1 1/24/16

1.1 Permutation testing

In order to test how the algorithm returns the permutation results, I set up sample matrices using `create_toy.m` and recorded the distance between the result returned by the algorithm (`eig_perm`) and that of the actual input data (`ss`).

Listing 1: Compare Permutations

```
rois = 33;
noise_mag = 0;
trace_type = 'sines';
shift = 10;
y = zeros(1,100);
for i = 1:length(y)
        [Z, ss] = create_toy(trace_type, 'rois', rois, ...
                'noisemag', noise_mag, ...
                'shift', shift);
        [eig_phases,eig_perm,slm,evals] = cyclic_analysis(Z);
        y(i) = cyclic_distance(ss,eig_perm);
end
plot(y)
```

The function `create_toy` produces a set of `rois` identical sine waves randomly spread over a `shift` time step period (i.e., each copy of the trace is shifted along the x axis so that all copies are within `shift` time steps of each other). The variable `ss` shows the ordering of the traces produced. Using the setting above, for example, a data set like that in on the left of Figure 1 is produced. The figure on the right shows that the algorithm returns the identical permutation each time. The distance is measured using `cyclic_distance(V1,V2)` which simply counts the how many steps V2 is from V1 (let V1 be (1:5), then [2,1,3,4,5] is a distance of 1 from V1 and [2,3,4,5,1] is a distance of 5 from V1.) This doesn't really count "cyclic distance" but works for this application since we would like to see if we are picking up the correct starting point and cycling in the right direction. Increasing `shift` to 18 (the period for the sine wave is 20) starts to mess up the permutation since the algorithm can't determine the proper starting point.

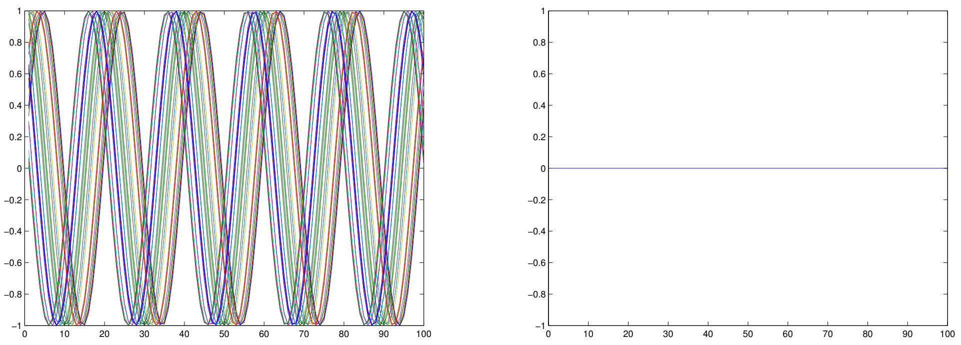

Figure 1: Left: Sample data created from sine waves shifted randomly at most 10 time steps apart (no added noise). Right: Comparison of the permutation returned by the algorithm with that of the actual data set for 100 separately generated data sets.

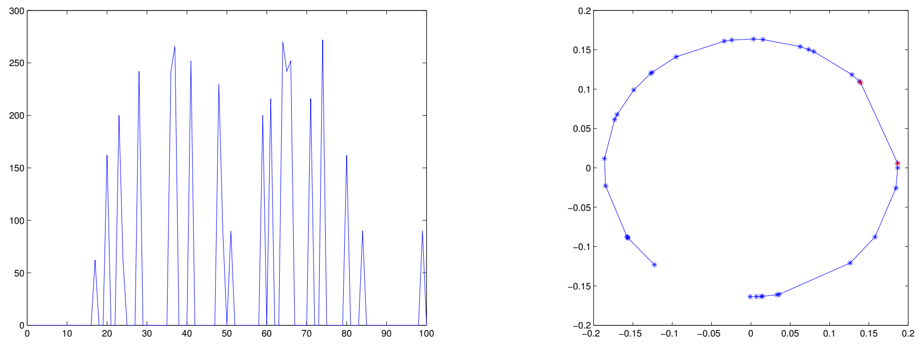

Figure 2: Left: Algorithm performance when shift=18. Right: Plot of phases for a permutation which the algorithm had shifted by 11 places (distance = 242). The red asterisks highlight the 11th and 12th points in the cycle.

