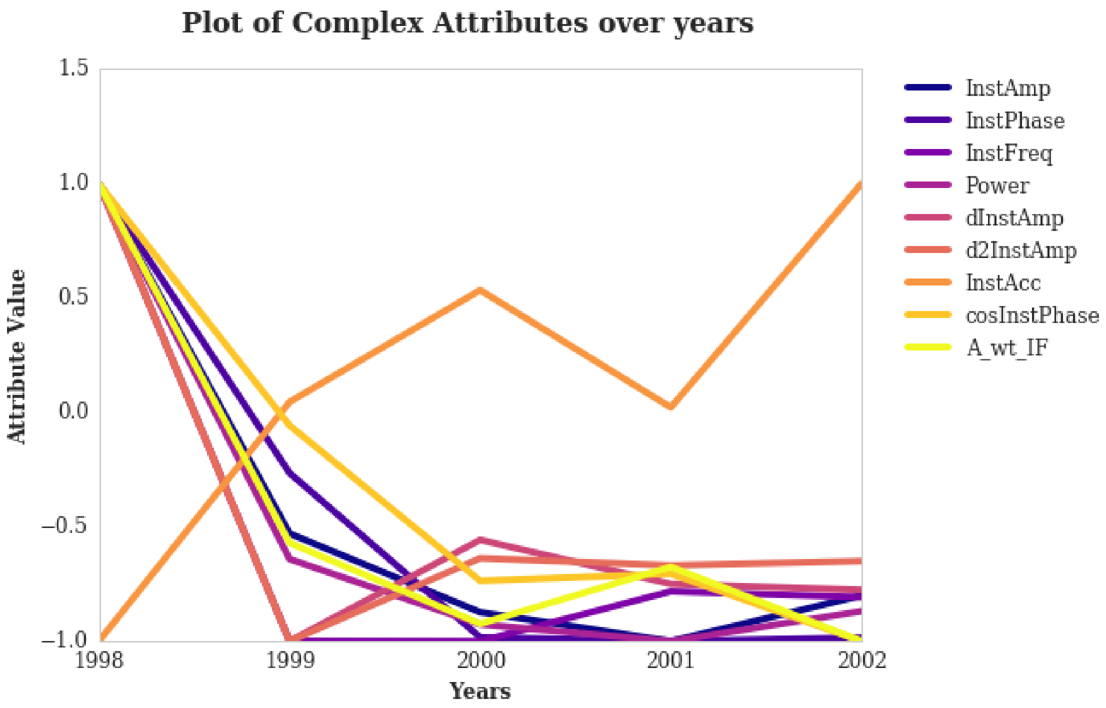
```
In [153]: attvol_y.iloc[:,:9].plot.line(use_index=True,cmap='plasma')
          plt.xlabel("Years", fontsize=12)
          plt.ylabel("Attribute Value", fontsize=12)
          plt.legend(fontsize=12, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
          plt.xticks(np.arange(len(years)), years,fontsize=12)
          plt.yticks(fontsize=12)
          plt.suptitle("Plot of Benchmark Measures over years", fontsize=16)

          attvol_m.iloc[:,:9].plot.line(use_index=True,cmap='plasma', rot=90)
          plt.xlabel("Years", fontsize=12)
          plt.ylabel("Attribute Value", fontsize=12)
          plt.legend(fontsize=12, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
          plt.xticks(np.arange(len(months)), months)
          plt.yticks(fontsize=12)
          plt.suptitle("Plot of Benchmark Measures over months", fontsize=16)
```

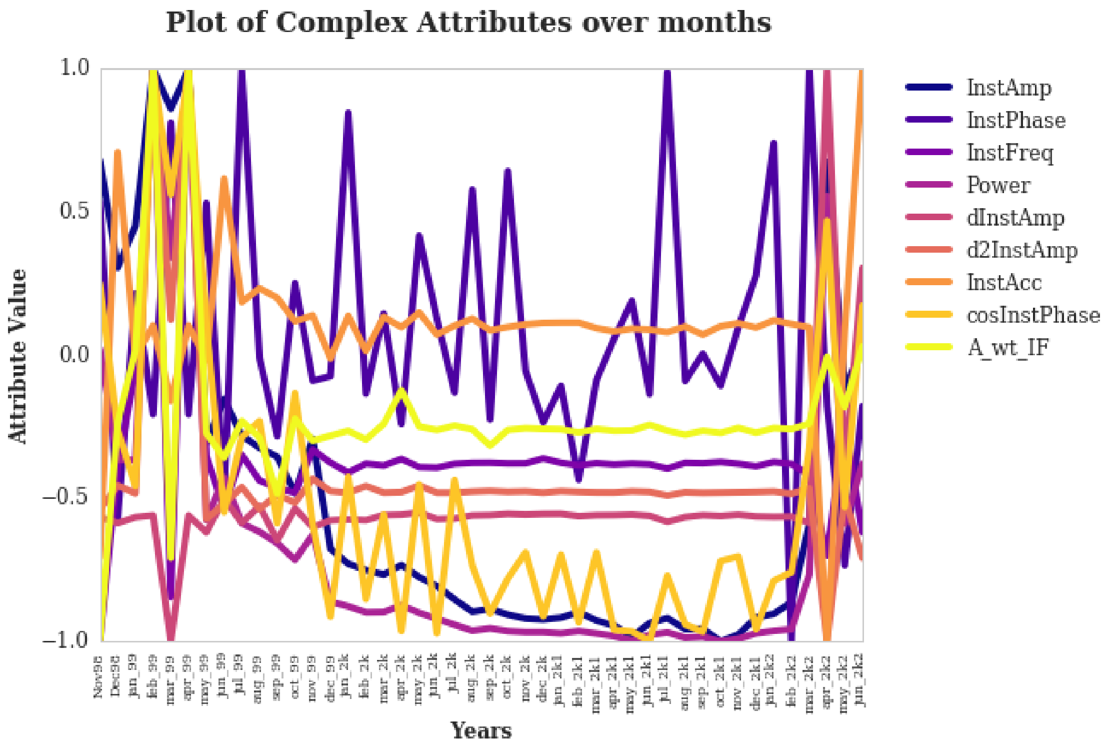
```
Out[153]: <matplotlib.text.Text at 0x7f625cd4bfd0>
```


Plot of Benchmark Measures over years
1.5
1.0
0.5
0.0
−0.5
−1.0
Attribute Value
1998
1999
2000
2001
2002
Years
AvgDeg
AvgBet
AvgClo
AvgLoad
AvgKatz
AvgDensity
AvgAlgCon
AvgClustCoff
AvgEig

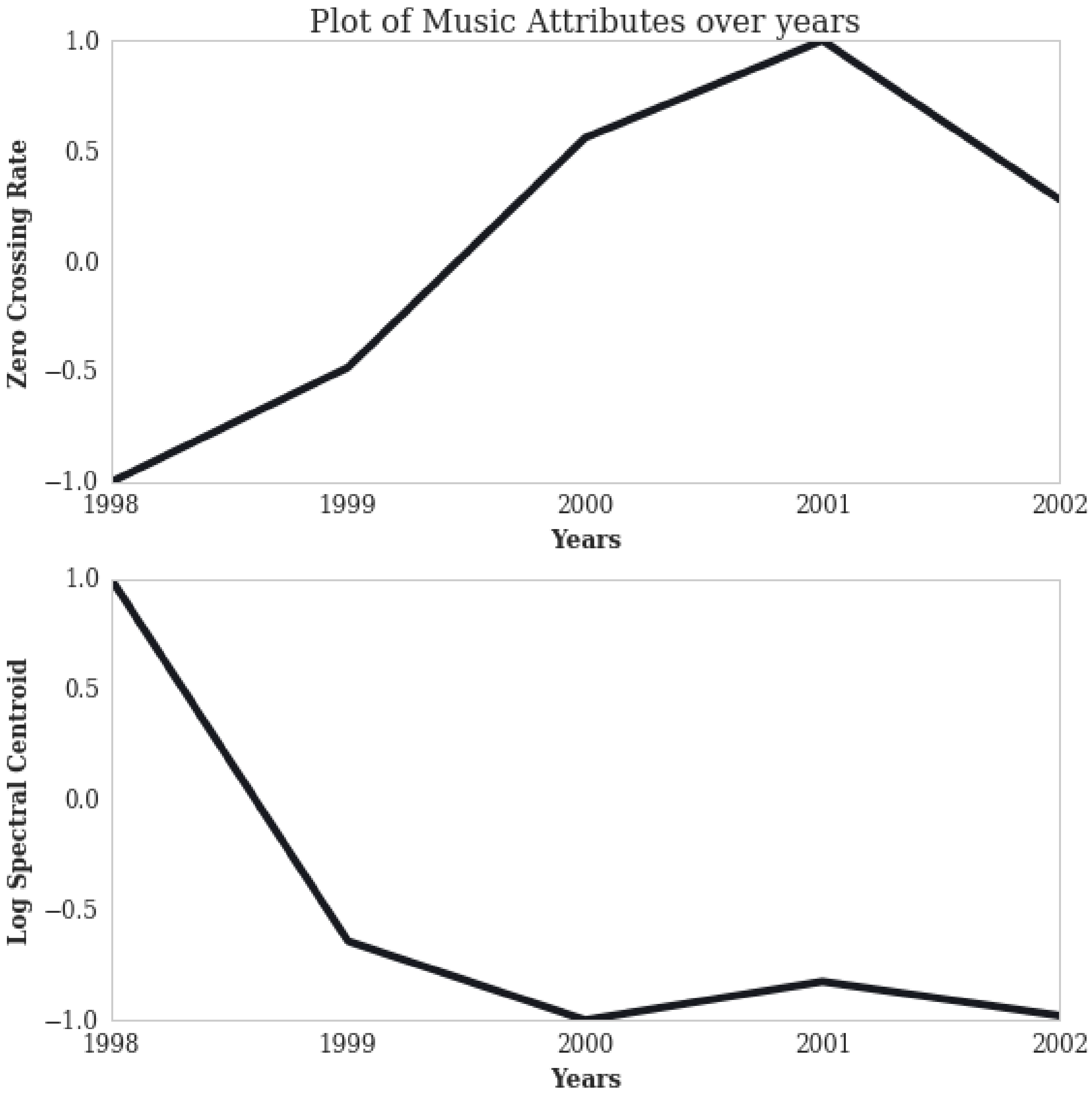

Plot of Benchmark Measures over months
1.0
0.5
0.0
−0.5
−1.0
Attribute Value
Nov98
Dec98
jan_99
feb_99
mar_99
apr_99
may_99
jun_99
jul_99
aug_99
sep_99
oct_99
nov_99
dec_99
jan_2k
feb_2k
mar_2k
apr_2k
may_2k
jun_2k
jul_2k
aug_2k
sep_2k
oct_2k
nov_2k
dec_2k
jan_2k1
feb_2k1
mar_2k1
apr_2k1
may_2k1
jun_2k1
jul_2k1
aug_2k1
sep_2k1
oct_2k1
nov_2k1
dec_2k1
jan_2k2
feb_2k2
mar_2k2
apr_2k2
may_2k2
jun_2k2
Years
AvgDeg
AvgBet
AvgClo
AvgLoad
AvgKatz
AvgDensity
AvgAlgCon
AvgClustCoff
AvgEig

```
In [155]: attvol_y.iloc[:,9:18].plot.line(use_index=True,cmap='plasma')
          plt.xlabel("Years", fontsize=12)
          plt.ylabel("Attribute Value", fontsize=12)
          plt.legend(fontsize=12, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
          plt.xticks(np.arange(len(years)), years,fontsize=12)
          plt.yticks(fontsize=12)
          plt.suptitle("Plot of Complex Attributes over years", fontsize=16)
          plt.autoscale()

          attvol_m.iloc[:,9:18].plot.line(use_index=True,cmap='plasma', rot=90)
          plt.xlabel("Years", fontsize=12)
          plt.ylabel("Attribute Value", fontsize=12)
          plt.legend(fontsize=12, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
          plt.xticks(np.arange(len(months)), months)
          plt.yticks(fontsize=12)
          plt.suptitle("Plot of Complex Attributes over months", fontsize=16)
          plt.autoscale()
```

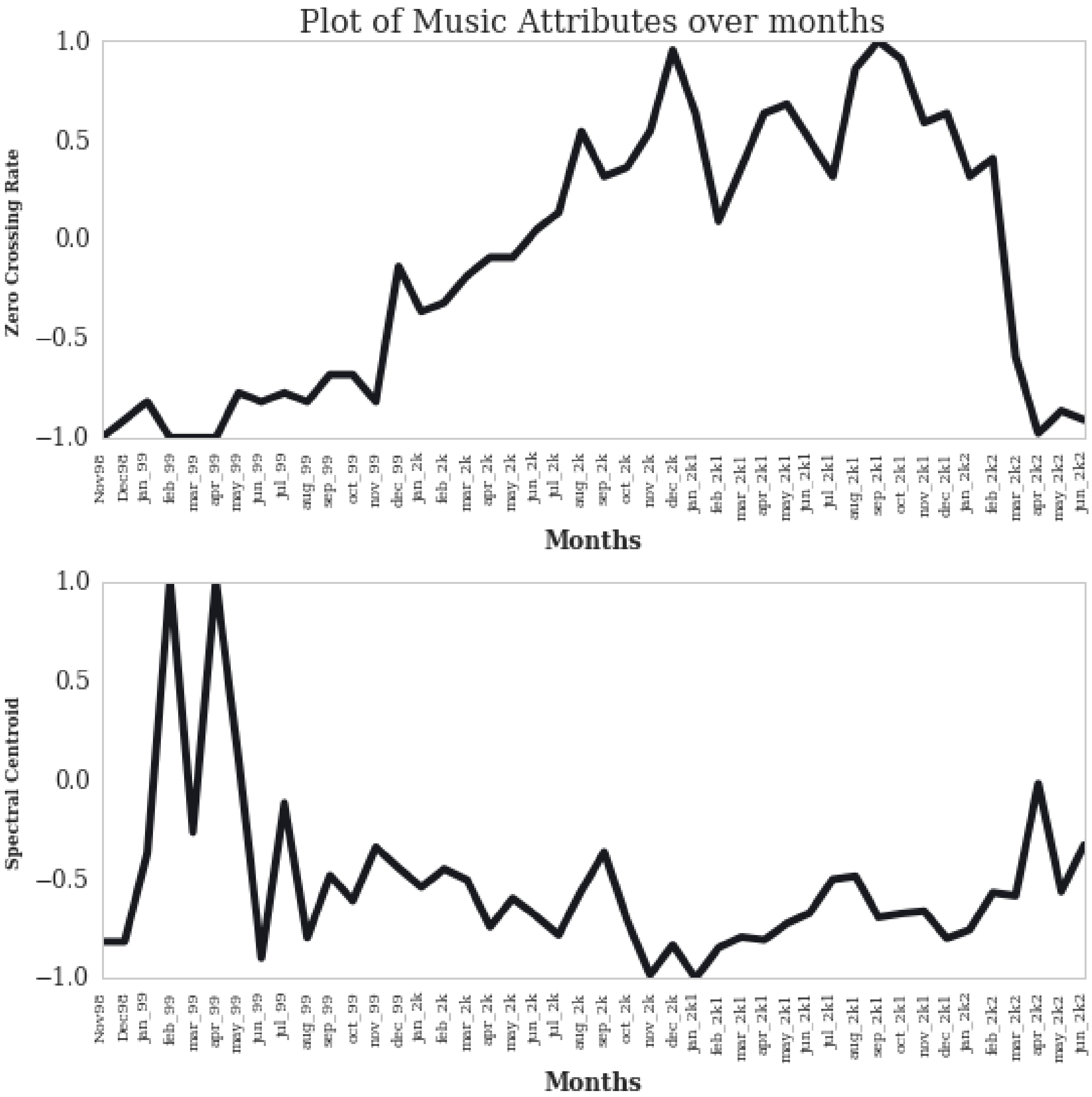




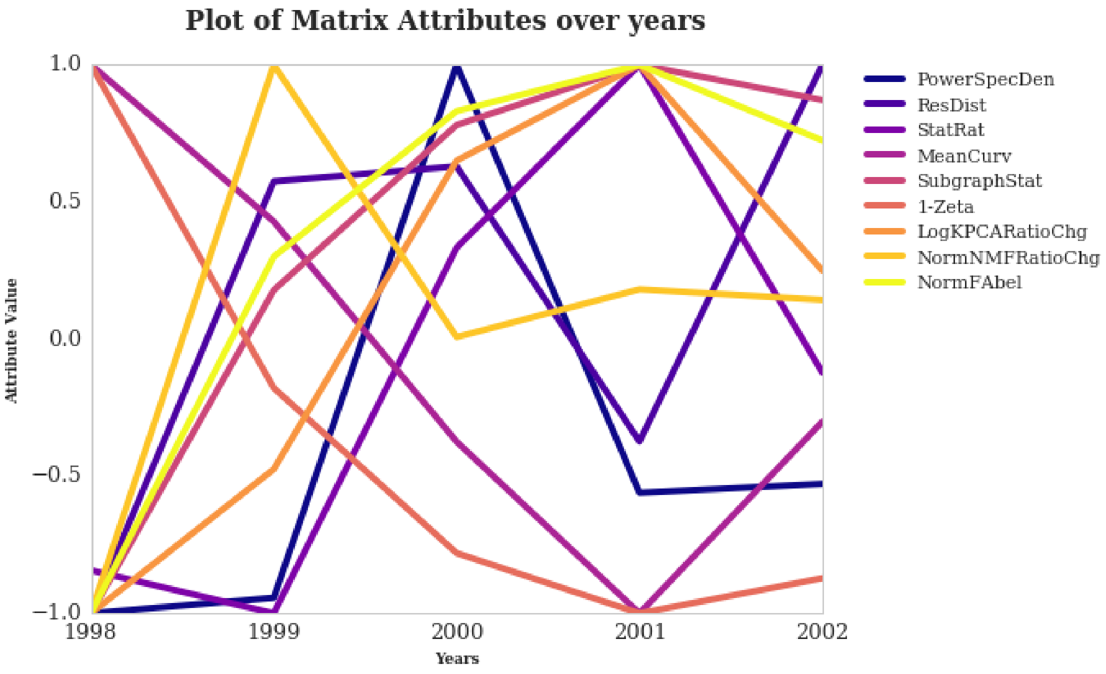
```
In [156]: plt.figure(figsize=(8,8))

          plt.subplot(2,1,1)
          plt.title("Plot of Music Attributes over years", fontsize=16)
          plt.plot(attvol_y.ZeroCrossRate)
          plt.xlabel("Years", fontsize=12)
          plt.ylabel("Zero Crossing Rate", fontsize=12)
          plt.xticks(np.arange(len(years)),years,fontsize=12)
          plt.yticks(fontsize=12)
          plt.autoscale()

          plt.subplot(2,1,2)
          plt.plot(attvol_y.LogSpecCentroid)
          plt.xlabel("Years", fontsize=12)
          plt.ylabel("Log Spectral Centroid", fontsize=12)
          plt.xticks(np.arange(len(years)),years,fontsize=12)
          plt.yticks(fontsize=12)
          plt.tight_layout()
          plt.savefig('images/musicatt_yrs.png')
```

