100 Time Series Data Mining Questions (with answers!)

Work in Progress

Keogh's Lab (with friends)

Dear Reader: This document offers examples of time series questions/queries, expressed in intuitive natural language, that can be answered using simple tools, like the Matrix Profile, and related tools such as MASS.

We show the step-by-step solutions. In most cases, the solutions require just a handful of lines of code.

As you may have noticed, we are not at 100 yet! This is a long term work-in-progress. We welcome suggestions and "donations" of questions.

The code and data is here: *www.cs.ucr.edu/~eamonn/HundredQuestions.zip*

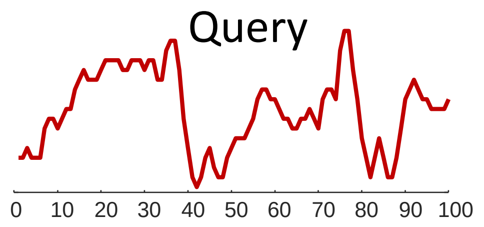
Corrections and suggestions to eamonn@cs.ucr.edu

In a handful of cases, we report *timing* results. These examples were made on an old machine, were optimized for simplicity, not speed. In any case the timing will become dated with Moore's Law. In addition, we are constantly optimizing our code. We only mean to produce *relative* numbers for your instruction. Please do **not** report the absolute numbers, run the experiments yourself, with the most optimized code available.

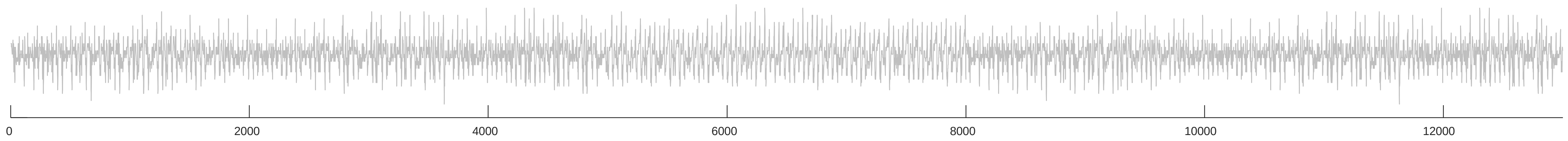
See also *www.cs.ucr.edu/~eamonn/MatrixProfile.html* and *www.cs.unm.edu/~mueen/FastestSimilaritySearch.html*

1. *Have we ever seen a pattern that looks just like this?*
2. *Are there any repeated patterns in my data?*
3. *What are the three most unusual days in this three month long dataset?*
4. *Is there any pattern that is common to these two time series?*
5. *How do these two time series differ in terms of alignment?*
6. *Find the most conserved pattern that happens at least once every two days in this two week long dataset.*
7. *If you had to summarize this long time series with just two shorter examples, what would they be?*
8. *Are there any patterns that appear as time reversed versions of themselves in my data?*
9. *When does the regime change in this time series?*
10. *How can I compare these time series of different lengths?*
11. *Are there any patterns that repeat in my data, but at two distinct lengths?*
12. *Have we ever seen a multidimensional pattern that looks just like this?*
13. *How do I quickly search this long dataset for this pattern, if an approximate search is acceptable?*
14. *How can I optimize similarity search in a long time series?*
15. *What is most likely to happen next?*
16. *What is the right length for motifs in this dataset?*
17. *I need to find motifs faster! Part I*
18. *I need to find motifs faster! Part II*
19. *Have we ever seen a pattern that looks just like this, but possibly at a different length?*
20. *How can I know which of these two classification approaches is best for time series?*
21. *Are there any evolving patterns in this dataset (time series chains)*
22. *(pending)*

Have we ever seen a pattern that looks just like this?

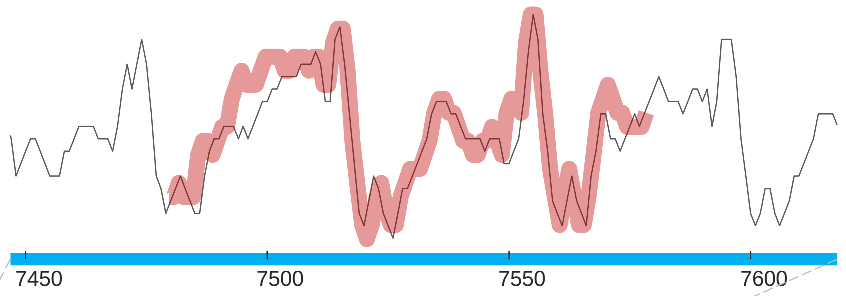


The dataset comes from an accelerometer inside a Sony AIBO robot dog. The query comes from a period when the dog was walking on carpet, the test data we will search comes from a time the robot walked on cement (for 5000 data points), then carpet (for 3000 data points), then back onto cement.



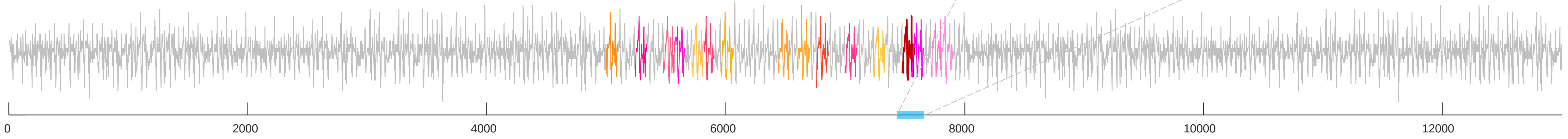
This task is trivial with Mueen's MASS code…

```
>> load robot_dog.txt , load carpet_query.txt % load the data
>> dist = findNN(robot_dog ,carpet_query );   % compute a distance profile
>> [val loc] = min(dist);                     % find location of match
>> disp(['The best matching subsequence starts at ',num2str(loc)])
The best matching subsequence starts at 7479
```



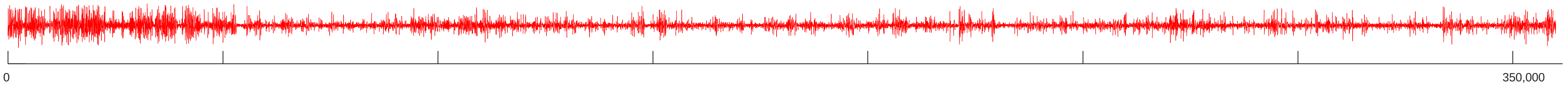
The best match, shown in context

Below we plot the *16* best matches. Note that they all occur during the carpet walking period. This entire process takes about 1/1000th of a second.



Are there any repeated patterns in my data?

The dataset is an hour of EOG (eye movement) data of a sleeping patient, sampled at 100 Hz. It looks very noisy, it is not obvious that there is any repeated structure…



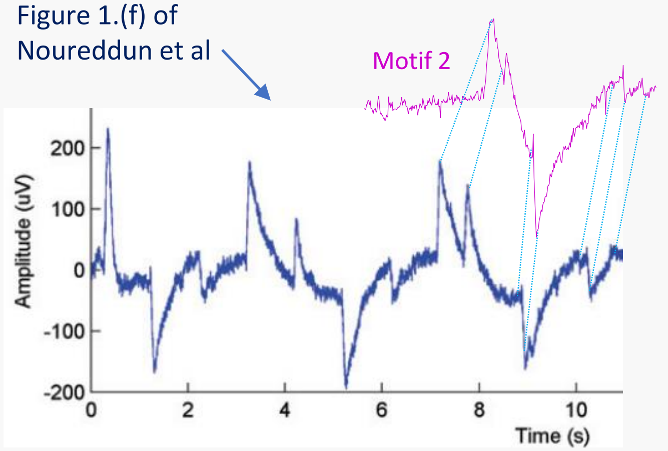
Let us run the Matrix Profile, looking for four-second long motifs…

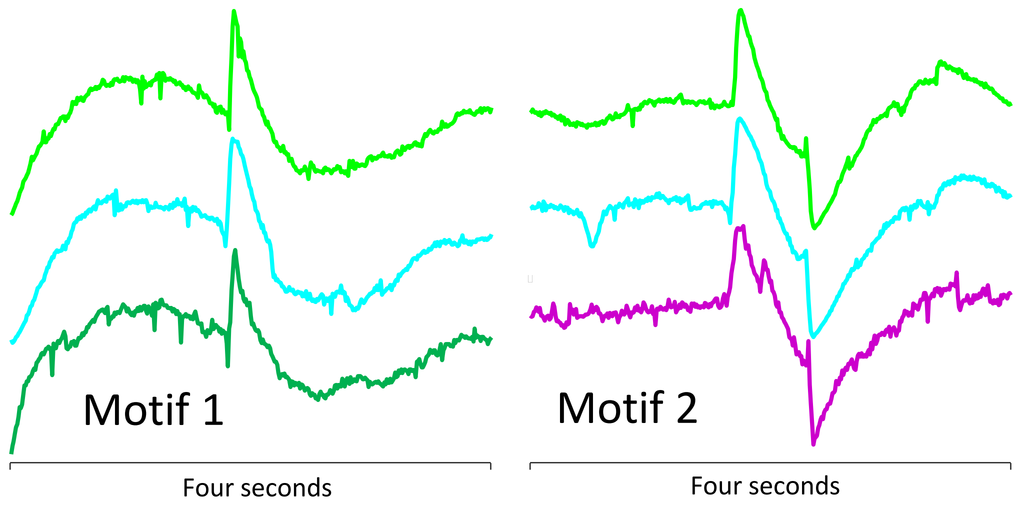
```
>> load eog_sample.mat
>> [matrixProfile profileIndex, motifIndex, discordIndex] = interactiveMatrixProfileVer3_website(eog_sample, 400);
```

The code takes a while to fully converge, but in just a few seconds, we see some stunningly well conserved motifs…

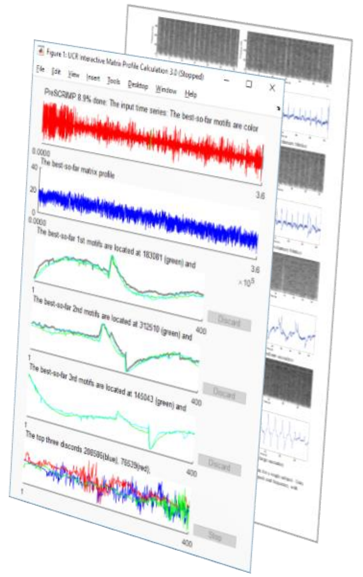
Having found the motifs, we can ask, what are they? A quick glance at a paper by Noureddin et. al. locates a very similar pattern (with some time warping) called `eye-blink-artifact`.

Figure 1.(f) of Noureddun et al

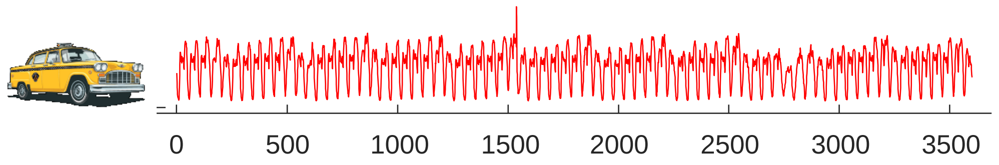




Note that there may be more examples of each motif. We should take one of the above, and use MASS to find the top 100 neighbors… See *Have we ever seen a pattern that looks just like this?*. We can also adjust the range parameter *r* inside the motif extraction code.



What are the three most unusual days in this three month long dataset?



The datasets is Taxi demand, in New York City, in the last three months of the year.