The algorithm determines the starting point by finding the largest gap between points in the phase plot (greatest difference in angle). Once the phases are too spread out, the largest gap can easily be between a pair of middle points. Note, however, that still the algorithm cycles in the correct direction in each case.

2 Listings

Listing 2: cyclic_distance

```
function [dist] = cyclic_distance(V1, V2)
[~, ss] = sort([V1(:), V2(:)]);
dist = sum(abs(diff(ss,1,2)))/2;
```

2.1 Movies

2.1.1 Data Treatment

In the original data set, there are gaps in the data for some genres and not all genres began at the same time so some initial data treatment was required. First, years in which there were gaps in the data were inpainted using linear interpolation. Next, the genre counts were taken on a log scale to account for the fact that movie production has grown exponentially. Finally, only years in which genres had data points were considered (1916-2015). The film-noir genre had to be excluded since the production years were so limited. The following table shows the results of `analyze_cyclicity` on the movie data (processed as described above).

quad results	zscore results
musical	musical
music	music
family	family
thriller	thriller
mystery	mystery
sci fi	sci fi
drama	drama
crime	crime
adventure	romance
action	adventure
fantasy	history
comedy	comedy
romance	action
animation	western
history	fantasy
biography	documentary
documentary	animation
horror	biography
war	horror
sport	sport
western	war

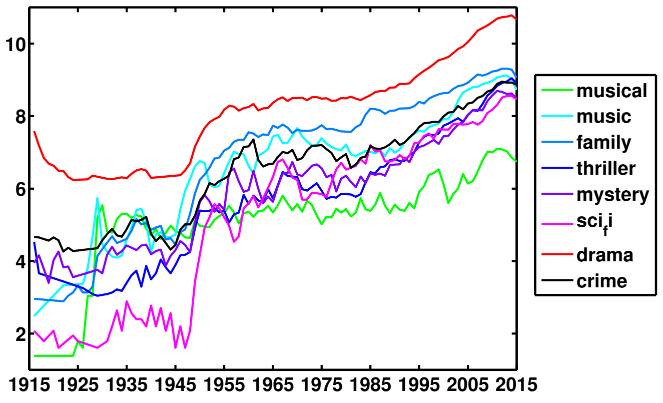

Figure 3: Traces from the first eight genres.

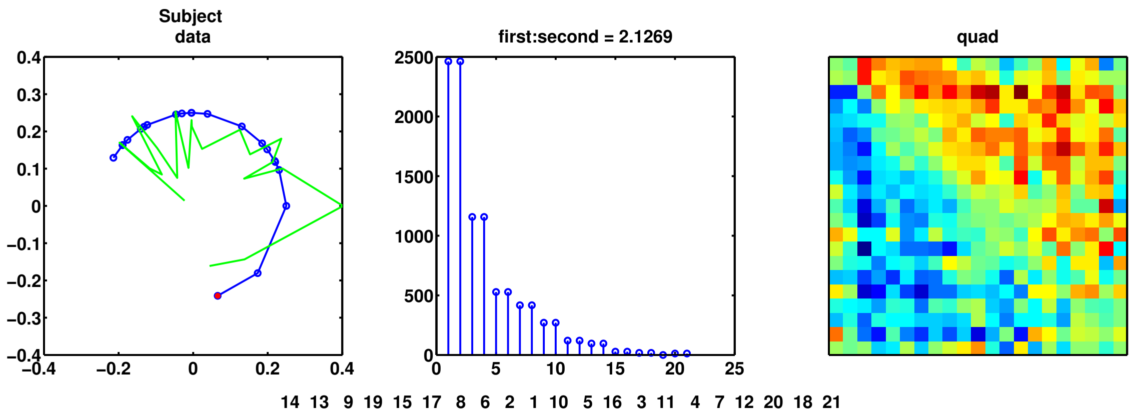

Figure 4: Cyclicity results using quadratic variation normalization

Note that the Musicals and Music genres look very similar to each other, but dissimilar to everything else and show a drastic jump around 1925. Since this is such a long time ago, it seemed worth considering more recent trends in the data.

