DATA VISUALIZATION

In [2]:
```
%matplotlib inline

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
```

In [3]:
```
# CASE 1: Potasium Ion Channel Kv1.2

# the input file has been generated by the following command:
# g_rmsf -f ../Trajs/mdTot_nojump.xtc -s ../steep2_out.gro -n index_anal
#
# g_rmsf is part of G R O M A C S:

inputfile1 = 'rmsf.xvg'

# Final plot: Root Mean Square Fluctuation vs Residue Number
```

# This file was created Thu Jan 7 12:59:20 2010
# by the following command:
# g_rmsf -f ../Trajs/mdTot_nojump.xtc -s ../steep2_out.gro -n index_anal.ndx -b 5000.0 -dt 10.0 #
# g_rmsf is part of G R O M A C S:
#
# Go Rough, Oppose Many Angry Chinese Serial killers #

@ title "RMS fluctuation"

@ xaxis label "Atom"

@ yaxis label "(nm)"

@TYPE xy

```
   1   0.3253


   2   0.2577

   3  0.2330

   4   0.1791

   5   0.1793
```

...

In [4]:

```

# read the input file:
rmsf = pd.read_csv(inputfile1,skiprows=12,sep='\s+',names=['Residue_Numb

#set the number of rows to be visualized
pd.set_option('max_rows',10)
# and inspect the input file:
rmsf
# or use head() and tail() methods
```

Out[4]:

	Residue_Number	RMSF
0	1	0.3253
1	2	0.2577
2	3	0.2330
3	4	0.1791
4	5	0.1793
...	...	...
2843	2844	0.1707
2844	2845	0.1805
2845	2846	0.1921
2846	2847	0.1935
2847	2848	0.2433

2848 rows × 2 columns

In [5]:

```
# to get information about our data frame:
rmsf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2848 entries, 0 to 2847
Data columns (total 2 columns):
Residue_Number    2848 non-null int64
RMSF              2848 non-null float64
dtypes: float64(1), int64(1)
memory usage: 44.6 KB
```

```
In [5]: # to get basic statistics on our data:
        rmsf.describe()
```

Out[5]:

	Residue_Number	RMSF
count	2848.000000	2848.000000
mean	1424.500000	0.205974
std	822.291108	0.078368
min	1.000000	0.085200
25%	712.750000	0.147175
50%	1424.500000	0.187800
75%	2136.250000	0.246525
max	2848.000000	0.654500

```
In [6]: # and attributes like number of rows and columns
        rmsf.shape
        #rmsf.shape[0]
        #rmsf.shape[1]

Out[6]: (2848, 2)
```

```
In [7]: # define the size of my dataset:
        mysize = int(rmsf.shape[0]/4)
        mysize

Out[7]: 712
```

```
In [8]: # create a new index for the new data structure
        pre_index=range(0,mysize,1)
        pre_index

Out[8]: range(0, 712)
```

```
In [9]: #create a new dataframe from our input data
        pre_data = {
                'ind':pd.Series(range(1,mysize+1,1),index=pre_index),
                'c1':rmsf.iloc[0:mysize,1].reset_index(drop=True),
                'c2':rmsf.iloc[mysize:mysize*2,1].reset_index(drop=True),
                'c3':rmsf.iloc[mysize*2:mysize*3,1].reset_index(drop=True),
                'c4':rmsf.iloc[mysize*3:,1].reset_index(drop=True)
                }

        pre_rmsf = pd.DataFrame(data=pre_data,index=pre_index)
        pre_rmsf.index

Out[9]: RangeIndex(start=0, stop=712, step=1)
```

```
In [10]: # show the new dataframe:
         pre_rmsf
```

Out[10]:

	c1	c2	c3	c4	ind
0	0.3253	0.2823	0.3282	0.2239	1
1	0.2577	0.1864	0.2394	0.1879	2
2	0.2330	0.1478	0.2023	0.1683	3
3	0.1791	0.1383	0.1724	0.1508	4
4	0.1793	0.1255	0.1579	0.1384	5
...	...	...	...	...	...
707	0.1490	0.1371	0.1553	0.1707	708
708	0.1734	0.1503	0.1593	0.1805	709
709	0.3099	0.1614	0.1698	0.1921	710
710	0.4350	0.2218	0.1769	0.1935	711
711	0.6545	0.2841	0.2572	0.2433	712

