SUPPLEMENTARY INFORMATION

SF3b1: A Dynamic Achilles Heel of Spliceosome Assembly

Debanjana Maji, Alan Grossfield, and Clara L. Kielkopf
Department of Biochemistry and Biophysics, University of Rochester School of Medicine
Correspondence: clara_kielkopf@urmc.rochester.edu

Table S1. Residue ranges used for analysis of the human SF3b1 superhelix in Fig. 6 and Fig. 7A.

α-Helical Repeat[1]	Residue Range
HR01	508-524
HR02	545-561
HR03	588-603
HR04	623-639
HR05	659-675
HR06	698-714
HR07	741-757
HR08	781-797
HR09	828-844
HR10	861-876
HR11	906-922
HR12	948-964
HR13	986-1002
HR14	1028-1045
HR15	1072-1088
HR16	1109-1125
HR17	1140-1160
HR18	1182-1198
HR19	1223-1239
HR20	1258-1277

[1]The C-terminal α-helix of each di-helical HEAT repeat is used for analysis, which avoids potential artifacts that could arise from inclusion of a disordered N-terminal α-helix in HR16 of PDB ID 5IFE (residues 1093 – 1106).

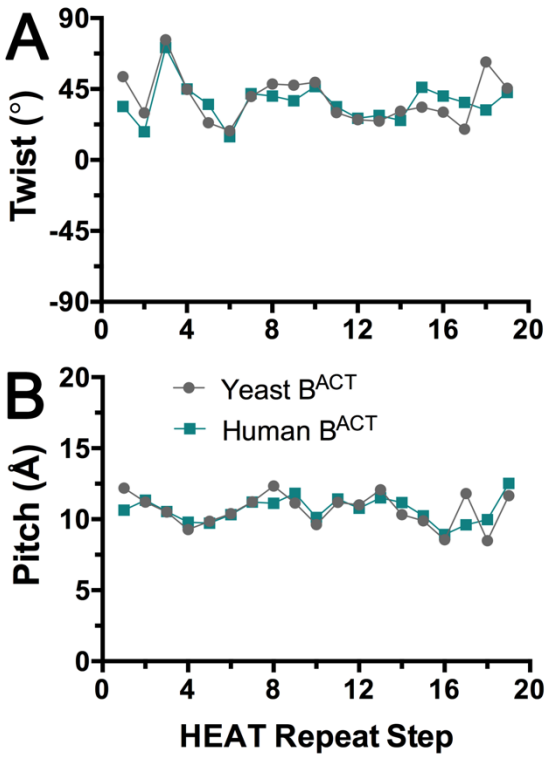

Fig. S1. Relationship between consecutive HEAT repeats of the SF3b1 superhelix in the context of the *Saccharomyces cerevisiae* (gray, PDB ID 5GM6) compared to human B^{ACT} spliceosomes (teal, PDB ID 6FF4). Since all relevant residues are assigned in both structures, the dihelical repeats are analyzed (representative scripts and residue ranges in Supplementary Materials, below). Otherwise, the plots are similar to Fig. 6. The intramolecular step from one HEAT repeat to the next is analyzed in the indicated superhelix. (**A**) The rotational angle (twist) between the principal axes of α-helices from consecutive HEAT repeats. (**B**) Translational shift (pitch) between centroids of α-helices from consecutive HEAT repeats.

Supplementary Materials for Maji, Grossfield, Kielkopf "SF3b1: a Dynamic Achilles Heel of the Spliceosome"

Movie S1. Morph of the human SF3b1 structure in the isolated SF3b particle (PDB ID 5IFE) to the B^{ACT} spliceosome (PDB ID 6FF4), colored by residues number in a gradient from blue at the N-terminus to red at the C-terminus.

Movie S2. The first major mode of SF3b1 dynamics, computed from an anisotropic network model of calculated using ANM 2.1 (http://anm.csb.pitt.edu, default settings). The HEAT domain of SF3b1 (residues 463-1274) from the SF3b particle structure (PDB ID 5IFE) was input as the starting coordinates.

Movie S3. The second major mode of SF3b1 dynamics, computed from the same anisotropic network model as Movie S2. The structure is viewed following a 90° rotation about the x-axis relative to Movie S2.