quad results	zscore results
romance	romance
family	family
adventure	adventure
documentary	documentary
thriller	musical
musical	history
history	thriller
drama	drama
sport	comedy
comedy	sport
music	music
mystery	mystery
crime	crime
western	western
biography	animation
horror	biography
war	horror
action	war
fantasy	action
animation	fantasy
sci-fi	sci-fi

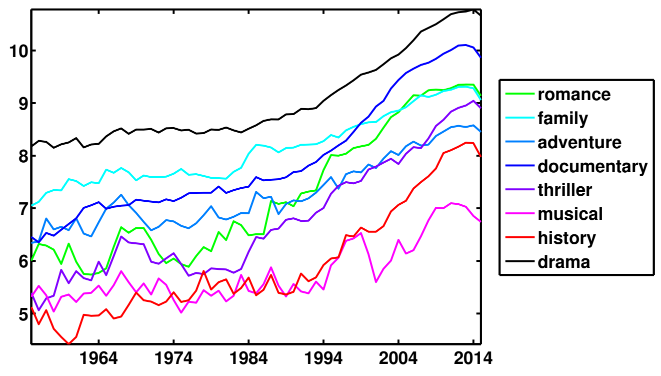

Figure 5: Traces from the first eight genres using only years 1955-2015 in the cyclicity calculation.

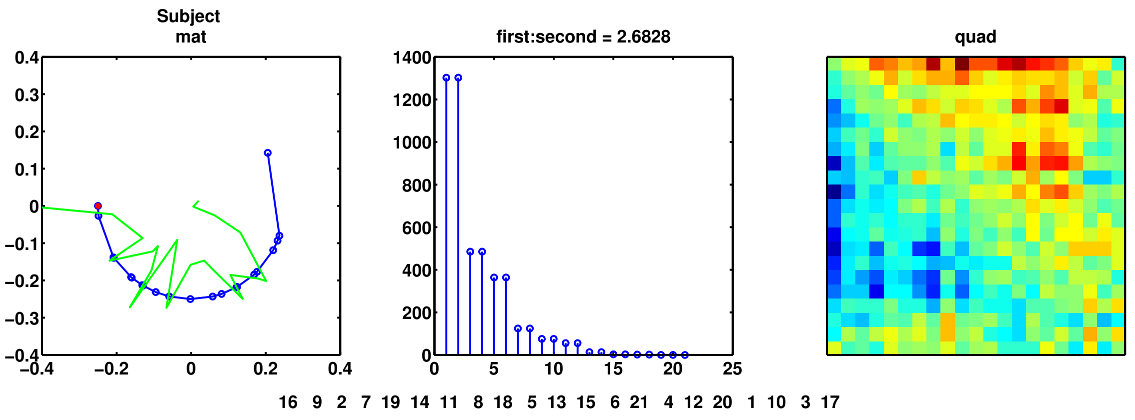

Figure 6: Cyclicity results from 1955-2015 movie data using quadratic variation normalization

3 2/2/16

3.1 Movie trends with words counts from New York Times

We added some word trends to the movie data, taking search counts for selected words on the New York Times website. The search was filtered by year so that we could get word counts for each year that we have movie data for (1916-2015). Word counts were pulled using `times_scrape.py` and looping through the years in a bash script (see scraping folder in Movies). The words searched were searched

crisis	terror	war	attack	expansion
growth	invasion	prosperity	shooting	

and the following results obtained:

perm results
EXPANSION
WAR
musical
ATTACK
music
INVASION
family
western
mystery
thriller

perm results
sci fi
drama
crime
adventure
action
fantasy
comedy
animation
biography
horror

perm results
war
history
romance
documentary
sport
TERROR
CRISIS
SHOOTING
GROWTH
PROSPERITY

Figure 7: Resulting permutation from full movie and word analysis

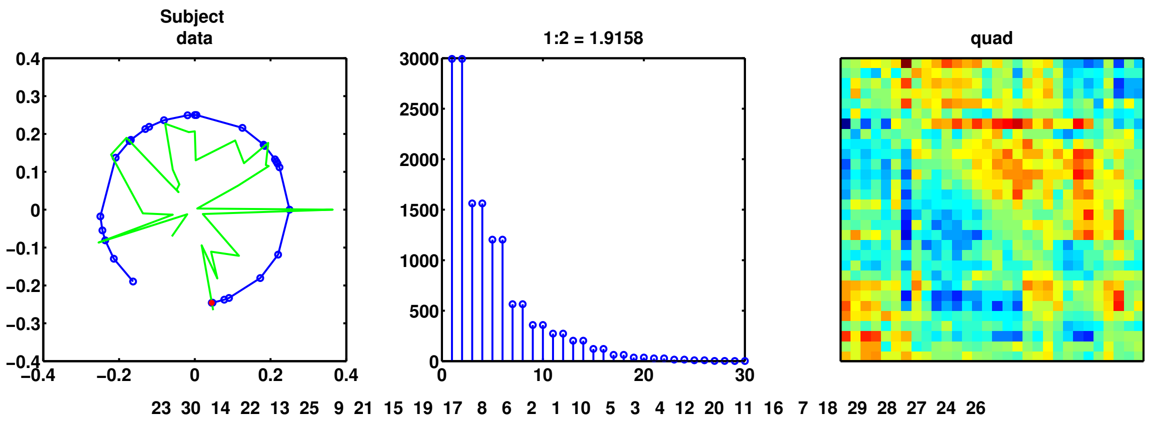

Figure 8: Results of full movie and word analysis

3.2 Refining Movie-word results

Next, we wish to consider smaller subsets of traces using the magnitude of the phase associated with a trace as an indicator of agreement with a particular ordering. For each set of figures, `p` indicates which eigenvector was used to generate the ordering , `n` indicates how many categories were used in the smaller group analysis and `group` is which subset of the full set. So, for example, if `p=1`, `n=8`, `group=1` then the first eigenvector was used to generate an ordering of full set of categories and then the eight categories with the highest magnitude of associated phase vector were pulled out and reanalyzed using the cyclicity algorithm to yield the ordering shown in the legend. If `group=2` the group 1 categories were pulled out of the original data set and then the same procedure was followed, ordering the categories associated with the top eight phases of this smaller subset. When p=2, the second eigenvector is used in the cyclicity algorithm to generate the ordering.