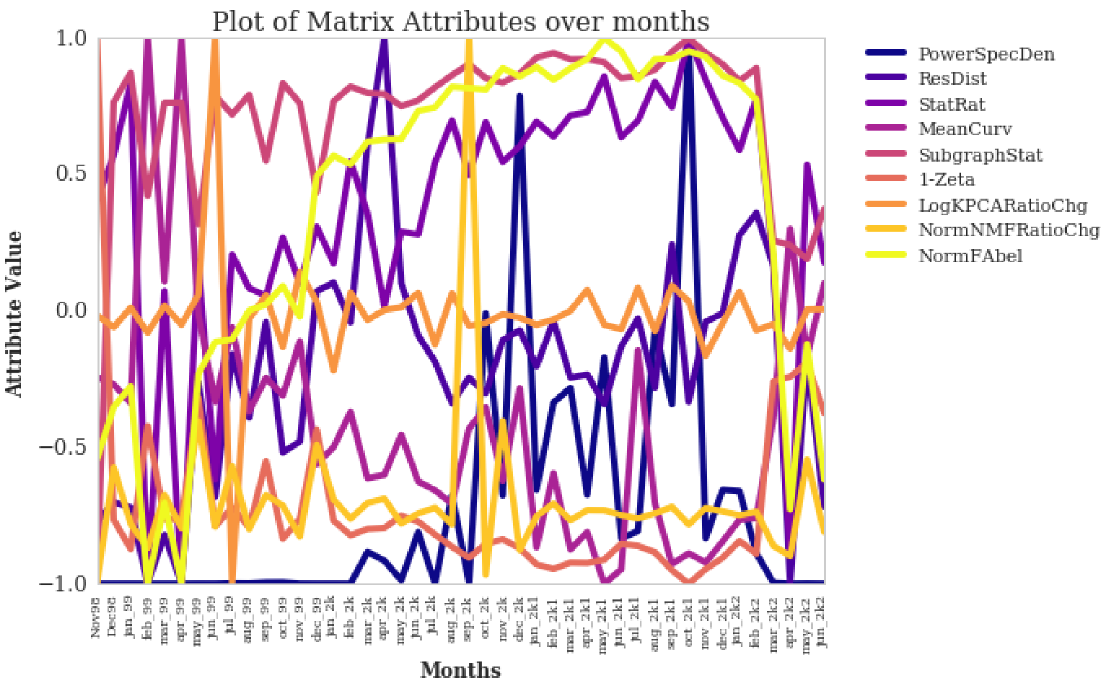


```
In [157]: plt.figure(figsize=(8,8))

          plt.subplot(2,1,1)
          plt.title("Plot of Music Attributes over months", fontsize=16)
          plt.plot(attvol_m.ZeroCrossRate.values)
          plt.xlabel("Months", fontsize=12)
          plt.ylabel("Zero Crossing Rate")
          plt.xticks(np.arange(len(months)), months, rotation=90)
          plt.yticks(fontsize=12)
          plt.autoscale()

          plt.subplot(2,1,2)
          plt.plot(attvol_m.LogSpecCentroid.values)
          plt.xlabel("Months", fontsize=12)
          plt.ylabel("Spectral Centroid")
          plt.xticks(np.arange(len(months)), months, rotation=90)
          plt.yticks(fontsize=12)
          plt.autoscale()
          plt.tight_layout()
          plt.savefig('images/musicatt_mth.png')
```

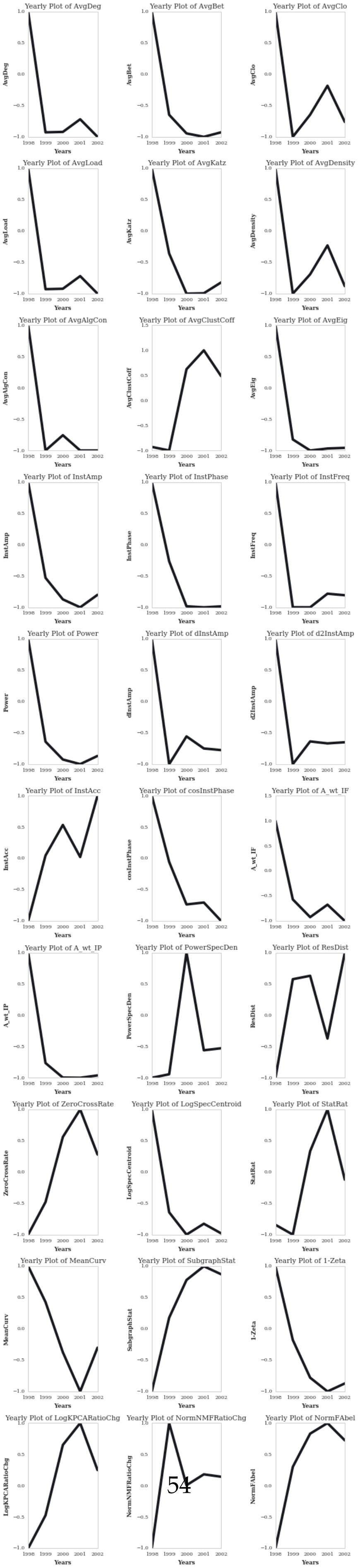


```
In [160]: attvol_y.drop(['ZeroCrossRate','LogSpecCentroid'], axis=1).iloc[:,19:].pl
          plt.xlabel("Years")
          plt.ylabel("Attribute Value")
          plt.legend(fontsize=11, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
          plt.xticks(np.arange(len(years)),years,fontsize=13)
          plt.yticks(fontsize=13)
          plt.suptitle("Plot of Matrix Attributes over years", fontsize=16)
          plt.autoscale()

          attvol_m.drop(['ZeroCrossRate','LogSpecCentroid'], axis=1).iloc[:,19:].pl
          plt.xlabel("Months", fontsize=12)
          plt.ylabel("Attribute Value", fontsize=12)
          plt.legend(fontsize=11, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.
          plt.xticks(np.arange(len(months)), months)
          plt.yticks(fontsize=13)
          plt.title("Plot of Matrix Attributes over months", fontsize=16)
          plt.autoscale()
```

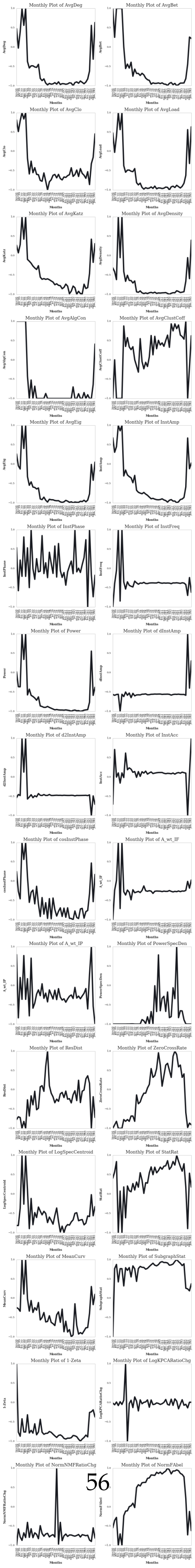




```
In [56]: all_cols = list(attvol_m.columns)

In [57]: attvol_y.shape

Out[57]: (5, 30)

In [162]: plt.figure(figsize=(8,38))

          for i in range(len(all_cols)):
              plt.subplot(11,3,i+1)
              attvol_y.iloc[:,i].plot.line(use_index=True)
              plt.xlabel("Years")
              plt.ylabel(all_cols[i])
              plt.xticks(np.arange(len(years)), years)
              plt.title("Yearly Plot of " +all_cols[i],fontsize=11)
              plt.tight_layout()
          plt.savefig('images/avg_allatt_yrs.png')
```


Yearly Plot of AvgDeg
AvgDeg
Yearly Plot of AvgBet
AvgBet
Yearly Plot of AvgClo
AvgClo
Yearly Plot of AvgLoad
AvgLoad
Yearly Plot of AvgKatz
AvgKatz
Yearly Plot of AvgDensity
AvgDensity
Yearly Plot of AvgAlgCon
AvgAlgCon
Yearly Plot of AvgClustCoff
AvgClustCoff
Yearly Plot of AvgEig
AvgEig
Yearly Plot of InstAmp
InstAmp
Yearly Plot of InstPhase
InstPhase
Yearly Plot of InstFreq
InstFreq
Yearly Plot of Power
Power
Yearly Plot of dInstAmp
dInstAmp
Yearly Plot of d2InstAmp
d2InstAmp
Yearly Plot of InstAcc
InstAcc
Yearly Plot of cosInstPhase
cosInstPhase
Yearly Plot of A_wt_IF
A_wt_IF
Yearly Plot of A_wt_IP
A_wt_IP
Yearly Plot of PowerSpecDen
PowerSpecDen
Yearly Plot of ResDist
ResDist
Yearly Plot of ZeroCrossRate
ZeroCrossRate
Yearly Plot of LogSpecCentroid
LogSpecCentroid
Yearly Plot of StatRat
StatRat
Yearly Plot of MeanCurv
MeanCurv
Yearly Plot of SubgraphStat
SubgraphStat
Yearly Plot of 1-Zeta
1-Zeta
Yearly Plot of LogKPCARatioChg
LogKPCARatioChg
Yearly Plot of NormNMFRatioChg
NormNMFRatioChg
Yearly Plot of NormFAbel
NormFAbel
Years
1998
1999
2000
2001
2002
1.5
1.0
0.5
0.0
−0.5
−1.0

```
In [163]: plt.figure(figsize=(8,64))

          for i in range(len(all_cols)):
              plt.subplot(15,2,i+1)
              attvol_m.iloc[:,i].plot.line(rot=90)
              plt.xlabel("Months")
              plt.ylabel(all_cols[i])
              plt.xticks(np.arange(len(months)), months)
              plt.title("Monthly Plot of " +all_cols[i],fontsize=13)
              plt.tight_layout()
          plt.savefig('images/avg_allatt_mth.png')
```

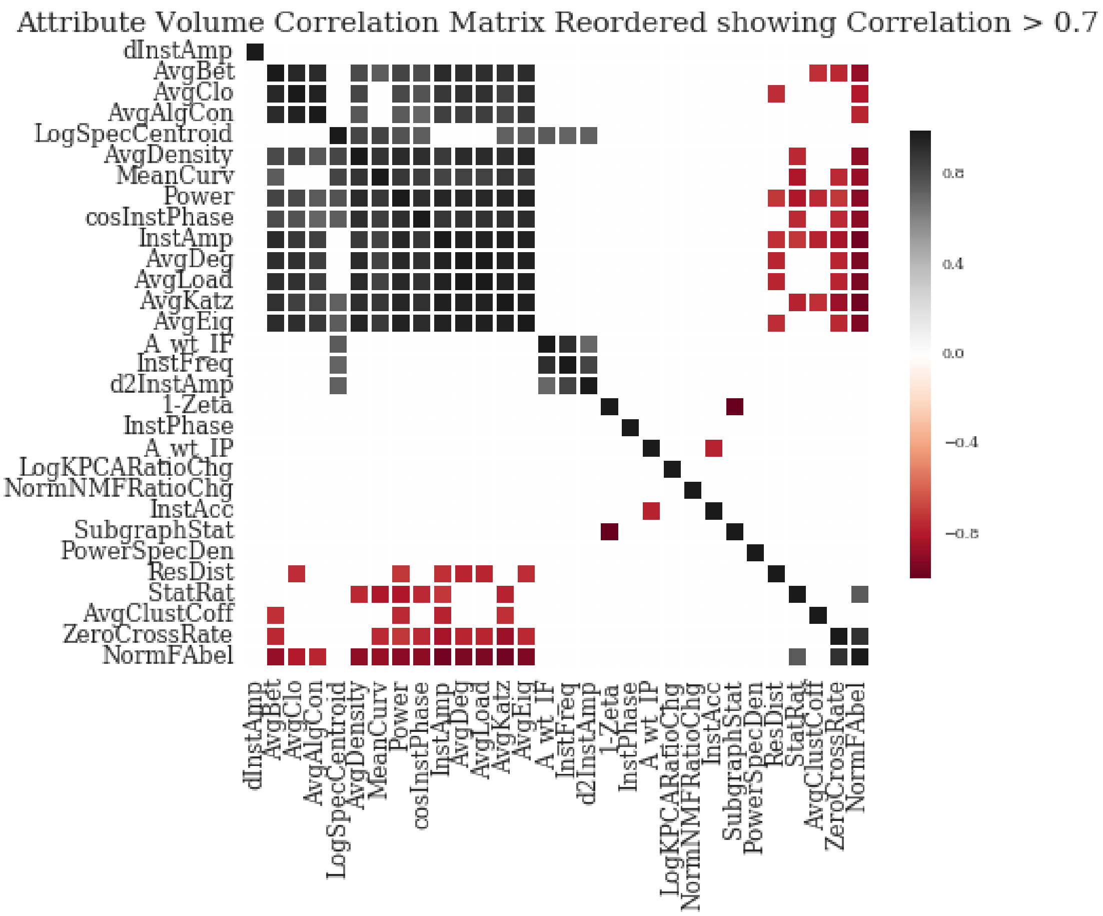

Monthly Plot of AvgDeg
Monthly Plot of AvgBet
Monthly Plot of AvgClo
Monthly Plot of AvgLoad
Monthly Plot of AvgKatz
Monthly Plot of AvgDensity
Monthly Plot of AvgAlgCon
Monthly Plot of AvgClustCoff
Monthly Plot of AvgEig
Monthly Plot of InstAmp
Monthly Plot of InstPhase
Monthly Plot of InstFreq
Monthly Plot of Power
Monthly Plot of dInstAmp
Monthly Plot of d2InstAmp
Monthly Plot of InstAcc
Monthly Plot of cosInstPhase
Monthly Plot of A_wt_IF
Monthly Plot of A_wt_IP
Monthly Plot of PowerSpecDen
Monthly Plot of ResDist
Monthly Plot of ZeroCrossRate
Monthly Plot of LogSpecCentroid
Monthly Plot of StatRat
Monthly Plot of MeanCurv
Monthly Plot of SubgraphStat
Monthly Plot of 1-Zeta
Monthly Plot of LogKPCARatioChg
Monthly Plot of NormNMFRatioChg
Monthly Plot of NormFAbel
1.0
0.5
0.0
−0.5
−1.0
Months