We choose 100 datapoints, which is about *two* days long (the exact values do not matter much here).

```
>> load taxi_3_months.txt
>> [matrixProfile, profileIndex, motifIndex, discordIndex] = interactiveMatrixProfileVer3_website(taxi_3_months ,100);
```

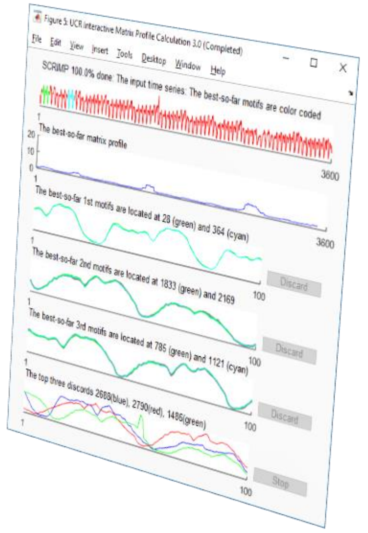


Figure 5: UCR interactive Matrix Profile Calculation 3.0 (Completed)

SCRIMP 100.0% done: The input time series: The best-so-far motifs are color coded

The best-so-far matrix profile

The best-so-far 1st motifs are located at 28 (green) and 364 (cyan)

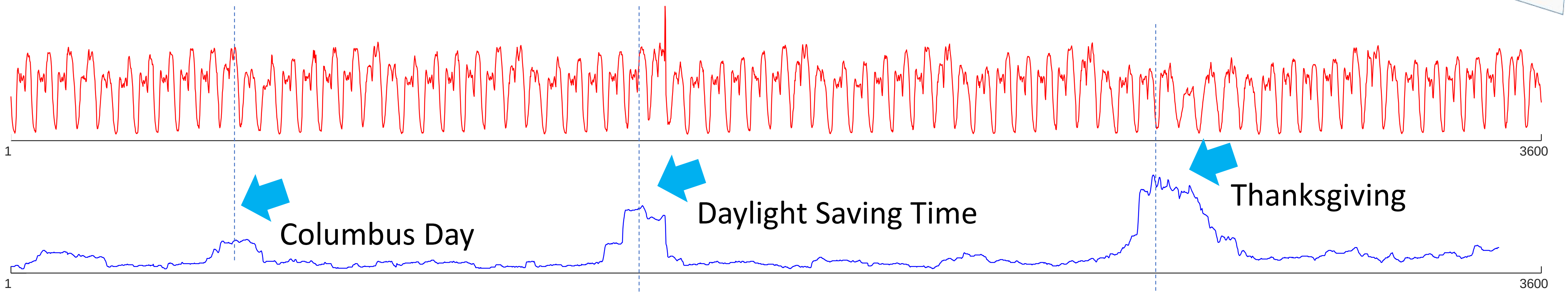
The best-so-far 2nd motifs are located at 1833 (green) and 2169

The best-so-far 3rd motifs are located at 785 (green) and 1121 (cyan)

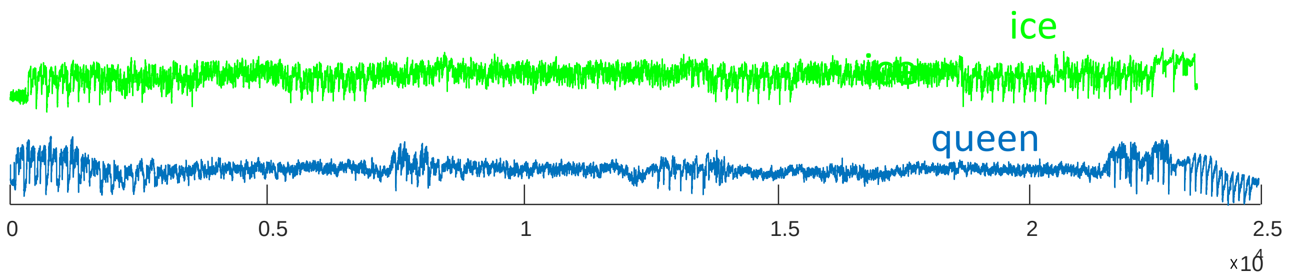
The top three discords 2688(blue), 2790(red), 1486(green)

The code pops up the matrix profile tool, and one second later, we are done! The three most unusual days correspond to the three highest values of the matrix profile (i.e. the *discords*), but what are they?

- The highest value corresponds to Thanksgiving
- We find a secondary peak around Nov 6th, what could it be? Daylight Saving Time! The clock going backwards one hour, gives an *apparent* doubling of taxi load.
- We find a tertiary peak around Oct 13th, what could it be? Columbus Day! Columbus Day is largely ignored in much of America, but still a big deal in NY, with its large Italian American community.



Is there any pattern that is common to these two time series?

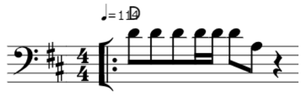


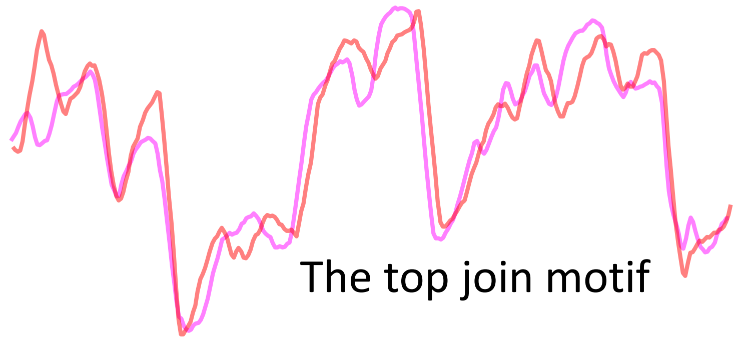
Lets assume that the common pattern is 3 seconds, or 300 datapoints long.
Let us concatenate the two time series, and smooth them (just for visualization purposes, we don't really need to)
Now let us find the top motif, but insist that one motif comes before 24289, and one after…

```
    >> load('Queen_vs_Ice.mat')
    >> whos
      Name                         Size                 Bytes  Class     Attributes
      mfcc_queen                   1x24289             194312  double
      mfcc_vanilla_ice             1x23095             184760  double
    >> interactiveMatrixProfileAB(smooth([mfcc_queen , mfcc_vanilla_ice]), 300, 24289); % This will spawn this plot ->
```

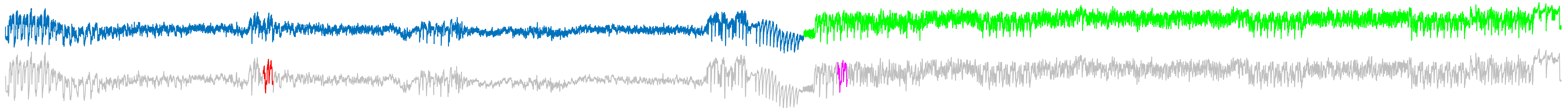
Figure 5: UCR Interactive Matrix Pr...
File Edit View Insert Tools Desktop Window Help
We are 52.8% done: The input time series: The
The best-so-far matrix profile
The best-so-far 1st motifs are
The best-so-far 2nd motifs are
The best-so-far 3rd motifs are
The top three discords
Discard
Discard
Discard
Stop

The top join motif shows a highly conserved pattern.

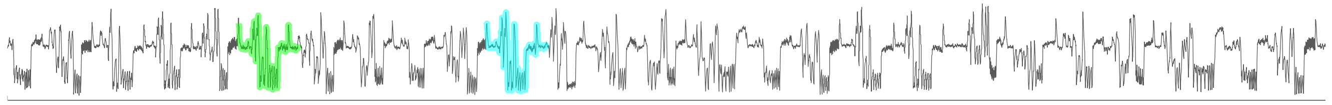
It is the famous bass line  from *Under Pressure* by Queen which was plagiarized by Vanilla Ice.



The concept for this example comes from Dr. Diego Furtado Silva.



How do these two time series differ in terms of alignment?

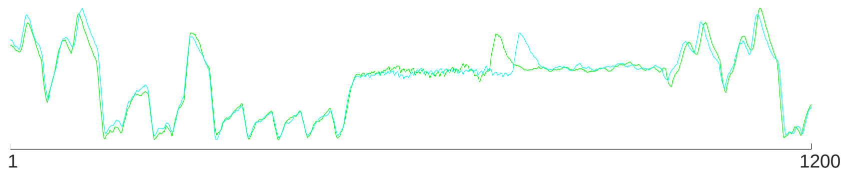


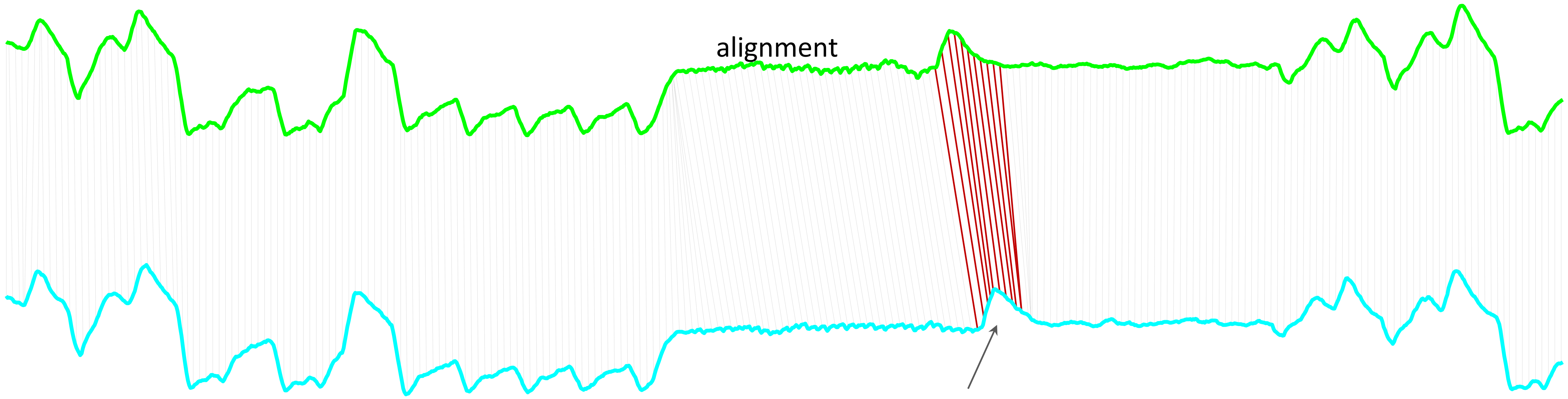
The data are two motifs discovered in the song of a bird[1], which we converted to MFCC. Let us load the data, and look at the DTW alignment.

```
>> load green.txt
>> load cyan.txt
>> DTW(green', cyan' ,1 ); % the '1' is just to force the plot
```

The DTW alignment clearly indicates where the differences lie, in the variability of the timing of a single note, about 2/3rds of the through the snippet. This example is trivial to see, but in more complex processes, this visual analysis can be very fruitful.

See *Multifractal analysis reveals music-like dynamic structure in songbird rhythms*, by Tina Roeske et al.

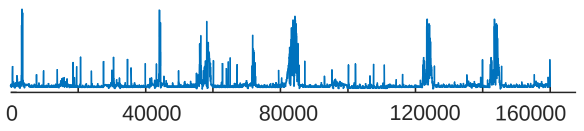




The second occurrence of this note happens much later, given how well the rest of the song snippet is preserved

[1]https://www.xeno-canto.org/415294

Find the most conserved pattern that happens at least once every two days in this dataset



The question is a little underspecified, as the length for the conserved patterns was not given. Let us try two hours, which is about 800 data points.
The full 20,000 datapoints represents about 14 days of electrical demand data for a house in the U.K. Thus we first need to divide it into approximate 2 day chunks.

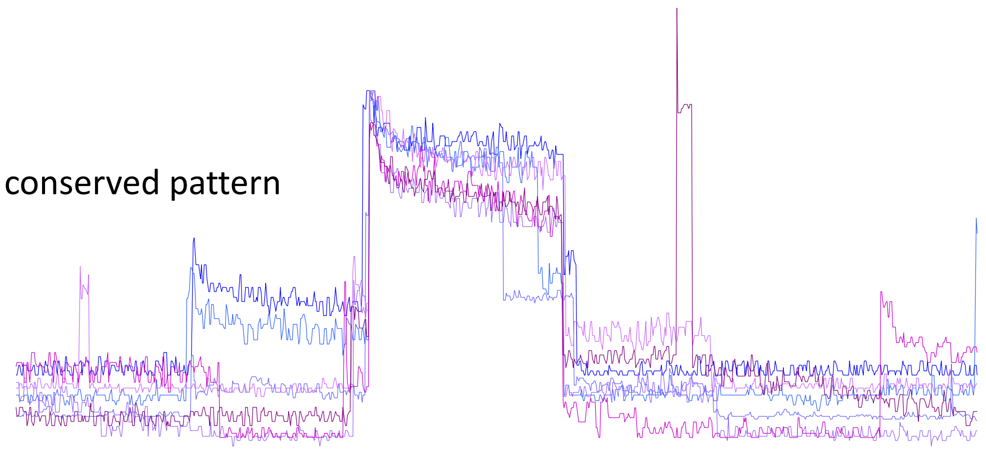
```
>> load TwoWeekElectrical
>> seven_two_day_chunks = divide_data(T);
```