712 rows × 5 columns

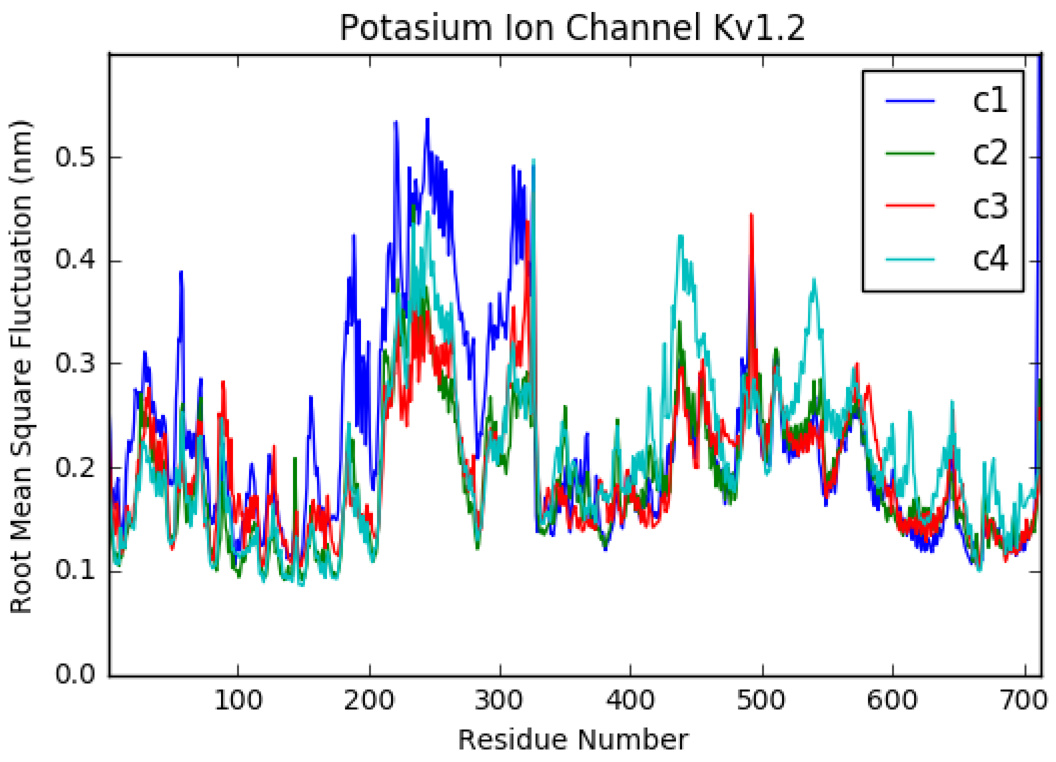
```
In [11]: # reindex the new dataframe
         new_rmsf = pre_rmsf.set_index('ind')
         new_rmsf
```

Out[11]:

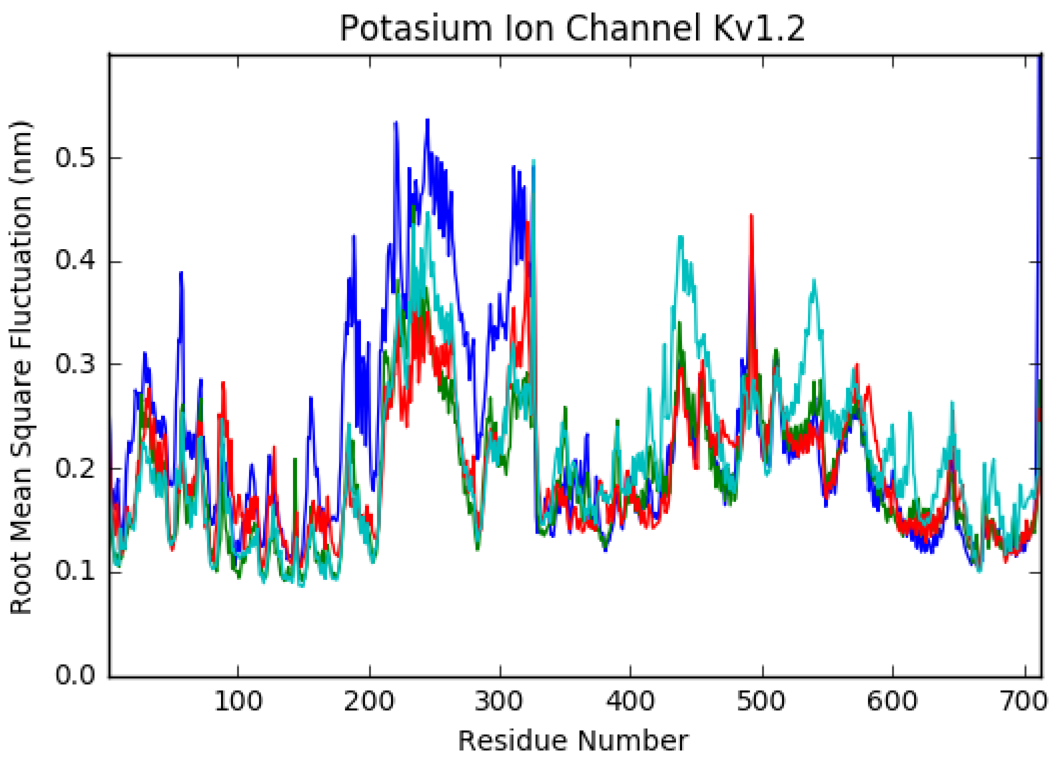
	c1	c2	c3	c4
ind				
1	0.3253	0.2823	0.3282	0.2239
2	0.2577	0.1864	0.2394	0.1879
3	0.2330	0.1478	0.2023	0.1683
4	0.1791	0.1383	0.1724	0.1508
5	0.1793	0.1255	0.1579	0.1384
...	...	...	...	...
708	0.1490	0.1371	0.1553	0.1707
709	0.1734	0.1503	0.1593	0.1805
710	0.3099	0.1614	0.1698	0.1921
711	0.4350	0.2218	0.1769	0.1935
712	0.6545	0.2841	0.2572	0.2433

712 rows × 4 columns

```
In [12]: # plot the data:
         new_rmsf.plot()
         plt.axis([1, 712, 0.0, 0.6])
         plt.xlabel('Residue Number')
         plt.ylabel('Root Mean Square Fluctuation (nm)')
         plt.title('Potasium Ion Channel Kv1.2')

Out[12]: <matplotlib.text.Text at 0x7f4176e605c0>
```



```
In [13]: # drop the legend:
         new_rmsf.plot(legend=False)
         plt.axis([1, 712, 0.0, 0.6])
         plt.xlabel('Residue Number')
         plt.ylabel('Root Mean Square Fluctuation (nm)')
         plt.title('Potasium Ion Channel Kv1.2')

Out[13]: <matplotlib.text.Text at 0x7f417553d7f0>
```