7 Correlation Analysis

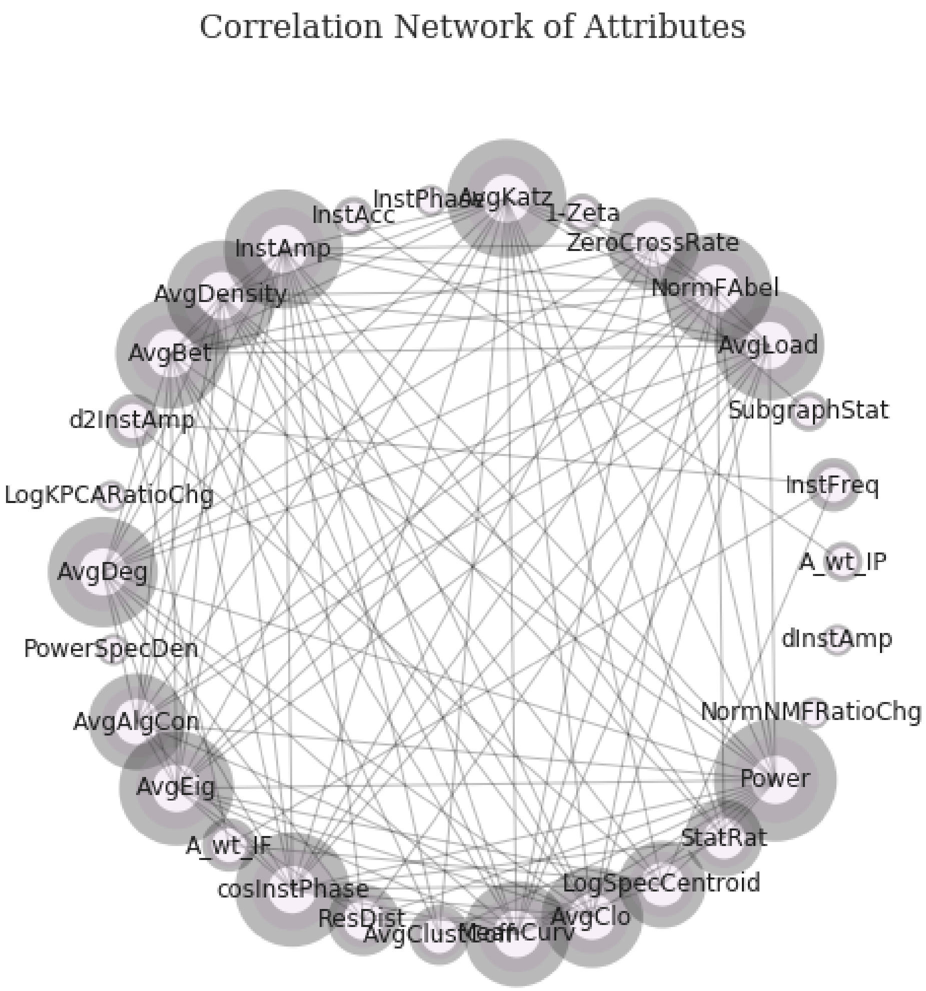
```
In [165]: corr_m = attvol_m.corr()
          g = sns.clustermap(corr_m, metric='chebyshev')
          plt.close()
```

7.1 Correlation > 0.7

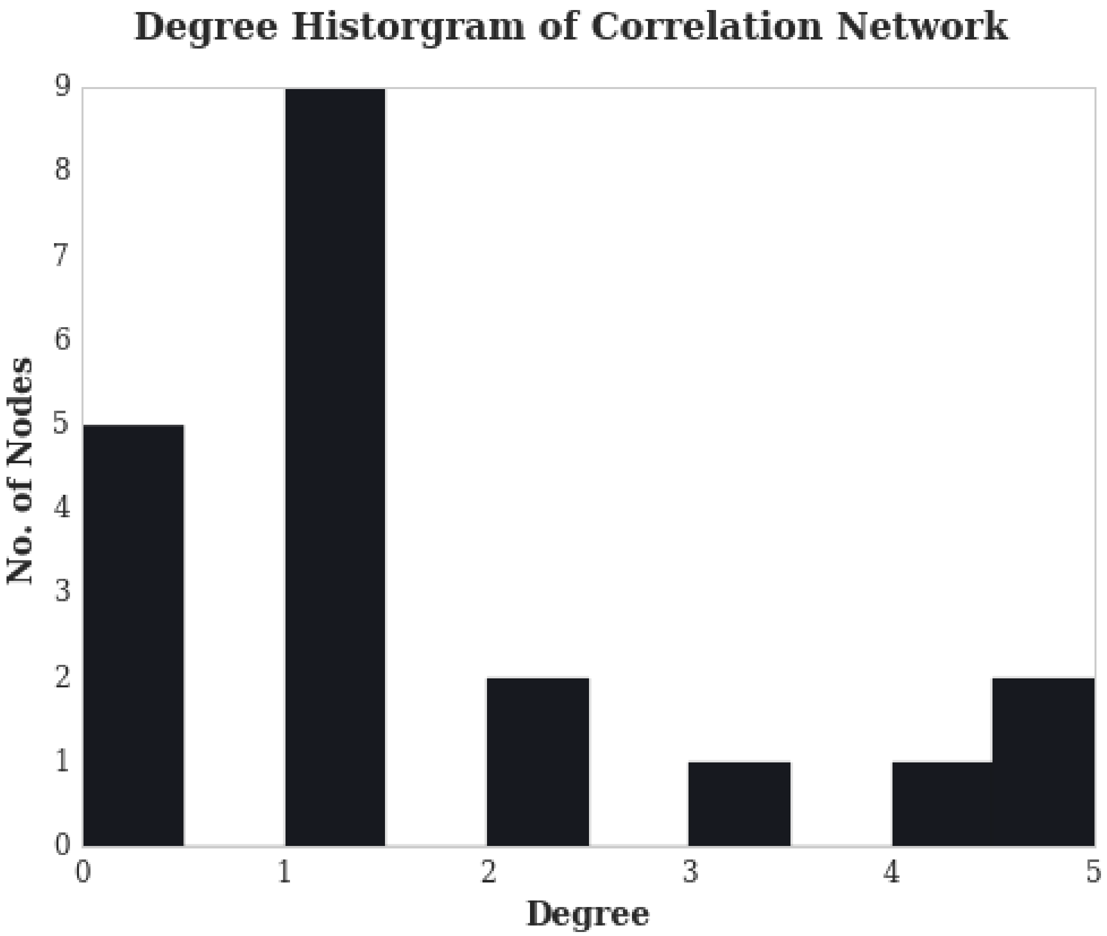
```
In [166]: threshold = 0.7
          corr_m.values[np.where(abs(corr_m.values) < threshold)] = 0

In [167]: plt.figure(figsize=(8,8))
          sns.heatmap(corr_m.iloc[g.dendrogram_col.reordered_ind, g.dendrogram_row.
                      cmap='RdGy',robust=True, fmt='d', linewidths=1, square=True,\
                      cbar_kws={'orientation':'vertical', 'shrink':0.5})
          plt.title("Attribute Volume Correlation Matrix Reordered showing Correlat
          plt.xticks(rotation=90, fontsize=12)
          plt.yticks(rotation=360, fontsize=12)
          plt.tight_layout()
          plt.savefig('images/reordered_corrmat.png')
```



```
In [169]: names = corr_m.index.values
          G_corr = nx.Graph(corr_m.values)
          pos=nx.fruchterman_reingold_layout(G_corr, iterations=1000, k=200)

In [172]: #ref: https://github.com/traims/correlation-based-networks/blob/master/cr
          components = nx.connected_components(G_corr)
          plt.figure(figsize=(8,8))
          plt.title("Correlation Network of Attributes", fontsize=16)
          plt.axis('off')
          for i in components:
              component = G_corr.subgraph(i)
              nx.draw_networkx(component, with_labels = True,node_size = [component
                  labels = dict([(x, names[x]) for x in component.nodes()]), pos=po
                                    edge_color='k', node_color='#E9CFEC', linewidths=[com
                                     fontsize=11)
          plt.tight_layout()
          plt.savefig('images/corrnet.png')
```



```
In [173]: plt.hist(nx.degree_histogram(G_corr))
          plt.suptitle("Degree Historgram of Correlation Network", fontsize=16)
          plt.xticks(fontsize=12)
          plt.yticks(fontsize=12)
          plt.xlabel("Degree", fontsize=14)
          plt.ylabel("No. of Nodes", fontsize=14)
          plt.savefig('images/corrnet_deghist.png')
```



8 Regression Analysis for Feature Ranking

```
In [66]: from sklearn.metrics import mean_squared_error
         from sklearn import ensemble

In [67]: def RMS(x, axis=None):
             rms = np.sqrt(np.mean(np.square(x), axis=axis))
             return rms

In [68]: X= attvol_m.drop(['AvgDeg'],axis=1)
         y = attvol_m.AvgDeg

In [70]: X[:5]

Out[70]:      AvgBet    AvgClo   AvgLoad   AvgKatz  AvgDensity  AvgAlgCon  \
         0  1.000000  0.810523  0.475524  0.267944   -0.318532        1.0
         1  0.250903  0.509903 -0.027972  0.069435   -0.419219        1.0
         2  1.000000  0.785809  0.388112  0.282044   -0.622808        1.0
```

```
3  1.000000  1.000000  1.000000  1.000000    1.000000        1.0
4  1.000000  0.835787  0.559441  0.423789   -0.054825        1.0

   AvgClustCoff    AvgEig   InstAmp  InstPhase    ...       ResDist  \
0     -1.000000  0.262262  0.679834   0.535455    ...     -0.774637
1     -0.000165 -0.044277  0.305093  -0.578269    ...     -0.701077
2     -1.000000 -0.127135  0.450396   0.214700    ...     -0.720470
3     -1.000000  1.000000  1.000000  -0.207114    ...     -1.000000
4     -1.000000  0.403511  0.860240   0.812100    ...     -0.820587

   ZeroCrossRate  LogSpecCentroid   StatRat  MeanCurv  SubgraphStat    1-Z
0      -1.000000        -0.815713  0.407864 -0.242635     -1.000000  1.000
1      -0.909091        -0.816057  0.570795 -0.269438      0.766117 -0.766
2      -0.818182        -0.371199  0.836955 -0.337965      0.873376 -0.873
3      -1.000000         1.000000 -1.000000  1.000000      0.421859 -0.421
4      -1.000000        -0.264125  0.074177  0.108395      0.763599 -0.763

   LogKPCARatioChg  NormNMFRatioChg  NormFAbel
0        -0.015620        -1.000000  -0.543745
1        -0.059551        -0.573875  -0.352246
2         0.012093        -0.780922  -0.276008
3        -0.079774        -0.869723  -1.000000
4         0.019315        -0.673657  -0.702049

[5 rows x 29 columns]
```

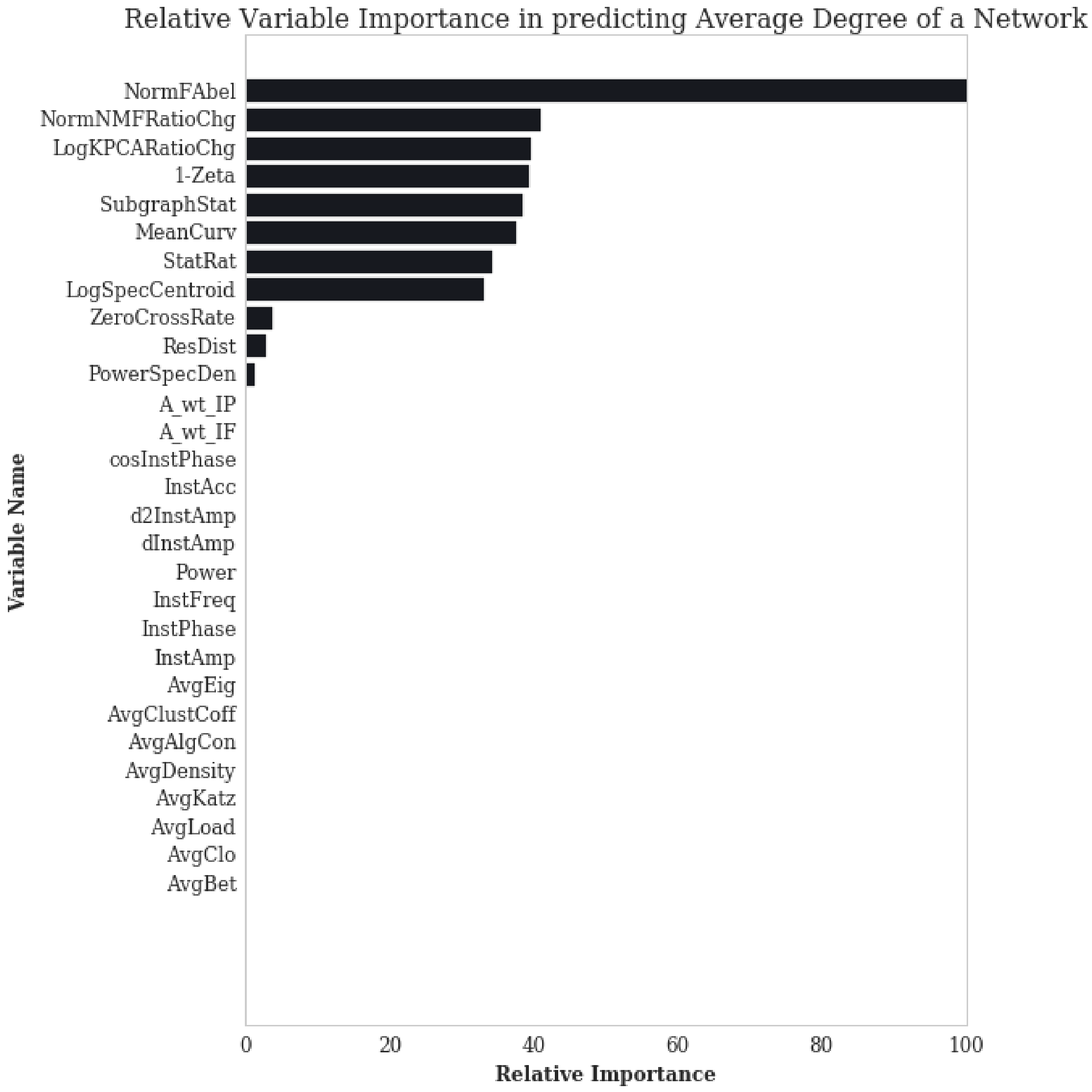
```
In [71]: X = X.astype(np.float32)
         offset = int(X.shape[0] * 0.5)
         X_train, y_train = X[:offset], y[:offset]
         X_test, y_test = X[offset:], y[offset:]
```

```
In [72]: params = {'n_estimators': 10, 'max_depth': 10, 'min_samples_split': 10,
                   'learning_rate': 0.1, 'loss': 'ls'}
         clf = ensemble.GradientBoostingRegressor(**params)
         clf.fit(X_train, y_train)
```

```
Out[72]: GradientBoostingRegressor(alpha=0.9, init=None, learning_rate=0.1, loss='l
                      max_depth=10, max_features=None, max_leaf_nodes=None,
                      min_samples_leaf=1, min_samples_split=10,
                      min_weight_fraction_leaf=0.0, n_estimators=10, presort='auto'
                      random_state=None, subsample=1.0, verbose=0, warm_start=False
```

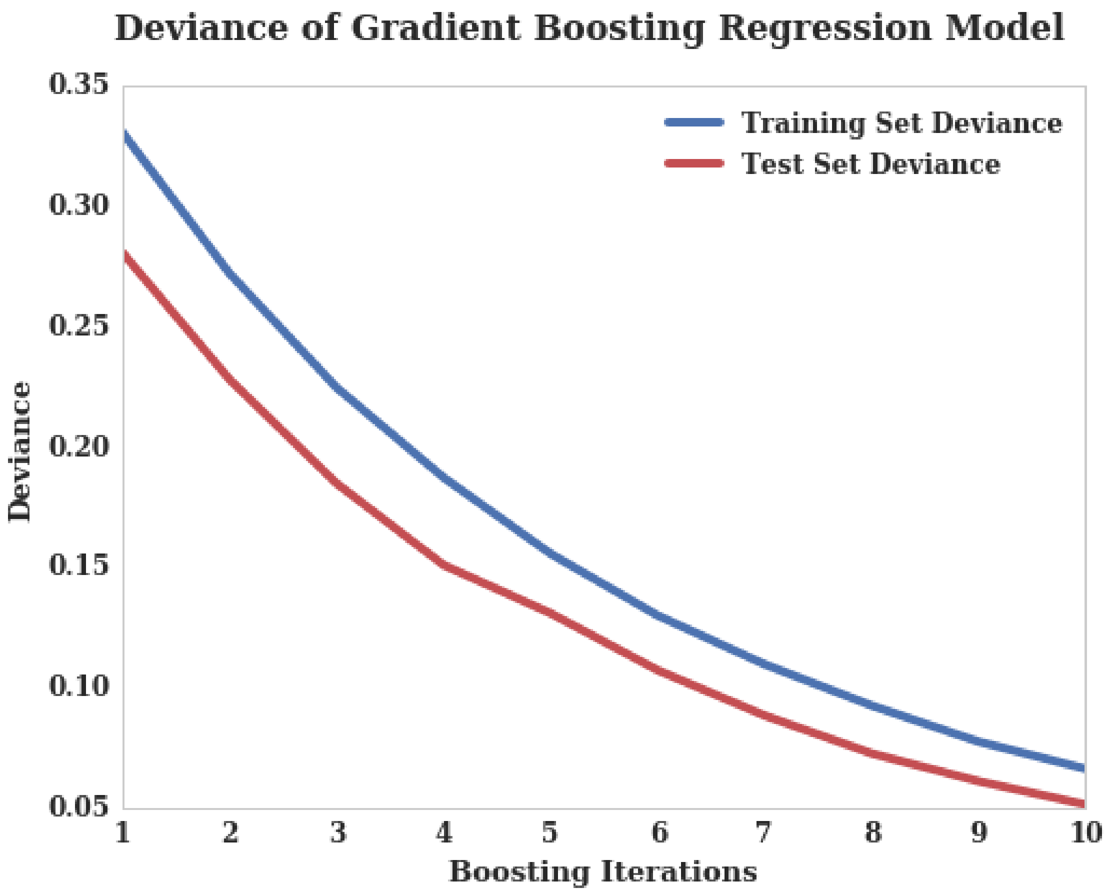
```
In [174]: feature_importance = clf.feature_importances_

          # make importances relative to max importance
          plt.figure(figsize=(8,11))
          feature_importance = 100.0 * (feature_importance / feature_importance.max
          sorted_idx = np.argsort(feature_importance)
          feature_names = list(X)
```

```
pos = np.arange(sorted_idx.shape[0])
plt.barh(pos, feature_importance[sorted_idx], align='center')
plt.yticks(pos, feature_names, fontsize=12)
plt.ylabel("Variable Name", fontsize=12)
plt.xlabel('Relative Importance', fontsize=12)
plt.title('Relative Variable Importance in predicting Average Degree of a
plt.xticks(fontsize=12)
plt.savefig('images/feature_ranking.png')
```



```
In [74]: test_score = np.zeros((params['n_estimators'],), dtype=np.float64)

         for i, y_pred in enumerate(clf.staged_predict(X_test)):
             test_score[i] = clf.loss_(y_test, y_pred)
```

```
plt.figure(figsize=(8, 6))
plt.suptitle('Deviance of Gradient Boosting Regression Model',fontsize=16)
plt.plot(np.arange(params['n_estimators']) + 1, clf.train_score_, 'b-',
         label='Training Set Deviance')
plt.plot(np.arange(params['n_estimators']) + 1, test_score, 'r-',
         label='Test Set Deviance')
plt.legend(loc=1, fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.xlabel('Boosting Iterations')
plt.ylabel('Deviance')
plt.savefig('images/reg_deviance.png')
```



```
In [75]: mse = mean_squared_error(y_test, clf.predict(X_test))
         print("Gradient Boosting MSE: " ,mse)

Gradient Boosting MSE:  0.0517054046745
```