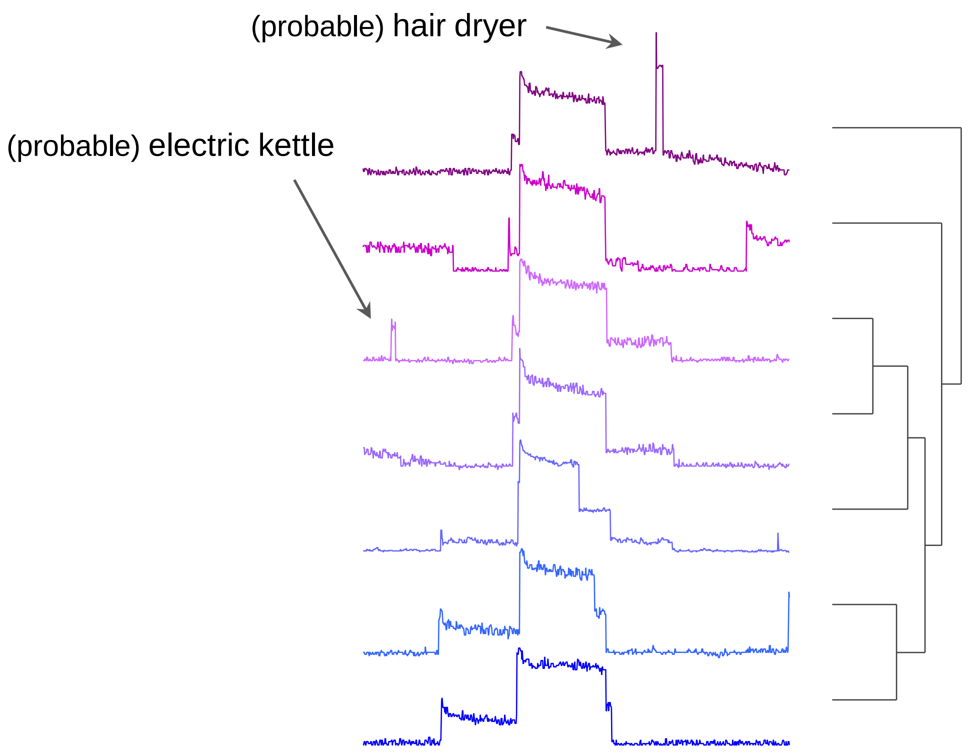
Now we just need to call the consensus motif code.

```
>> consensus_motifs = consensusMotifs(seven_two_day_chunks,800); % 800 is the length of subsequence
```

The code returns the seven time series below. Note that the basic pattern is highly conserved, given how noisy the data is.
The similarity between the items can be better seen if we cluster the time series with a single linkage dendrogram.

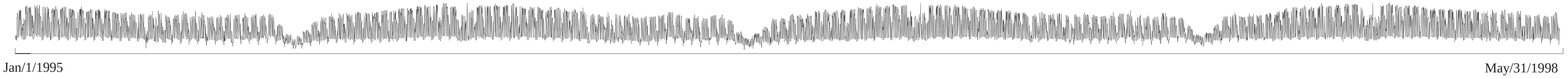
The most conserved pattern





If you had to summarize this long time series with just two shorter examples, what would they be?

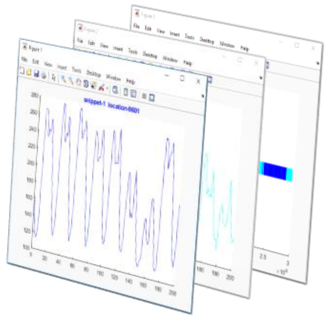
The dataset is 3 years of Italian power demand data which represents the hourly electrical power demand of a small Italian city for 3 years beginning on Jan 1st 1995.



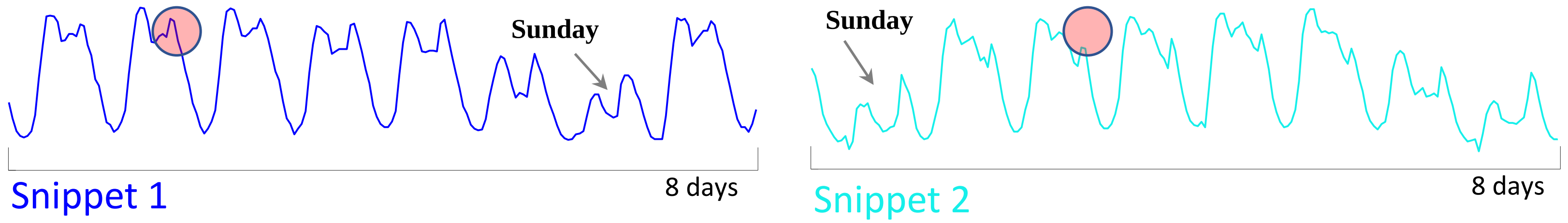
We just need to call Time Series Snippets algorithm…

```
>> load('ItalianPowerDemand.mat')
>> [fraction,snippet,snippetidx]= snippetfinder(data(:,4),2,200,30);
```

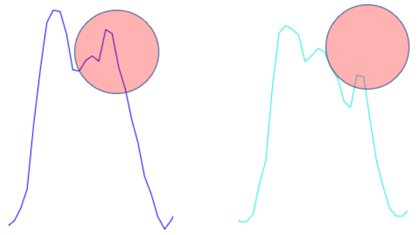
It will pop open three windows, which are snippet 1, snippet 2 and the regime bar.



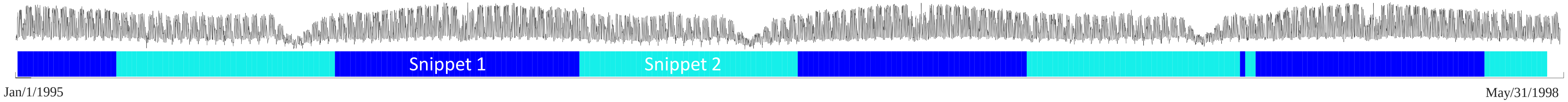
We searched for the top-2 snippets of length 200. This was our quick "eyeballing" guess as to the length of a week, but it is actually about 8.3 days. Note that the snippets are not align to start at the same day of the week (this is a trivial constraint to add if desired).



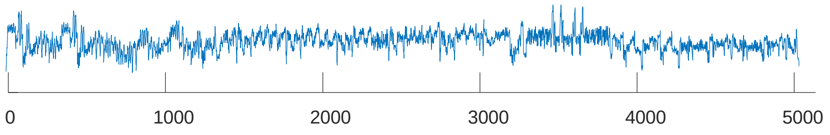
What makes the snippets different? (tentative answer) In the winter, people go home after work (and turn on heaters/appliances). In the summer, people do more leisure activities after work and don't return home until it is cooler.



We obtain the "regime bar," which tells us which snippet "explains" which region of data. As it happens, Snippets seem to represent *summer* and *winter* regimes respectively.



Are there any patterns that appear as time reversed versions of themselves in my data?



Lets us load the data, and concatenate it to itself, after flipping left to right.
We can then search for a join motif, that spans 5046, the length of the original time series.
If we find a good join motif, it means that the conserved pattern is time reversed!

```
>> load('mfcc.mat')
>> length(mfcc1(1,:))
ans = 5046
>> interactiveMatrixProfileAB(([mfcc1(1,:)'; flipud(mfcc1(1,:)')]), 150, 5046); % This will spawn this plot ->
```

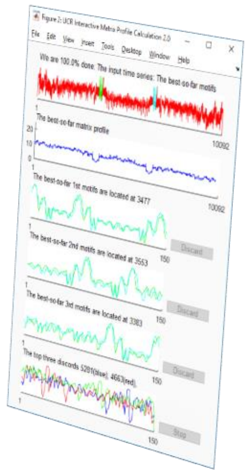
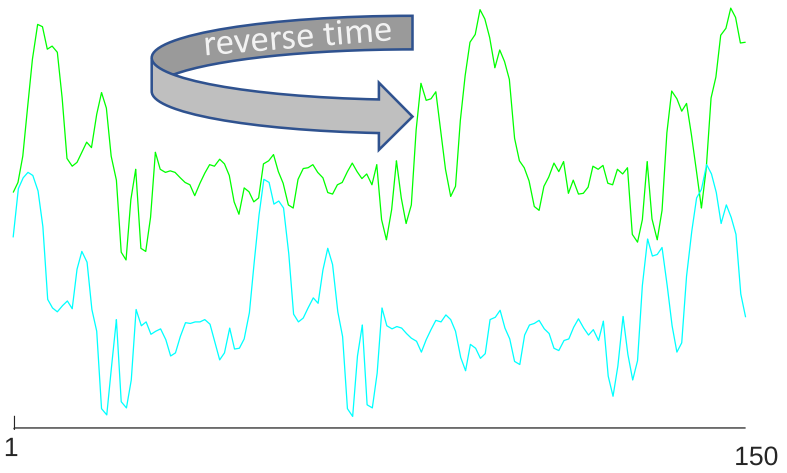


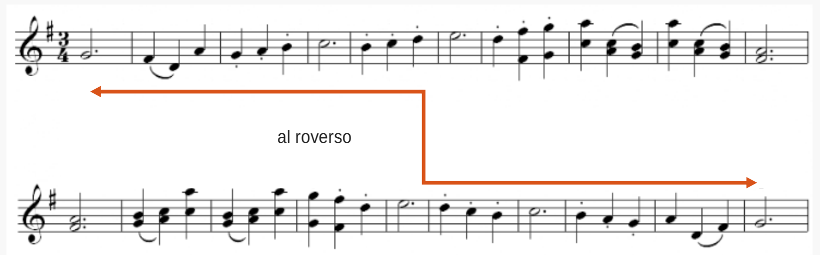
Figure 1: UCR Interactive Matrix Profile Calculation 2.0
File Edit View Insert Tools Desktop Window Help
We are 100.0% done. The input time series: The best-so-far motifs
The best-so-far matrix profile
The best-so-far 1st motifs are located at 3477
The best-so-far 2nd motifs are located at 3563
The best-so-far 3rd motifs are located at 3383
The top three discords
Discard

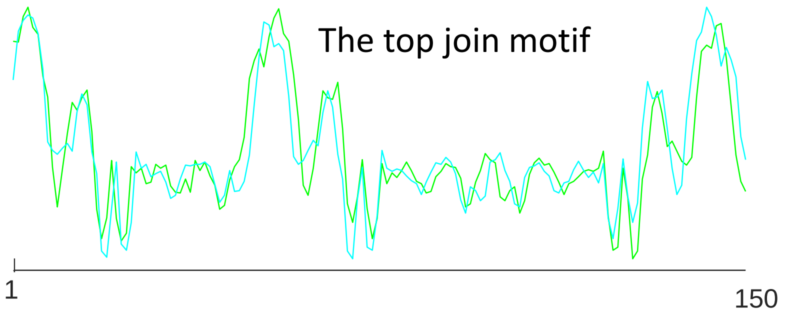
The top join motif shows a highly conserved pattern.

Why would a pattern occur time reversed?

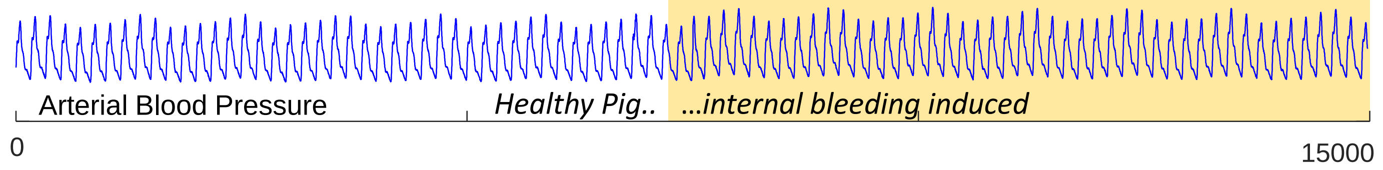


"The most extraordinary of all canonic movements from this time is of course from Symphony No. 47. Here Haydn writes out only one reprise of a two-reprise form, and the performer must play the music 'backward' the second time around".
The data is the 1st MFCC of this piece of music.





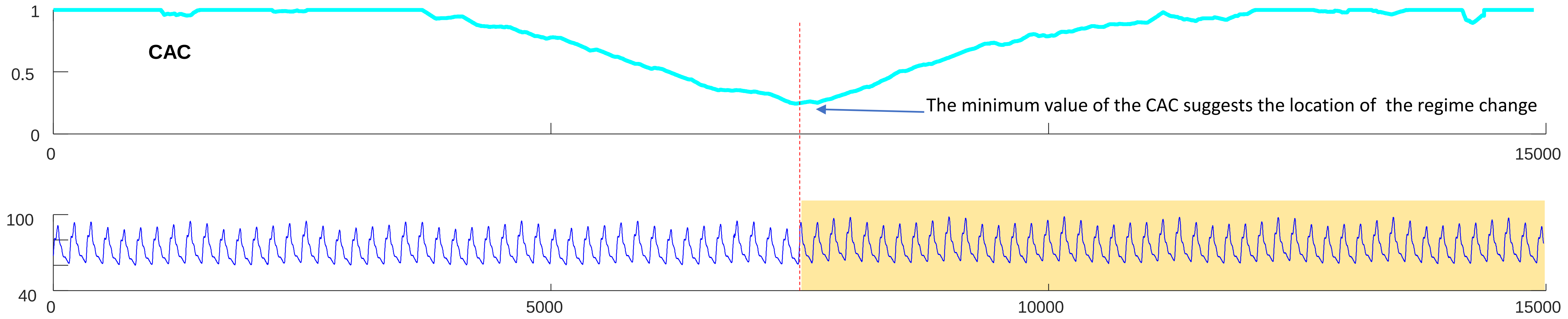
When does the regime change in this time series?