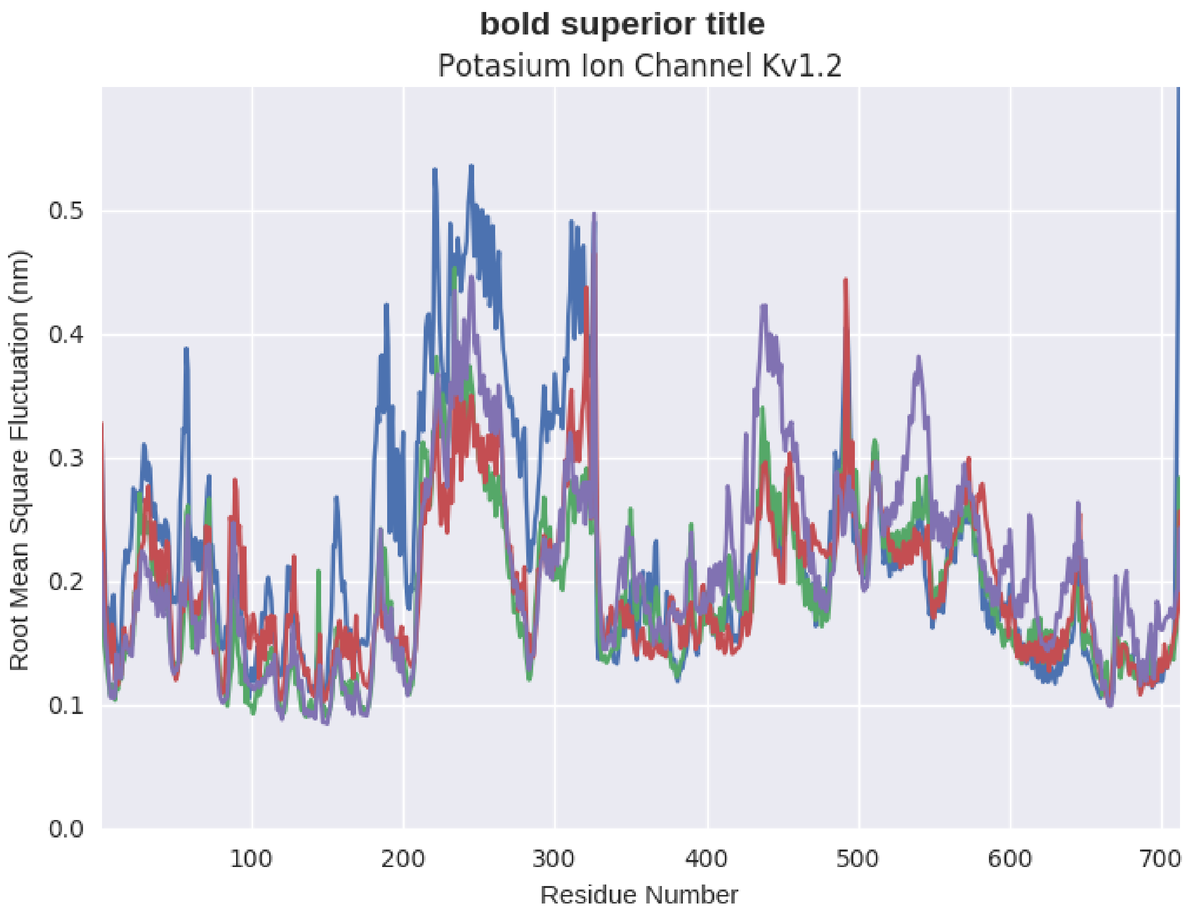
A level up!