9 Aggregation Measures

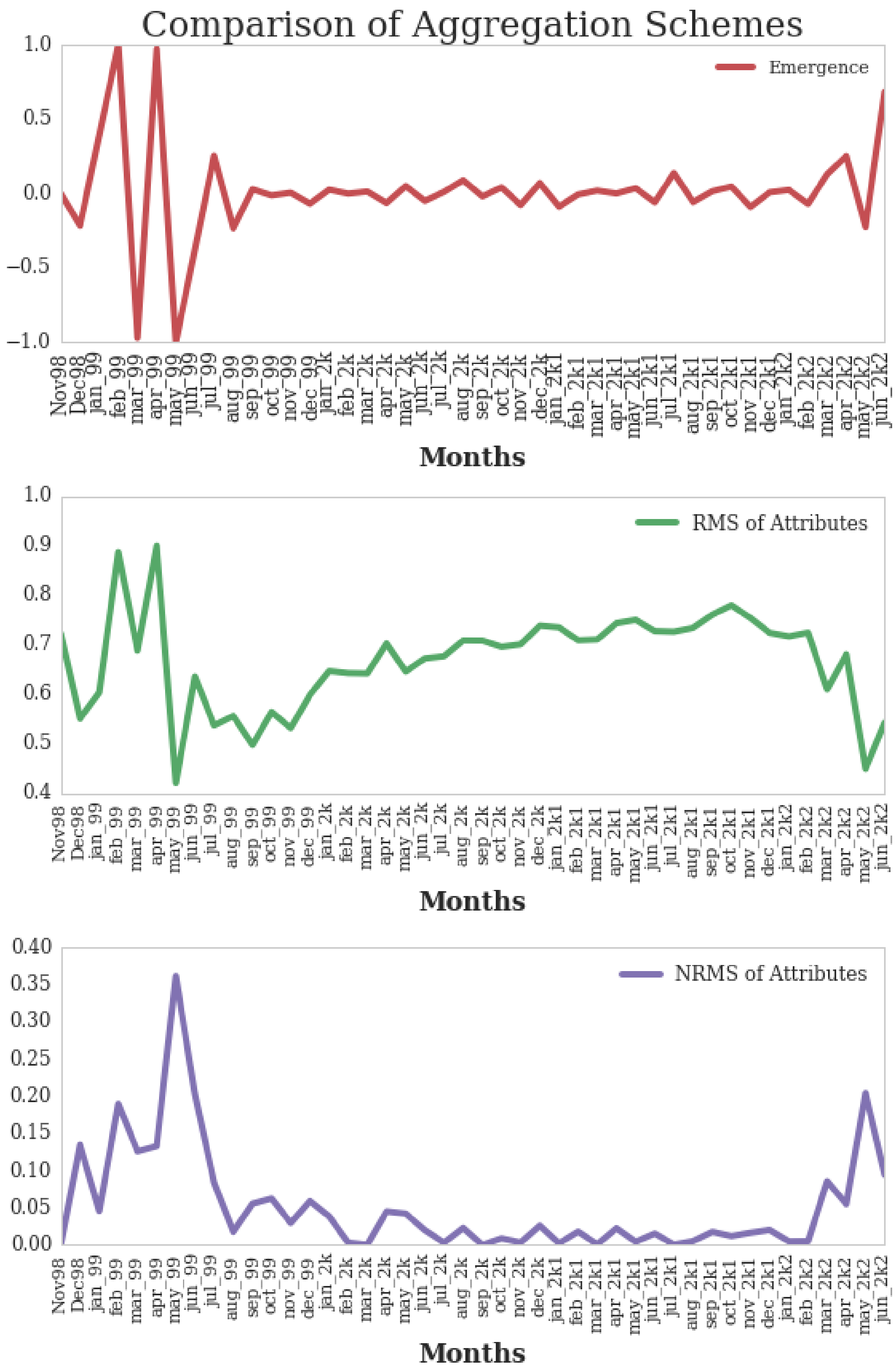
```
In [77]: persistence_m = attvol_m.mean(axis=1)/(attvol_m.shape[0]-1)
         persistence_y = attvol_y.mean(axis=1)/(attvol_y.shape[0]-1)
```

```
In [78]: def emergence(per):
             tmp = np.asarray(per)
             emerg = []
             for i in range(len(tmp)-1):
                 x= int(i)
                 y = x +1
                 #print(tmp[y], tmp[x])
                 if tmp[y]==tmp[x]:
                     emerg.append(0)
                 elif tmp[y] < (0) or tmp[x] < 0:
                     res = (tmp[y]-tmp[x])/(abs(tmp[y])+abs(tmp[x]))
                     emerg.append(res)
                 else:
                     res = (tmp[y]-tmp[x])/max([tmp[y],tmp[x]])
                     emerg.append(res)
             tmp2 = np.zeros(len(emerg)+1)
             tmp2[1:] = emerg

             return tmp2


         def NRMS(n):
             nrms = []
             for i in range(len(n)-1):
                 x= int(i)
                 y = x +1
                 a = n[x]
                 b = n[y]
                 nrms_ = np.divide((RMS(a-b)),(RMS(a)+RMS(b)))
                 nrms.append(nrms_)
             tmp2 = np.zeros(len(nrms)+1)
             tmp2[1:] = nrms

             return tmp2
```

```
In [79]: rms_m = attvol_m.apply(lambda x: RMS(x), axis=1)
         nrms_m = NRMS(rms_m)
         emerg_m = emergence(persistence_m)
```

```
In [80]: emerg_y = emergence(persistence_y)
```

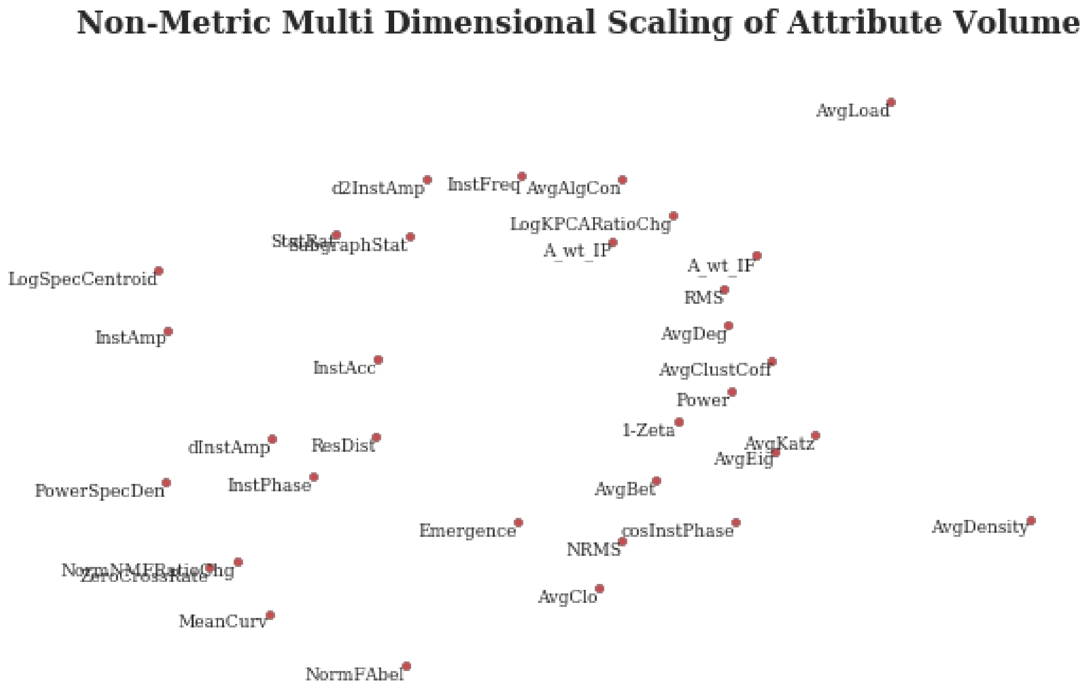
```
In [181]: plt.figure(figsize=(8,11))

           plt.subplot(3,1,1)
           plt.plot(emerg_m,'r', label='Emergence')
           plt.xticks(np.arange(len(months)), months,fontsize=12, rotation=90)
           plt.yticks(fontsize=12)
           plt.xlabel('Months', fontsize=16)
           plt.legend(fontsize=11, loc=1)
           plt.tight_layout()
           plt.title("Comparison of Aggregation Schemes", fontsize=22)
           plt.autoscale()

           plt.subplot(3,1,2)
           plt.plot(rms_m,'g',label='RMS of Attributes',)
           plt.yticks(fontsize=12)
           plt.xticks(np.arange(len(months)), months,fontsize=11, rotation=90)
           plt.xlabel('Months', fontsize=16)
           plt.legend(fontsize=12, loc=1)
           plt.tight_layout()
           plt.autoscale()

           plt.subplot(3,1,3)
           plt.plot(nrms_m,'m',label='NRMS of Attributes')

           plt.xticks(np.arange(len(months)), months,fontsize=11, rotation=90)
           plt.yticks(fontsize=12)
           plt.xlabel('Months', fontsize=16)
           plt.legend(fontsize=12, loc=1)
           plt.autoscale()
           plt.savefig('images/agg_comp.png')
```


Comparison of Aggregation Schemes
Emergence
1.0
0.5
0.0
−0.5
−1.0
Nov98
Dec98
jan_99
feb_99
mar_99
apr_99
may_99
jun_99
jul_99
aug_99
sep_99
oct_99
nov_99
dec_99
jan_2k
feb_2k
mar_2k
apr_2k
may_2k
jun_2k
jul_2k
aug_2k
sep_2k
oct_2k
nov_2k
dec_2k
jan_2k1
feb_2k1
mar_2k1
apr_2k1
may_2k1
jun_2k1
jul_2k1
aug_2k1
sep_2k1
oct_2k1
nov_2k1
dec_2k1
jan_2k2
feb_2k2
mar_2k2
apr_2k2
may_2k2
jun_2k2
Months
RMS of Attributes
1.0
0.9
0.8
0.7
0.6
0.5
0.4
Nov98
Dec98
jan_99
feb_99
mar_99
apr_99
may_99
jun_99
jul_99
aug_99
sep_99
oct_99
nov_99
dec_99
jan_2k
feb_2k
mar_2k
apr_2k
may_2k
jun_2k
jul_2k
aug_2k
sep_2k
oct_2k
nov_2k
dec_2k
jan_2k1
feb_2k1
mar_2k1
apr_2k1
may_2k1
jun_2k1
jul_2k1
aug_2k1
sep_2k1
oct_2k1
nov_2k1
dec_2k1
jan_2k2
feb_2k2
mar_2k2
apr_2k2
may_2k2
jun_2k2
Months
NRMS of Attributes
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Nov98
Dec98
jan_99
feb_99
mar_99
apr_99
may_99
jun_99
jul_99
aug_99
sep_99
oct_99
nov_99
dec_99
jan_2k
feb_2k
mar_2k
apr_2k
may_2k
jun_2k
jul_2k
aug_2k
sep_2k
oct_2k
nov_2k
dec_2k
jan_2k1
feb_2k1
mar_2k1
apr_2k1
may_2k1
jun_2k1
jul_2k1
aug_2k1
sep_2k1
oct_2k1
nov_2k1
dec_2k1
jan_2k2
feb_2k2
mar_2k2
apr_2k2
may_2k2
jun_2k2
Months

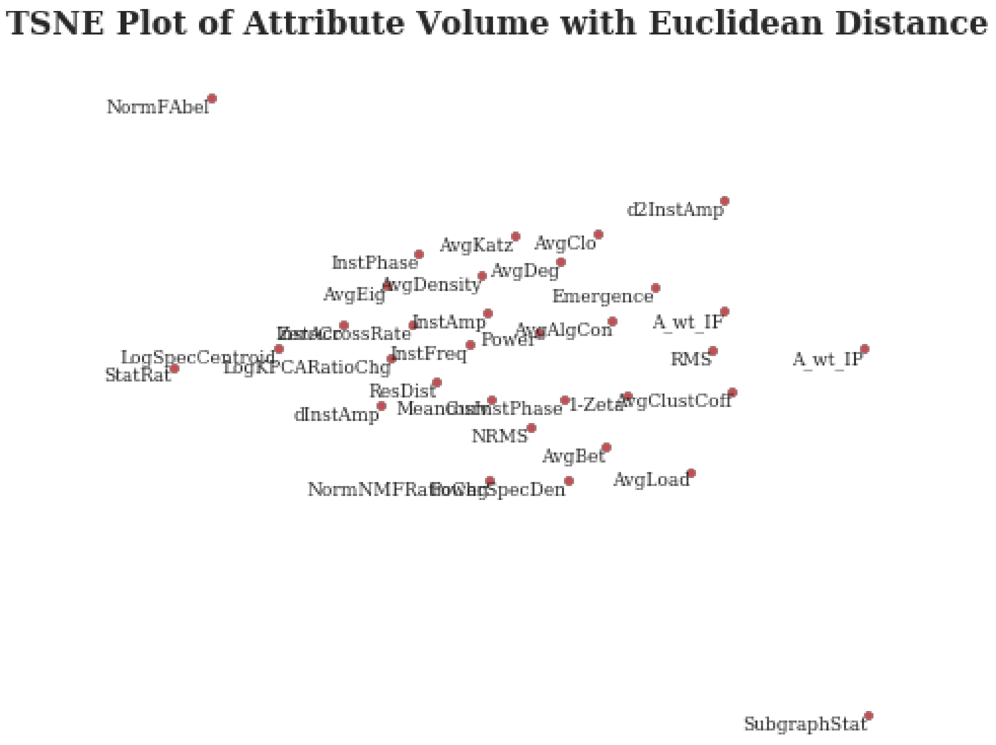
```
In [82]: final_attvol_m = attvol_m.copy()

         final_attvol_m['NRMS'] = nrms_m
         final_attvol_m['RMS']=rms_m
         final_attvol_m['Emergence'] =emerg_m
```

10 MDS and TSNE

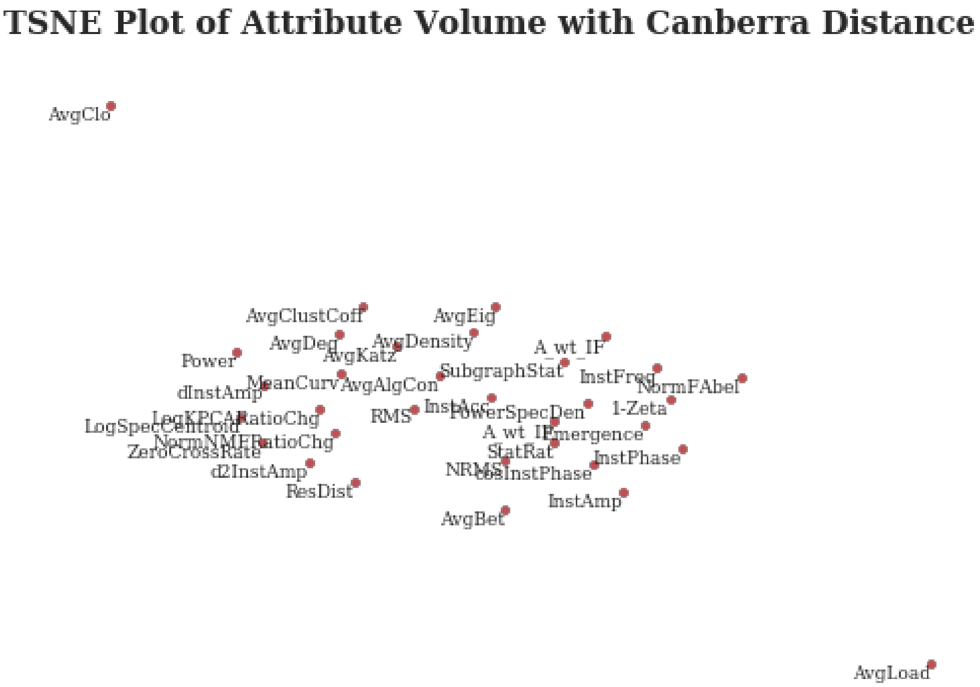
```
In [83]: from sklearn.manifold import *

In [184]: mds = MDS(n_components=2, metric=False, random_state=0)
          Y = mds.fit_transform(final_attvol_m)

          plt.axis('off')
          plt.suptitle("Non-Metric Multi Dimensional Scaling of Attribute Volume",
          for i, txt in enumerate(final_attvol_m.columns):
              plt.scatter(Y[:, 1][i],Y[:, 0][i], c='r')
              plt.annotate(txt, (Y[:, 1][i],Y[:, 0][i]), horizontalalignment='right

          plt.autoscale()
          plt.tight_layout()
          plt.savefig('images/mds_plot.png')
```