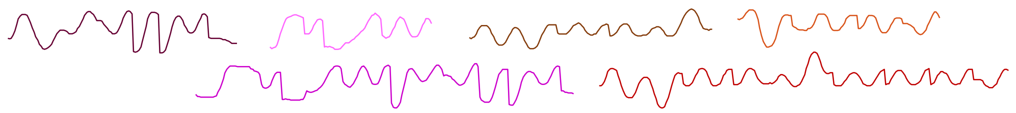
In this dataset, at time stamp 7,500, bleeding was induced in an otherwise healthy pig. This changes the pig's APB measurement, but only *very* slightly. Could we find the location of the change, if we were not told it? Moreover, can we do this with no domain knowledge? In other words, can we detect regime changes in time series?

```
>> TS = load('PigInternalBleedingDatasetArtPressureFluidFilled_100_7501.txt');
>> CAC = RunSegmentation(TS, SL);  %SL is the length of subsequence
>> plot(CAC,'c')
>> [~, loc] = min(CAC) %value of loc is 7460 which is the approximation of exact value 7500
```

Here, we choose SL to be 100, approximately the length of one period of arterial pressure (or the period of whatever repeated patterns you have in your data), however, up to half or twice that value would work just as well. The output curve, the CAC, minimizes at just the right place.
How does it do it? In brief, if we examine the pointers in the Matrix Profile Index, we will find that very few will cross *over* the location of a regime change (most healthy beats have a nearest neighbor that is another healthy beat, most "bleeding" beats have a nearest neighbor that is another "bleeding" beat), it is this lack of pointers that cross over the regime change that is what the CAC is measuring.



How can I compare these time series of different lengths?

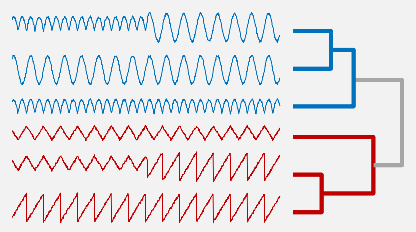


If you have data that are different lengths, you could make them the same length (using truncation or interpolation) or use DTW. However, for some datasets, that would be a very bad idea. To see why, consider *text* instead of time series for a moment.

For example, to find the similarity between (Lisa, Lisabeth), truncation of the second half of Lisabeth works well. However, to find the similarity between (Beth, Lisabeth), truncation of the second half of Lisabeth is clear wrong.

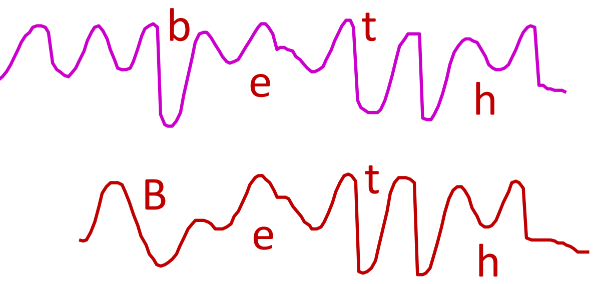
One trick to solve this issue is the Mpdist, a distance measure that automatically solves the above dilemma, by only comparing the most similar parts of the sequence. Below we demonstrate it on the Y-axis of the time series recording the location of the tip of a pen as it writes six girls names.

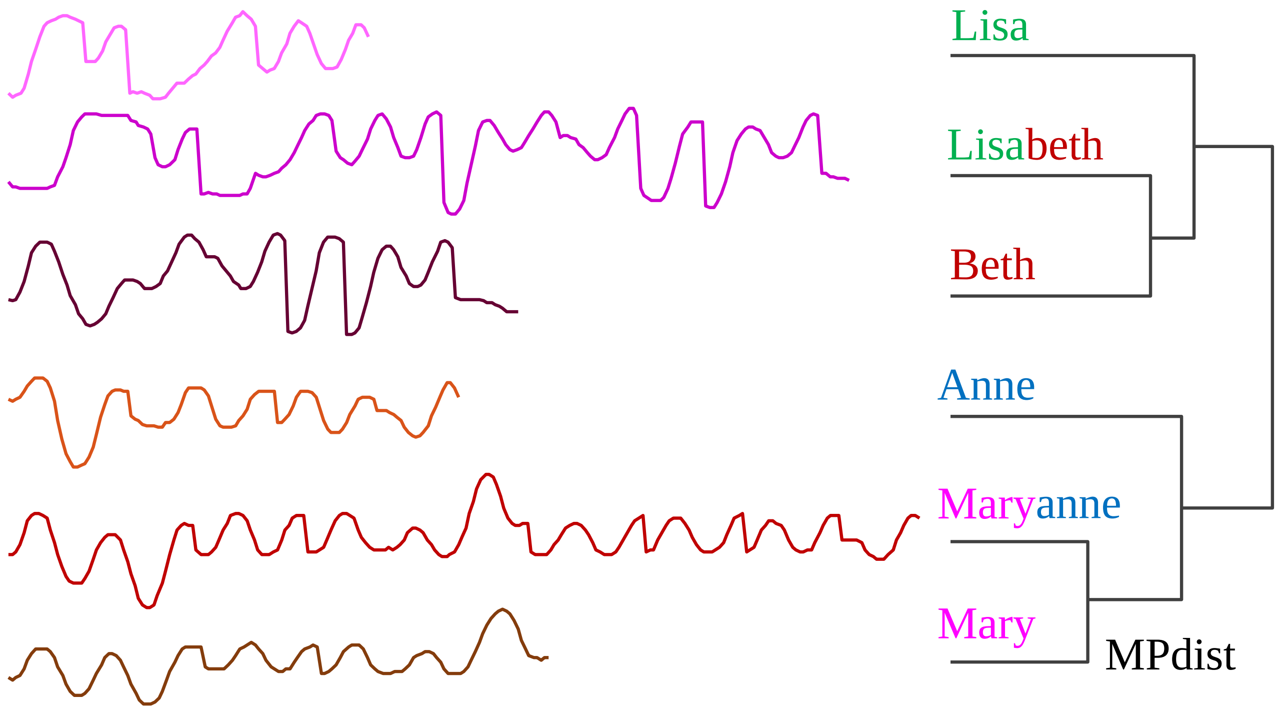
```
>> load('TSs')
>> MPdist_Clustering(TSs)
```



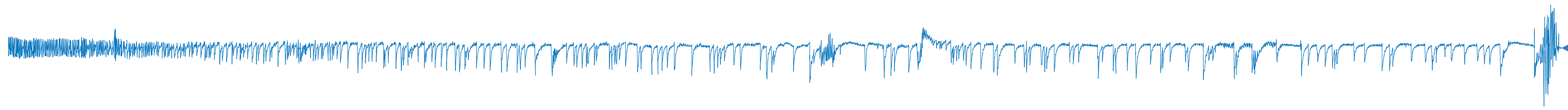
(We also have done this experiment on synthetic data to make this result even clearer)

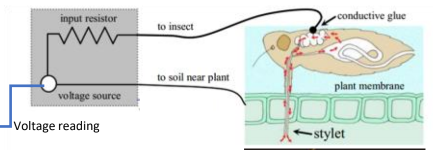
Note the that names in our example *piecewise* match. However, there are differences, for example due to the capitalization of ‘b’ in Beth vs.Lisabeth





Are there any patterns that repeat in my data, but at two distinct lengths?





See also "*Is there any pattern that is common to these two time series?*"

We can solve this with a quick and dirty trick. The code `interactiveMatrixProfileAB(T,m,crossover)` searches time series `T` for a motif of length `m`, such that one of the motif pair occurs before `crossover` and one occurs after `crossover`.

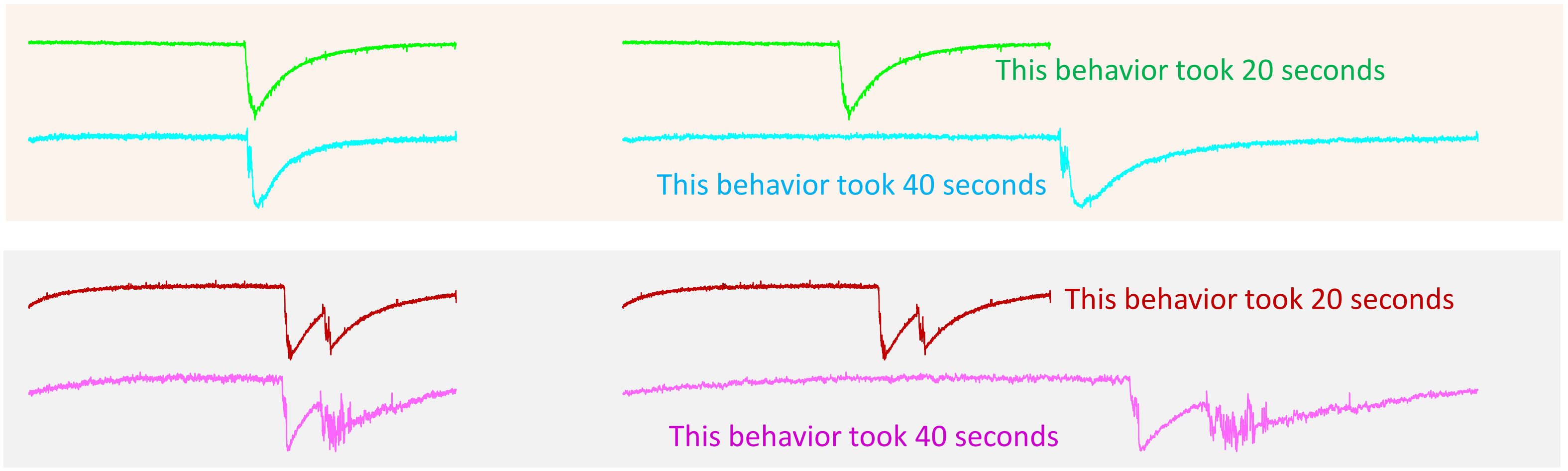
We can take a time series and append it to a *rescaled* copy itself, setting the to the length of the original time series. Now when we find motifs, we are finding one at the original scale, *and* one at the rescaled size.

In this case, I want to know if any of my insect behaviors happens at length 5,000 and at 10,000, so I type…

```
>> load insectvolts.mat   % load some insect epg data
>> interactiveMatrixProfileAB([insectvolts ; insectvolts(1:2:end)], 5000, length(insectvolts));  % search the appended data
```

No need to let it converge, after a few seconds we have our answer…

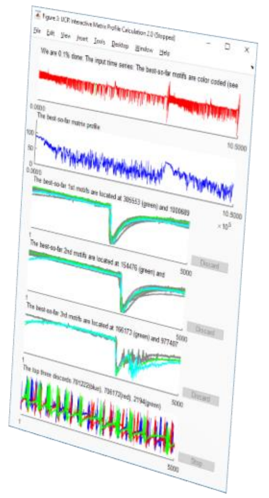
Two motifs in the rescaled space   Two motifs in the original, true space



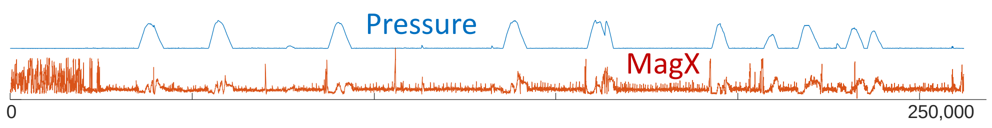
Note you *can* do this for non integer rescaling. Matlab will warn you, but it is defined and allowed.

Note that the bottom motif is discovered in spite of having a lot of noise in one of the occurrences.

Note that the dimensionality of the motifs is 5,000! This would have been unthinkable before the Matrix Profile.



Have we ever seen a multidimensional pattern that looks just like this?



I have 262,144 data points that record a penguin's orientation (MagX) and the water/air pressure as he hunts for fish.

Question: Does he ever change his bearing leftwards as he reaches the apex of his dive?

