Is Drumming Fractal?

Supervisor – Philip Moriarty
Project Number - 99

Oliver Gordon - 4224942
F33PJU

Partner – Dominic Coy
Due – 17/05/2017

Abstract

The act of playing the drums is repeated with a correlated degree of randomness that is fractal and statistically self-similar. This self-similarity is quantified by the Hurst exponent, α. These fractal variations are yet to be implemented in music generation software, resulting in music that sounds "robotic".
Whilst listening to "Tom Sawyer" by Rush, 22 drummers play semiquavers on a closed hi-hat. This is performed with one hand, two hands, and then in an unaccented, metronomic fashion. To calculate α, Detrended Fluctuation Analysis is performed on the variations between the times when the drums are hit.
One aim of this experiment is to verify if drumming is fractal. The other is to investigate the claim that two values of α can be universally seen in music.
We find the mean value of α to be 0.52 ± 0.10, corresponding to non-fractal noise. However, the large range, 0.44, and error, 0.10, suggest otherwise. It is therefore difficult to confirm if drumming is fractal. Because 84.84% of graphs produced are straight, we also find that the production of two values of α is unlikely to be universal. Playing experience and technique are found to have no effect on this finding.

Acknowledgements

Cover Image: *Courtesy of Brittany Bednarik.*
University of Nottingham: *Jonathan Gosling, Andy McKeown, Emma Stafford.*
Nottingham Trent University: *Molly Parker, Cameron, Sam Croft, Harvey Neizer-arku, Lili Popper, George Hadden, Janice Chan, Bua Nateejarurat.*
Staffordshire University: *Aaron Smith, Steve Barwell, Ben Miles, Ben Beilby, Aaron Dolton, Ashley Matthews, Dave Hickman, Paul Hodson.*
Saturday Music School & Leeds College of Music: *Karen Gourlay, Connor Small, Nathan Birch, Ellie M, Herbie Allen, Tom Gibbins, Matt Martin, Louis Tissiman, Tom Bortlik, Eliza Wheatley.*

Contents

1 Introduction

Although computers can repeat given tasks perfectly, human actions are performed with an inherent degree of variability for reasons that are mostly obvious. One may also be tempted to expect that these variations are independent and can be thought of as random white noise. However, it has instead been found that these variations are fractal and contain long-range correlations (LRCs)[1]. This phenomenon can be observed in a variety of places ranging from music[2] and painting[3] to walking[4], band gap materials[5] and even the layout of Japanese gardens[6].

The relationship between LRCs and music was first proposed in 1975 by Voss and Clarke[1]. Before this, a variable in a system was originally thought to be fractal if it was inversely proportional to signal frequency, f. This would produce an equation of the form[7] $1/f$. Voss and Clarke, however, found that several aspects of music fit the form $1/f^{\alpha}$, where α corresponds to the Hurst exponent. This was itself proposed by Harold Hurst in 1951 to quantify the self-affinity of a system, otherwise thought of as its long-term memory[8].

This $1/f^{\alpha}$ noise has been demonstrated during recent studies[9][10]. Participants were asked to listen to drum patterns generated with no variations, white noise variations and fractal variations in timing. Subjects showed an overwhelming preference for the fractal versions of the patterns[10]. Some even went as far as to describe them as more human[9]. As such, there are clear implications in using fractal patterns and LRCs to humanise computer generated music.

Modern commercial drumming software, however, currently only generates white noise variations in the times when drums are hit[10], or 'onsets'. This is likely because the nature in which they present themselves in music is not well understood. A greater understanding of LRCs could lead to their implementation in such software. This would result in a more pleasurable listening experience.

There have been many recent attempts to better understand the potential correlation between α and various components of music. One attempt regards the coupling together of musicians performing together, and the subsequent affect on α[9]. Other potential links include the effect of genre[11][12], melody[13] and different instruments[14]. However, one promising example of interest regards a recent study by Esa Räsänen et al. into the hi-hats played in Michael McDonald's "I Keep Forgettin'"[2][15]. By using detrended fluctuation analysis (DFA)[16], it was found that for the variation in onset times, $\alpha = 0.31$ and $\alpha = 0.72$. It was proposed that this double-barrelled result may be universal when a musician tries to drum with a "groove". This result is reproduced in Figure 1. Here, a groove is not referred to in a purely musical sense, but instead in the sense of playing in a more human and varied fashion.

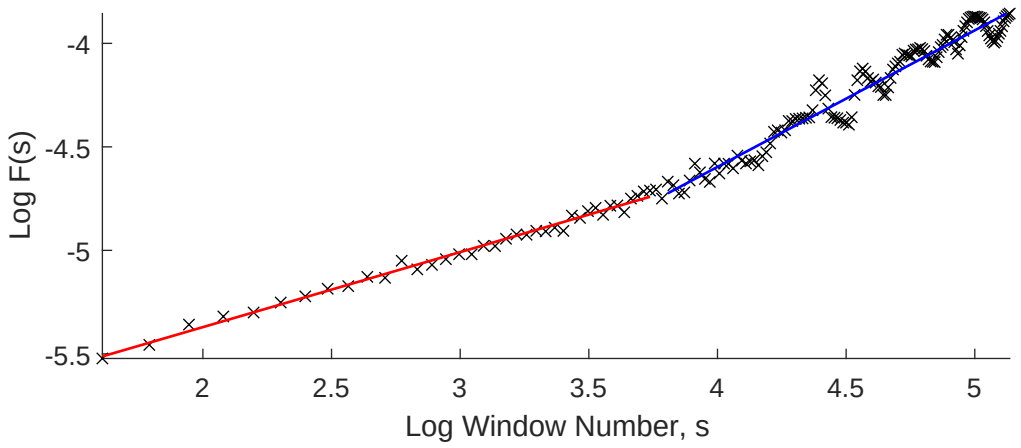


Figure 1: Figure to reproduce work by Esa Räsänen et al. during their analysis of the hi-hats during Michael McDonald's "I Keep Forgettin'". Here, Detrended Fluctuation Analysis was performed on the variations in times when the drums were hit. This allowed them to calculate the Hurst exponent, α, which measures the self-similarity of the data. Two gradients were found, with $\alpha = 0.31$ for the red line, and $\alpha = 0.72$ for the blue. They proposed that this "double-barrelled" shaped graph may be universal in music. Data courtesy of Räsänen et al.[2].

Given that this study involved only one piece of music, it is possible that this double-barrelled nature may not be fully universal. It is also reasonable to hypothesise that the Hurst exponent may vary with different drumming techniques and levels of talent, as this will change the groove of the drumming. This change in groove is suggested by the drummer in "I Keep Forgettin'", Jeff Porcaro, who noted that *"I tried doing the alternating stroke method, and it sounded just too stiff and staccato [short and separated] for me."*[17]

As α is thought to be affected by how a musician plays it is therefore affected by the groove of the performance. However, many studies into music and α often included metronomes[2], or were based on recordings with multiple instruments. Metronomes have been shown to affect the $1/f$ behaviour[18] of a sample. This is because the musician is forced to play with a certain drift and groove, suppressing the drift of onsets and thus changing α. This may partly explain why such studies have found a wide range of Hurst exponents between[2,19,20] 0 and 1. The piece of interest in this study, "Tom Sawyer", by Rush[21], was not recorded with a metronome and therefore does not suffer from this drawback.

In this project, 22 musicians of differing talents are recorded while playing semiquavers along to "Tom Sawyer" [21] on the closed hi-hats of a drum kit. This is then repeated for single handed and double handed drumming techniques, before playing in as metronomic and unaccented a fashion as possible. A fully automated process developed in MATLAB then detects onset times by calculating the power spectral density of samples as a function of time. DFA is then performed on the variation between onset times to calculate the Hurst exponent.

The main aim of this report includes demonstrating that drumming is a fractal process through the calculation of the Hurst exponent, α. The potentially double-barrelled nature of α suggested by Räsänen et al.[2] is also investigated. Both of these metrics are also analysed as functions of ability, playing technique, and the presence of a groove in the performance. Finally, weaknesses in Räsänen et al.'s work are briefly discussed.

2 Theory & Methods

2.1 Recording & Pre-Processing

In "Tom Sawyer", each bar consists of four beats. These four beats can be broken down into four crotchets or 16 semiquavers, amongst other musical parts. To record samples, musicians are asked to play these semiquavers on the closed hi-hat of a drum kit whilst simultaneously listening to the song. This is done for a minimum of two minutes. This minimum time is required to give each performance sufficient time to evolve and generate different patterns in volume and onset times. It is also needed as enough data must be produced for the DFA algorithm to form a sufficient comparison against. A short performance would therefore reduce the effectiveness of the DFA algorithm and the final results.

Each performance of semiquavers is then repeated once with a double-handed method, once with a single-handed method, and once with a technique of their choice but in as metronomic and unaccented a fashion as possible. This is done with their own sticks and as if they are giving a live performance, instead of just trying to keep time. This ensures that they are playing and grooving in a style that they prefer, which better reflects a real-world performance.

Immediately before hearing "Tom Sawyer", eight crotchets are played to each musician through a click track. After hearing four of these crotchets, each musician then plays four crotchets, with one on each beat. This is done to ensure that each musician begins in time with the music and not have to catch up or slow down to the beat, affecting results. Also, it provides an obvious starting location to the onset detection algorithm, which will be discussed later.

To avoid the false detection of onsets due to other instruments or noises, recordings are done in a quiet room free of significant background noise and reverberation, with the microphone facing the hi-hats. The microphone is also calibrated to avoid peaking out and making the detection of onsets difficult. Each sample is recorded in mono at a sampling frequency of 44.1 kHz, and then saved in the uncompressed 32-bit .wav file format.

The output frequency range of the hi-hat is of great use when detecting the onset times when the hi-hats are hit. This is as not only is the frequency range of the hi-hat well above the range of human voice of 85-255 Hz[22][23], but it is also well-defined and temporally short[2]. These properties make it easier to isolate hi-hats from surrounding noises. To isolate the hi-hats, a 100^{th} order FIR bandpass filter is applied for frequencies between 0.6-16 kHz on the waveforms of samples collected. This produces waveforms such as those shown in Figure 2.

