A DATA SCIENCE TUTORIAL

Four ways to quantify synchrony between time series data

Sample code and data to compute synchrony metrics including Pearson correlation, time-lagged cross correlations, dynamic time warping, and instantaneous phase synchrony.

Jin Hyun Cheong May 13, 2019 · 7 min read ★

Airplanes flying in synchrony, photo by Gabriel Gusmao on Unsplash

In psychology, synchrony between individuals can be an important signal that provides information about the social dynamics and potential outcomes of social interactions. Synchrony between individuals has been observed in numerous domains including bodily movement (Ramseyer & Tschacher, 2011), facial expressions (Riehle, Kempkensteffen, & Lincoln, 2017), pupil dilations (Kang & Wheatley, 2015), and neural signals (Stephens, Silbert, & Hasson, 2010). However, the term *synchrony* can take on many meanings as there are various ways to quantify synchrony between two signals.

In this article, I survey the pros and cons of some of the most common synchrony metrics and measurement techniques including the Pearson correlation, time lagged cross correlation (TLCC) and windowed TLCC, dynamic time warping, and instantaneous phase synchrony. To illustrate, the metrics are calculated using sample data in which smiling facial expressions were extracted from a video footage of two participants engaging in a 3 minute conversation (screenshot below). To follow along, feel free to download the sample extracted face data and the Jupyter notebook containing all the example codes.

Outline

1. Pearson correlation
2. Time Lagged Cross Correlation (TLCC) & Windowed TLCC
3. Dynamic Time Warping (DTW)
4. Instantaneous phase synchrony

Sample data is the smiling facial expression between two participants having a conversation.

1. Pearson correlation — simple is best

The Pearson correlation measures how two continuous signals co-vary over time and indicate the linear relationship as a number between -1 (negatively correlated) to 0 (not correlated) to 1 (perfectly correlated). It is intuitive, easy to understand, and easy to interpret. Two things to be cautious when using Pearson correlation is that 1) outliers can skew the results of the correlation estimation and 2) it assumes the data are homoscedastic such that the variance of your data is homogenous across the data range. Generally, the correlation is a snapshot measure of global synchrony. Therefore it does not provide information about directionality between the two signals such as which signal leads and which follows.

The Pearson correlation is implemented in multiple packages including Numpy, Scipy, and Pandas. If you have null or missing values in your data, correlation function in Pandas will drop those rows before computing whereas you need to manually remove those data if using Numpy or Scipy's implementations.

The following code loads are sample data (in the same folder), computes the Pearson correlation using Pandas and Scipy and plots the median filtered data.

Once again, the Overall Pearson r is a measure of *global* synchrony that reduces the relationship between two signals to a single value. Nonetheless there is a way to look at moment-to-moment, *local* synchrony, using Pearson correlation. One way to compute this is by measuring the Pearson correlation in a small portion of the signal, and repeat the process along a rolling window until the entire signal is covered. This can be somewhat subjective as it requires arbitrarily defining the window size you'd like to repeat the procedure. In the code below we use a window size of 120 frames (~4 seconds) and plot the moment-to-moment synchrony in the bottom figure.

Sample data on top, moment-to-moment synchrony from moving window correlation on bottom.

Overall, the Pearson correlation is a good place to start as it provides a very simple way to compute both global and local synchrony. However, this still does not provide insights into signal dynamics such as which signal occurs first which can be measured via cross correlations.

such as a leader-follower relationship in which the leader initiates a response which is repeated by the follower. There are couple ways to do investigate such relationship including Granger causality, used in Economics, but note that these still do not necessarily reflect true causality. Nonetheless we can still extract a sense of which signal occurs first by looking at cross correlations.

http://robosub.eecs.wsu.edu/wiki/ee/hydrophones/start

As shown above, TLCC is measured by incrementally shifting one time series vector (red) and repeatedly calculating the correlation between two signals. If the peak correlation is at the center (offset=0), this indicates the two time series are most synchronized at that time. However, the peak correlation may be at a different offset if one signal leads another. The code below implements a cross correlation function using pandas functionality. It can also *wrap* the data so that the correlation values on the edges are still calculated by adding the data from the other side of the signal.

Peak synchrony is not at the center, suggesting a leader-follower signal dynamic.

In the plot above, we can infer from the negative offset that Subject 1 (S1) is leading the interaction (correlation is maximized when S2 is pulled forward by 47 frames). But once again this assesses signal dynamics at a global level, such as who is leading during the entire 3 minute period. On the other hand we might think that the interaction may be even *more* dynamic such that the leader follower roles vary from time to time.

To assess the more fine grained dynamics, we can compute the *windowed* time lagged cross correlations (WTLCC). This process repeats the time lagged cross correlation in multiple windows of the signal. Then we can analyze each window or take the sum over the windows would provide a score comparing the difference between the leader follower interaction between two individuals.

Windowed time lagged cross correlation for discrete windows

in the interaction. For example, in the first window (first row), the red peak to the right suggests S2 initially leads the interaction. However by the third or fourth window (row), we can see that S1 starts to lead the interaction more. We can also compute this continuously resulting in a smoother plot as shown below.

Rolling window time lagged cross correlation for continuous windows

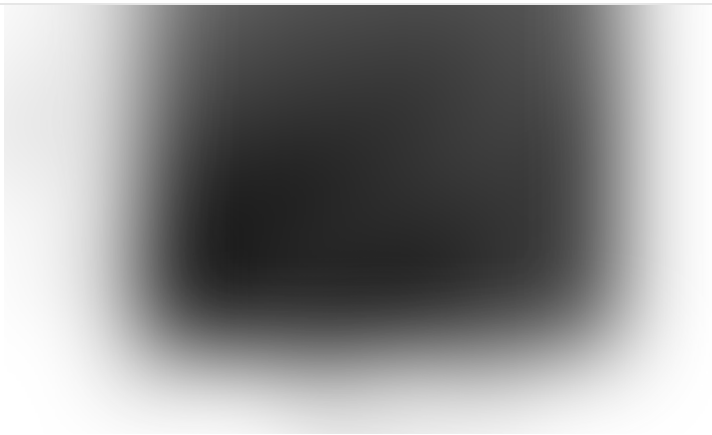
Time lagged cross correlations and windowed time lagged cross correlations are a great way to visualize the fine-grained dynamic interaction between two signals such as the leader-follower relationship and how they shift over time. However, these signals have

3. Dynamic Time Warping — synchrony of signals varying in lengths

Dynamic time warping (DTW) is a method that computes the path between two signals that minimize the distance between the two signals. The greatest advantage of this method is that it can also deal with signals of different length. Originally devised for speech analysis (learn more in this video), DTW computes the euclidean distance at each frame across every other frames to compute the minimum path that will match the two signals. One downside is that it cannot deal with missing values so you would need to interpolate beforehand if you have missing data points.

XantaCross [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0)]

To compute DTW, we will use the `dtw` Python package which will speed up the calculation.



Here we can see the minimum path shown in the white convex line. In other words, earlier Subject2 data is matched with synchrony of later Subject1 data. The minimum path cost is $d=.33$ which can be compared with that of other signals.

4. Instantaneous phase synchrony.

Lastly, if you have a time series data that you believe may have oscillating properties (e.g. EEG, fMRI), you may also be able to measure instantaneous phase synchrony. This measure also measures moment-to-moment synchrony between two signals. It can be somewhat subjective because you need to filter the data to the wavelength of interest but you might have theoretical reasons for determining such bands. To calculate phase synchrony, we need to extract the phase of the signal which can be done by using the Hilbert transform which splits the signal into its phase and power (learn more about Hilbert transform here). This allows us to assess if two signals are in phase (moving up and down together) or out of phase.

Gonfer at English Wikipedia [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0)]

Filtered time series (top), angle of each signal at each moment in time (middle row), and instantaneous phase synchrony measure (bottom).

The instantaneous phase synchrony measure is a great way to compute moment-to-moment synchrony between two signals without arbitrarily deciding the window size as done in rolling window correlations. If you'd like to know how instantaneous phase synchrony compares to windowed correlations, check out my earlier blog post here.

Conclusion

Here we covered four ways to measure synchrony between time series data: Pearson correlation, time lagged cross correlations, dynamic time warping, and instantaneous phase synchrony. Deciding the synchrony metric will be based on the type of signal you have, the assumptions you have about the data, and your objective in what synchrony information you'd like from the data. Feel free to leave any questions or comments below!

See all the code in a Jupyter Notebook below and use with the sample data available here.

Data Science Synchrony Facial Expressions Signal Processing Python