

REF1II5_master_w-
ADD_432seqs_centralTM_removed

PDB: 1II5 - May 25, 2023

1 SCA 6.1 - REF1II5_master_w-ADD_432seqs_centralTM_removed

```
[ ]: import os
import time
import matplotlib.pyplot as plt
import math
import numpy as np
import copy
import scipy.cluster.hierarchy as sch
from scipy.stats import scoreatpercentile
from pysca import scaTools as sca
import colorsys
# import mpld3
import pickle as pickle
from optparse import OptionParser

if not os.path.exists('../output/'):
    os.makedirs('../output/')

%matplotlib inline
```

Read in the results of the above three scripts (scaProcessMSA, scaCore and scaSectorID), stored as three dictionaries in the database REF1II5_master_w-ADD_432seqs_centralTM_removed.db. To see what variables are stored in each dictionary, use:

```
>>> list(db)
```

```
[ ]: db = pickle.load(open('../output/REF1II5_master_w-ADD_432seqs_centralTM_removed.
↳db', 'rb'))
Dseq = db['sequence']
Dsca = db['sca']
Dsect = db['sector']
```

1.0.1 I. Statistical Structure of the Multiple Sequence Alignment (MSA)

Plot a histogram of all pairwise sequence identities (*left panel*) and a global view of the sequence similarity matrix (defined by $S \equiv \frac{1}{L}XX^T$) (*right panel*). From the matrix at right, it is clear that the alignment is composed of several distinct sequence families.

```
[ ]: # List all elements above the diagonal (i<j):
listS = [Dzca['simMat'][i,j] for i in range(Dzca['simMat'].shape[0]) \
        for j in range(i+1, Dzca['simMat'].shape[1])]

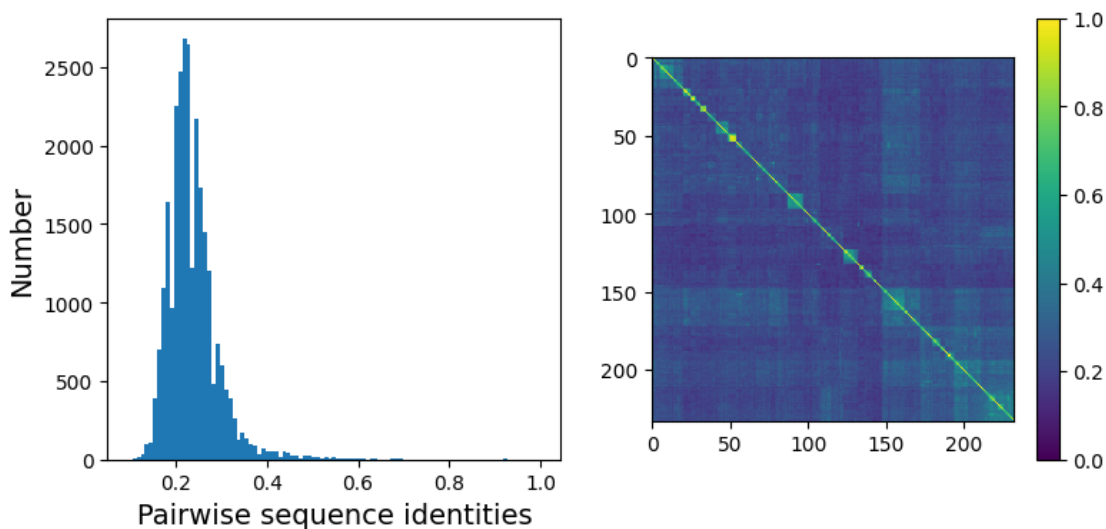
# Cluster the sequence similarity matrix
Z = sch.linkage(Dzca['simMat'], method = 'complete', metric = 'cityblock')
R = sch.dendrogram(Z,no_plot = True)
ind = R['leaves']
```

```
[ ]: # Plotting
plt.rcParams['figure.figsize'] = 9, 4

plt.subplot(121)
plt.hist(listS, math.floor(Dseq['Npos']/2))
plt.xlabel('Pairwise sequence identities', fontsize=14)
plt.ylabel('Number', fontsize=14)

plt.subplot(122)
plt.imshow(Dzca['simMat'][np.ix_(ind,ind)], vmin=0, vmax=1)
plt.colorbar()
```

```
[ ]: <matplotlib.colorbar.Colorbar at 0x7f791dbf9fe0>
```



