Wavelet Algorithms

The wavelet calculations were completed using Fourier transforms. The exact algorithm is given below in Mathematica code. The `FourierConvolve[]` function performs a circular convolution of two signals using the convolution theorem. The `ScaleWavelet[f, n, s]` function takes a wavelet function `f`, an ideal length `n`, and a stretch `s`, and returns the wavelet as a signal of length `n` stretched by `s`. The function `CWT[signal, f, s0, ds, S]` takes the signal, wavelet function `f`, and variables `s0`, `ds`, and `S`, and returns the continuous wavelet transform of the signal with wavelets scaled according to the formula in Equation S1.

$$s_k = s_0 2^{kds}, \quad k = 0,1,...S-1 \tag{S1}$$

For the wavelets used in this paper, the values s_0=100, ds=1/8, and S=60 were used, and all values were measured in picoseconds.

```
FourierConvolve[a_List, b_List] :=
  Times[Sqrt[Length[a]],
   InverseFourier[
    Times[
     Fourier[a, FourierParameters -> {0, -1}],
     Fourier[b, FourierParameters -> {0, -1}]],
    FourierParameters -> {0, -1}]];
ScaleWavelet[wltfunc_, n_Integer, scale_] :=
  Times[
   1/Sqrt[scale],
   Table[N[wltfunc[k/scale]], {k, -Floor[n/2], Floor[(n - 1)/2]}]];
CWT[signal_List, wltfunc_, s0_, ds_, S_Integer] :=
  Module[{n = Length[signal], scales = Table[s0*2^(k*ds), {k, 0, S - 1}]},
   Map[
    FourierConvolve[
     signal,
     RotateLeft[ScaleWavelet[wltfunc, n, #], Floor[n/2]]] &,
    scales]];
Morlet[t_] := Pi^(-1/4)*Exp[-t^2/2]*Exp[2*Pi*I*t];
Paul[t_] := 8*Sqrt[2/(35*Pi)]*(1 - I*t)^(-5);
```

For example, if a the variable `x` contained the *x*-coordinate of an atom over time, then the continuous wavelet transform using the Paul wavelet as described in this paper could be obtained with the command `wlt=CWT[x, Paul, 105, 1/8, 60]`. The value of `wlt[[1]]` is the wavelet coefficient vector with a scale of 105 ps while the the value of `wlt[[41]]` is the wavelet coefficient vector with a scale of $105*2^{40/8}$ = 3.36 ns.

Once the wavelet coordinates are collected, significance testing is performed using the algorithm below. The `WaveletSignificance[wlt, scales, x, pval, correction]` function returns the wavelength of the frequency most strongly matched by the model described in this paper. The `wlt` parameter is the wavelet coordinates as generated by the `CWT[]` function while the `scales` variable should be a list of the scales calculated in `CWT[]`. The `x` parameter is the same as that passed to `CWT[]` while the `pval` is the minimum *p*-value acceptable for the significance test. Finally the `correction` parameter is the scale-to-wavelength factor described in the paper (1.01 for Morlet, 1.389 for Paul).

```
WaveletSignificance[wlt_List, scales_List, x_List, pval_, correction_] :=
  Module[
   {n = Length[wlt[[1]]],
    chival = (InverseCDF[ChiSquareDistribution[2], t] /. t -> (1 - pval))/2,
    res = Table[{0, 0}, {Length[wlt[[1]]]}],
    var = Variance[x],
    tmp, min},
   Scan[
    Function[{s},
     min = (0.00647*(correction*scales[[s]])^1.41344 + 19.7527)*chival;
     res = Table[
       (tmp = Abs[wlt[[s, k]]]^2/var;
        If[res[[k, 2]] < tmp && min < tmp,
         {correction*scales[[s]], tmp},
         res[[k]]]),
       {k, 1, n}]],
    Range[1, Length[scales]]];
   First[Transpose[res]]];
```