This is easy to describe as a multidimensional search. The apex of a dive is just an approximately parabolic shape. I can create this with `query_pressure = zscore([[-500:500].^2]*-1)';` it looks like this

I can create *bearing leftwards* with a straight rising line, like this `query_MagX = zscore([[-500:500]])';` It looks like this

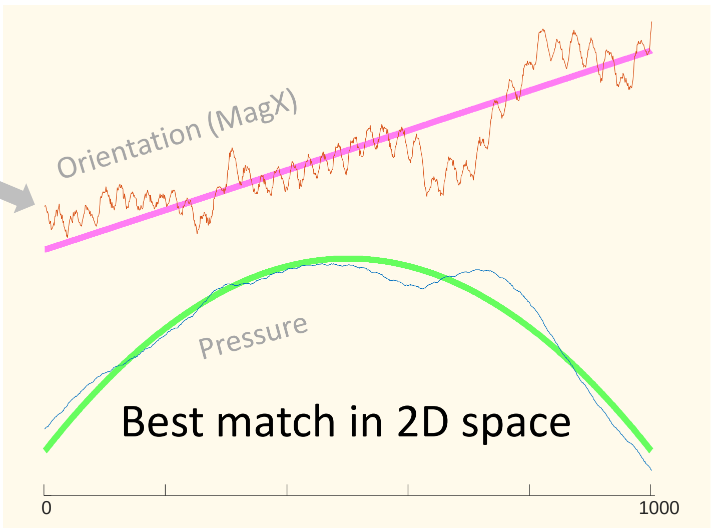
We have seen elsewhere in this document how to search for a 1D pattern. For this 2D case, all we have to do is add the two distance profiles together, before we find the minimum value.
Note that the best match location in 2D is different to either of the 1D queries.

We can do this for 3D or 4D…
However, there are some caveats. In brief, it almost never makes sense to do multidimensional time series search in more than 3 or 4D. See Matrix Profile VI: M. Yeh ICDM 2017 and "Weighting" B. Hu, ICDM 2013. In addition, in some cases we may want to weight the dimensions differently, even though they are both z-normalized Euclidean Distance.

What are the periodic bumps? They are wingstokes as the bird "flies" underwater

```
load penguintest.mat
figure;, hold on;

  query_pressure = zscore([[-500:500].^2]*-1)';
  dist_p = MASS_V2(penguintest(:,1),query_pressure);
  query_MagX = zscore([[-500:500]])';
  dist_m = MASS_V2(penguintest(:,2),query_MagX);
  [val,loc] = min([dist_m + dist_p]);          % find best match location in 2D
  plot(zscore(penguintest(loc:loc+length(query_MagX),2)),'color',[0.85 0.32 0.09])
  plot(zscore(query_MagX),'m')
  plot(zscore(penguintest(loc:loc+length(query_pressure),1)),’b’)
  plot(zscore(query_pressure),'g')
  title(['Best matching sequence, pressure/MagX, is at ', num2str(loc)])
```



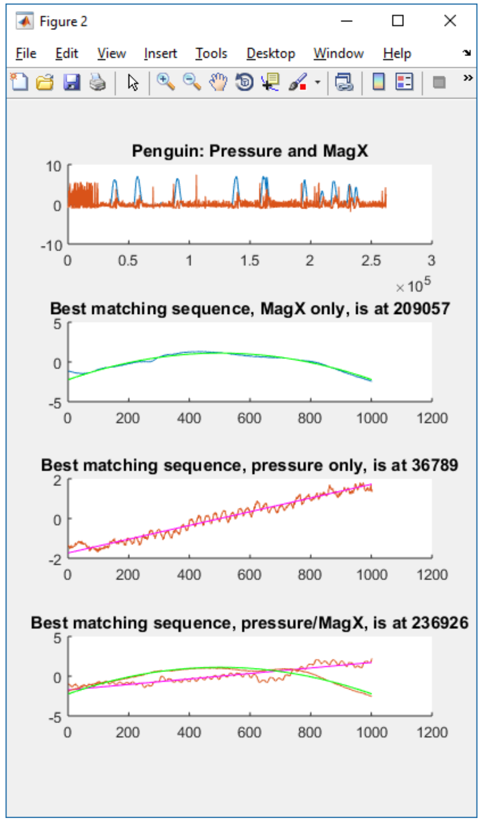


Figure 2
File Edit View Insert Tools Desktop Window Help
Penguin: Pressure and MagX
Best matching sequence, MagX only, is at 209057
Best matching sequence, pressure only, is at 36789
Best matching sequence, pressure/MagX, is at 236926

How do I quickly search this long dataset for this pattern, if an approximate search is acceptable?

As shown elsewhere in this document, exact search is surprisingly fast under Euclidean Distance. However, let us suppose that you want to do even faster search, and you are willing to do an approximate search (but want a high quality answer). A simple trick is to downsample both the data and the query by the same amount (in the below, by 1 in 64) and search the downsampled data. If the data has low intrinsic dimensionality, this will typically give you very good results.
Let us build a long dataset, with 67,108,864 datapoints, and a long query, with 8,192 datapoints

A full exact search takes 12.4 seconds, an approximate search takes 0.24 seconds, and produces (at least in this example) almost exactly the same answer. The answer is just slightly shifted in time.

How well this will work for you depends on the *intrinsic* dimensionality of your data.

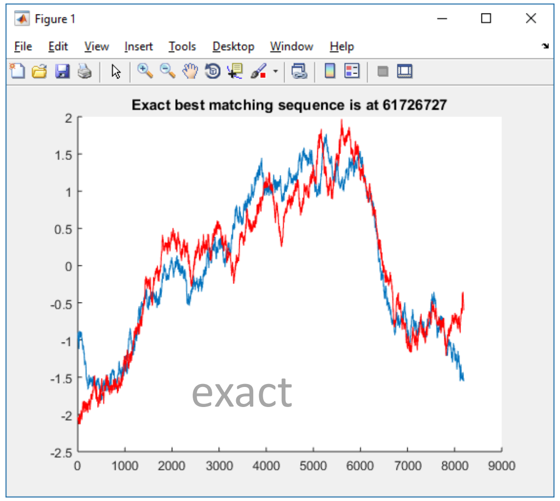
```
rng('default')  % set seed for reproducibility
data= cumsum(randn(1,2^26));  % make data
query= cumsum(randn(1,2^13)); % make a query
tic
  dist = MASS_V2(data ,query );
  [val,loc] = min(dist);      % find best match location
      hold on
      plot(zscore(data(loc:loc+length(query))))
      plot(zscore(query),'r')
  title(['Exact best matching sequence is at ', num2str(loc)])
  disp(['Exact best matching sequence is at ', num2str(loc)])
toc
```

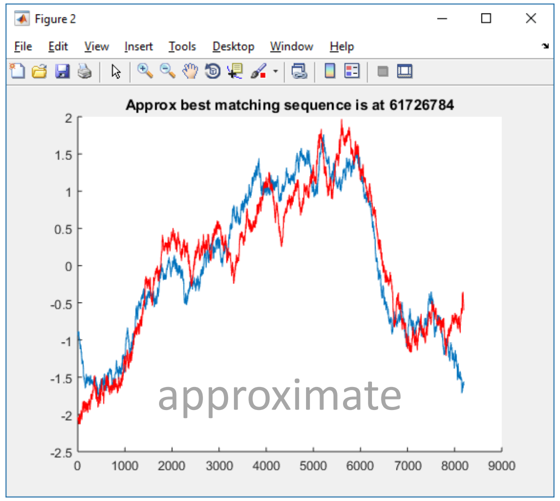
exact

```
figure;
downsampled_data  = data(1:64:end);  % create a downsampled version of the data
downsampled_query = query(1:64:end); % create a downsampled version of the query
tic
  dist = MASS_V2(downsampled_data ,downsampled_query );
  [val,loc] = min(dist);
      hold on
      plot(zscore(data((loc*64):(loc*64)+length(query)))) % multiply the 'loc' by 64 to index correctly
      plot(zscore(query),'r')
  title(['Approx best matching sequence is at ', num2str(loc*64)])
  disp(['Approx best matching sequence is at ', num2str(loc*64)])
toc
```

approximate

```
Exact best matching sequence is at 61726727,  Elapsed time is 12.40 seconds.
Approx best matching sequence is at 61726784, Elapsed time is 0.240 seconds.
```





How can I optimize similarity search in a long time series?

see also "*How do I quickly search this long dataset for this pattern, if an approximate search is acceptable?*"

Suppose you want to find a query inside a long time series, say of length 67,000,000.

First trick: MASS (and several other FFT and DWT ideas) have their best case when the data length is a power of two, so pad the data to make it a power of two (padding with zeros works fine).

Second trick: MASS V3 is a piecewise version of MASS that performs better when the size of the pieces are well aligned with the hardware. You need to tune a single parameter, but the parameter can only be a power of two, so you can search over say 2^{10} to 2^{20}. Once you find a good value, you can hardcode it for your machine.

Naive

```
rng('default')  % Set seed for reproducibility
data= cumsum(randn(1,67000000));  % make data
query= cumsum(randn(1,2^13));     % make a long query
tic
  dist = MASS_V2(data ,query );
  [val,loc] = min(dist);          % find best match location
  hold on
  plot(zscore(data(loc:loc+length(query))))
  plot(zscore(query),'r')
  disp(['Best matching sequence is at ', num2str(loc)])
toc
```

Trick 1

```
figure
data = [data zeros(1,2^nextpow2(67000000) -67000000)]; % pad data to
tic                                                    % next pow of 2
  dist = MASS_V2(data ,query );
  [val,loc] = min(dist);           % find best match location
  hold on
  plot(zscore(data(loc:loc+length(query))))
  plot(zscore(query),'r')
  disp(['After padding: Best matching sequence is at ', num2str(loc)])
toc
```

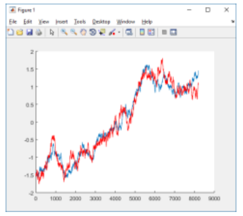
Trick 1 & 2

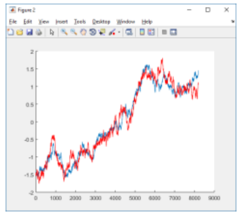
```
figure
tic
  dist = MASS_V3(data ,query, 2^16 );
  [val,loc] = min(dist);       % find best match location
  hold on
  plot(zscore(data(loc:loc+length(query))))
  plot(zscore(query),'r')
  disp(['MASS V3 & padding: Best matching sequence is at ', num2str(loc)])
toc
```

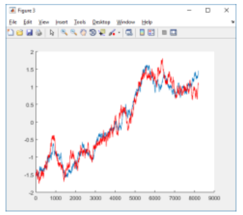
If you run this code, it will output…

```
Best matching sequence is at 32463217
Elapsed time is 14.30 seconds.
After padding: Best matching sequence is at 32463217
Elapsed time is 12.31 seconds.
MASS V3 & padding: Best matching sequence is at 32463217
Elapsed time is 5.82 seconds.
```

Note that it outputs the exact same answer, regardless of the optimizations, but it is fast, then faster, then super fast.







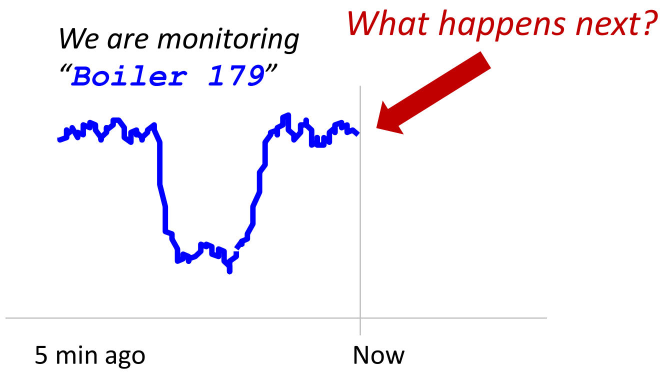
What is most likely to happen next?

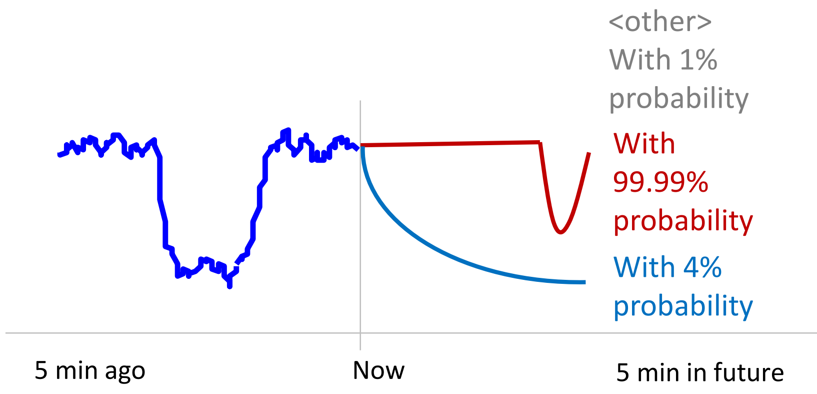
Can we do "predictive text" for time series? It is meant to be *very* "light weight", no model building, no training, no tweaking.

It is not predicting the *value*, only the *shape,* of things to come.

Such predictions can justify themselves, like this...

We predicted this pattern because the last two times we saw a 24-hour prefix that looked like the last day (May 2, June 9), it was followed by this shape.





This dataset is one year of electrical power demand in an Dutch facility city. The key features for our demo are weekends look different and there are some national holidays.

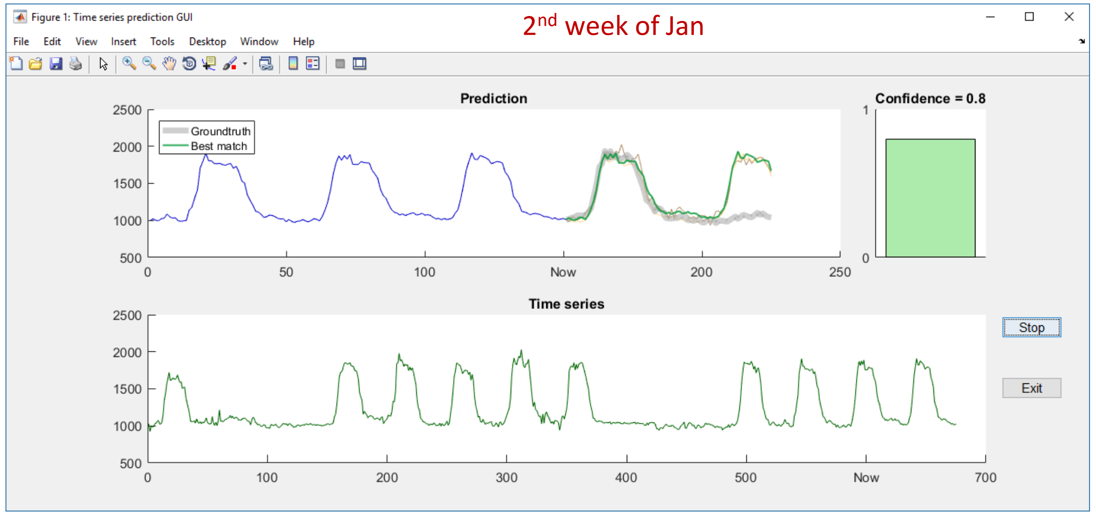
Here our prediction is good for 24 hours, then it is poor.

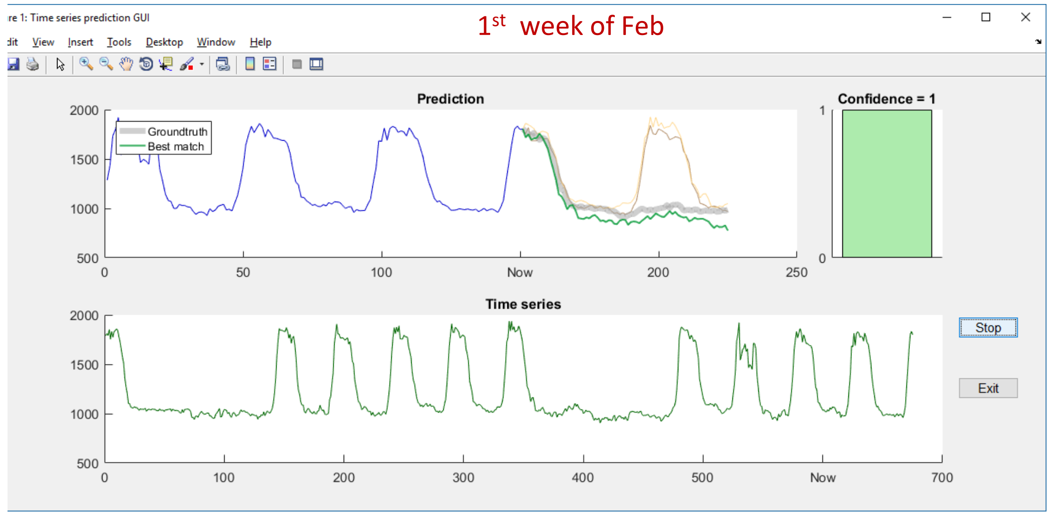
Up to this point, we have seen too few weekends to make good predictions….

(contrast right figure, later in the year)

```
>> data = load('power_data.txt');
>> sub_len = 150;
>> predicting_tool(data, sub_len);
```

(contrast left figure) With just a little more experience, we can predict weekends. How? It is *very* subtle, but on Fridays some folk leave early, and the power demand scales down a little faster in the afternoon….





What is the right length for motifs in this dataset?

See also Are there any repeated patterns in my data?

This is a very interesting question, which more than most, deserves a long explanation. However, to be brief and pragmatic. Let us revisit the EOG dataset. Recall that we choose 4 seconds as the motif length, which I happen to know (from reading papers on the topic) is a good choice.

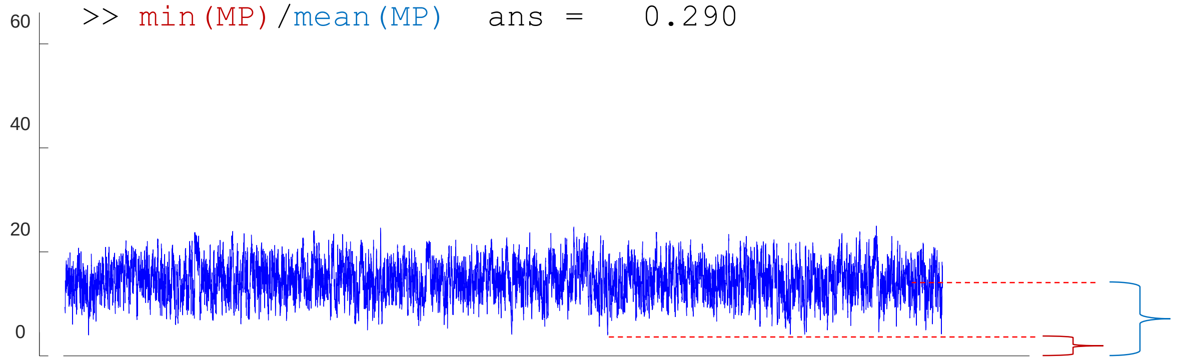
- `>> load eog_sample.mat`
- `>> [MP profileIndex, motifIndex, discordIndex] = interactiveMatrixProfileVer3_website(eog_sample, 400);`

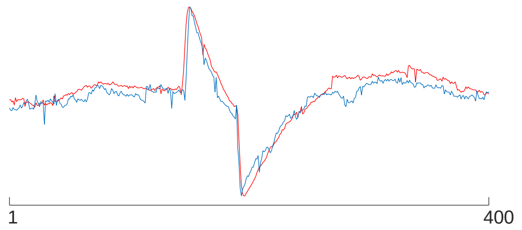
Let us look at the Matrix Profile, and the top motif we find (bottom left). The results seem to make sense.

However, suppose in contrast that we knew nothing about the domain, and had chosen a motif length that was much too long, say length 3,000 (bottom right). How could we know that we had picked a length that was too long? There are two clues:

- Obviously, the motifs themselves will be less well conserved visually.
- The Matrix Profile itself offers useful clues. It tells us how "specially well conserved" the motif is (`min(MP)`) relative to the *average* subsequence (`mean(MP)`). As the ratio of these two numbers is approaches zero, it suggests a stunningly well conserved motif in the midst of others unconserved data. However, as the ratio of these two numbers is approaches one, it suggest that the "motif" is no better conserved than we would expect by random chance. In practice, we rarely compute these ratios, as is visually obvious that the MP looks "flat".

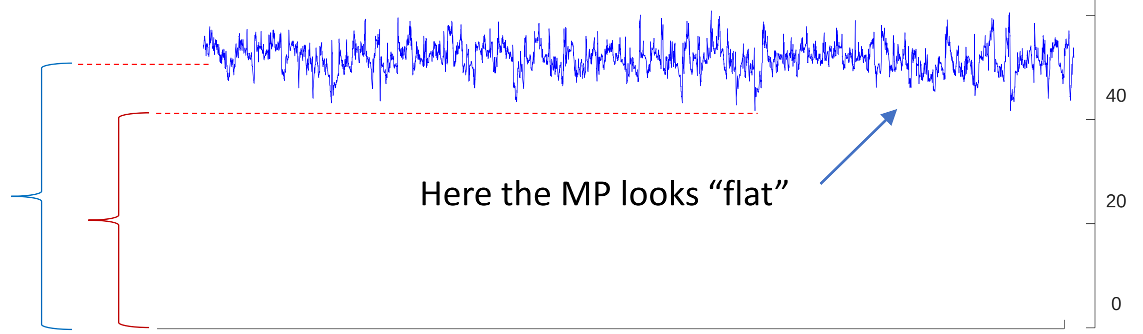
```
>> min(MP)/mean(MP)   ans =    0.290
```



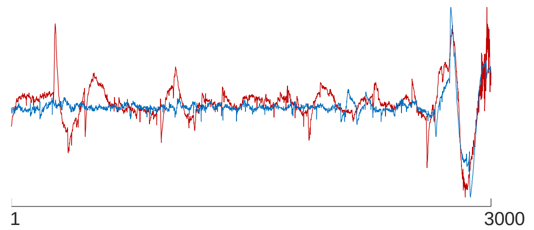


The motifs *are* well conserved visually..

```
>> min(MP)/mean(MP)   ans =    0.756
```



The motif are *not* well conserved visually..



I need to find motifs faster! Part I

Part of the solution might be to use GPUs, see [a][b].

Moreover, it is important to understand, we almost *never* need to compute the Matrix Profile to completion, the anytime SCRIMP++ converges so fast, that in general we just run it 1% (or less) of convergence.

Nevertheless, sometimes you might want to compute the converged Matrix Profile. There is a faster algorithm for this. It exploits some of the ideas in [b], and it exploits the fact that it does not need to waste the overhead needed to make anytime updates, to achieve about an order of magnitude speedup.

For consistency with our other tools, when the fast code finish, it pops open the same plot.

```
>> tic,  [MP MPIndex, mIndex, dIndex] = interactiveMatrixProfileVer3_website(insectvolts(1:35:700000), 300);, toc
Elapsed time is 61.44 seconds.
```

```
>> tic, [MP MPIndex, mIndex, dIndex] = mpx(insectvolts(1:35:700000), 200, 300); , toc
Elapsed time is 6.22 seconds.
```

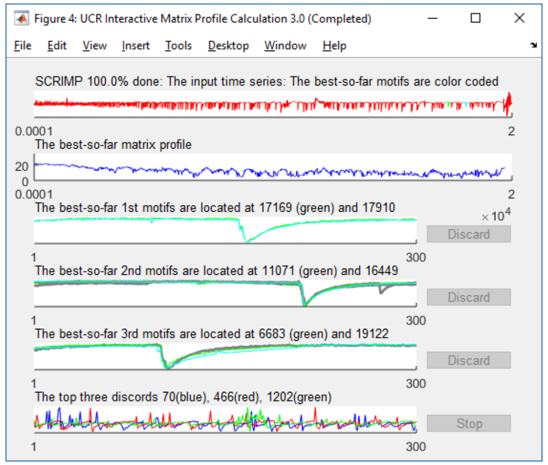


Figure 4: UCR Interactive Matrix Profile Calculation 3.0 (Completed)

SCRIMP 100.0% done: The input time series: The best-so-far motifs are color coded

The best-so-far matrix profile

The best-so-far 1st motifs are located at 17169 (green) and 17910

The best-so-far 2nd motifs are located at 11071 (green) and 16449

The best-so-far 3rd motifs are located at 6683 (green) and 19122

The top three discords 70(blue), 466(red), 1202(green)

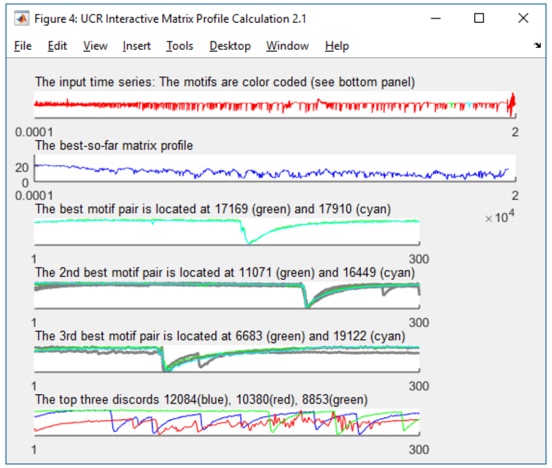


Figure 4: UCR Interactive Matrix Profile Calculation 2.1

The input time series: The motifs are color coded (see bottom panel)

The best-so-far matrix profile

The best motif pair is located at 17169 (green) and 17910 (cyan)

The 2nd best motif pair is located at 11071 (green) and 16449 (cyan)

The 3rd best motif pair is located at 6683 (green) and 19122 (cyan)

The top three discords 12084(blue), 10380(red), 8853(green)

Why are there *very* slightly different results? The value of the exclusion zone, and of ‘r’, the radius were different here. See elsewhere to understand how these effect the motifs returned.