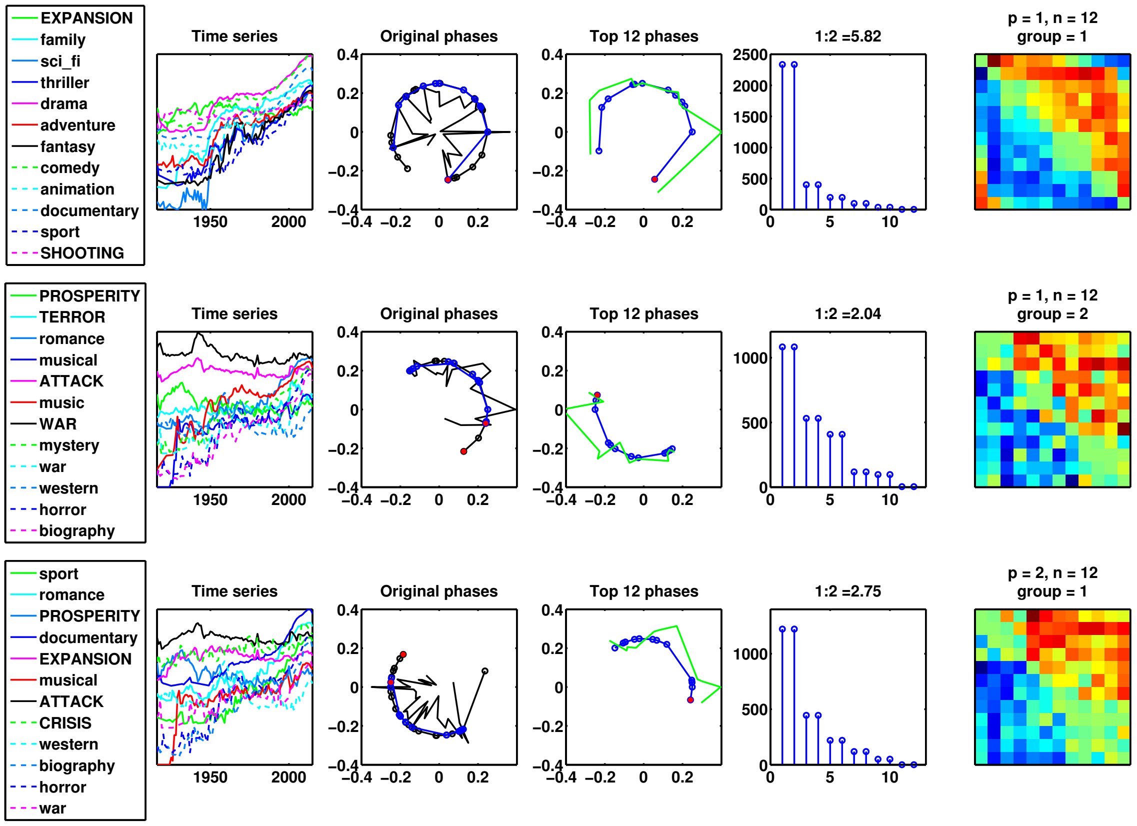

Figure 9: Results with n=12.

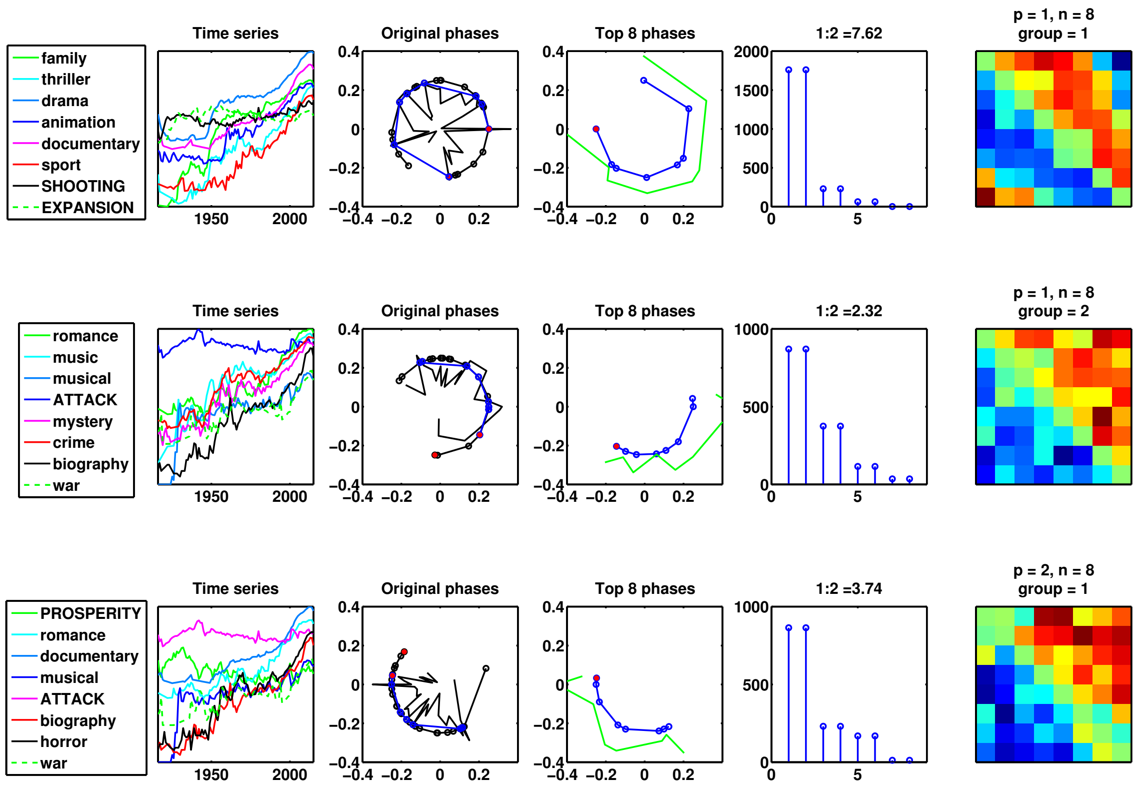

Figure 10: Results with n=8.