SUPPLEMENTARY METHODS

We calculated the distance between the centroids and the angle between the principal axes for the indicated α-helices from consecutive SF3b1 HEAT repeats, using Python scripts that implement the Lightweight Object-Oriented Structure (LOOS) library (Romo et al. 2014). Using the `centroid()` and `principalAxes()` functions from the `AtomicGroup` class, we calculated centroid and principal axis for each HEAT Repeat helix.

The centroid for a group of coordinates (here the Cα atoms of each residue in the indicated HEAT repeat α-helix) is given by:

$$(X_G, Y_G, Z_G) = \left(\frac{\sum_{i=1}^{N} X_i}{N}, \frac{\sum_{i=1}^{N} Y_i}{N}, \frac{\sum_{i=1}^{N} Z_i}{N}\right)$$

where (X_G, Y_G, Z_G) is the coordinate of the centroid and N is the number of Cα atoms used in the calculation.

The principal axes ($\mathbf{p_1}$, $\mathbf{p_2}$, $\mathbf{p_3}$) are obtained from the diagonalization of the moment of inertia tensor:

$$\Lambda = U^T I U, \text{ where } U = (\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3})$$

U is a rotation matrix whose columns are the eigenvectors that form the principal axes. Λ is the diagonal matrix of eigenvalues (sorted from largest to smallest) known as the principal moments of inertia and $I = \sum_{i=1}^{N} m_i \left[(\boldsymbol{r_i} \cdot \boldsymbol{r_i}) \sum_{a=1}^{3} \boldsymbol{e_a} \otimes \boldsymbol{e_a} - \boldsymbol{r_i} \otimes \boldsymbol{r_i}\right]$ is the tensor of inertia.

The angle between the principal axes of two consecutive helices is calculated by taking the dot product of their respective first principal axes:

$$\theta = \cos^{-1}(p_i \cdot p_i') \times \frac{180°}{\pi}$$

where p_i *and* p_i' are the principal axes of two consecutive HEAT repeat helices.

Representative script for intramolecular relationship between consecutive HEAT repeats (Used for Fig. 6 and Fig. S1)

```
#!/usr/bin/env python3

import sys
import loos
import sel_residues #imports a file of residue selections exemplified below
import math

def orientation_vector(group):
    """
    Returns orientation vector from the beginning of the group to the
    middle of the group
    """
    num_atoms = len(group)
    middle = num_atoms / 2
    v = s[int(middle)].coords() - s[0].coords()
    return v

if __name__ == '__main__':

    system1 = loos.createSystem("5gm6_chainG.pdb")
#uses a pdb file to create the input structure.
    dihelices = [] #initializing an array to store the coordinates for the
helix residues
    orientations = [] #initializing an array to store the orientation vector
for each helix
    fop = open('5gm6_calc_cter.txt', 'w')


    for selections in sel_5gm6_cter.selections:
# selecting the helices from selections_5ife_cter/selections_5gm6_cter
        s = loos.selectAtoms(system1, selections)
#storing the coordinates using the residue selections
        s = loos.selectAtoms(s, 'name=="CA"') #storing the coordinates for
only the C-alpha atoms
        dihelices.append(s) #appends the coordinate selections to the
initialized array

        v = make_orientation_vector(s)
        orientations.append(v)

    #d = len(dihelices)  #to print the length of the array
    #print(d)

    def calculate_centroid_distance(a1, a2):
# calculate the centroid distance between two consecutive C-terminal helices
        cen1 = a1.centroid()
        cen2 = a2.centroid()
        dif = cen2-cen1
        dist = dif.length() #to calculate the distance between the centroids
        return dist
```

```
    def get_angle(b1, b2): # This function is defined to calculate the angle
between the principal axes of two consecutive C-terminal helices
        vec1 = b1.principalAxes()
        ax1 = vec1[0]
        #uses the orientation of each helix and calculates the principal axis
vectors accordingly
        if (ax1 * orientations[i] < 0):
            ax1 *= -1
        vec2 = b2.principalAxes()
        ax2 = vec2[0]
        if (ax2 * orientations[i+1] < 0):
            ax2 *= -1
        ang = math.acos(ax1 * ax2) * 180/math.pi #to calculate the angle
between to principal axis
        return ang

    fop.write('# i\t distance\tangle\ttorsion\n')

    for i in range(0,20): # this for loop is to count each of the member of
the list for centroid calculations
        s1 = dihelices[i] #picking out each member of the helix 1 list
        s2 = dihelices[i+1] #picking out the next member of the helix 1 list
        distance = calculate_centroid_distance(s1, s2) #calculates the
distance between the centroids

        str1 = '%f\t' % (i+1) #writing helix indices into the output file
        fop.write(str1)

        str1 = '%f\t' % distance #writing distance between centroids into the
output file
        fop.write(str1)

        angle = get_angle(s1, s2) #writing angle between the principal into
the output file
        str1 = '%f\t' % angle
        fop.write(str1)

        fop.write('\n')

    fop.close()
```

Representative script for intermolecular distance between HEAT repeats (Used for Fig. 7A)

```
#!/usr/bin/env python3
import sys
import loos
import sel_residues #imports a file of residue selections exemplified below
import math

def calculate_centroid_distance(a1, a2):
    #this function is defined to calculate the distance between centroids
    cen1 = a1.centroid() #calculates centroid, answer is a coordinate
    cen2 = a2.centroid()
    dif = cen2-cen1 #calculate the difference between the coordinates
    dist = dif.length() #calculates the distance between the coords
    return dist

if __name__ == '__main__':
    system1 = loos.createSystem("pdb_5ife_chain_C.pdb")
#uses PDB1 to create an input structure
    system2 = loos.createSystem("6ff4_chainU_clean.pdb")
#uses PDB2 to create an input structure
    dihelices1 = []
#creates list of the coordinates of the C-alpha atoms from PDB1
    dihelices2 = []
#creates list of the coordinates of the C-alpha atoms from PDB1
    fop =  open('pdb_5ife_6ff4_centroid.txt', 'w')
#to write the results in an output file

    for selections in sel_Cter_5ife.selections: #using the cterminal side
helix residues of PDB1 and PDB2
        sys1 = loos.selectAtoms(system1, selections) #storing the coordinates
using the residue selections for PDB1
        sys1 = loos.selectAtoms(sys1, 'name=="CA"') #selects only the C-alpha
atoms from PDB1
        sys2 = loos.selectAtoms(system2, selections) #storing the coordinates
using the residue selections for PDB2
        sys2 = loos.selectAtoms(sys2, 'name=="CA"') #selects only the C-alpha
atoms from PDB2

        dihelices1.append(sys1) #separate lists of the coordinates of C-alpha
atoms of helix residues of PDB1
        dihelices2.append(sys2) #separate lists of the coordinates of C-alpha
atoms of helix residues of PDB2

    fop.write('# i\t distance\n')

# loop to count each of the member of the list for centroid calculations
    for i in range(0,20):
        s1 = dihelices1[i] #picking out each member of the helix 1 list
        s2 = dihelices2[i] #picking out each member of the helix 2 list
        distance = calculate_centroid_distance(s1, s2)

        str1 = '%f\t' % (i+1) #writing the results in an output file
```

```
        fop.write(str1)

        str1 = '%f\t' % distance #writing the results in an output file
        fop.write(str1)

        fop.write('\n') #print (distance)

    fop.close() #closing the output file
```

Representative script for human HEAT C-terminal α-helix selection (Used for Fig. 6 and Fig. S1)

```
selections = ['(resid >= 508 && resid <= 524)',
              '(resid >= 545 && resid <= 561)',
              '(resid >= 588 && resid <= 603)',
              '(resid >= 623 && resid <= 639)',
              '(resid >= 659 && resid <= 675)',
              '(resid >= 698 && resid <= 714)',
              '(resid >= 741 && resid <= 757)',
              '(resid >= 781 && resid <= 797)',
              '(resid >= 828 && resid <= 844)',
              '(resid >= 861 && resid <= 876)',
              '(resid >= 906 && resid <= 922)',
              '(resid >= 948 && resid <= 964)',
              '(resid >= 986 && resid <= 1002)',
              '(resid >= 1028 && resid <= 1045)',
              '(resid >= 1072 && resid <= 1088)',
              '(resid >= 1109 && resid <= 1125)',
              '(resid >= 1140 && resid <= 1160)',
              '(resid >= 1182 && resid <= 1198)',
              '(resid >= 1223 && resid <= 1239)',
              '(resid >= 1258 && resid <= 1277)']
```

Representative script for human HEAT dihelix selection (Used for Fig. S1)

```
selections = ['(resid >= 490 && resid <= 506) || (resid >= 508 && resid <= 524)',
              '(resid >= 525 && resid <= 541) || (resid >= 545 && resid <= 561)',
              '(resid >= 567 && resid <= 584) || (resid >= 588 && resid <= 603)',
              '(resid >= 605 && resid <= 621) || (resid >= 623 && resid <= 639)',
              '(resid >= 640 && resid <= 656) || (resid >= 659 && resid <= 675)',
              '(resid >= 680 && resid <= 696) || (resid >= 698 && resid <= 714)',
              '(resid >= 721 && resid <= 737) || (resid >= 741 && resid <= 757)',
              '(resid >= 760 && resid <= 776) || (resid >= 781 && resid <= 797)',
              '(resid >= 802 && resid <= 818) || (resid >= 828 && resid <= 844)',
              '(resid >= 843 && resid <= 860) || (resid >= 861 && resid <= 876)',
              '(resid >= 885 && resid <= 901) || (resid >= 906 && resid <= 922)',
              '(resid >= 925 && resid <= 941) || (resid >= 948 && resid <= 964)',
              '(resid >= 967 && resid <= 983) || (resid >= 986 && resid <= 1002)',
              '(resid >= 1010 && resid <= 1025) || (resid >= 1028 && resid <= 1045)',
              '(resid >= 1052 && resid <= 1068) || (resid >= 1072 && resid <= 1088)',
              '(resid >= 1092 && resid <= 1108) || (resid >= 1109 && resid <= 1125)',
              '(resid >= 1126 && resid <= 1138) || (resid >= 1140 && resid <= 1160)',
              '(resid >= 1161 && resid <= 1177) || (resid >= 1182 && resid <= 1198)',
              '(resid >= 1204 && resid <= 1220) || (resid >= 1223 && resid <= 1239)',
              '(resid >= 1242 && resid <= 1255) || (resid >= 1258 && resid <= 1277)']
```

Representative script for yeast HEAT dihelix selection (Used for Fig. S1)

```
selections = ['(resid >= 156 && resid <= 173) || (resid >= 179 && resid <= 193)',
              '(resid >= 196 && resid <= 210) || (resid >= 215 && resid <= 231)',
              '(resid >= 237 && resid <= 252) || (resid >= 256 && resid <= 272)',
              '(resid >= 275 && resid <= 287) || (resid >= 291 && resid <= 308)',
              '(resid >= 310 && resid <= 322) || (resid >= 327 && resid <= 344)',
              '(resid >= 351 && resid <= 364) || (resid >= 368 && resid <= 385)',
              '(resid >= 390 && resid <= 405) || (resid >= 411 && resid <= 426)',
              '(resid >= 428 && resid <= 446) || (resid >= 451 && resid <= 465)',
              '(resid >= 472 && resid <= 488) || (resid >= 496 && resid <= 512)',
              '(resid >= 516 && resid <= 525) || (resid >= 531 && resid <= 548)',
              '(resid >= 555 && resid <= 571) || (resid >= 577 && resid <= 590)',
              '(resid >= 592 && resid <= 609) || (resid >= 614 && resid <= 633)',
              '(resid >= 637 && resid <= 651) || (resid >= 656 && resid <= 672)',
              '(resid >= 682 && resid <= 694) || (resid >= 699 && resid <= 714)',
              '(resid >= 721 && resid <= 736) || (resid >= 740 && resid <= 756)',
              '(resid >= 759 && resid <= 770) || (resid >= 775 && resid <= 791)',
              '(resid >= 792 && resid <= 805) || (resid >= 810 && resid <= 826)',
              '(resid >= 838 && resid <= 847) || (resid >= 851 && resid <= 868)',
              '(resid >= 875 && resid <= 888) || (resid >= 893 && resid <= 910)',
              '(resid >= 912 && resid <= 923) || (resid >= 930 && resid <= 944)']
```

Supplemental References

Romo TD, Leioatts N, Grossfield A. 2014. Lightweight object oriented structure analysis: tools for building tools to analyze molecular dynamics simulations. *J Comput Chem* **35**: 2305-2318.