To make this compatible with 2016 MATLAB, we replace the built-in “maxk” with a third party version called “maxk1” (by Salam Ismaeel). If you have later versions of MATLAB, you may wish to undo this change.

[a] https://www.cs.ucr.edu/~eamonn/ten_quadrillion.pdf
[b] https://www.cs.ucr.edu/~eamonn/public/GPU_Matrix_profile_VLDB_30DraftOnly.pdf

I need to find motifs faster! Part II

See also, How do I quickly search this long dataset for this pattern, if an approximate search is acceptable?

Part of the solution might be to use GPUs, see [a][b].

Many datasets are oversampled. For example, the `insectvolts` dataset that accompanies these notes is *greatly* oversampled.

By downsampling, all algorithms have some speedup, but for motif discovery, that speed up is most dramatic.

You need to remember to downsample the query length by the same factor. For example, in the below we want to find motifs of five seconds, in a 100 Hz dataset. So we should use..

- `eog_sample(1:1:end)` with a motif length of **500**. This in the original data, or..
- `eog_sample(1:2:end)` with a motif length of **250**. or…
- `eog_sample(1:5:end)` with a motif length of **100**. or…

(500 is the original motif length, note that in every case, X times Y = 500)

Below, we obtain speedup by downsampling, and get essentially the same results. However, it is important to note that in both cases we found the motifs well before even preSHRIMP finished. In this case, in a few seconds.

For greatly oversampled datasets, this simple trick can get you speedups of 2 or 3 orders of magnitude!

```
>> tic, [MP MPIndex, mIndex, dIndex] = interactiveMatrixProfileVer3_website(eog_sample(1:1:end), 500);, toc
Elapsed time is 186.02 seconds.
```

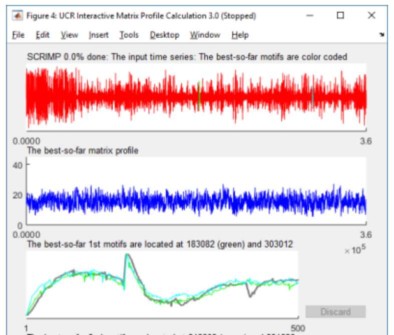


Figure 4: UCR Interactive Matrix Profile Calculation 3.0 (Stopped)
File Edit View Insert Tools Desktop Window Help
0.0000
3.6
40
20
0
0.0000
3.6
×10^5
1
500
Discard

SCRIMP 0.0% done: The input time series: The best-so-far motifs are color coded

The best-so-far matrix profile

The best-so-far 1st motifs are located at 183082 (green) and 303012

```
>> tic, [MP MPIndex, mIndex, dIndex] = interactiveMatrixProfileVer3_website(eog_sample(1:5:end), 100);, toc
Elapsed time is 43.81 seconds.
```

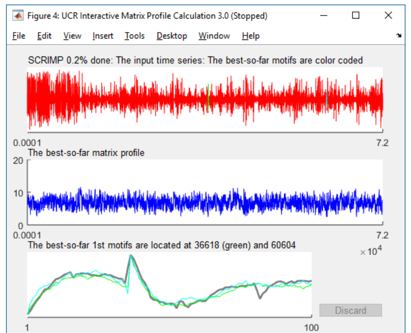


Figure 4: UCR Interactive Matrix Profile Calculation 3.0 (Stopped)
File Edit View Insert Tools Desktop Window Help
0.0001
7.2
20
10
0
0.0001
7.2
×10^4
1
100
Discard

SCRIMP 0.2% done: The input time series: The best-so-far motifs are color coded

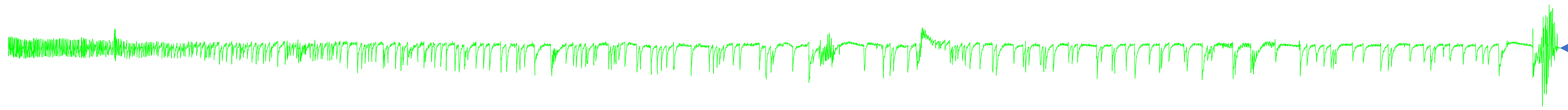
The best-so-far matrix profile

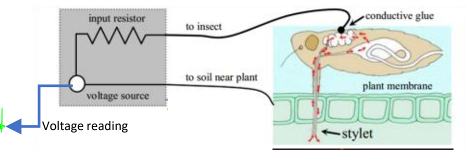
The best-so-far 1st motifs are located at 36618 (green) and 60604

Note: These are the times for preSHRIMP *only*

[a] https://www.cs.ucr.edu/~eamonn/ten_quadrillion.pdf
[b] https://www.cs.ucr.edu/~eamonn/public/GPU_Matrix_profile_VLDB_30DraftOnly.pdf

Have we ever seen a pattern that looks just like this, but possibly at a different length?





In our insect data, a basic feeding primitive looks like this: , we can model it with something like: [1:600].^0.2

We have a theory that a certain higher level behavior will result in "*A long primitive, followed by a shorter and smaller primitive, followed by another long primitive*", like this.. , we can model it with: [[[1:600].^0.2] [[1:300].^0.2] [[1:600].^0.2]]

However, we don't know how long the whole pattern could be…

The function to the right can solve this problem. It simply brute forces a MASS test for all lengths within a range (here 100 to 300%) at a given step size (here 5%).
There may be faster techniques, but MASS is so fast, they may not be worth bothering with.
A critical trick is to *normalize* the comparisons at different lengths (see Appendix).

```
>> load insectvolts.mat    % load some insect epg data
>> query = ([[[1:600].^0.2] [[1:300].^0.2] [[1:600].^0.2]]);
>> uniform_scaling_search(smooth(insectvolts,10), query);
```

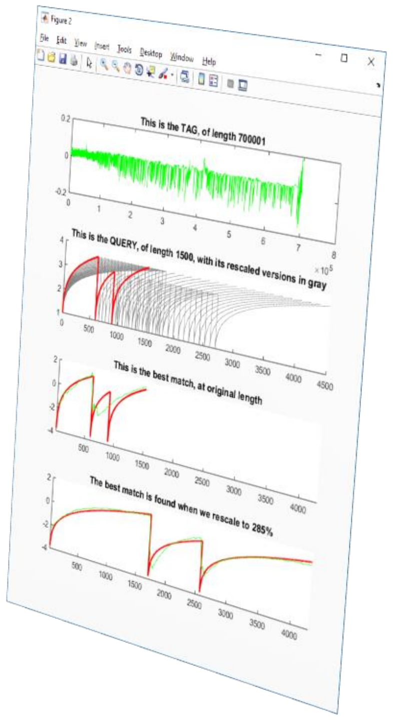
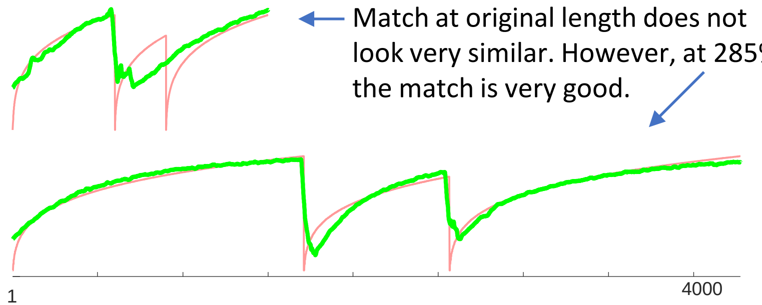


Figure 2
File Edit View Insert Tools Desktop Window Help
This is the TAG, of length 700001
This is the QUERY, of length 1500, with its rescaled versions in gray
This is the best match, at original length
The best match is found when we rescale to 285%



Match at original length does not look very similar. However, at 285% the match is very good.

Even with this unoptimized approach. We can search two hours of data (at 100hz), with a long query (upto to 4,500 datapoints), in well under 10 seconds

This looks like a lot of code, but most of it is for plotting

```
function [] = uniform_scaling_search(TAG, QUERY)
figure;                                               % Spawn a blank figure
subplot(4,1,1);                                       % Plot panel 1
plot(TAG,'g')                                         % Plot the TAG/time series in green
title(['This is the time series, of length ',num2str(length(TAG))]);

subplot(4,1,2);                                       % Plot panel 2
hold on;
title(['This is the QUERY, of length ',num2str(length(QUERY)), ', rescaled versions in gray']);
for i = 110:10:300                                    % Plot the rescaled queries
    plot(QUERY(1:100/i:end),'color',[0.5 0.5 0.5])
end
plot(QUERY,'LineWidth',2,'color','r');                % Plot the query

subplot(4,1,4);                                       % Plot panel 4
hold on;
best_match_val = inf;
for i = 100:5:300                                    % Loop over all scalings
    NewQUERY = (QUERY(1:100/i:end));
    distprofile = MASS_V3(TAG, NewQUERY, 1024);
    [val,loc] = min(distprofile);
    val = val * 1/sqrt(i);                            % This normalizaiton step is critical see Appendix
    if val < best_match_val                           % Record the best scaling
        best_match_val = val;
        best_match_loc  = loc;
        best_match_scale = i;
    end
end

plot(zscore(QUERY(1:100/best_match_scale:end)),'LineWidth',2,'color','r');   % Plot bestscaled match
plot(zscore(TAG(best_match_loc:best_match_loc+length(QUERY(1:100/best_match_scale:end))-1)),'g');
set(gca,'Xlim',[1 length(QUERY(1:100/best_match_scale:end))]);
title(['The best match is found when we rescale to ',num2str(best_match_scale),'%']);

subplot(4,1,3);                                       % Plot panel 3
hold on;
distprofile = MASS_V3(TAG, QUERY, 1024);              % Compute the distance profile
[val,loc] = min(distprofile);                         % Find were the best match was
plot(zscore(QUERY),'LineWidth',2,'color','r');        % Plot the query
plot(zscore(TAG(loc:loc+length(QUERY)-1)),'g');
set(gca,'Xlim',[1 length(QUERY(1:100/best_match_scale:end))]);
title(['This is the best match, at original length'])

end
```

How can I know which of these two classification approaches is best for time series?

This is a very tricky question. A sophisticated statistical test is probably the answer, but beyond the scope of this document.

However, here we consider a simple way to *visualize* the answer.

Many papers have essentially said "*we tested on many datasets, we are better on some, so we are useful sometimes*". However, it is not useful to be better sometimes, unless you know *in advance* you are going to be better!

In brief, after computing the strawman/baseline (here, the Euclidean distance) we then compute the **expected improvement** we would get using the proposed algorithm (here cDTW, learning any parameters and settings on just the training data), then compute **the actual improvement** obtained (using these *now* hardcoded parameters and settings). We then plot a point for each dataset, using the expected and actual improvement (either of which could be less than one)