built on top of matplotlib

import seaborn as sns

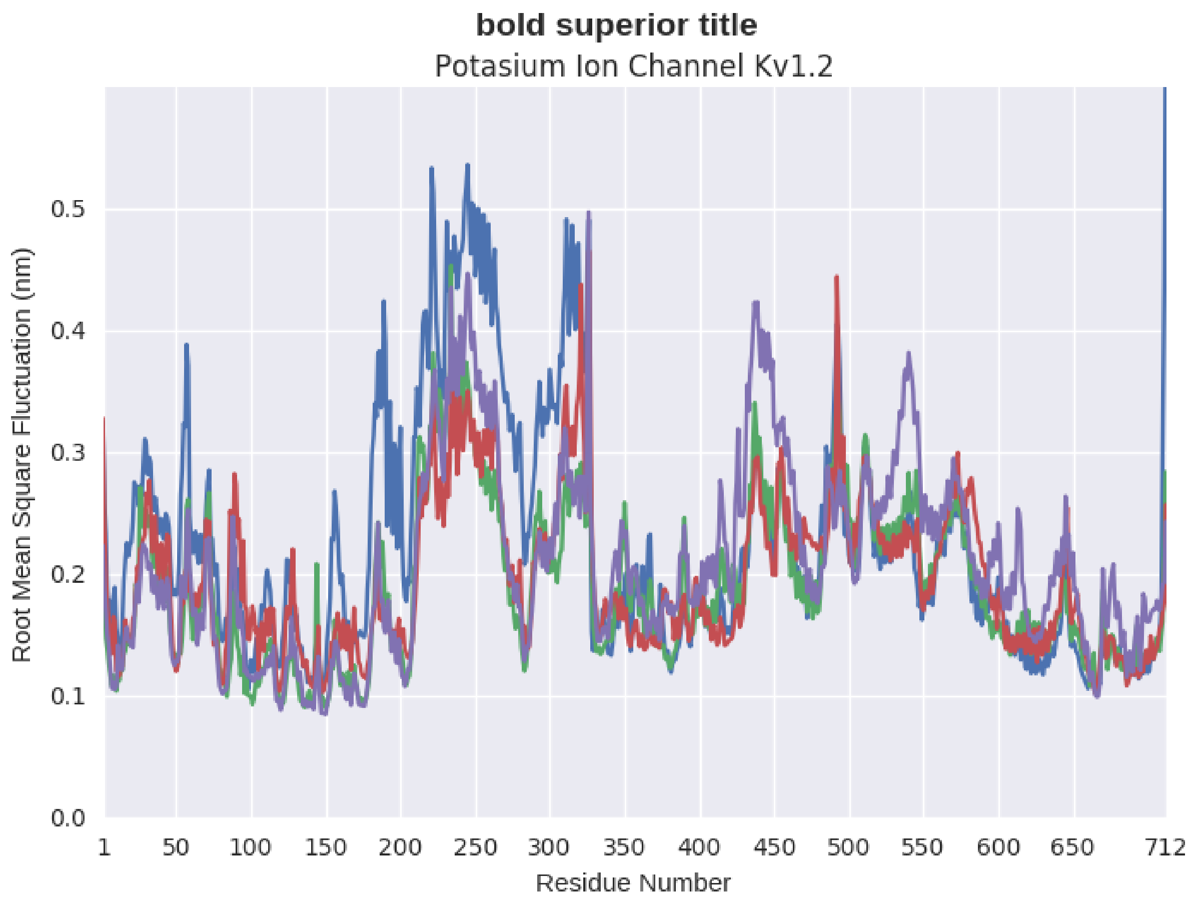
```
In [14]: import seaborn as sns

         new_rmsf.plot(legend=False)
         plt.axis([1, 712, 0.0, 0.6])
         plt.xlabel('Residue Number')
         plt.ylabel('Root Mean Square Fluctuation (nm)')
         plt.suptitle('bold superior title', fontsize=14, fontweight='bold')
         plt.title('Potasium Ion Channel Kv1.2')
```

```
Out[14]: <matplotlib.text.Text at 0x7f4165b38860>
```

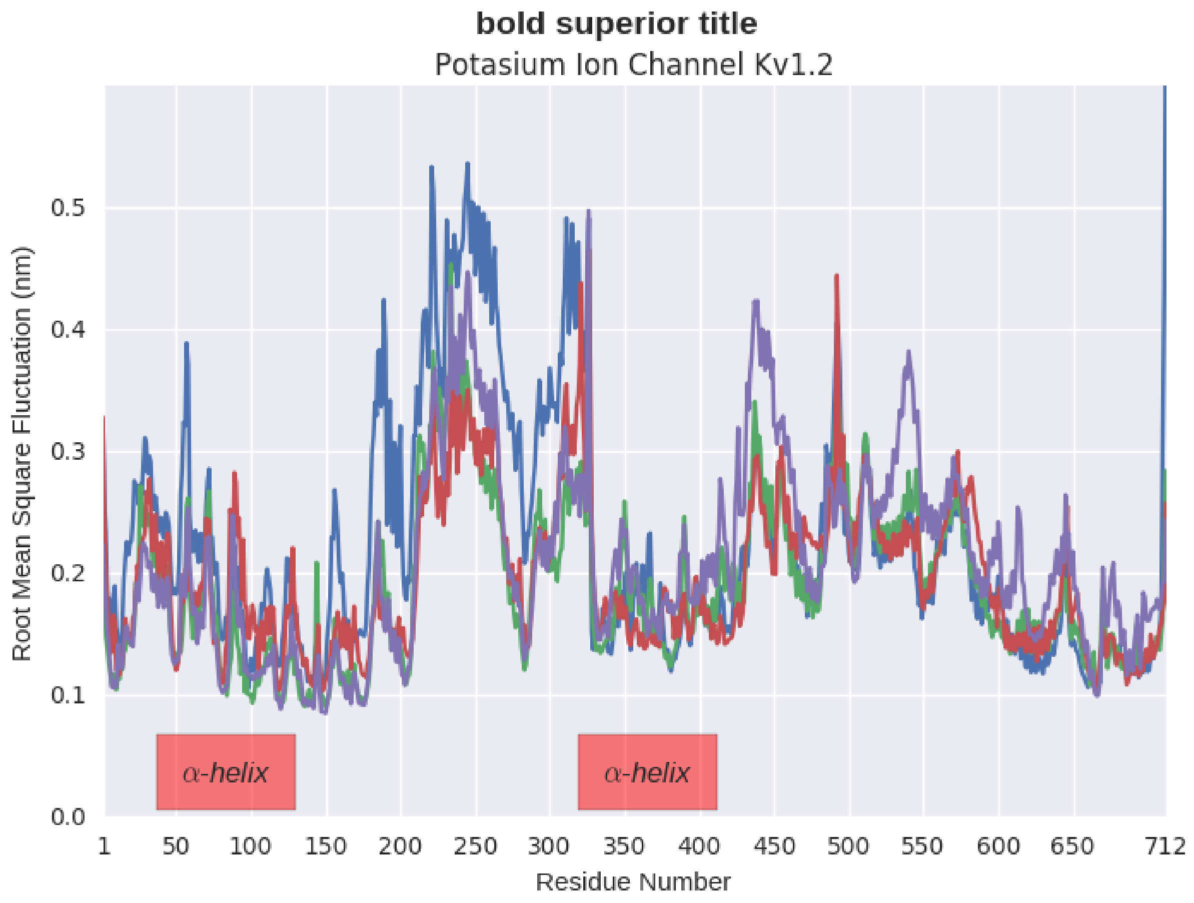


```
In [38]: new_rmsf.plot(legend=False)
         plt.axis([1, 712, 0.0, 0.6])
         plt.xlabel('Residue Number')
         plt.ylabel('Root Mean Square Fluctuation (nm)')
         plt.suptitle('bold superior title', fontsize=14, fontweight='bold')
         plt.title('Potasium Ion Channel Kv1.2')
         # set the major and minor xticks and their labels for the graph
         a=np.append([1],range(50,700,50))
         b=np.append(a,[712])
         plt.xticks(b)
```

```
Out[38]: ([<matplotlib.axis.XTick at 0x7f41651c6198>,
           <matplotlib.axis.XTick at 0x7f41656c7dd8>,
           <matplotlib.axis.XTick at 0x7f41650d8f28>,
           <matplotlib.axis.XTick at 0x7f4164fb7da0>,
           <matplotlib.axis.XTick at 0x7f4164fbb8d0>,
           <matplotlib.axis.XTick at 0x7f4164fbf400>,
           <matplotlib.axis.XTick at 0x7f4164fbfef0>,
           <matplotlib.axis.XTick at 0x7f4164fc1a20>,
           <matplotlib.axis.XTick at 0x7f4164fc3550>,
           <matplotlib.axis.XTick at 0x7f41652671d0>,
           <matplotlib.axis.XTick at 0x7f4164f95860>,
           <matplotlib.axis.XTick at 0x7f4164fb7668>,
           <matplotlib.axis.XTick at 0x7f416528cc88>,
           <matplotlib.axis.XTick at 0x7f4164fcbf98>,
           <matplotlib.axis.XTick at 0x7f4164fc3240>],
          <a list of 15 Text xticklabel objects>)
```

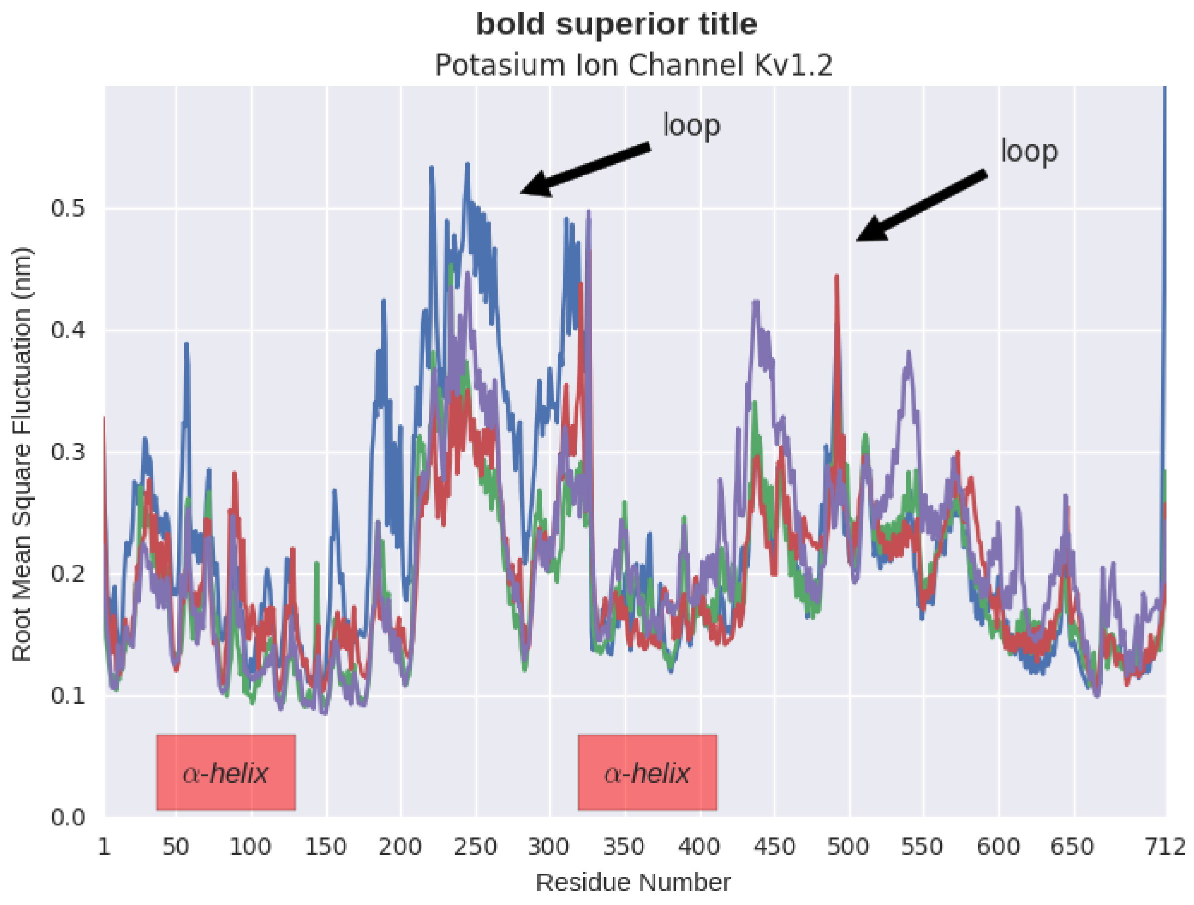


```
In [37]: # insert text in data coordinates
         # usage of greek letters
         from matplotlib import rc
         new_rmsf.plot(legend=False)
         plt.axis([1, 712, 0.0, 0.6])
         plt.xlabel('Residue Number')
         plt.ylabel('Root Mean Square Fluctuation (nm)')
         plt.suptitle('bold superior title', fontsize=14, fontweight='bold')
         plt.title('Potasium Ion Channel Kv1.2')
         # set the major and minor xticks and their labels for the graph
         a=np.append([1],range(50,700,50))
         b=np.append(a,[712])
         plt.xticks(b)
         plt.text(53,0.03, r'$\alpha$-helix', style='italic',
             bbox={'facecolor':'red', 'alpha':0.5, 'pad':10})

         plt.text(335,0.03, r'$\alpha$-helix', style='italic',
             bbox={'facecolor':'red', 'alpha':0.5, 'pad':10})

Out[37]: <matplotlib.text.Text at 0x7f4164fe7080>
```



```
In [36]: # other annotations
         new_rmsf.plot(legend=False)
         plt.axis([1, 712, 0.0, 0.6])
         plt.xlabel('Residue Number')
         plt.ylabel('Root Mean Square Fluctuation (nm)')
         plt.suptitle('bold superior title', fontsize=14, fontweight='bold')
         plt.title('Potasium Ion Channel Kv1.2')
         # set the major and minor xticks and their labels for the graph
         a=np.append([1],range(50,700,50))
         b=np.append(a,[712])
         plt.xticks(b)
         plt.text(53,0.03, r'$\alpha$-helix', style='italic',
             bbox={'facecolor':'red', 'alpha':0.5, 'pad':10})
         plt.text(335,0.03, r'$\alpha$-helix', style='italic',
             bbox={'facecolor':'red', 'alpha':0.5, 'pad':10})

         plt.annotate('loop', xy=(275, 0.51), xytext=(375, 0.56),
             arrowprops=dict(facecolor='black', shrink=0.05))

         plt.annotate('loop', xy=(500, 0.47), xytext=(600, 0.54),
             arrowprops=dict(facecolor='black', shrink=0.05))
```

```
Out[36]: <matplotlib.text.Annotation at 0x7f4165066e80>
```



In [18]:
```
# CASE 2: correlation map visualization
inputfile2='covfarb.dat'

# inspect the raw data:
```

In [19]:
```
data_1 = pd.read_csv('covfarb.dat',skiprows=2,sep='\s+',header=None)
data_1
```

Out[19]:

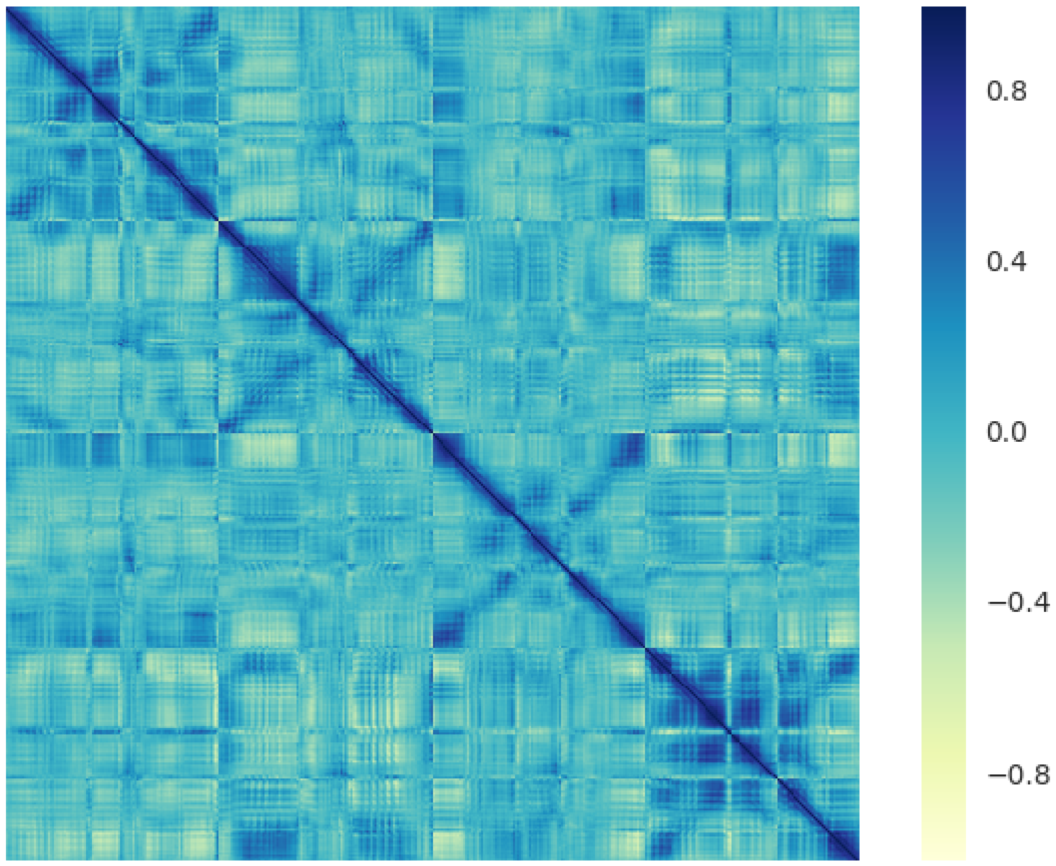
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	1.000000	0.775776	0.621853	0.672662	0.596661	0.489385	0.446742	0.418091
1	0.775776	1.000000	0.797844	0.587230	0.570801	0.492044	0.439680	0.366097
2	0.621853	0.797844	1.000000	0.790597	0.679336	0.661124	0.620815	0.496696
3	0.672662	0.587230	0.790597	1.000000	0.847841	0.759177	0.742630	0.684790
4	0.596661	0.570801	0.679336	0.847841	1.000000	0.867817	0.736696	0.712400
...	...	...	...	...	...	...	...	...
431	-0.058320	-0.075145	-0.018434	-0.075299	-0.173802	-0.153937	-0.083545	-0.183857
432	-0.016182	-0.041609	0.022400	-0.010943	-0.121460	-0.106748	0.002092	-0.088404
433	-0.056576	-0.085644	-0.039047	-0.057091	-0.132342	-0.113577	-0.022250	-0.099796
434	-0.015967	-0.083077	-0.050013	-0.037743	-0.105828	-0.095469	-0.015076	-0.079748
435	0.052056	-0.014082	0.009690	0.021066	-0.061008	-0.063926	0.012068	-0.049495