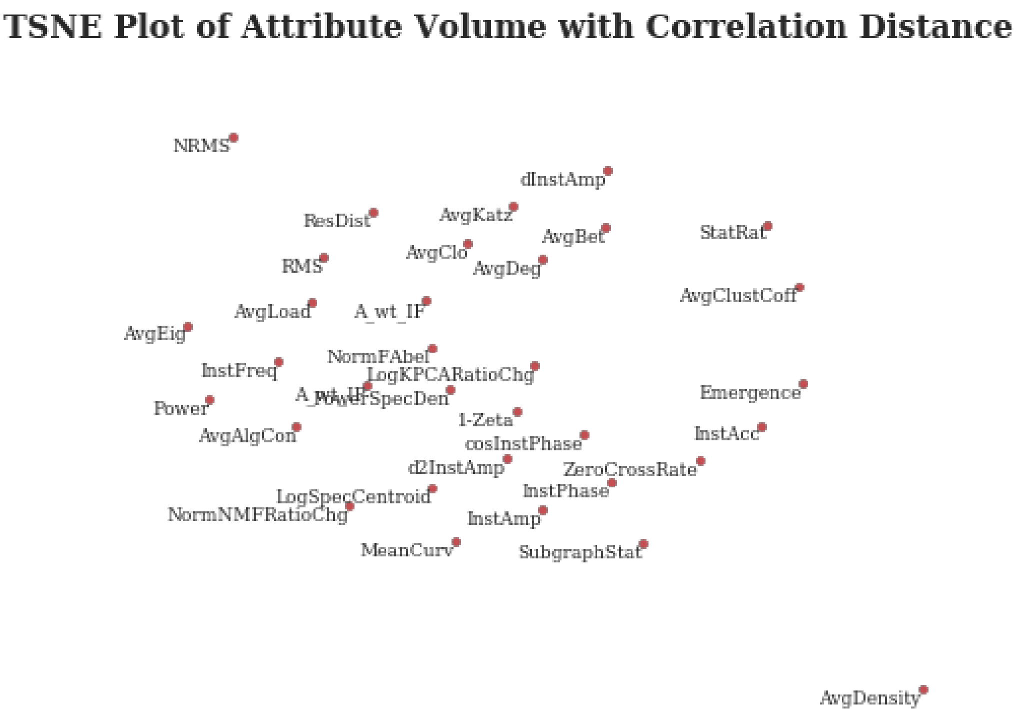
```
In [185]: tsne = TSNE(metric='euclidean', random_state=0)

          y_tsne = tsne.fit_transform(final_attvol_m)

          plt.axis('off')
          plt.suptitle("TSNE Plot of Attribute Volume with Euclidean Distance", for
          for i, txt in enumerate(final_attvol_m.columns):
              plt.scatter(y_tsne[:, 1][i],y_tsne[:, 0][i], c='r')
              plt.annotate(txt, (y_tsne[:, 1][i],y_tsne[:, 0][i]), horizontalalignm

          plt.autoscale()
          plt.tight_layout()
          plt.savefig('images/tsne_plot_euc.png')
```



```
In [186]: tsne = TSNE(metric='canberra', random_state=0)

          y_tsne = tsne.fit_transform(final_attvol_m)
          plt.axis('off')
          plt.suptitle("TSNE Plot of Attribute Volume with Canberra Distance", font
          for i, txt in enumerate(final_attvol_m.columns):
              plt.scatter(y_tsne[:, 1][i],y_tsne[:, 0][i], c='r')
              plt.annotate(txt, (y_tsne[:, 1][i],y_tsne[:, 0][i]), horizontalalignm

          plt.autoscale()
          plt.tight_layout()
          plt.savefig('images/tsne_plot.png')
```



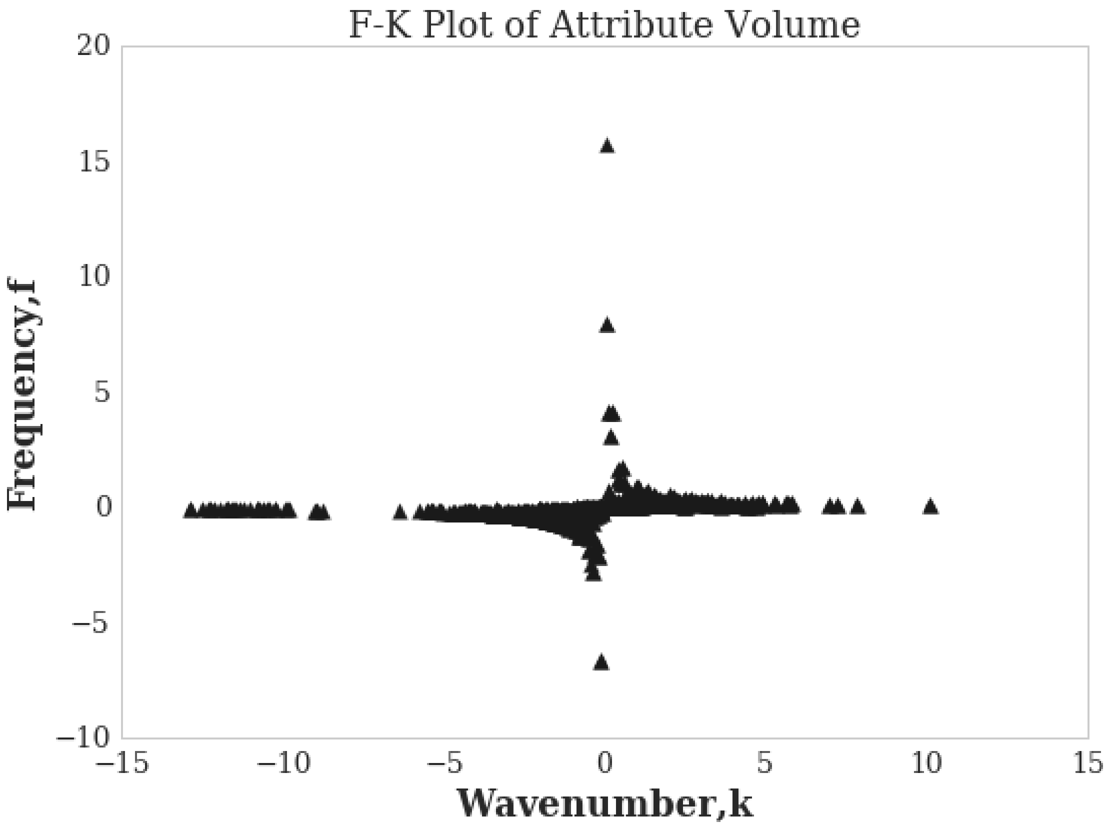
```
In [187]: tsne = TSNE(metric='correlation', random_state=0)

          y_tsne = tsne.fit_transform(final_attvol_m)

          plt.axis('off')
          plt.suptitle("TSNE Plot of Attribute Volume with Correlation Distance", f
          for i, txt in enumerate(final_attvol_m.columns):
              plt.scatter(y_tsne[:, 1][i],y_tsne[:, 0][i], c='r')
              plt.annotate(txt, (y_tsne[:, 1][i],y_tsne[:, 0][i]), horizontalalignm

          plt.autoscale()
          plt.tight_layout()
          plt.savefig('images/tsne_plot_corr.png')
```



11 FK and Radon Plot

```
In [191]: def radon(m):
              from skimage.transform import radon
              theta = np.linspace(0., 180., max(m.shape), endpoint=False)
              sinogram = radon(m, theta=theta, circle=True)
              return sinogram

In [232]: def fk_plot(f):
              freq = sc.fft(f)
              wavnum = 1/freq

              return [freq, wavnum]

In [234]: f,k, = fk_plot(final_attvol_m)
          plt.scatter(f,k, s=60, marker='^', c='k')
          plt.title("F-K Plot of Attribute Volume", fontsize=18)
          plt.xticks(fontsize=14)
          plt.yticks(fontsize=14)
          plt.xlabel("Wavenumber,k", fontsize=18)
          plt.ylabel("Frequency,f", fontsize=18)
          plt.savefig('images/fkplot.png')
```

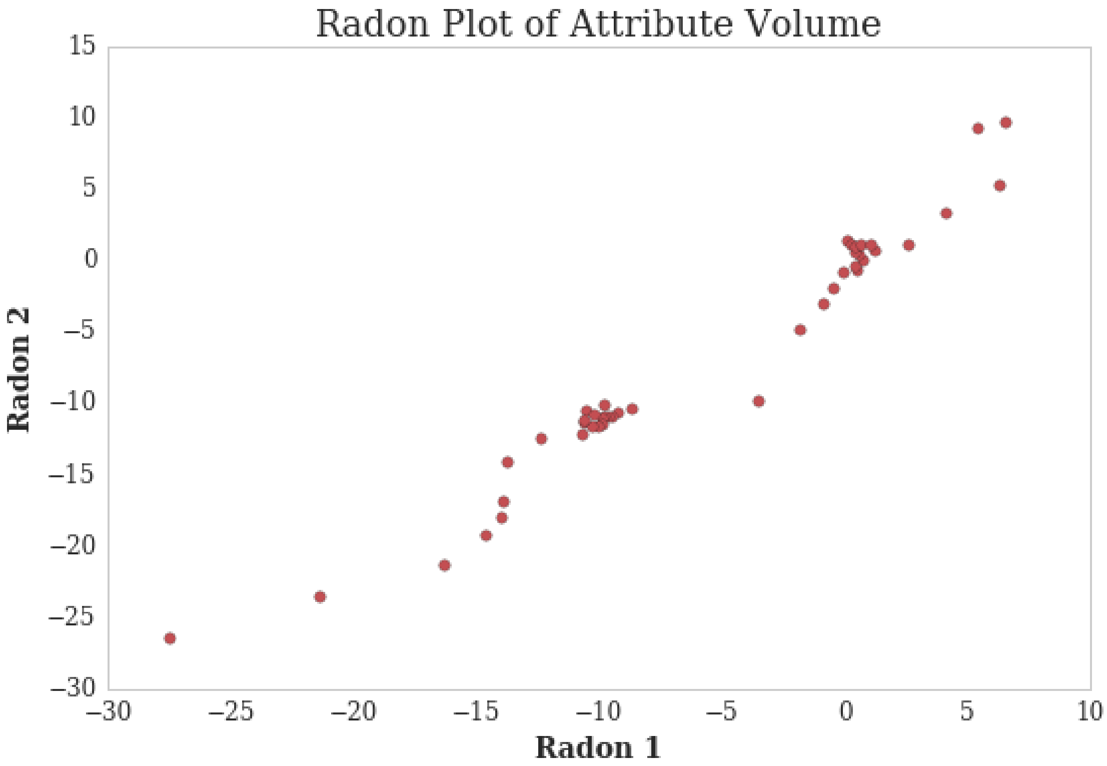
```
/home/arshad/anaconda3/lib/python3.5/site-packages/numpy/core/numeric.py:533: Compl
  return array(a, dtype, copy=False, order=order, subok=True)
/home/arshad/anaconda3/lib/python3.5/site-packages/matplotlib/figure.py:1742: UserW
  warnings.warn("This figure includes Axes that are not "
```

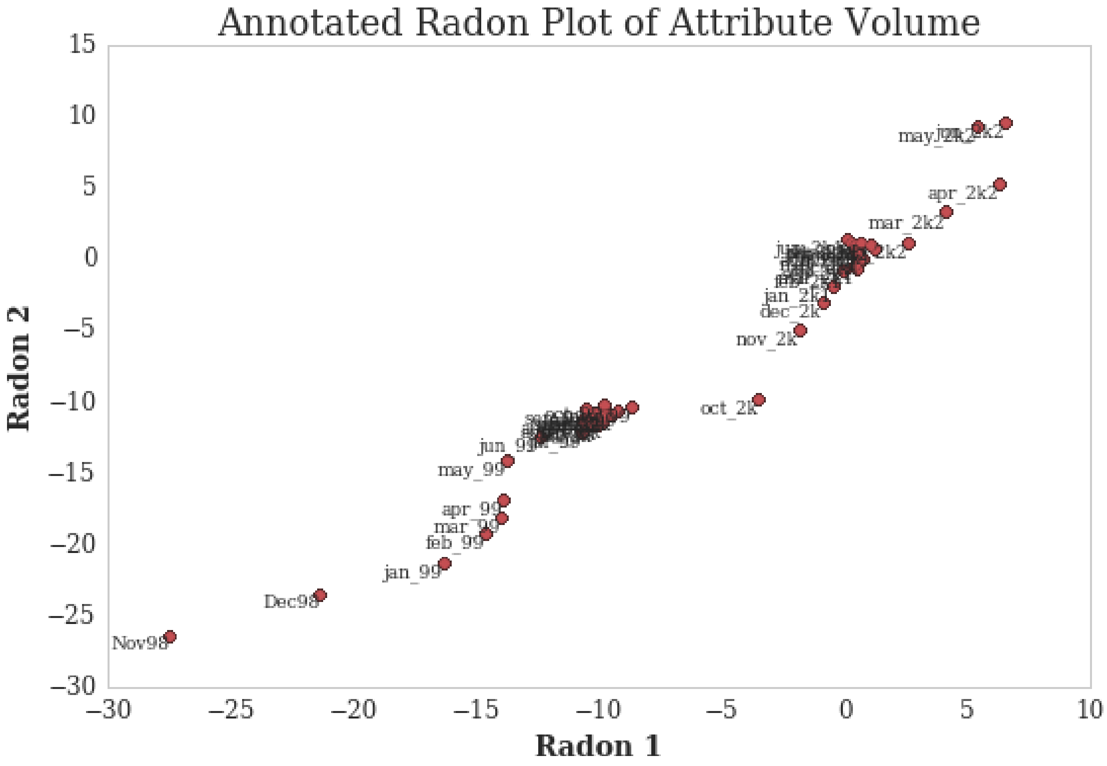


```
In [192]: sgram = radon(final_attvol_m.values)
```

```
/home/arshad/anaconda3/lib/python3.5/site-packages/skimage/transform/radon_transfor
  warn('Radon transform: image must be zero outside the '
```

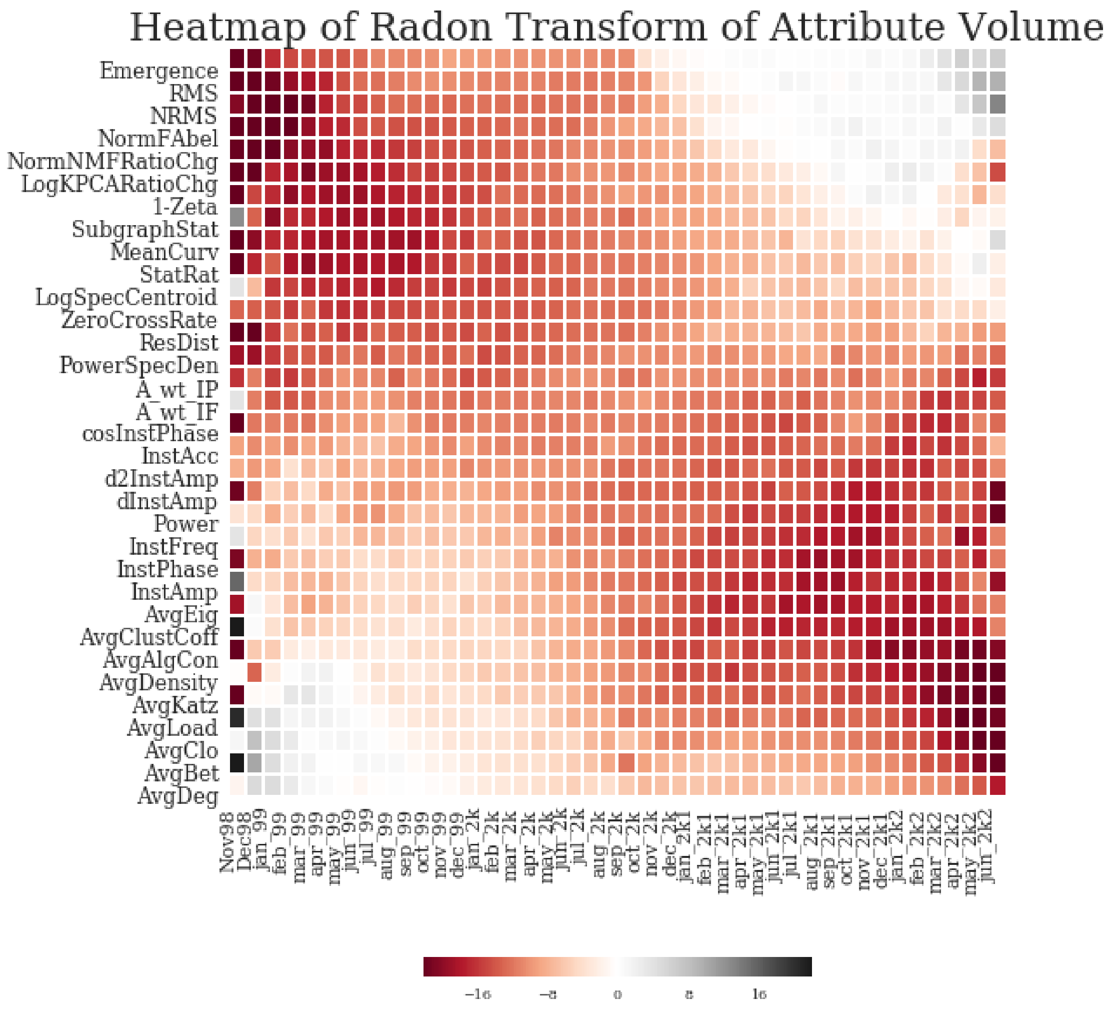
```
In [190]: plt.figure(figsize=(8,11))

          plt.subplot(211)
          plt.scatter(sgram[0],sgram[1], c='r', s=30)
          plt.xticks(fontsize=12)
          plt.yticks(fontsize=12)
          plt.xlabel("Radon 1", fontsize=14)
          plt.ylabel("Radon 2", fontsize=14)
          plt.title("Radon Plot of Attribute Volume", fontsize=18)
          plt.tight_layout()

          plt.subplot(212)
          plt.title("Annotated Radon Plot of Attribute Volume", fontsize=18)
          for i, txt in enumerate(months):
              plt.scatter(sgram[0],sgram[1], c='r', s=30)
              plt.annotate(txt, (sgram[0][i],sgram[1][i]), horizontalalignment='rig
          plt.xticks(fontsize=12)
          plt.yticks(fontsize=12)
          plt.xlabel("Radon 1", fontsize=14)
          plt.ylabel("Radon 2", fontsize=14)
          plt.tight_layout()
          plt.savefig('images/radonplot.png')
```