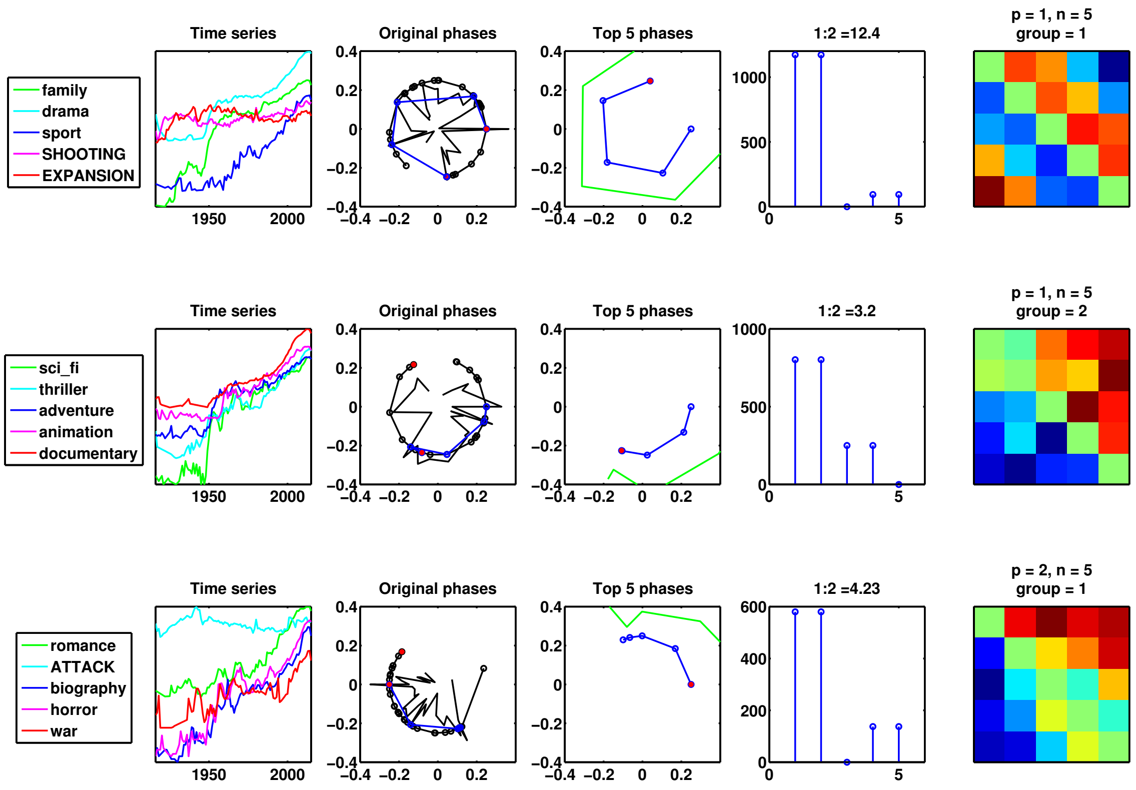

Figure 11: Results with n=5.

In each figure, the first subplot on the left shows the original traces (log scale) of the n categories. The next subplot shows the original phase plot in black with the final n categories highlighted in blue with the new ordering. Mostly the ordering from the initial analysis is preserved, but there are a few places (`p=1`, `n=5`, `group=2`, for example) where if two categories were initially close together, they get exchanged in the final ordering. The final three plots are the same thing that we have been looking at all along for the subset of n categories. Note that for the eigenvector ratio label on subplot 4, I the label says 1:2, which is actually 1:3 since the eigenvalues are complex conjugates so 1 and 2 have the same absolute value. If the third eigenvector was equal to 0 (odd man out), then I took the ratio of the first to the next non-zero (first:fourth technically).

3.3 Impressions

- I like the set of five. It is easy to digest and has high e1:e2 ratios. Until I have some reason to do otherwise, I'll look at sets of 5.

- Group 2 is not fantastic. Of the three analyses shown, it has the lowest e1:e2 ratio. I will probably run it anyway when we look at the brain permutations.
- I would like to cut off the first 40 years as before and see what those trends look like.