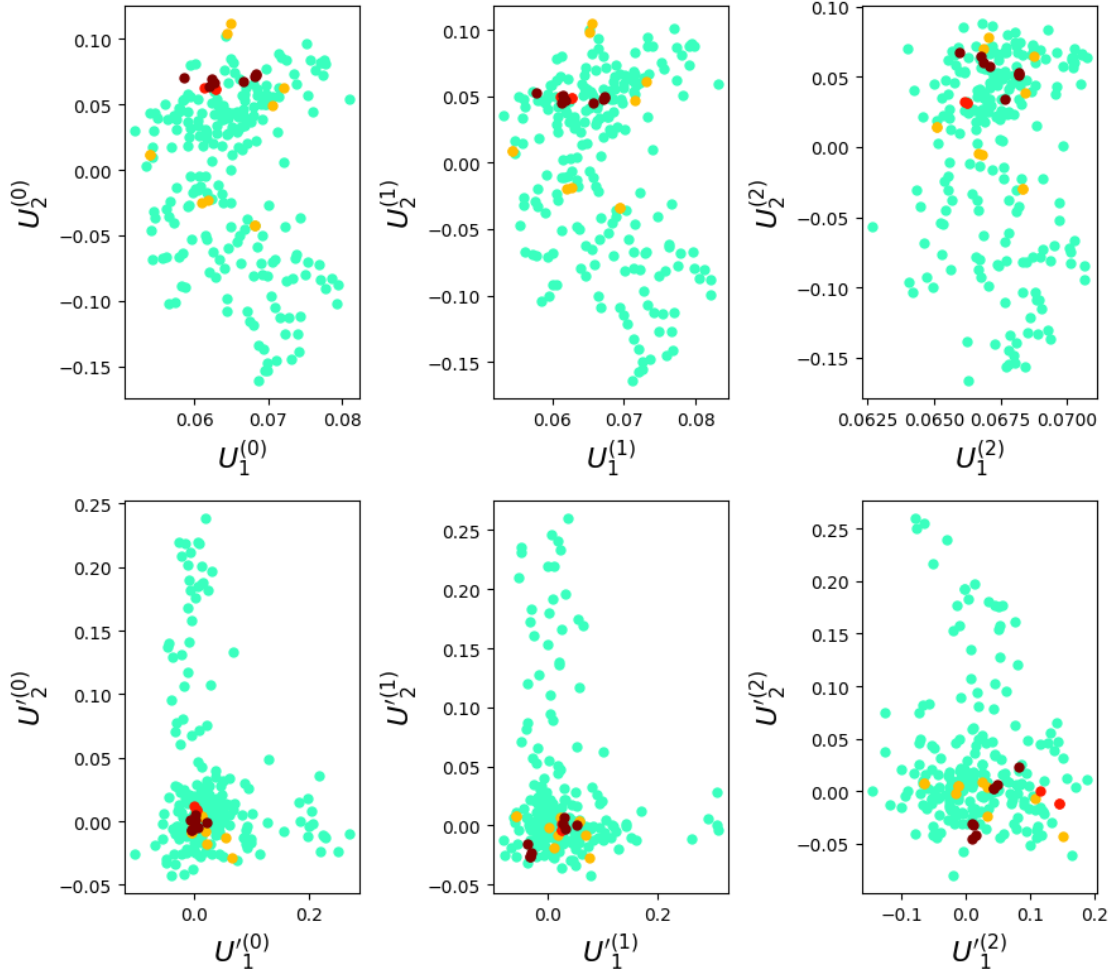
To examine the role of sequence and position weighting on the structure of the sequence space, we compute correlation matrices between all pairs of sequences, either with or without sequence and position weights and project the corresponding sequence space (by eigenvalue decomposition) down to a small set of top modes that contain the statistically dominant relationships between sequences. Since eigenvalue decomposition does not necessarily provide the best representation of sequence groups (for reasons described in “xx”), we also apply independent components analysis (or ICA)

to the top few eigenmodes; this manipulation provides a representation in which the top groupings of sequences in the alignment (if such exists) should separate along the so-called independent components (or ICs). Below we plot the following eigenmodes (*top row*) and independent components (*bottom row*):

- $U^{(0)}$ and $U'^{(0)}$, the top eigenmodes and ICs without any weights;
- $U^{(1)}$ and $U'^{(1)}$ the top eigenmodes and ICs with sequences weights;
- $U^{(2)}$ and $U'^{(2)}$ the top eigenmodes and ICs with both sequences and positional weights.