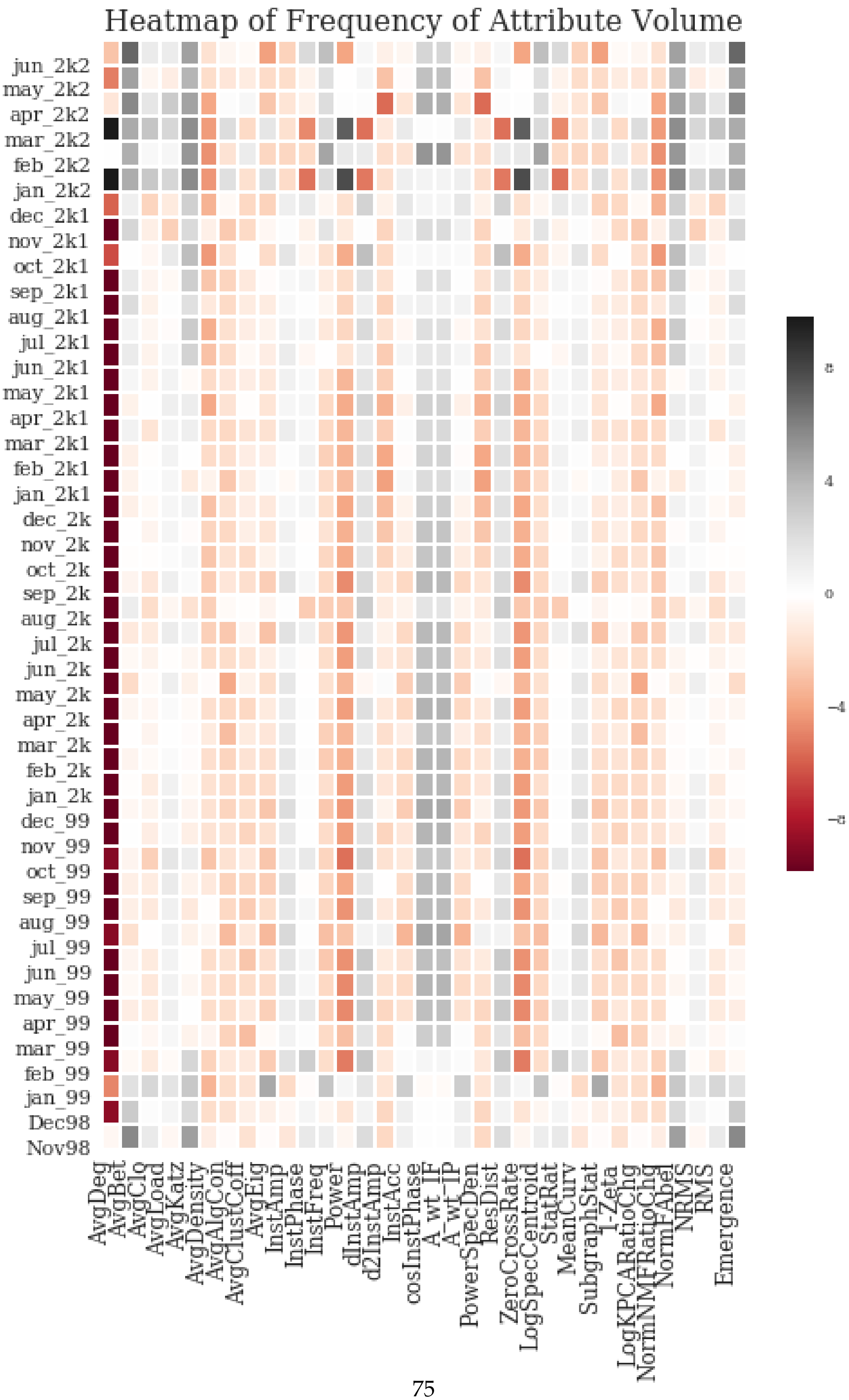


```
In [194]: plt.figure(figsize=(8,11))
```

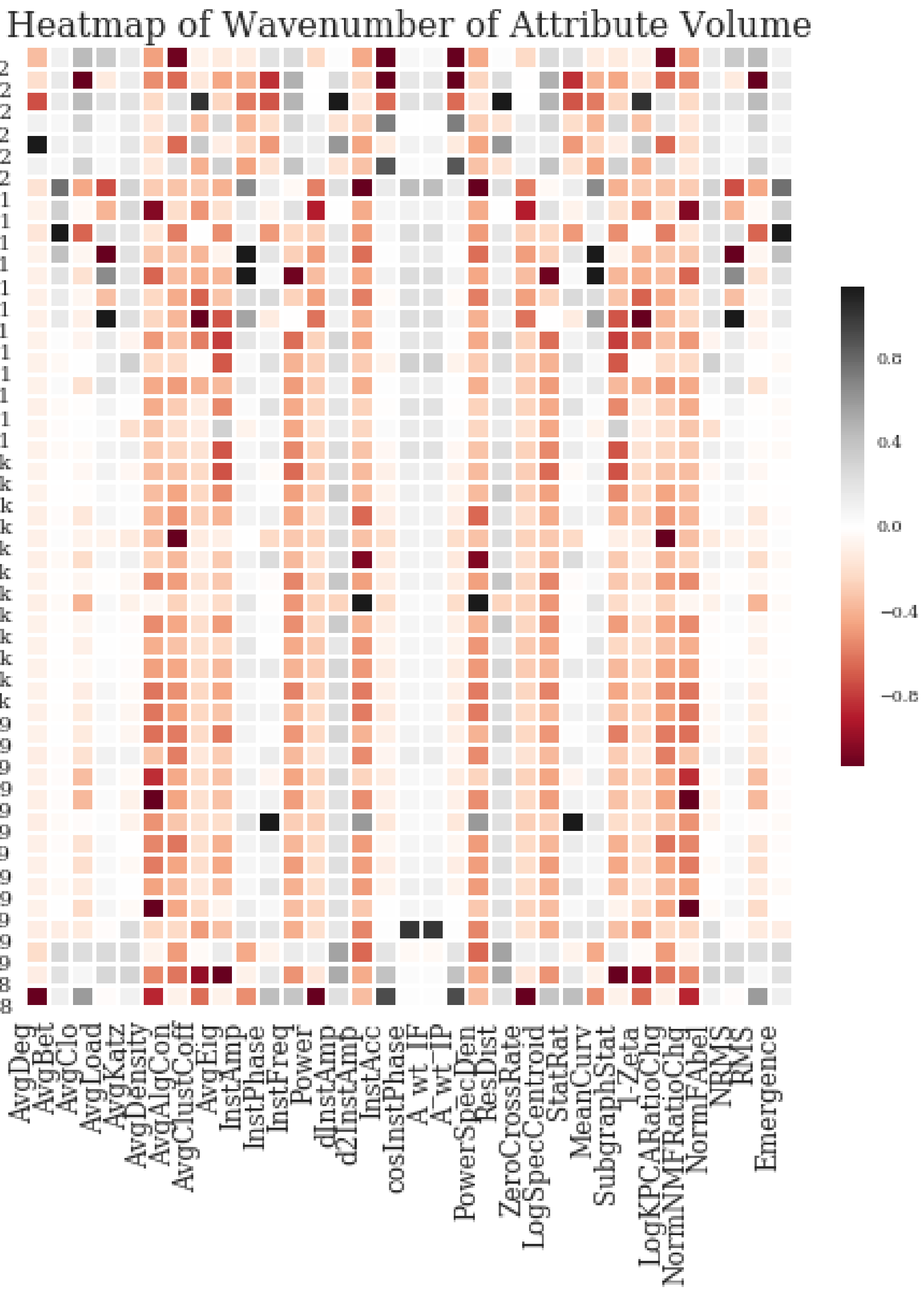
```
         sns.heatmap(sgram, cmap='RdGy', robust=True,cbar_kws={'orientation':'hori
         plt.xticks(np.arange(len(months)), months,fontsize=11, rotation=90);
         plt.yticks(np.arange(len(final_attvol_m.columns)), final_attvol_m.columns
         plt.title("Heatmap of Radon Transform of Attribute Volume", fontsize=16)
         plt.savefig('images/radon_heat.png')
```



```
In [197]: plt.figure(figsize=(8,11))
          sns.heatmap(np.real(f), cmap='RdGy', robust=True,cbar_kws={'orientation':
          plt.yticks(np.arange(len(months)), months,fontsize=11, rotation=360);
          plt.xticks(np.arange(len(final_attvol_m.columns)), final_attvol_m.columns
          plt.title("Heatmap of Frequency of Attribute Volume", fontsize=16)
          plt.savefig('images/freq_heat.png')
```

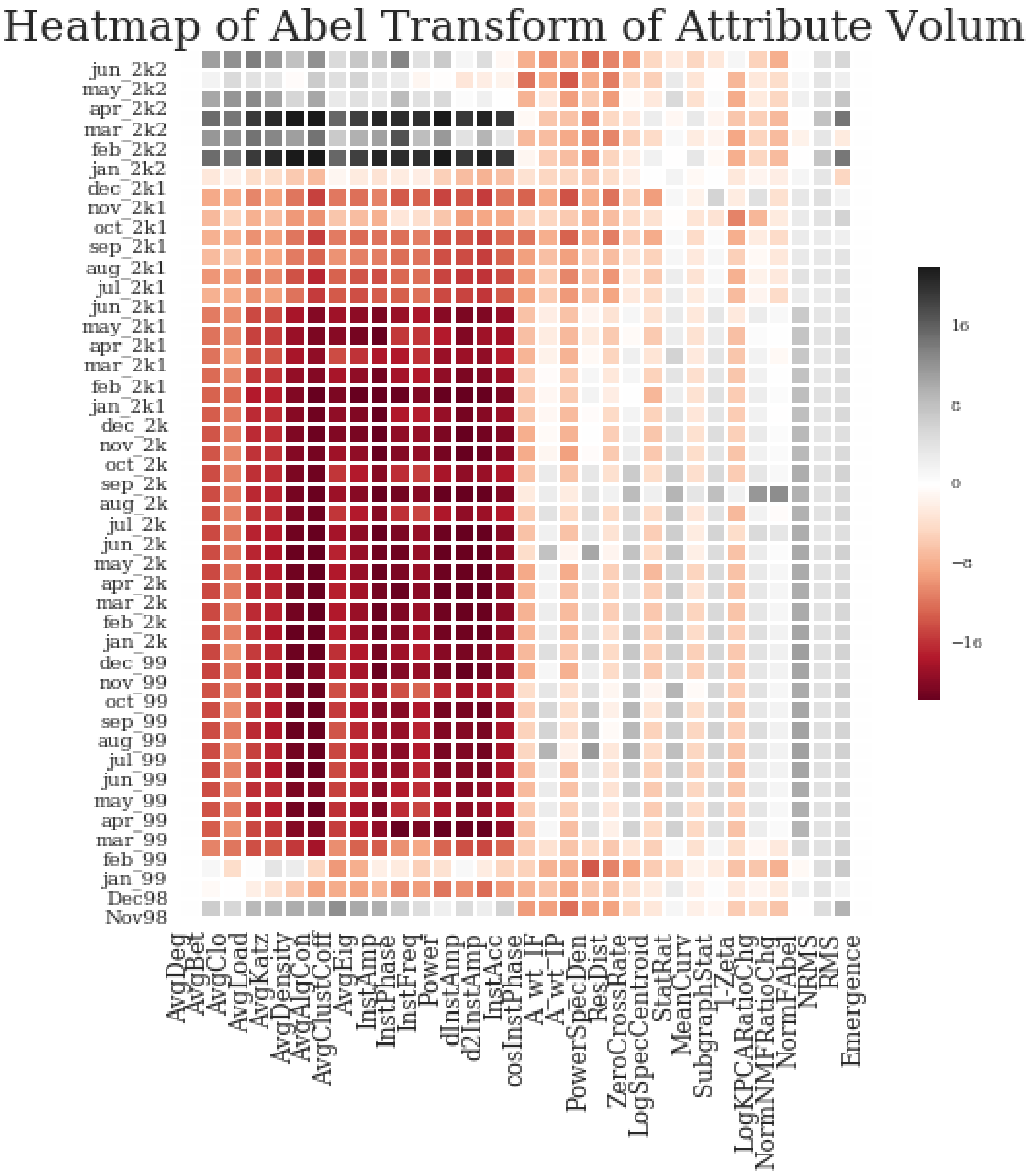

Heatmap of Frequency of Attribute Volume
jun_2k2
may_2k2
apr_2k2
mar_2k2
feb_2k2
jan_2k2
dec_2k1
nov_2k1
oct_2k1
sep_2k1
aug_2k1
jul_2k1
jun_2k1
may_2k1
apr_2k1
mar_2k1
feb_2k1
jan_2k1
dec_2k
nov_2k
oct_2k
sep_2k
aug_2k
jul_2k
jun_2k
may_2k
apr_2k
mar_2k
feb_2k
jan_2k
dec_99
nov_99
oct_99
sep_99
aug_99
jul_99
jun_99
may_99
apr_99
mar_99
feb_99
jan_99
Dec98
Nov98
AvgDeg
AvgBet
AvgClo
AvgLoad
AvgKatz
AvgDensity
AvgAlgCon
AvgClustCoff
AvgEig
InstAmp
InstPhase
InstFreq
Power
dInstAmp
d2InstAmp
InstAcc
cosInstPhase
A_wt_IF
A_wt_IP
PowerSpecDen
ResDist
ZeroCrossRate
LogSpecCentroid
StatRat
MeanCurv
SubgraphStat
1-Zeta
LogKPCARatioChg
NormNMFRatioChg
NormFAbel
NRMS
RMS
Emergence
8
4
0
−4
−8

```
In [201]: plt.figure(figsize=(8,8))
          sns.heatmap(k, cmap='RdGy', robust=True,cbar_kws={'orientation':'vertical
          plt.yticks(np.arange(len(months)), months,fontsize=10, rotation=360);
          plt.xticks(np.arange(len(final_attvol_m.columns)), final_attvol_m.columns
          plt.title("Heatmap of Wavenumber of Attribute Volume", fontsize=16)
          plt.savefig('images/wavnum_heat.png')

/home/arshad/anaconda3/lib/python3.5/site-packages/numpy/ma/core.py:3095: ComplexWa
  output = self._data.astype(newtype).view(type(self))
```