Note that in the case of a wavelet with no imaginary part, the fourth line would be as follows.

```
chival = (InverseCDF[ChiSquareDistribution[1], t] /. t -> (1 - pval)),
```

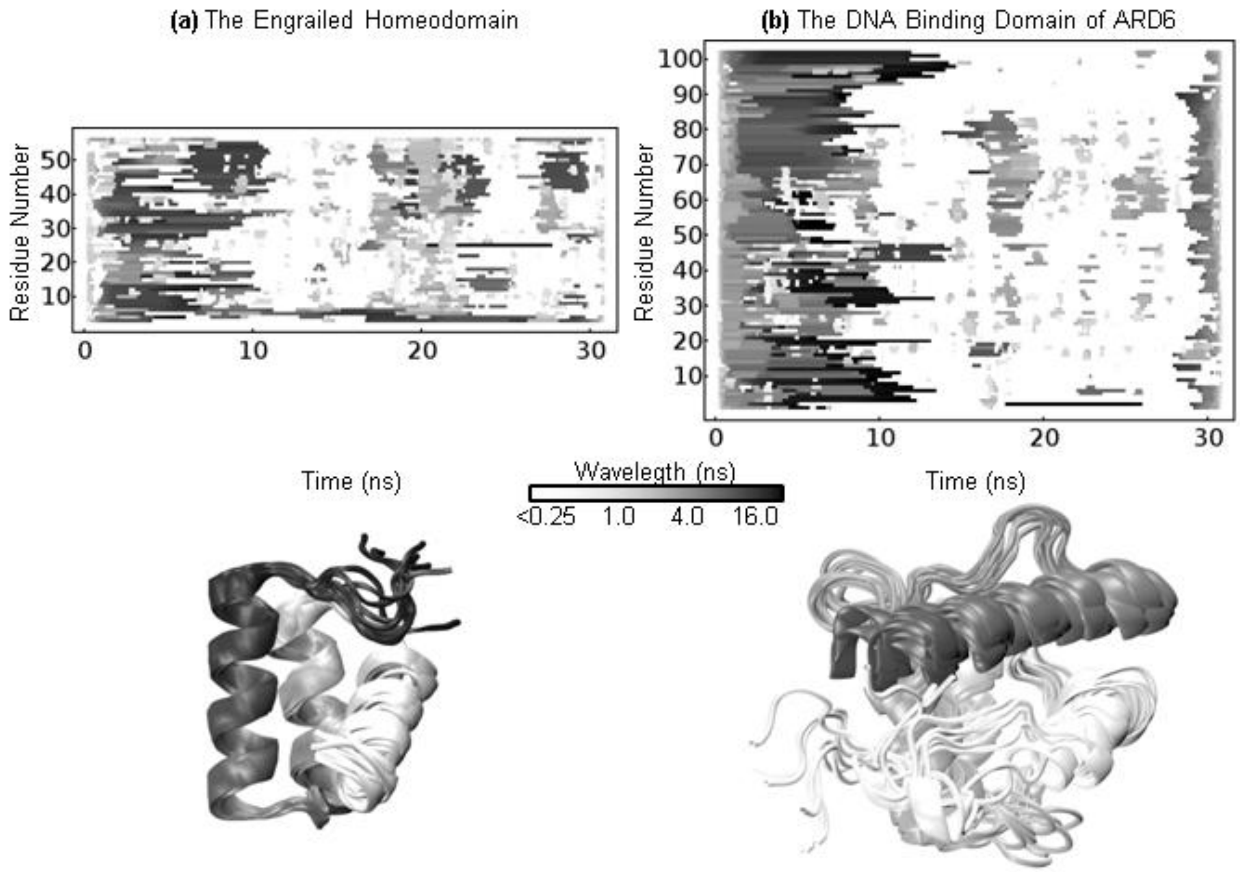

Figure S1. Comparison of the engrailed homeodomain **(a)** to the DNA-binding domain of ADR6 **(b)**.