The sequences are colored by weight, with red indicating the most strongly downweighted sequences.

```
[ ]: Useq = Dsca['Useq']
      Uica = Dsca['Uica']
      plt.rcParams['figure.figsize'] = 9, 8
      ica = ["", "", "", "", "", "", ""]
      for k,U in enumerate(Useq+Uica):
          plt.subplot(2,3,k+1)
          sca.figWeights(U[:,0], U[:,1], Dseq['seqw'][0])
          plt.xlabel(r'${U}s}^{\{i\}}_1$(ica[k],k%3), fontsize=16)
          plt.ylabel(r'${U}s}^{\{i\}}_2$(ica[k],k%3), fontsize=16)
      plt.tight_layout()
```

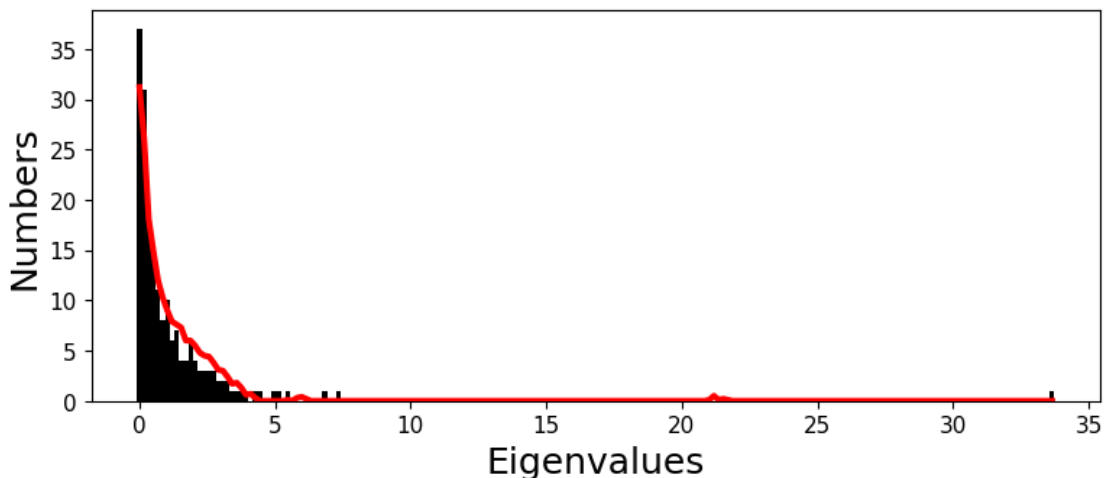


1.0.2 II. SCA conservation and coevolution

Plot the eigenspectrum of the SCA positional coevolution matrix ($\tilde{C}_{ij}$) (*black bars*) and 10 trials of matrix randomization for comparison (*red line*). This graph is used to choose the number of significant eigenmodes.

```
[ ]: plt.rcParams['figure.figsize'] = 9, 3.5
hist0, bins = np.histogram(Dsca['Lrand'].flatten(), bins=Dseq['Npos'], \
                           range=(0,Dsect['Lsca'].max()))
hist1, bins = np.histogram(Dsect['Lsca'], bins=Dseq['Npos'], \
                           range=(0,Dsect['Lsca'].max()))
plt.bar(bins[:-1], hist1, np.diff(bins),color='k')
plt.plot(bins[:-1], hist0/Dsca['Ntrials'], 'r', linewidth=3)
plt.tick_params(labelsize=11)
plt.xlabel('Eigenvalues', fontsize=18); plt.ylabel('Numbers', fontsize=18);
print('Number of eigenmodes to keep is %i' %(Dsect['kpos']))
```

Number of eigenmodes to keep is 3



To define the positions with significant contributions each of the independent components (ICs), we make an empirical fit for each IC to the t-distribution and select positions with greater than a specified cutoff on the CDF. We choose $p = 0.95$ as our cutoff. Note that since some positions might contribute significantly to more than one IC (and indication of non-independence of ICs), we apply a simple algorithm to assign such positions to one IC. Specifically, we assign positions to the IC with which it has the greatest degree of co-evolution.