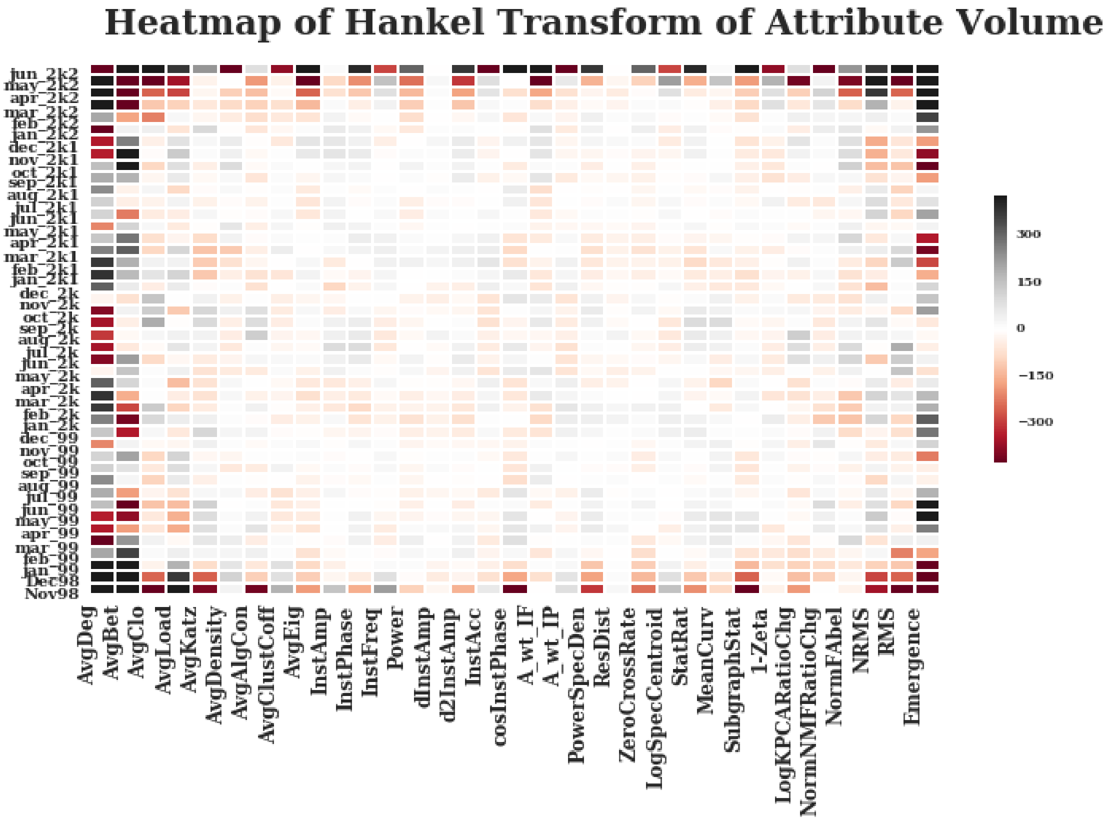


12 Exotic Transforms

```
In [96]: fabel = abel.Transform(final_attvol_m.values, direction='forward', method=
In [199]: plt.figure(figsize=(8,8))
          sns.heatmap(fabel, cmap='RdGy', robust=True,cbar_kws={'orientation':'vert
          plt.yticks(np.arange(len(months)), months,fontsize=10, rotation=360);
          plt.xticks(np.arange(len(final_attvol_m.columns)), final_attvol_m.columns
          plt.title("Heatmap of Abel Transform of Attribute Volume", fontsize=16)

Out[199]: <matplotlib.text.Text at 0x7f625b656fd0>
```



```
In [98]: H = sc.fftpack.fft2(fabel)
In [99]: sns.heatmap(H, cmap='RdGy', robust=True,cbar_kws={'orientation':'vertical'
         plt.yticks(np.arange(len(months)), months,fontsize=10, rotation=360);
         plt.xticks(np.arange(len(final_attvol_m.columns)), final_attvol_m.columns,
         plt.suptitle("Heatmap of Hankel Transform of Attribute Volume", fontsize=2

/home/arshad/anaconda3/lib/python3.5/site-packages/numpy/ma/core.py:3095: ComplexWa
  output = self._data.astype(newtype).view(type(self))

Out[99]: <matplotlib.text.Text at 0x7f625b115f98>
```



13 Which nodes are common to all years?

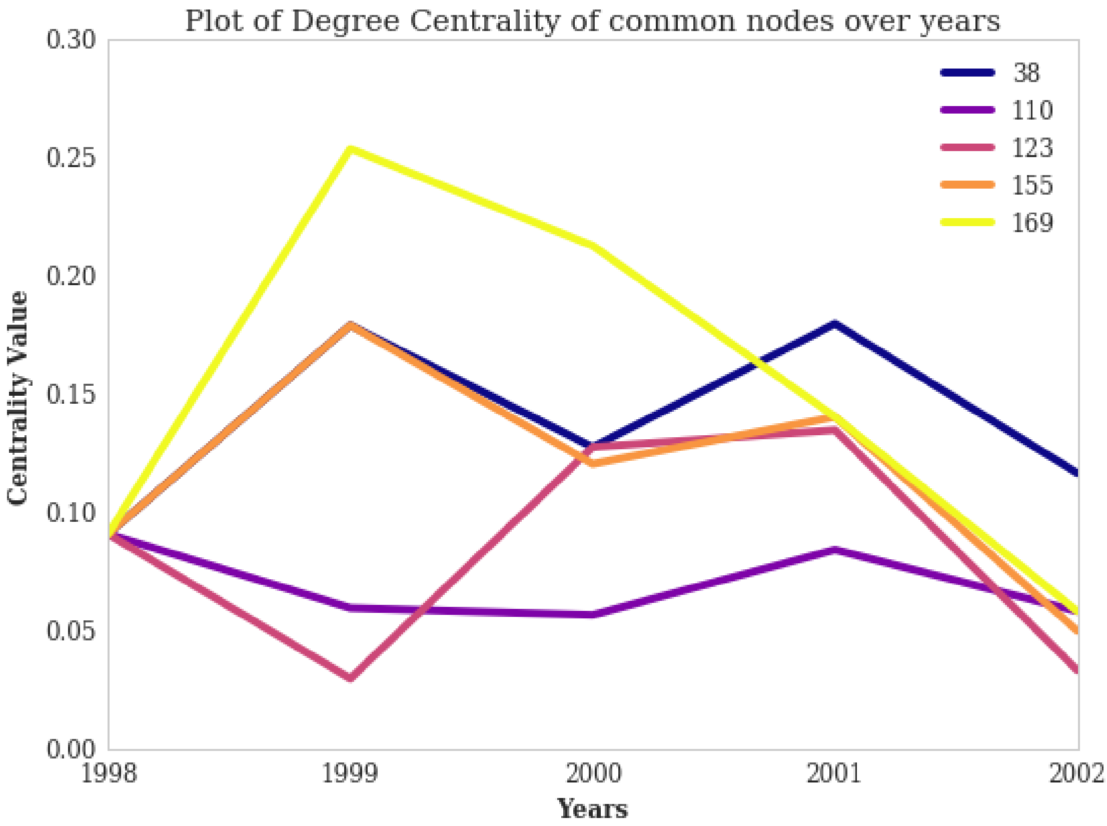
```
In [100]: int_0 = np.intersect1d(Gt0.nodes(), Gt1.nodes())
          int_1 = np.intersect1d(int_0, Gt2.nodes())
          int_2 = np.intersect1d(int_1, Gt3.nodes())
          int_all = np.intersect1d(int_2, Gt4.nodes())
          int_all

Out[100]: array([ 38, 110, 123, 155, 169])
```

```
In [101]: 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)
              katzC = nx.katz_centrality_numpy(net)
              loadC = nx.load_centrality(net)

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

```
In [102]: degC0, cloC0, betC0, eigC0,  katzC0, loadC0 = get_cent(Gt0)
          degC1, cloC1, betC1, eigC1,  katzC1, loadC1 = get_cent(Gt1)
          degC2, cloC2, betC2, eigC2,  katzC2, loadC2 = get_cent(Gt2)
          degC3, cloC3, betC3, eigC3,  katzC3, loadC3 = get_cent(Gt3)
          degC4, cloC4, betC4, eigC4,  katzC4, loadC4 = get_cent(Gt4)
```

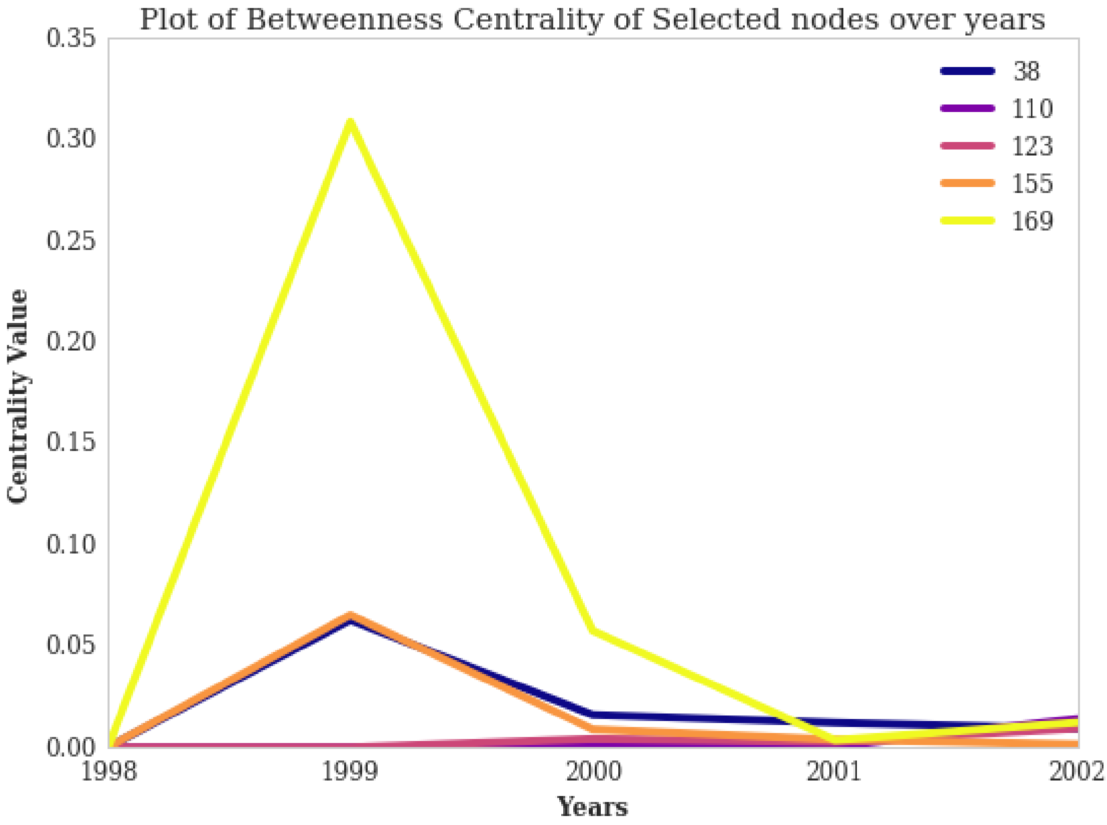
```
In [236]: node_deg = pd.DataFrame.from_dict(degC0, orient='index').\
          join(pd.DataFrame.from_dict(degC1,orient='index'), rsuffix = 'Gt1', lsuff
          join(pd.DataFrame.from_dict(degC2,orient='index'), rsuffix = 'Gt2', lsuff
          join(pd.DataFrame.from_dict(degC3,orient='index'), rsuffix = 'Gt3', lsuff
          join(pd.DataFrame.from_dict(degC4,orient='index'), rsuffix = 'Gt4', lsuff

          node_deg = node_deg.T
          node_deg.columns = years
          node_deg = node_deg.T

          node_deg.plot.line(figsize=(8,6), cmap='plasma')
          plt.title("Plot of Degree Centrality of common nodes over years", fontsiz
          plt.xlabel("Years", fontsize=12)
          plt.ylabel("Centrality Value", fontsize=12)
          plt.legend(fontsize=12, loc=1)
          plt.xticks(years,[i for i in years],fontsize=12)
          plt.yticks(fontsize=12)
          plt.tight_layout()
          plt.savefig('images/node_deg.png')
```

```
/home/arshad/anaconda3/lib/python3.5/site-packages/matplotlib/figure.py:1742: UserW
  warnings.warn("This figure includes Axes that are not "
```

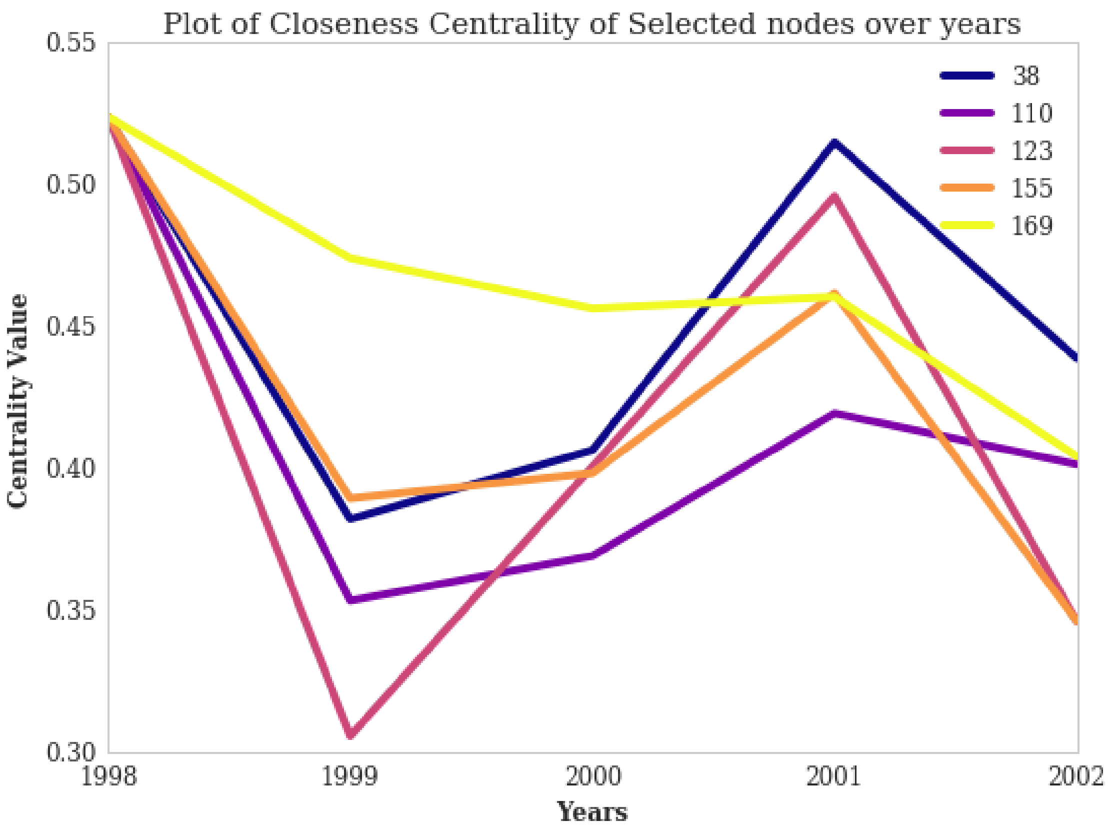


```
In [237]: node_bet = pd.DataFrame.from_dict(betC0, orient='index').\
          join(pd.DataFrame.from_dict(betC1,orient='index'), rsuffix = 'Gt1', lsuff
          join(pd.DataFrame.from_dict(betC2,orient='index'), rsuffix = 'Gt2', lsuff
          join(pd.DataFrame.from_dict(betC3,orient='index'), rsuffix = 'Gt3', lsuff
          join(pd.DataFrame.from_dict(betC4,orient='index'), rsuffix = 'Gt4', lsuff
          node_bet = node_bet.T
          node_bet.columns = years
          node_bet = node_bet.T
          node_bet.plot(cmap='plasma')
          plt.title("Plot of Betweenness Centrality of Selected nodes over years",
          plt.xlabel("Years", fontsize=12)
          plt.ylabel("Centrality Value", fontsize=12)
          plt.legend(fontsize=12, loc=1)
          plt.xticks(years,[i for i in years],fontsize=12)
          plt.yticks(fontsize=12)
          plt.tight_layout()
          plt.savefig('images/node_bet.png')
```

```
/home/arshad/anaconda3/lib/python3.5/site-packages/matplotlib/figure.py:1742: UserW
  warnings.warn("This figure includes Axes that are not "
```

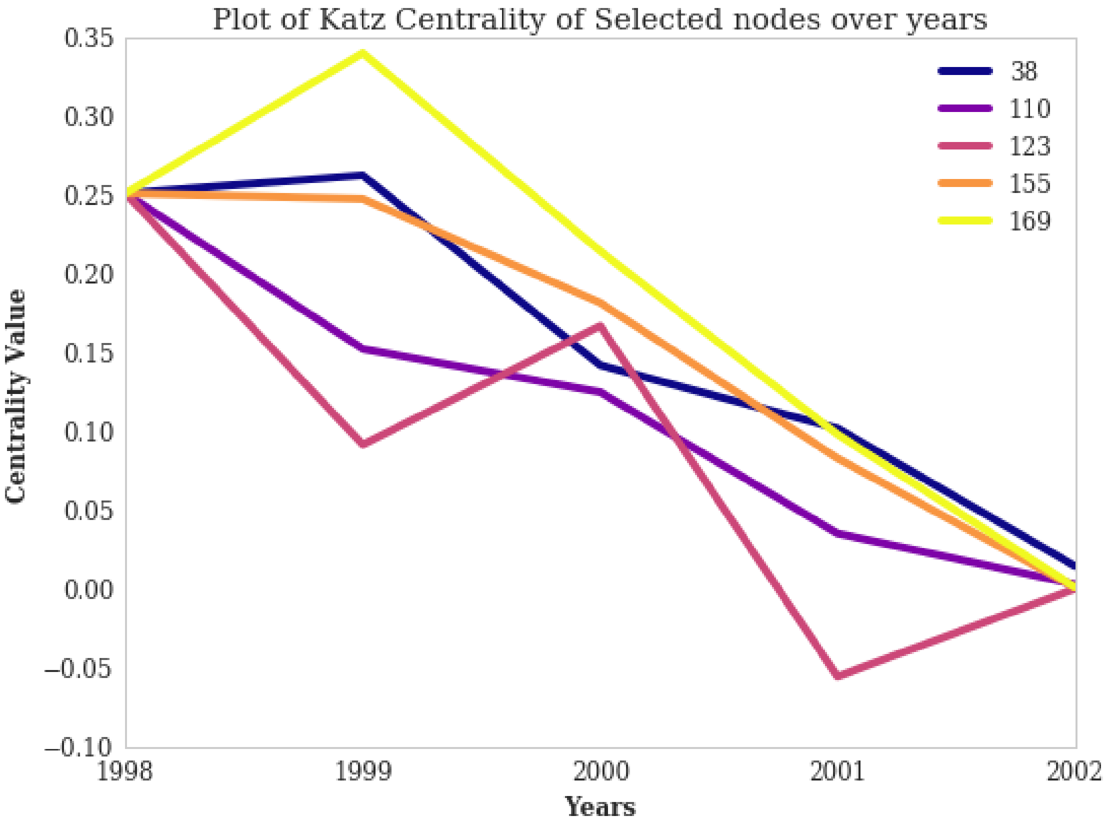