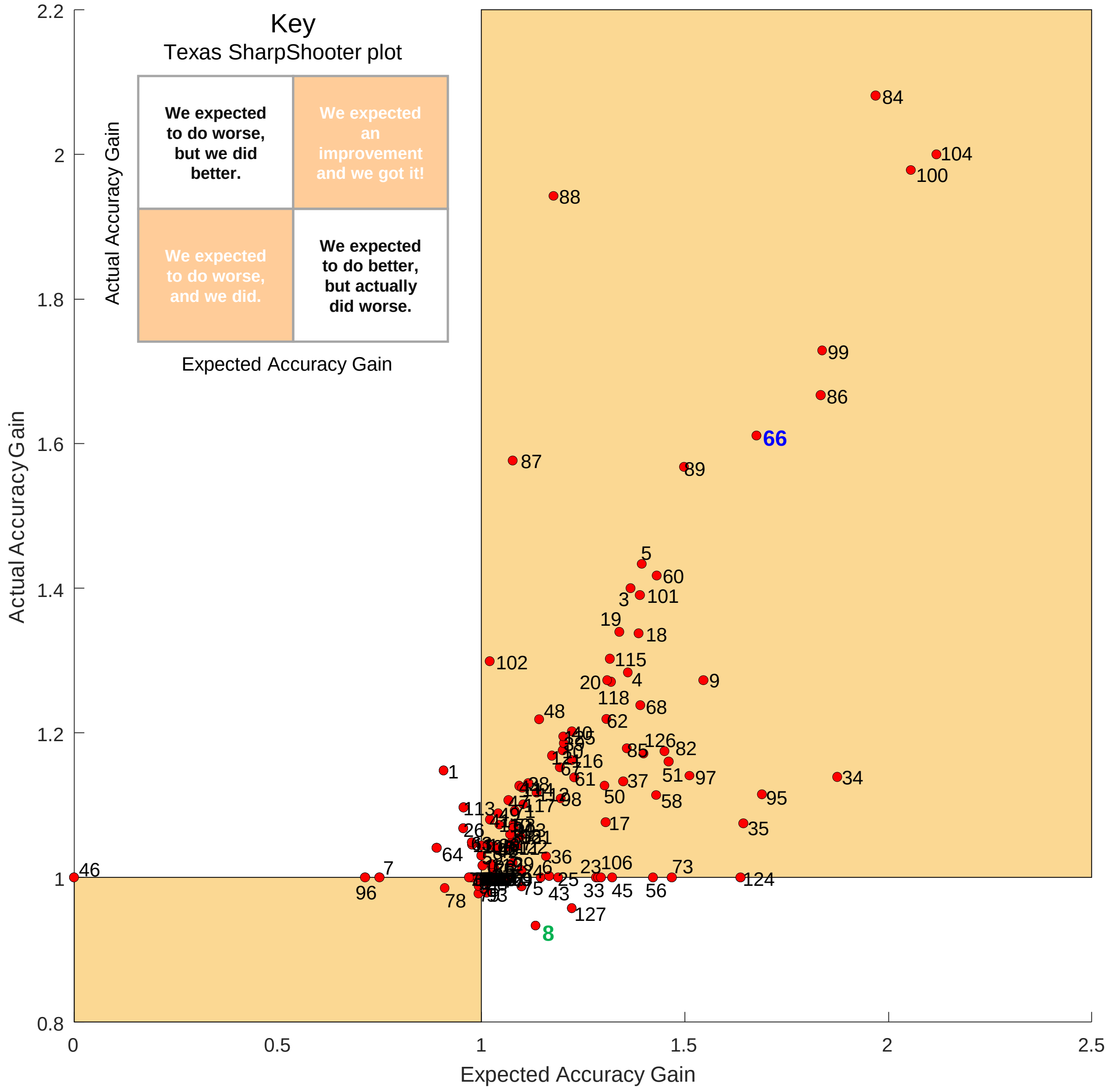
As the key shows, there are four possible outcomes for each dataset, we would prefer to be in a yellow region (ideally, the upper right region).

Let us answer the question for 1NN-ED vs, 1NN-DTW. We did all the experiments and saved them into a file called texas_plot_2018.csv.

```
    >> result_file = texas_plot_2018.csv
    >> plot_texas_sharpshooter(result_file)
```

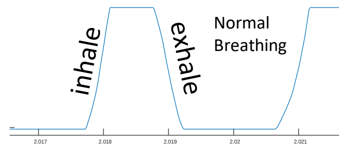
This plot show forceful evidence that cDTW is superior to ED. This is unsurprising, since cDTW subsumes ED as a special case. The rare examples where cDTW is worse is because we learned an unsuitable warping window*.

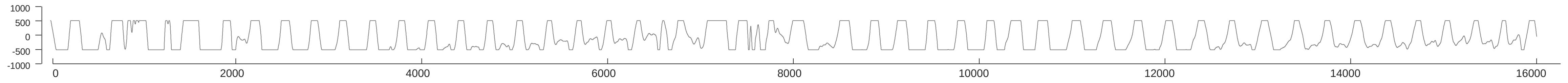
See arxiv.org/abs/1810.07758 for more details



*For example, 8 is BeetleFy, with just 20 train and 20 test instances. Here we expected to do a little better, but we did a little worse.
In contrast, for 66 (LargeKitchenAppliances) we had 375 train and 375 test instances, and where able to more accurately predict the warping window that gives a large improvement.

Are there any evolving patterns in this dataset (time series chains)

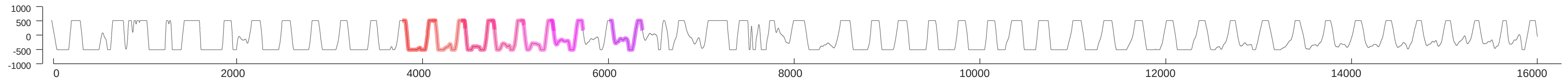




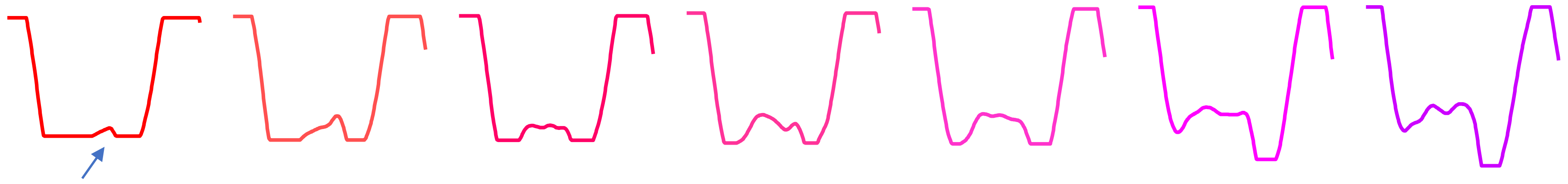
This is a dataset of respiration from a sleep study. Each breath appears to be about 360 data points long. So lets search *for time series chains* of length 360…

```
>> load respiration.mat
>> TSC1_demo(respiration , 360);
```

The algorithm finds the highlighted chains below.



Let us zoom in on the chains, to better see what is going on…



Note the increasing "gulp" artifact that happens between cycles. Also note that it begins to happen earlier and earlier in the cycle. What does this mean?

Here is the (lightly edited) annotation of Dr. Gregory Mason (LA BioMed/UCLA) an expert on cardiopulmonary interactions.

"The gulps are attempts to inspire against an obstruction coming the back of the tongue. The large signals are from the machine which do not necessarily reach the patient, the small gulps are pathologic attempts to breathe. Why does it increase? With each successive breath the patient tries harder to inspire. It finally is 'synchronized' and you don't see the small patient signal, and this event cycles over and over. The cycling is best seen without treatment if one looks up "crescendo snoring," a hallmark of obstructive sleep apnea."

Please consider “donating” question

- Even better if you can donate data
- Even better if you can donate an answer!

Appendices

- Below are miscellaneous appendices

In some of our examples we needed to compare the Euclidian distances of pair of time series, that are of different lengths.

How should we normal to compensate for the length differences?

We could:

1) Not normalize at all, but this strongly biases us to *short* patterns.

2) Normalize by dividing my the length of the time series. This seems to make sense, but it strongly biases us towards *long* patterns.

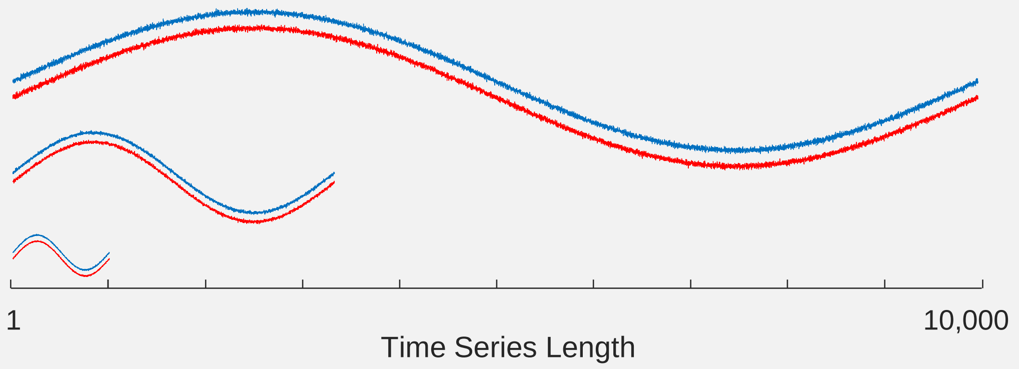
3) Normalize by the reciprocal of square root of length of the time series. Without explanation (here) we claim that this is correct.

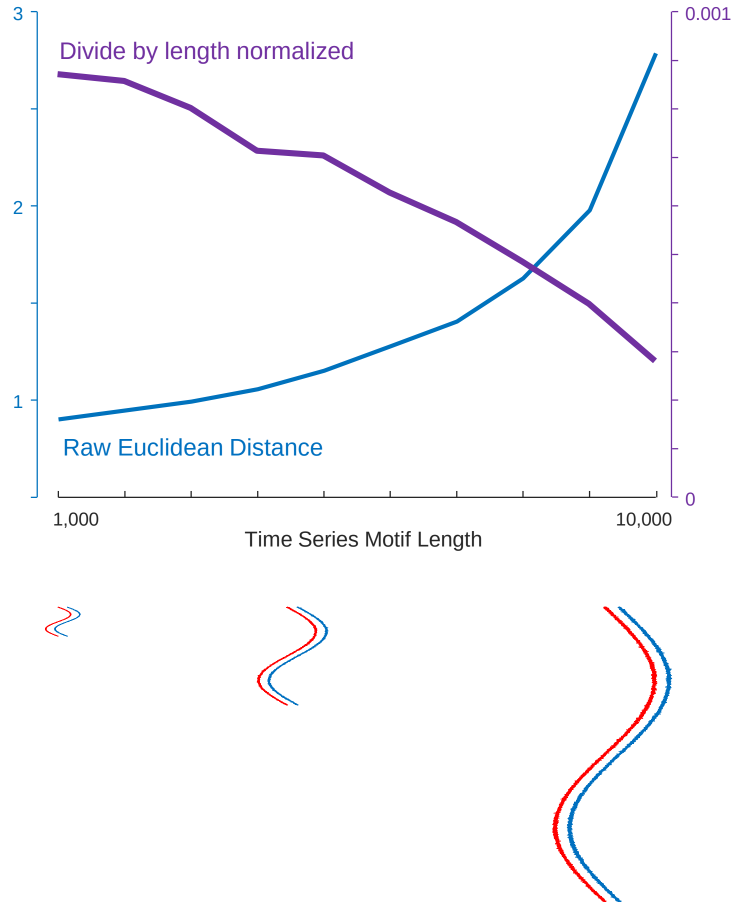
In the figures on the *bottom right*, we compare the three above ideas on slightly nosily sine waves of different lengths (shown *top right*).

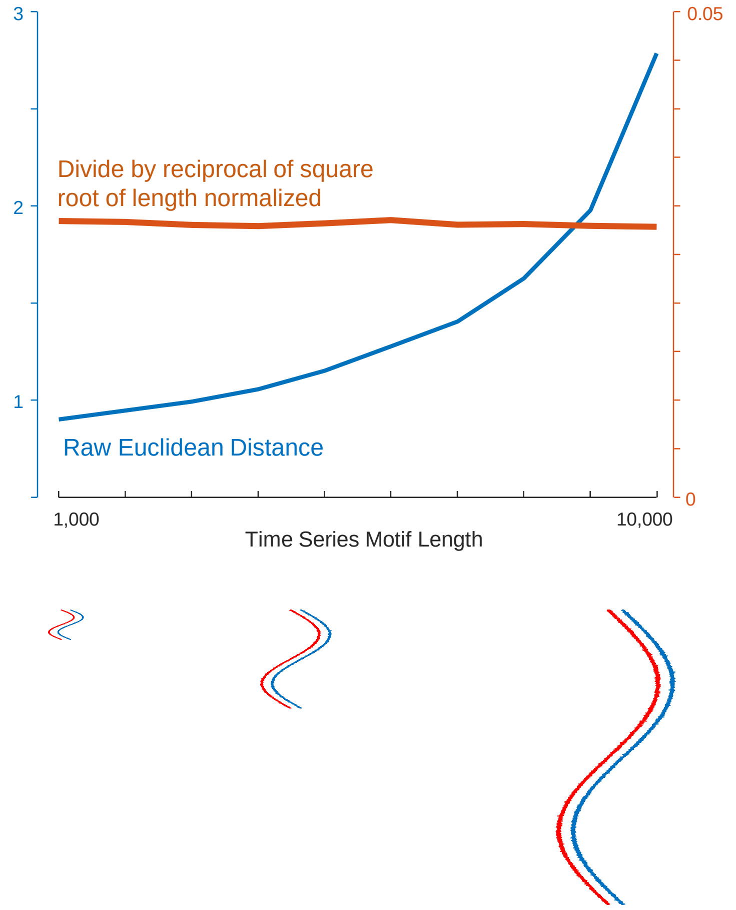
As you can see, the "Normalize by the reciprocal of square root of length of the time series" approach is invariant to the length of the patterns.

Which of these three pairs is closest?

Under mild assumptions, we might claim that they are *all* equally similar.


Time Series Length


Divide by length normalized
Raw Euclidean Distance
Time Series Motif Length


Divide by reciprocal of square root of length normalized
Raw Euclidean Distance
Time Series Motif Length