436 rows × 436 columns

In [20]:
```
data_1.shape
```

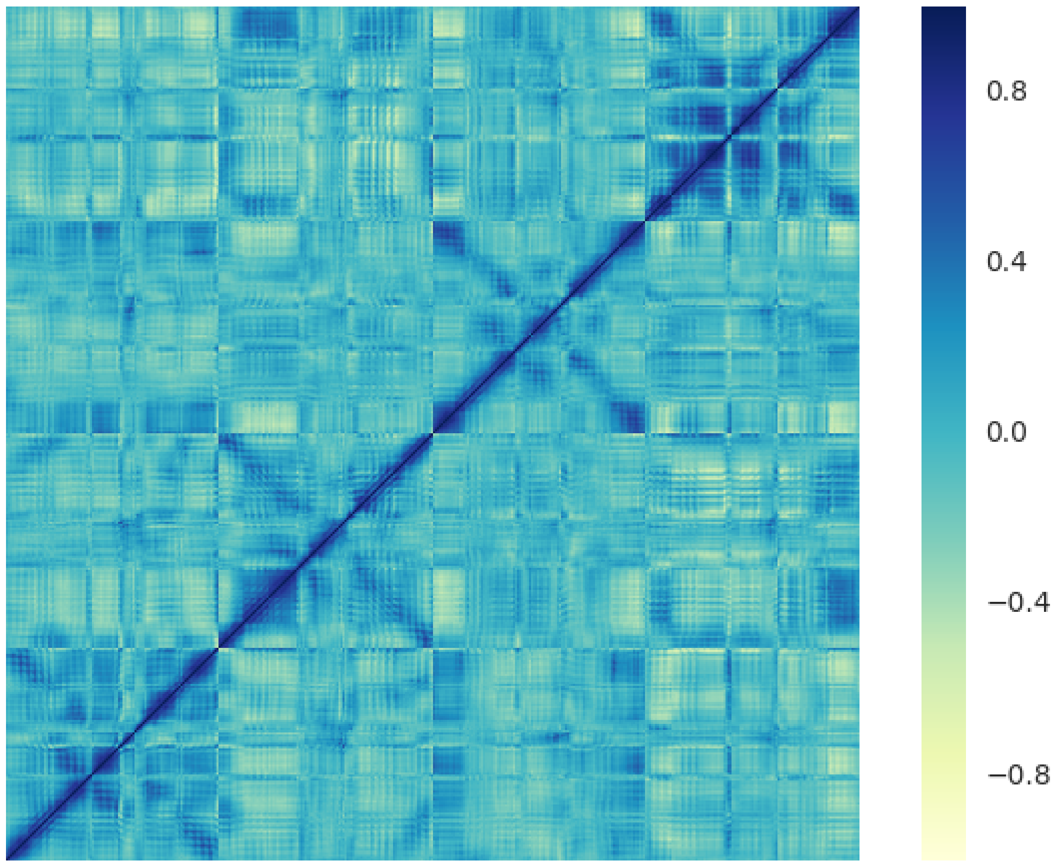
Out[20]:
```
(436, 436)
```

```
In [39]: sns.heatmap(data_1.iloc[:],square=True,vmin=-1, vmax=1,cmap="YlGnBu",xti

Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x7f416526b940>
```



In [21]:
```
sns.heatmap(data_1.iloc[::-1],square=True,vmin=-1, vmax=1,cmap="YlGnBu",
```

Out[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4165903b00>
```



In [22]:
```
# CASE 3: Principal Component Analysis
inputfile3='projection2_mod.xvg'

# inspect the raw data:
# @ comment rows
# & end line for each dataset

# grep '&' projection2_mod.xvg | wc -l
# 2 different dataset in the raw data
```

```
In [23]: # read the raw data:
         data = pd.read_csv('projection2.xvg',skiprows=2,sep='\s+',header=None,
             comment='@',names=['Col1','Col2'])

         # explore the dataset:
         data
```

Out[23]:

	Col1	Col2
0	100.0000	2.22866
1	105.0000	2.21076
2	110.0000	2.24238
3	115.0000	2.28658
4	120.0000	2.19461
...	...	...
23375	145750.0000	1.38455
23376	145760.0000	1.11736
23377	145770.0000	1.34684
23378	145780.0000	1.33750
23379	&	NaN

23380 rows × 2 columns

```
In [24]: data[data['Col1'] == '&']
```

Out[24]:

	Col1	Col2
11689	&	NaN
23379	&	NaN

```
In [25]: nr_ds1 = data[data['Col1'] == '&'].index[0]
         nr_ds1

Out[25]: 11689

In [26]: nr_ds2 = data[data['Col1'] == '&'].index[1]
         nr_ds2

Out[26]: 23379

In [27]: dif = nr_ds2 - nr_ds1
         dif

Out[27]: 11690
```

In [28]:
```
# read the same raw data file with no comment lines:
ds = pd.read_csv('projection2_mod.xvg',sep='\s+',header=None,nrows=nr_ds
ds
```

Out[28]:

	ds1
0	2.22866
1	2.21076
2	2.24238
3	2.28658
4	2.19461
...	...
11684	-2.29655
11685	-2.40724
11686	-2.39260
11687	-2.17884
11688	-2.10903

11689 rows × 1 columns

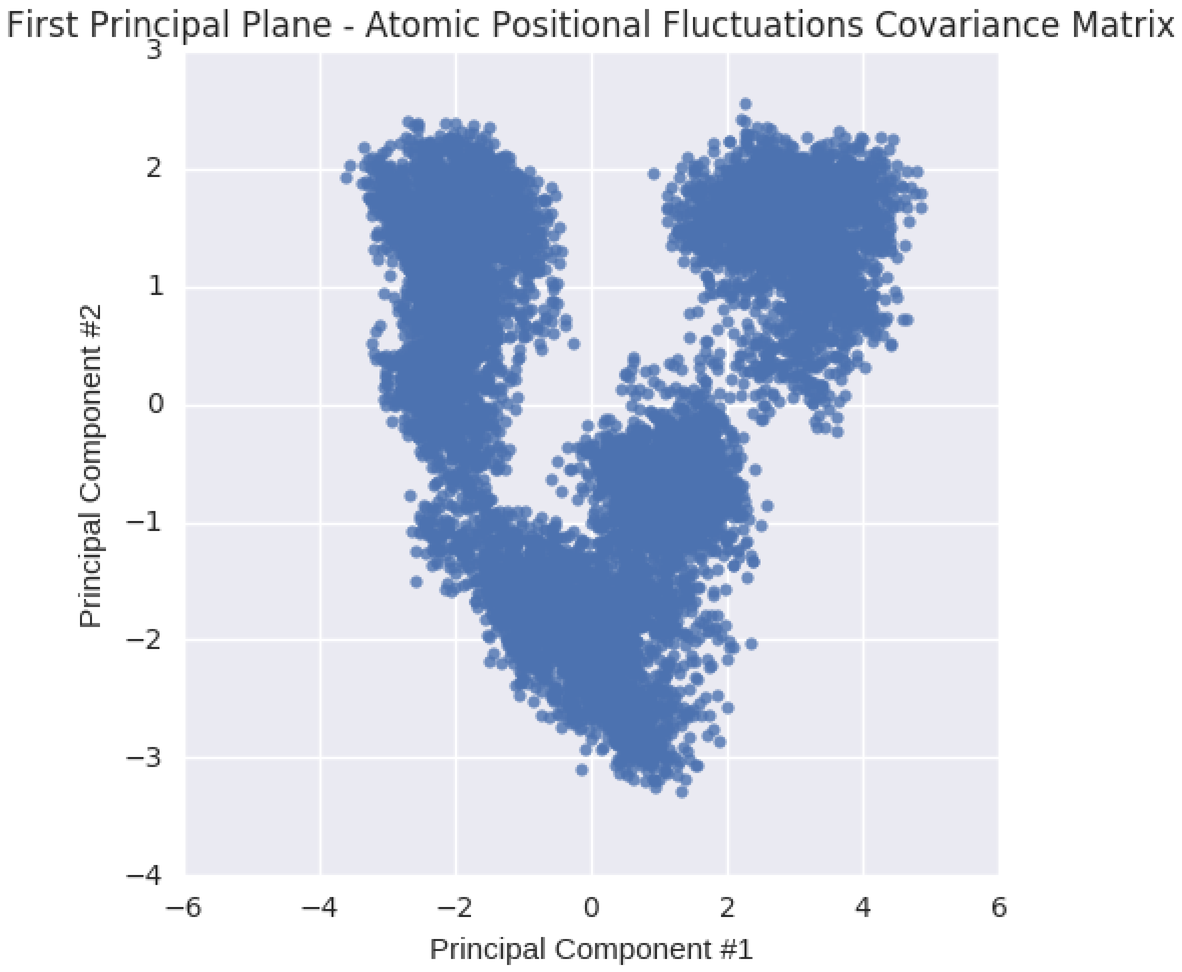
In [29]:
```
# read the second dataset:
ds['ds2'] = pd.read_csv('projection2_mod.xvg',sep='\s+',header=nr_ds1,nr
ds
```

Out[29]:

	ds1	ds2
0	2.22866	1.89379
1	2.21076	1.90674
2	2.24238	1.91264
3	2.28658	1.89820
4	2.19461	1.85189
...	...	...
11684	-2.29655	1.13738
11685	-2.40724	1.38455
11686	-2.39260	1.11736
11687	-2.17884	1.34684
11688	-2.10903	1.33750

11689 rows × 2 columns

```
In [30]: # plot dataset 1 vs dataset 2:
         sns.lmplot(x="ds1", y="ds2",data=ds,fit_reg=False)
         plt.xlabel('Principal Component #1')
         plt.ylabel('Principal Component #2')
         plt.title('First Principal Plane - Atomic Positional Fluctuations Covari

Out[30]: <matplotlib.text.Text at 0x7f416573af60>
```


First Principal Plane - Atomic Positional Fluctuations Covariance Matrix
Principal Component #2
Principal Component #1