```
In [238]: node_clo = pd.DataFrame.from_dict(cloC0, orient='index').\
          join(pd.DataFrame.from_dict(cloC1,orient='index'), rsuffix = 'Gt1', lsuf
          join(pd.DataFrame.from_dict(cloC2,orient='index'), rsuffix = 'Gt2', lsuf
          join(pd.DataFrame.from_dict(cloC3,orient='index'), rsuffix = 'Gt3', lsuf
          join(pd.DataFrame.from_dict(cloC4,orient='index'), rsuffix = 'Gt4', lsuf
          node_clo = node_clo.T
          node_clo.columns = years
          node_clo = node_clo.T
          node_clo.plot(figsize=(8,6), cmap='plasma')
          plt.title("Plot of Closeness Centrality of Selected nodes over years", fo
          plt.xlabel("Years", fontsize=12)
          plt.ylabel("Centrality Value", fontsize=12)
          plt.legend(fontsize=12, loc=1)
          plt.xticks(years,[i for i in years],fontsize=12)
          plt.yticks(fontsize=12)
          plt.tight_layout()
          plt.savefig('images/node_clo.png')

/home/arshad/anaconda3/lib/python3.5/site-packages/matplotlib/figure.py:1742: UserW
  warnings.warn("This figure includes Axes that are not "
```



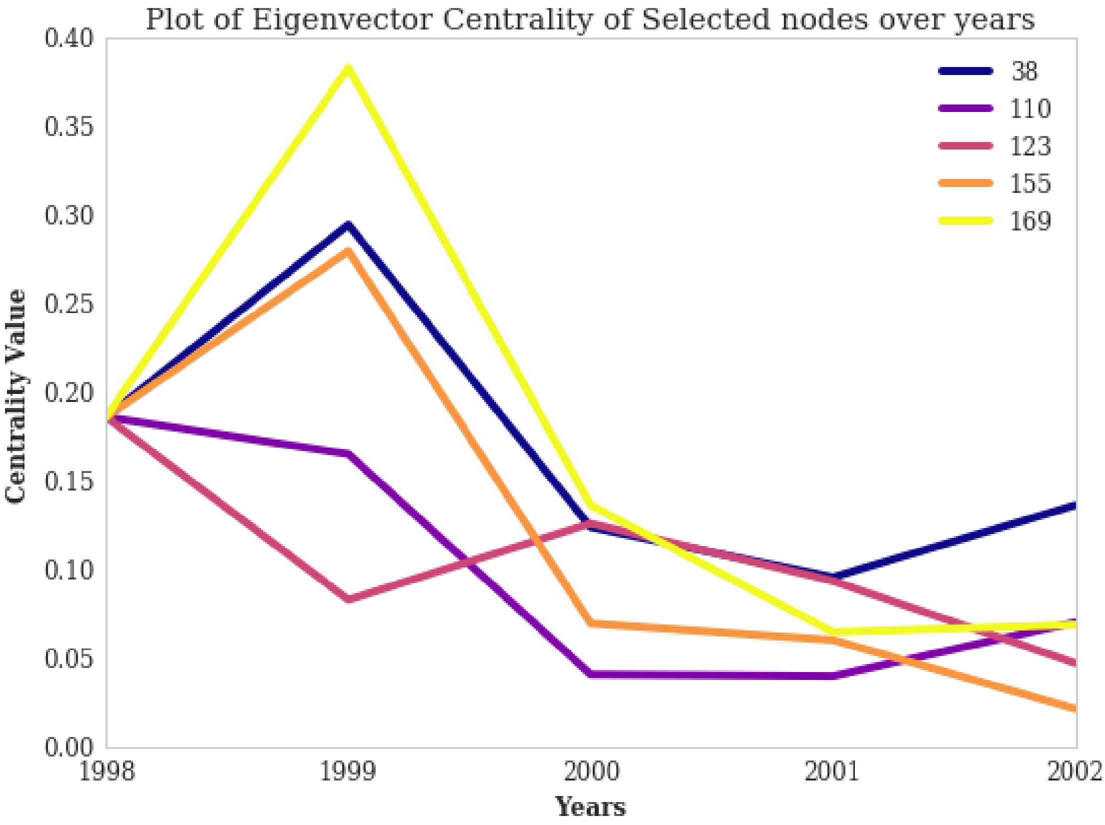
```
In [239]: node_katz = pd.DataFrame.from_dict(katzC0, orient='index').\
          join(pd.DataFrame.from_dict(katzC1,orient='index'), rsuffix = 'Gt1', lsuf
          join(pd.DataFrame.from_dict(katzC2,orient='index'), rsuffix = 'Gt2', lsuf
          join(pd.DataFrame.from_dict(katzC3,orient='index'), rsuffix = 'Gt3', lsuf
          join(pd.DataFrame.from_dict(katzC4,orient='index'), rsuffix = 'Gt4', lsuf
          node_katz = node_katz.T
          node_katz.columns = years
          node_katz = node_katz.T
          node_katz.plot(figsize=(8,6), cmap='plasma')
          plt.title("Plot of Katz Centrality of Selected nodes over years", fontsiz
          plt.xlabel("Years", fontsize=12)
          plt.ylabel("Centrality Value", fontsize=12)
          plt.legend(fontsize=12, loc=1)
          plt.xticks(years,[i for i in years],fontsize=12)
          plt.yticks(fontsize=12)
          plt.tight_layout()
          plt.savefig('images/node_katz.png')
```

```
/home/arshad/anaconda3/lib/python3.5/site-packages/matplotlib/figure.py:1742: UserW
  warnings.warn("This figure includes Axes that are not "
```



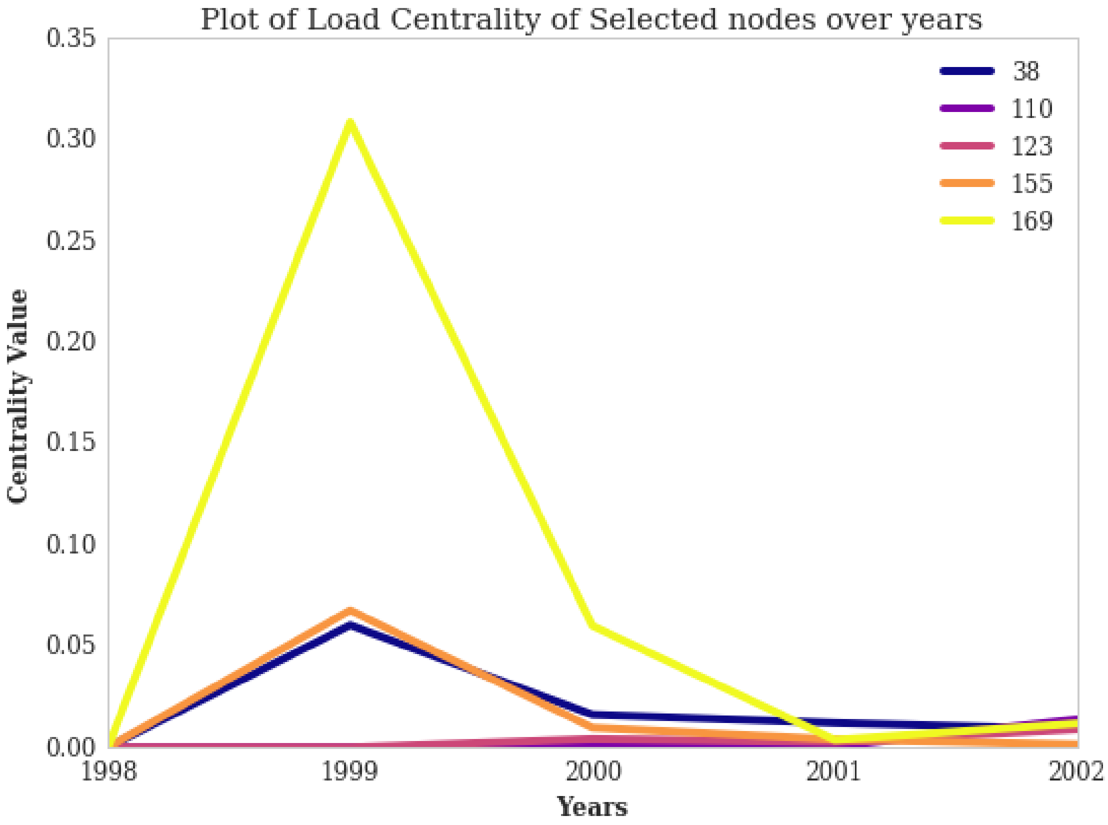
```
In [240]: node_eig = pd.DataFrame.from_dict(eigC0, orient='index').\
          join(pd.DataFrame.from_dict(eigC1,orient='index'), rsuffix = 'Gt1', lsuff
          join(pd.DataFrame.from_dict(eigC2,orient='index'), rsuffix = 'Gt2', lsuff
          join(pd.DataFrame.from_dict(eigC3,orient='index'), rsuffix = 'Gt3', lsuff
          join(pd.DataFrame.from_dict(eigC4,orient='index'), rsuffix = 'Gt4', lsuff
          node_eig = node_eig.T
          node_eig.columns = years
          node_eig = node_eig.T
          node_eig.plot(figsize=(8,6), cmap='plasma')
          plt.title("Plot of Eigenvector Centrality of Selected nodes over years",
          plt.xlabel("Years", fontsize=12)
          plt.ylabel("Centrality Value", fontsize=12)
          plt.legend(fontsize=12, loc=1)
          plt.xticks(years,[i for i in years],fontsize=12)
          plt.yticks(fontsize=12)
          plt.tight_layout()
          plt.savefig('images/node_eig.png')
```

```
/home/arshad/anaconda3/lib/python3.5/site-packages/matplotlib/figure.py:1742: UserW
  warnings.warn("This figure includes Axes that are not "
```



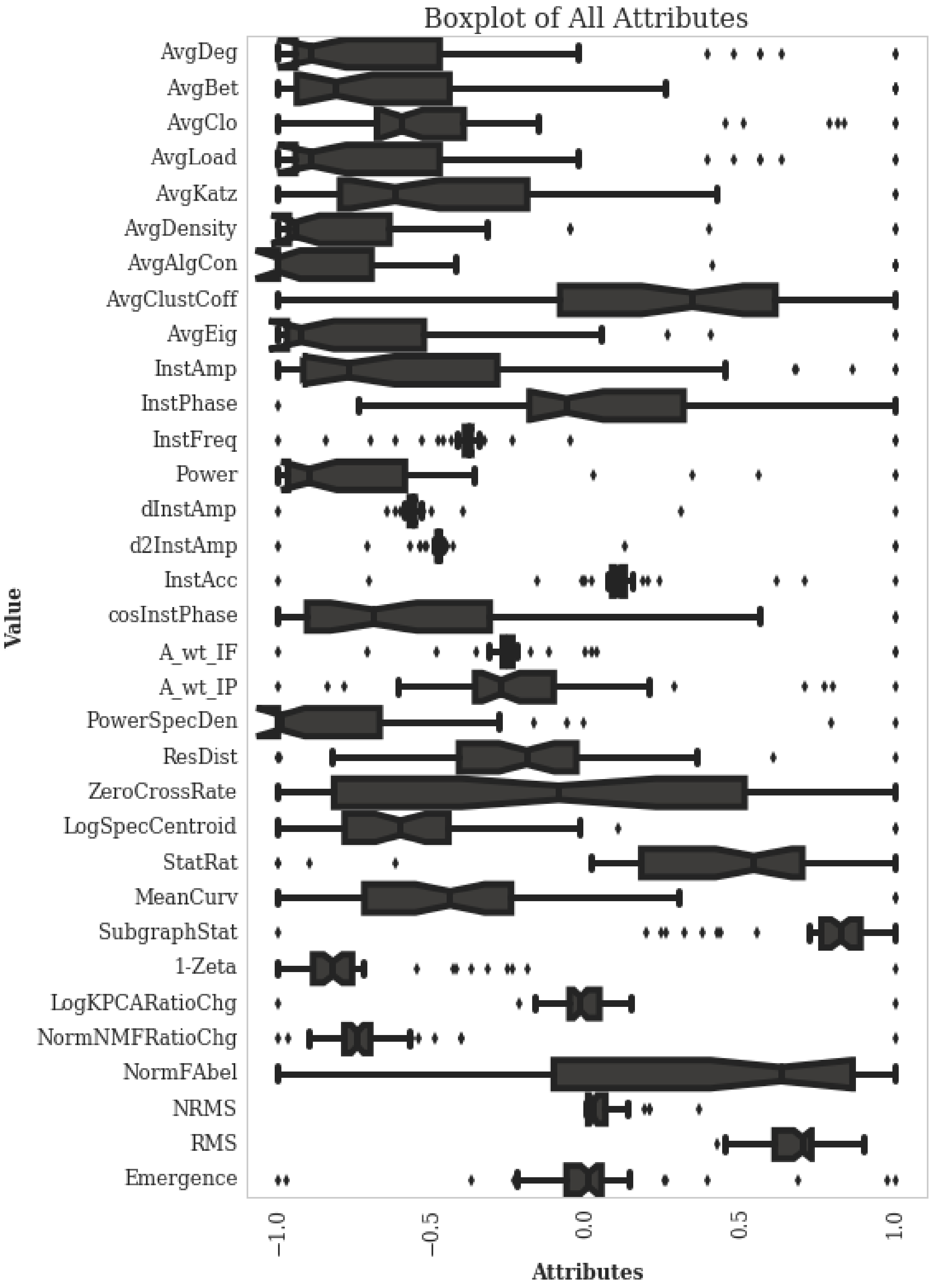
```
In [241]: node_load = pd.DataFrame.from_dict(loadC0, orient='index').\
          join(pd.DataFrame.from_dict(loadC1,orient='index'), rsuffix = 'Gt1', lsuf
          join(pd.DataFrame.from_dict(loadC2,orient='index'), rsuffix = 'Gt2', lsuf
          join(pd.DataFrame.from_dict(loadC3,orient='index'), rsuffix = 'Gt3', lsuf
          join(pd.DataFrame.from_dict(loadC4,orient='index'), rsuffix = 'Gt4', lsuf
          node_load = node_load.T
          node_load.columns = years
          node_load = node_load.T
          node_load.plot(figsize=(8,6),cmap='plasma')
          plt.title("Plot of Load Centrality of Selected nodes over years", fontsiz
          plt.xlabel("Years", fontsize=12)
          plt.ylabel("Centrality Value", fontsize=12)
          plt.legend(fontsize=12, loc=1)
          plt.xticks(years,[i for i in years],fontsize=12)
          plt.yticks(fontsize=12)
          plt.tight_layout()
          plt.savefig('images/node_load.png')
```

```
/home/arshad/anaconda3/lib/python3.5/site-packages/matplotlib/figure.py:1742: UserW
  warnings.warn("This figure includes Axes that are not "
```



```
In [210]: plt.figure(figsize=(8,11))
          plt.title("Boxplot of All Attributes", fontsize=16)
          sns.boxplot(final_attvol_m, orient='h', notch=True, color='#3D3C3A')
          plt.xlabel('Attributes', fontsize=12)
          plt.ylabel('Value', fontsize=12)
          plt.xticks(fontsize=12, rotation=90)
          plt.yticks(fontsize=12)
          plt.xlim(-1.1,1.1);
          plt.tight_layout()
          plt.savefig("images/attvol_boxplot_all.png")
```

```
/home/arshad/anaconda3/lib/python3.5/site-packages/seaborn/categorical.py:2171: Use
  warnings.warn(msg, UserWarning)
```


Boxplot of All Attributes
Value
AvgDeg
AvgBet
AvgClo
AvgLoad
AvgKatz
AvgDensity
AvgAlgCon
AvgClustCoff
AvgEig
InstAmp
InstPhase
InstFreq
Power
dInstAmp
d2InstAmp
InstAcc
cosInstPhase
A_wt_IF
A_wt_IP
PowerSpecDen
ResDist
ZeroCrossRate
LogSpecCentroid
StatRat
MeanCurv
SubgraphStat
1-Zeta
LogKPCARatioChg
NormNMFRatioChg
NormFAbel
NRMS
RMS
Emergence
−1.0
−0.5
0.0
0.5
1.0
Attributes

```
In [ ]:
```

In [ ]: