Teamwork Visualisation

# Overview

This document describes a set of routines to create visualisations that provide an insight into teamwork. The visualisations are based upon data collected through sociometers, making it possible to do an objective assessment of how well a team works together and making it possible to identify possible improvements to how the team interacts. After a review of past literature on the topic, it was decided that two kinds of visualisation will be used:

1. Visualising dynamic complexity through heatmaps, which will aid the identification of key points in time when the dynamics of the team change.
2. Visualising the ‘energy, engagement and exploration’ of teams, which is intended to show the overall quality of communication in the team and show the (im)balance of how members contribute to the team.

These two kinds of visualisation complement each other. Dynamic complexity identifies ‘critical instabilities’ where the dynamics of the group change, and the group behaviour before and after each critical instability is visualised using the visualisations for energy, engagement and exploration.

This document describes the concepts on which the visualisation process is based, as well as technical details of how the visualisation routines are run. Note that the visualisation routines are run from a Jupyter notebook which includes some options to manually adjust parameters.

# Data from the Rhythm badges

The Rhythm badges from MIT are lightweight sociometers that collect audio and proximity data. They will be the primary source of the data to be used in this project.

One thing to note is that there does not appear to be any detailed documentation of the badges, how they work and how they make recordings. This means that it is potentially difficult to obtain the answers to questions about the data that ends up being extracted from them. Nevertheless, enough is understood about the Rhythm badges for them to be used in this project.

The Rhythm badges make two main types of recordings: (1) proximity recordings, in which each badge records what other badges it can detect nearby, and (2) vocal volume recordings, which record the volume of the sound reaching the microphone in averages across 50ms windows of time. Voltage is also recorded, which can help to identify badges that run low on battery, but this has little value for measuring social behaviour. Seemingly, the voltage data has no other purpose. Proximity and audio records are saved by the hub in ‘json’ (JavaScript Object Notation) format, where each record is structured a bit like a set of folders on the laptop.

The proximity data is sent from the badges to the hub (which records the data) as entries per time point per badge, with each entry containing a list of badges that were detected nearby and associated RSSI values. According to a 2018 paper by MIT[[1]](#footnote-1), the badges scan for other nearby badges every 60 seconds using Bluetooth. Each entry in the extracted data includes two values for each of the other badges detected: the RSSI and a ‘count’ value. It is not clear what ‘count’ represents. The proximity data may be useful in future enhancements to the visualisation routines, but currently only the audio data is used. The audio information from Rhythm is described below.

According to the 2018 paper by MIT[[2]](#footnote-2), the audio data comes from microphone recordings at 700Hz split into 50ms sections and averaged. This results in 20 values per second each representing the average volume in a 50ms interval. Every update that the badge sends to the hub contains a list of ‘samples’ stating the average volumes recorded. From inspecting an example extract of the data, it seems that the maximum number of ‘samples’ in a single entry is 114. An average volume of zero has not been witnessed in examples of previously-collected data, raising the question of whether it is a valid value. When less than the maximum number of samples is present in an entry, the badge sends a second (replacement) message for the same time stamp but with a full set of 114 samples.

The data is quite dirty when extracted from the hub. Two issues are:

* Timestamps are neither synchronised for different badges nor consistently spaced out within a single badge. This makes it harder to align the ‘samples’ which also need to be unpacked from the messages that the badge sends to the hub. Please be aware that this means that the recording periods of one badge could typically have an offset of 25ms from another badge (for example). This means that while one badge records two samples of [1,11], the other badge potentially records a sample of [6] (reflecting 25ms of each of the previous badge’s two sampling periods). The consequence of this is that badges may give different samples even if they are at the same location recording the same thing. In practice, this may not be an issue since it will be much less noticeable for fluctuations in volume that do not occur at a rate of 20Hz (which is probable for speech).
* There can be two messages for the same time period, with newer messages superseding the older ones. For example, a first message might include only 40 of the samples for the time period but then a second message arrives with the full 114 samples. Ultimately, there appear to be no gaps in the recordings; messages with fewer than 114 samples seem to always be superseded eventually.

Fortunately, there are existing pre-processing routines provided by MIT that seem able to cope with these issues. The only thing to bear in mind is that the analysis routines from MIT slightly obscure the underlying nature of the data extracted from the hub.

During pre-processing, the audio samples are aligned by the MIT routines such that each sample is ‘rounded down’ to the nearest 50ms time point. So, a sample for the time 16:05:01:0625 would be changed in order to have the time 16:05:01:0500. To do this, the audio data points are assembled into a single Pandas DataFrame and then ‘resampled’ to a rate of 20Hz.

It is worth noting that nonverbal vocal information besides volume is not available through rhythm badges. It is not possible to obtain information about (for example) pitch, voice quality and the use of nonverbal vocalisations (such as ‘ah’ or ‘um’).

## Manual setting of parameters

When running the visualisation routines from the Jupyter notebook, there are various parameters relating to pre-processing that can be manually adjusted.

First, it is necessary to specify the location of the data to be processed. This can be done using absolute or relative paths. An absolute path includes the full address from the root directory, and typically looks like “C:\Users\[…]\audio\_data.txt”. If the data file is in the same folder as the notebook running the visualisations, then a simpler relative path can be used, which typically would look like “./audio\_data.txt”.

The name of the data file will be used to construct a name for the folder where the visualisation outputs will be saved. The output folder name will be a combination of the input file name, the current date and the current time (to a precision of one second).

For the following pre-processing parameters, simple heatmaps of the volume data are plotted after each pre-processing step has been applied. This is to allow for a quick visual inspection of the data, making it easier to identify possible issues with the parameter settings.

Rhythm always saves data with the time zone of GMT/UTC. When pre-processing the data, it may be desirable to convert this to the local time zone that was in effect when making the recordings. The time zone to use is a parameter that is manually set while running the visualisation routines. Standard time zone codes such as “CET” for Central European Time can be used. It is also possible to use time zones with an offset from GMT using codes of the form “Etc/GMT+1” or “Etc/GMT-2”.

If desired, a manual start and/or end point can be provided. This makes it possible to exclude segments of the recording from before or after the experiment.

In the next step, the notebook will print the names of the badges included in the data set. These names can be replaced with more user-friendly names if desired. New names can either be specified as a list (of the form [“A”, “B”, “C”]) or can be provided in a csv file where each badge name in the data set is mapped to a new name.

Badges can be excluded from the data, for example if they stopped recording early on. The list of badges to exclude is provided in the form [“C”, “B”]. Note that any excluded badges will not appear in the visualisations at all, which may be misleading since a meeting of (e.g.) seven people will look like a meeting of six or fewer people.

‘Key points’ can be added to the visualisations. These are points in time that have been noted by the experimenters, and for example could reflect moments at which the task changed or observations made by those present during the recording period. If any key points are provided, they will be added as extra labels when plotting the dynamic complexity heatmap. They can be displayed as times or as text labels in the heatmap. More details on dynamic complexity heatmaps are provided in the relevant section below.

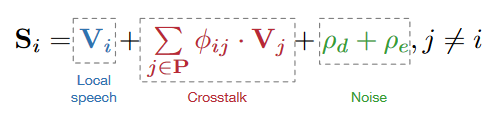
Other parameters exist and are described in the relevant sections below.

# Voice activity detection (VAD)

When processing the audio data, one tricky matter is to figure out who was the speaker. Other speakers might be detected by my badge (aka ‘crosstalk’). The algorithms provided by MIT’s analysis package[[3]](#footnote-3) do ‘multichannel voice activity detection’, in order to determine who is speaking at each moment. Note however that it is not evident that MIT’s method has been tested on anything but a simple, brief and artificial test recording. It is not clear how reliable the VAD method is, or what impact errors at this stage might have on downstream processing steps. Nevertheless, since the algorithm for multichannel VAD was written by the same person who developed Rhythm, is capable of identifying multiple simultaneous speakers, and has had at least some testing on Rhythm, it has been chosen for this project.

The multichannel VAD approach taken by the existing MIT analysis package relies on correlations between the volumes recorded by different badges. The process is applied on one-second frames, meaning that 20 data points are used to calculate correlations. This is a low number when calculating correlations, but is hopefully accurate enough on the majority of frames.

Multichannel VAD is based on the idea that in each frame (of one second) the volume measurements are made up of four components, which are: the badge wearer’s own speech, crosstalk, environmental noise and badge noise. Here is the formula from Oren Lederman’s PhD thesis[[4]](#footnote-4), in which he introduces the method:



Where:

* **Si** is the badge data for the frame for person ***i***
* **Vi** is the voice signal for ***i***
* **φij** is the reduction in signal strength over distance between ***i*** and ***j***
* **ρd** is noise from the device
* **ρe** is noise from the environment

The idea is that when crosstalk is the dominant component for a particular badge in a particular frame, the signals will be highly correlated, but otherwise will on average have a correlation of zero. The algorithm follows a three-step process:

1. For each frame, look at the strongest average volume and then check if all other badges strongly correlate with this badge; if so, then declare the strongest badge to be the only speaker in that given moment. The purpose of this is to find occurrences of one person speaking.
2. Then decide ‘classification rules’ for each speaker by looking at the intersection between (average volume during) frames when the speaker was the only person talking and frames when someone else was the only person talking. Apply this as a threshold for average volume; in each frame, if the speaker is above the threshold then consider then to be talking, otherwise not.
3. Then remove false positives. Go through at each frame and look at the correlations between speakers; if the correlation is high, then remove the quieter speaker.

One problem with the algorithm is that it assumes that if one person speaks, there will be sufficient crosstalk in all the other badges that they are all highly correlated. Future work might seek to validate how well the VAD algorithm works in different scenarios, and compare the outputs of VAD to human annotation.

## Setting of parameters

The multichannel VAD method is based upon a single parameter: the correlation threshold. The algorithm will attempt to automatically find a reasonable value for this parameter. It considers the following correlation thresholds: 0.3, 0.4, 0.5, 0.6 and 0.7. Values less than 0.3 or higher than 0.7 are considered extreme and excluded. To choose between the possible correlation thresholds, it uses two values that are estimated by routines from MIT:

1. The number of moments in which only one person was speaking
2. The KL divergence for volume measurements when comparing speaking moments to non-speaking moments

The KL divergence reflects the degree of difference between the distribution of values, and could be thought of (perhaps imprecisely) as summarising the differences in how frequent each value is. According to documentation from MIT, there needs to be a trade-off between maximising the number of moments with one speaker and minimising the KL divergence. To find a reasonable trade-off, our routines simply multiply one value by the other, and selects the correlation threshold leading to the highest result.

This automatically-chosen threshold can be overridden if desired.

Future work may want to consider gathering additional data to help improve the selection of the correlation threshold and/or improve the VAD algorithm itself.

# Dynamic complexity

The original work on dynamic complexity was by psychologists (called Schiepek and Strunk) who were interested in psychological data. They created dynamic complexity as a way to monitor ongoing therapy, and identify significant points in the evolution of the patient’s condition. Dynamic complexity is useful for identifying significant moments in the evolution of a phenomenon because it can be used to find critical instabilities, which are sometimes also referred to as critical fluctuations or phase transitions. These critical instabilities are a good indicator of a change from one pattern of organisation to another, such as when water shifts from liquid to vapor, there is a shift in quadrupedal locomotion from walking to trotting, or when there is a change in group interaction processes.

The dynamic complexity measure applies to time series and is calculated within a moving window, quantifying the complexity exhibited by the system over time. It is similar to a score for how erratic the observations are in the data. Observations that keep on increasing over time (or decreasing) are not considered complex, nor are observations nearby in time that have similar values. Instead, periods containing moment-to-moments changes from increases to decreases and a wide range of values are considered complex.

Once dynamic complexity values have been computed for a time series, it is possible to search for moments of unusually high dynamic complexity, which are likely to happen at the same time as a critical instability and an associated phase transition – in other words, the points at which we expect to have a change in the dynamics of the interaction between team members.

Although this is not necessary in order to understand the teamwork visualisations, it might be worthwhile to explain some key ideas from the technical background of dynamic complexity. It arises from the theory of dynamical systems, a mathematical method for describing systems that evolve over time according to a set of interdependent rules. Dynamical systems can notably have ‘attractors’, which can be thought of as patterns in the evolution of the system: they consist of a reduced set of evolutions (compared to what is theoretically possible within the system) and once the system enters an attractor state, it continues to stay within the attractor. For any combination of parameters describing the dynamical system, the attractors that exist within the system are determinate; however, if the parameters of the system change then it is possible for the attractors to also change. It is expected that most small changes to the parameters of a system will only result in small modifications of the attractors, but sometimes it is possible that the attractors will change drastically. Points at which changing parameters cause considerable changes to attractors (possibly removing or creating attractors in the system) are called ‘critical instabilities’. The dynamic complexity measure is intended to make it easier to identify critical points – moments of change at which the expected patterns of behaviour before are different from the expected patterns of behaviour after.

Note that other methods for identifying critical instabilities exist, but they require much longer sequences of data. Dynamic complexity is attractive because it can be calculated on shorter/more coarse-grained data.

Dynamic complexity is calculated by combining two components called the 'Fluctuation' and the 'Distribution', both of which are also calculated over a window of values. Both values range from 0 to 1.

* *Fluctuation* (F) is based on changes in value between ‘points of return’ where the gradient stops being positive, zero, or negative (i.e., looking in between these points, we see periods when the value is consecutively increasing, staying the same, or decreasing). For each period between points of return, the difference between values at the beginning and end of the period is divided by the number of time points included in the period, and the outcomes are summed across the entire window. The result is then divided by a ‘maximum possible fluctuation’ for the window size, which is what would happen if there were only oscillation between the maximum and minimum values. F is high when there are frequent oscillations between high and low values, and is lower when oscillations are less frequent or have smaller amplitude.
* *Distribution* (D) compares how regular the differences are between sorted values in a window. First, the values in the window are sorted in ascending order. Then, the sorted observed values are compared to artificial series of values that were all evenly-spaced across the possible measurement scale. The disparity in the differences (e.g., in the differences between the first and third values within each sorted list) is the basis for D. Positive disparities are summed. This addition continues to be calculated across the whole window for different sub-window sizes. So, first, the differences at an offset of 1 (e.g., first and second sorted values) will be considered, followed by differences at an offset of 2 (e.g., first and third sorted values). Before being added into the summation, each difference is normalised by dividing it by the evenly-spaced expectation of what the difference should be. D is highest when there is an equal distribution of values, and it decreases when there is a preference for certain values or certain ranges of values.

Dynamic complexity within a given window is simply the product of the F and D measures.

The visualisation process calculates dynamic complexity upon time series for each participant. The Rhythm Badges themselves produce a time series in which each data point is the average volume recorded over a 50ms period. This series is resampled to a rate of 0.2 Hz and the average is taken within each 5s period. This is done for two reasons: first, it reduces the size of the time series and so lessens the computational demand of the process, and second, it lowers the impact of fluctuations in volume that occur naturally in speech (for example due to pauses for breath) that are incidental and not indicative of changes in team dynamics. To calculate the dynamic complexity, a sliding window size of 12 is used with a increment of 1, which corresponds to a one-minute period.

Before moving on, there are a few things to consider, although these hopefully will not have any practical consequence.

First, past research does not give many details on how robust the dynamic complexity measure is against noise or when using measurements that naturally fluctuate rapidly. The original usage was for repeated psychological assessment through questionnaires, where the factors measured (like general happiness) are perhaps expected to be fairly consistent from day to day.

Second, Rhythm audio data can have periods of low-to-zero values, during which any change in complexity has a chance of being considered significant. By contrast, psychological ratings are unlikely to have long periods of zero values. This is one motivation for excluding periods at the beginning or end of a recording when the group was not interacting.

Third, dynamic complexity is reliant upon there being a known range of possible values. However, none of the documentation for Rhythm suggests a theoretical maximum value. Therefore, our algorithm infers the minimum and maximum volume values from the data (by taking the minimum and maximum values observed in the recording period), and does so individually for each badge. So far, this has not caused problems on the data sets for which the visualisation routines have been tested.

# Critical instabilities

The next step in the visualisation routines is to calculate the average dynamic complexity across the time series for all team members. This results in a single series of dynamic complexity values from which critical instabilities can be identified by moments of high complexity. Since local maxima of complexity are indicative of critical instabilities, we emphasise local maxima over the global maximum. We use a moving window within which dynamic complexity values larger than two standard deviations from the mean for that given window will be identified as points of critical instability. For this purpose, we use a window size of 60, which corresponds to the complexity scores encountered over a five-minute period.

Looking between critical instabilities, the communication patterns between team members are depicted in network visualisations.

# Energy and engagement

After the identification of critical instabilities, teamwork is further visualised using two measures that capture different communication patterns, and that not only reflect the quality of the social interaction, but that also make it easier to understand the team dynamics in a visual manner. Energy and engagement are measures of the quality of social interaction in teams, described by Alex ‘Sandy’ Pentland in an article for the Harvard Business Review[[5]](#footnote-5). That article posited that high-performing teams could be identified by certain characteristics of their communication patterns. It has not been possible to find other publications on the topic. The lack of published details may be because these techniques for quantifying social behaviour are being used by Humanyze, a company founded by PhD students of Pentland (that Pentland himself is also is involved in), and they wish to keep their intellectual property private. Energy and engagement are appealing measures, since they have already been developed by experts and used in a commercial setting.

Because precise details of how the measures were calculated do not appear to be available, the visualisation process is inspired by, rather than an exact match to the measures presented by Pentland and colleagues.

The first measure is the *individual utterance rate*, which will also be called *energy* in this document even though our implementation may differ from Pentland’s original work. It reflects how much each individual is contributing to the overall communication for a given time window. Any continuous period of speaking, as identified by VAD, is considered an utterance. The rate is the number of utterances started divided by the duration of the time period (in seconds). In the visualisations, each team member is represented by a node and their energy (relative to other team members) is depicted through the size of the node.

The second measure used is the *pairwise rate of responses between two individuals*, which will also be called *engagement* in this document even though our implementation may differ from Pentland’s original work. A response is considered to occur when the first individual begins an utterance within five seconds of the second individual speaking. The rate is the number of times this occurs in the time period divided by the length of the time period. This measure gives an indication of which team members are interacting with which other team members, and which people are involved in the same discussions. In the network visualizations, the rate of responses is portrayed as the thickness of the edge connecting two nodes.

Note that our routines consider a response to occur when one person speaks within 5 seconds of another person speaking. Since details of previous implementations are not available, 5 seconds was chosen as a seemingly-reasonable value. However, it may be desirable to try other values in future work. The pairwise rate of responses is undirected/symmetrical, so it does not matter if person A responds to B or B responds to A.

One thing to consider when using large groups of people is that the routines currently do not attempt to identify which individuals were in the same conversations before calculating the rate of responses. If multiple conversations occur at the same time, then an individual involved in one conversation may coincidentally speak after an individual in another conversation; this would increase the engagement between them even though in reality they were not conversing. Future work may consider whether this is an issue, and if so how to improve the calculations.

# Plotting the visualisations

Before plotting the dynamic complexity heatmap, the algorithm attempts to find an appropriate resample rate to apply to the results so that they display correctly. This is to avoid there being too many cells to display in the heatmap, which would make the heatmap difficult to read or even obscure valuable information. The algorithm aims to find a nice, round period (ranging from 5 seconds to 5 minutes) that produces close to 200 observations. This makes the heatmap more readable and solves some potential display issues. The heatmap shows the average dynamic complexity values within each period of the resampling.

Key points, reflecting important moments in time that were manually recorded (rather than automatically identified as critical instabilities), will be added to the heatmap if they were defined during the manual setting of parameters. These appear as extra labels along the top of the heatmap, but otherwise have no function.

The network visualisations are produced based on the data that appears in between critical instabilities. Each one visualises energy as the size of nodes and engagement as the thickness of edges connecting two nodes. The outputs are saved in SVG format. Note that the period between critical points will not be visualised if it is less than one minute in duration; this is (1) because in such cases it is not clear that the period of critical instability actually ended, and (2) because there would be an insufficient period of time from which to reasonably calculate energy and engagement.

There are two ways to determine the scale (what visual size is used to represent each numeric value) used in the network visualisations. The first is to give numeric values a fixed size in the visualisations, so that (for example) an energy of 0.10 is always displayed as the same node width, no matter what dataset is being used. The second, ‘relative’ sizing approach is to decide what node width represents an energy of (for example) 0.10 by using the data to find an appropriate width. This approach starts by inferring the maximum energy that needs to be displayed across all network visualisations, and represents this number as a maximum node width that never varies. This approach means that the largest node in a series of visualisations will always be the same size, regardless of what data set is used. The sizes of other elements will be chosen relative to the maximum, so that (for example) an energy of 0.05 will be represented through a node half as wide as the maximum if the maximum was 0.10. By default, the ‘relative’ approach is used in the algorithm, since it more reliably produces readable visualisations. The only drawback to the relative approach is that it is not possible to compare visualisations produced from different data sets, since the maximum value will be inferred independently for each. The alternative, ‘absolute’ sizing can optionally be used instead, but due to a lack of data sets on which to test, it is not certain that the approach will work well on future data and produce readable visualisations. Note that even when using the absolute approach, it may be problematic to compare visualisations produced from different data sets. This is because differences in seating positions, number of participants and VAD (for example a different correlation threshold) might affect the results in unknown ways.

Note that alongside visualising the energy and engagement for each period between critical instabilities, the output will save the numeric values of energy and engagement to csv files.

1. https://dam-prod.media.mit.edu/x/2018/02/05/Rhythm%202.pdf [↑](#footnote-ref-1)
2. <https://dam-prod.media.mit.edu/x/2018/02/05/Rhythm%202.pdf> [↑](#footnote-ref-2)
3. <https://github.com/HumanDynamics/openbadge-analysis-examples/blob/master/notebooks/multi-channel_VAD_illustration.ipynb> [↑](#footnote-ref-3)
4. <https://dspace.mit.edu/handle/1721.1/123566> [↑](#footnote-ref-4)
5. <https://hbr.org/2012/04/the-new-science-of-building-great-teams> [↑](#footnote-ref-5)