The data indicate generally good fits for the top six ICs, and we return the positions contributing to each IC in a format suitable for cut and paste into PyMol.

```
[ ]: plt.rcParams['figure.figsize'] = 10,5

Vpica = Dsect['Vpica']
for k in range(Dsect['kpos']):
    iqr = scoreatpercentile(Vpica[:,k],75) - scoreatpercentile(Vpica[:,k],25)
    binwidth=2*iqr*(len(Vpica)**(-0.33))
    nbins=int(round((max(Vpica[:,k])-min(Vpica[:,k]))/binwidth))
    plt.subplot(1,Dsect['kpos'],k+1)
    h_params = plt.hist(Vpica[:,k], nbins)
    x_dist = np.linspace(min(h_params[1]), max(h_params[1]), num=100)
    plt.plot(x_dist,Dsect['scaled_pd'][k], 'r', linewidth = 2)
    plt.xlabel(r'$V^p_{\%i}$' % (k+1), fontsize=14)
    plt.ylabel('Number', fontsize=14)

for n,ipos in enumerate(Dsect['ics']):
    sort_ipos = sorted(ipos.items)
    ats_ipos = ([Dsect['ats'][s] for s in sort_ipos])
    ic_pymol = ('+'.join(ats_ipos))
    print('IC %i is composed of %i positions:' % (n+1,len(ats_ipos)))
```

```
print(ic_pymol + "\n")
```

IC 1 is composed of 19 positions:

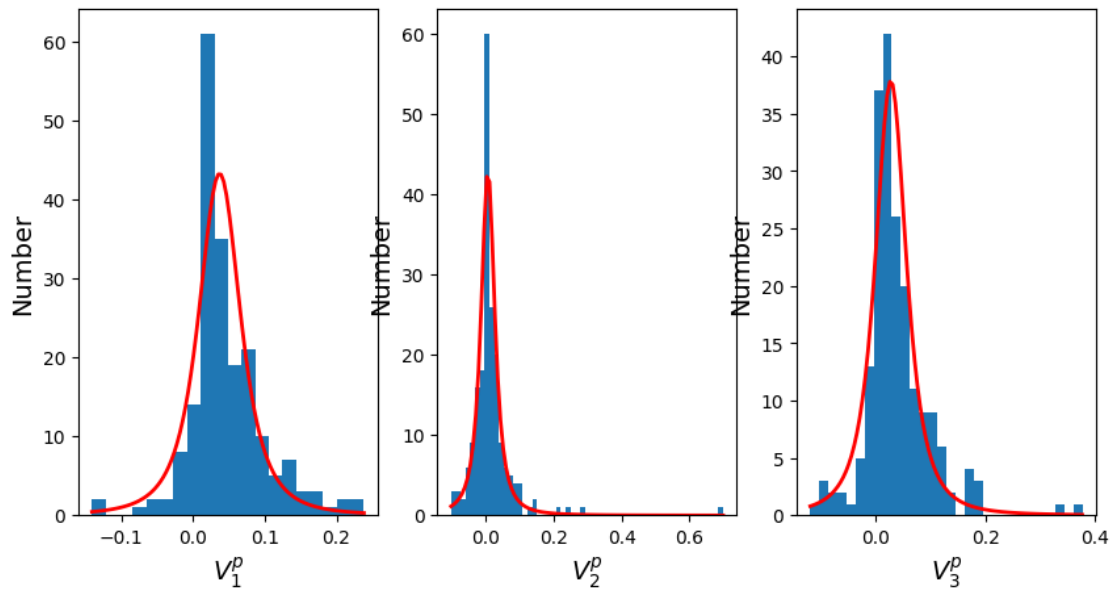
15+29+67+75+76+79+87+90+95+100+114+126+151+158+162+190+191+205+208

IC 2 is composed of 10 positions:

27+30+33+116+124+154+215+217+223+224

IC 3 is composed of 16 positions:

16+31+35+73+89+91+96+98+121+125+160+166+168+196+198+209



Sector 1: ICs 1 and 3 Sector 2: IC 2

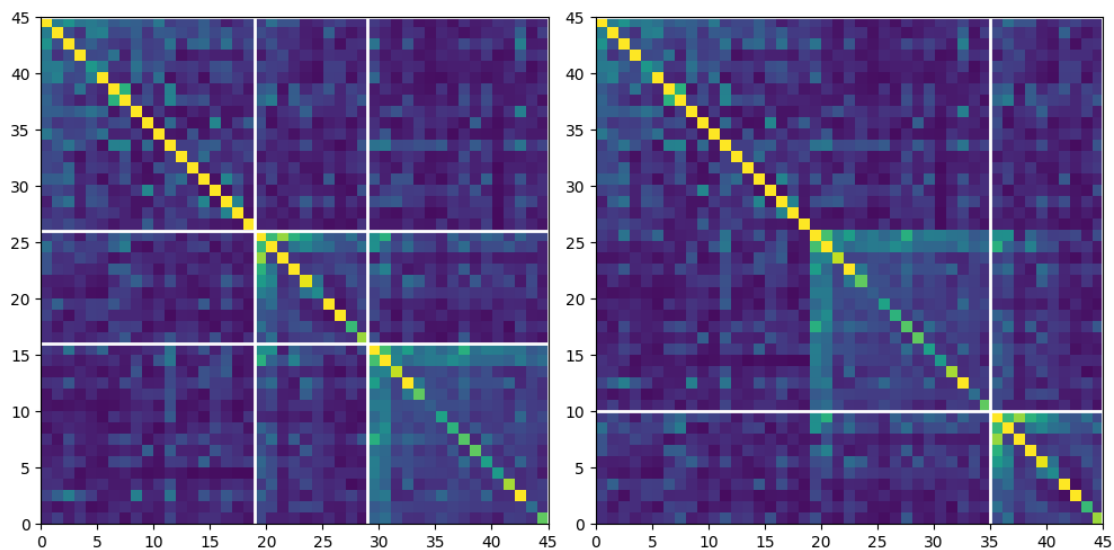
```
[ ]: # plot the SCA positional correlation matrix, ordered by contribution to the
      ↪ top ICs
plt.rcParams['figure.figsize'] = 10, 10
plt.subplot(121)
plt.imshow(Dsca['Csca'][np.ix_(Dsect['sortedpos'], Dsect['sortedpos'])], \
           vmin=0, vmax=2, interpolation='none', aspect='equal', \
           extent=[0, sum(Dsect['icsize']), 0, sum(Dsect['icsize'])])
line_index=0
for i in range(Dsect['kpos']):
    plt.plot([line_index+Dsect['icsize'][i], line_index+Dsect['icsize'][i]], \
             [0, sum(Dsect['icsize'])], 'w', linewidth = 2)
    plt.plot([0, sum(Dsect['icsize'])], [sum(Dsect['icsize'])-line_index, \
                                         sum(Dsect['icsize'])-line_index], 'w', linewidth = 2)
    line_index += Dsect['icsize'][i]
```

```

# **** MANUAL SELECTION **** define the new sector groupings - 2 total
sec_groups = ([0,2],[1])
sectors = list()
for n,k in enumerate(sec_groups):
    s = sca.Unit()
    all_items = list()
    for i in k: all_items = all_items+Dsect['ics'][i].items
    s.items = all_items
    s.col = (1/len(sec_groups))*n
    sectors.append(s)

# plot the re-ordered matrix
plt.subplot(122)
line_index=0
sortpos = list()
for s in sectors:
    sortpos.extend(s.items)
plt.imshow(Dsca['Csca'][np.ix_(sortpos, sortpos)], vmin=0, vmax=2,\
           interpolation='none',aspect='equal',\
           extent=[0,len(sortpos),0,len(sortpos)])
for s in sectors:
    plt.plot([line_index+len(s.items),line_index+len(s.items)],\
             [0,len(sortpos)],'w', linewidth = 2)
    plt.plot([0,sum(Dsect['icsize'])],[len(sortpos)-line_index, \
                                         len(sortpos)-line_index], 'w', linewidth_
↵= 2)
    line_index += len(s.items)
plt.tight_layout()

```



Print the sector positions, in a format suitable for pyMol, and create a pyMol session with the sectors (and decomposition into independent components) as separate objects.

```
[ ]: for i,k in enumerate(sectors):
    sort_ipos = sorted(k.items)
    ats_ipos = ([Dseq['ats'][s] for s in sort_ipos])
    ic_pymol = ('+'.join(ats_ipos))
    print('Sector %i is composed of %i positions:' % (i+1,len(ats_ipos)))
    print(ic_pymol + "\n")
sca.writePymol('../data/1iii5.pdb', sectors, Dsect['ics'], Dseq['ats'], \
               '../output/REF1III5_master_w-ADD_432seqs_centralTM_removed.pml', \
               ↪ 'A', '../data/', 0)
```

Sector 1 is composed of 35 positions:

15+16+29+31+35+67+73+75+76+79+87+89+90+91+95+96+98+100+114+121+125+126+151+158+160+162+166+168+190+191+196+198+205+208+209

Sector 2 is composed of 10 positions:

27+30+33+116+124+154+215+217+223+224