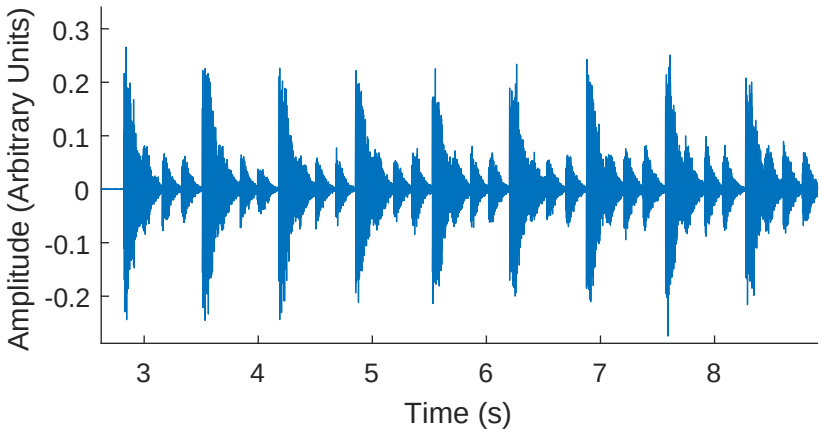


Figure 2: Figure to demonstrate a typical filtered waveform. Musicians were asked to play semiquavers on a closed hi-hat along to "Tom Sawyer" by Rush. Samples were recorded at 44.1 kHz, before being filtered with a 100^{th} order FIR filter between 0.6-16 kHz. Sharp changes in amplitude correspond to the hitting of the drums, known as 'onsets'.

2.2 Onset Detection

For obvious reasons, it is difficult to both accurately and consistently determine onsets by detecting where the amplitude of a waveform begins to increase. As the hi-hat is an energetic instrument which instantaneously produces noise when hit, each onset is represented by a sharp increase in the power spectral density (PSD) of the waveform. The PSD of the waveform is therefore used instead to determine onsets.

To calculate the PSD of a waveform, a spectrogram must first be produced. A spectrogram visually demonstrates the power spectrum present in a signal as a function of both frequency and time[24]. The window size used to calculate the spectrogram effectively acts as its resolution. A smaller window size has greater temporal resolution but smaller frequency resolution and vice versa. As a compromise between computing time and temporal clarity, a window size of 64 is used. The intensity of the spectrogram represents the power of a small frequency range during a small range of time. A hi-hat onset would therefore be observed as a very intense narrow vertical line. A typical spectrogram produced is demonstrated in Figure 3.

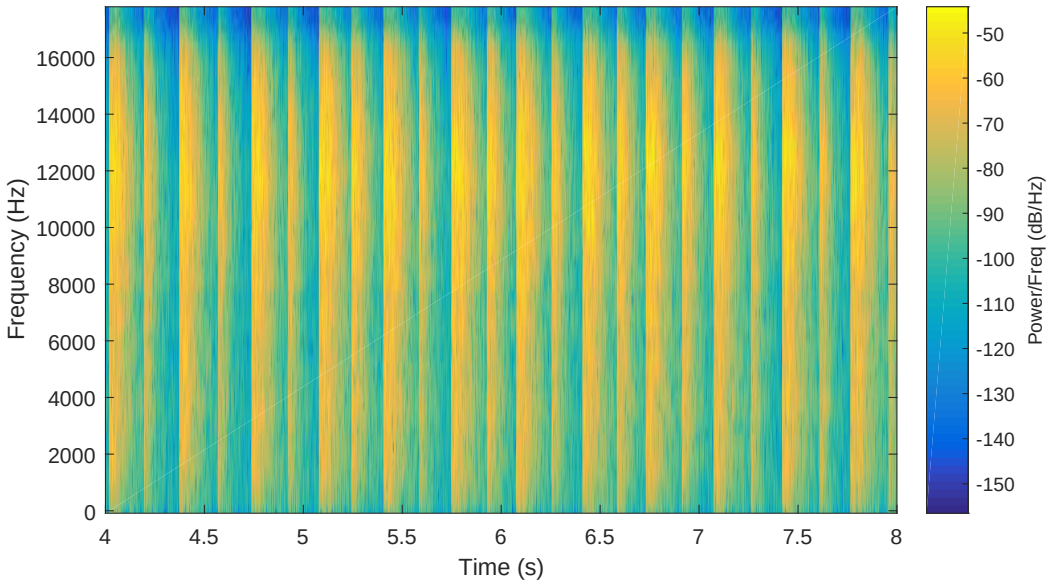


Figure 3: Figure to demonstrate a typical spectrogram produced by a musician playing semiquaver hi-hats along to "Tom Sawyer" by Rush. Sharp sections of yellow correspond to high power at a particular frequency and time. These therefore correspond to times when the hi-hats were hit. The sharp bright yellow region trails off in intensity at frequencies where the hi-hats do not produce sound. As such, the data corresponding to frequencies above 14 kHz is removed before extracting the power spectral density of the data.

After spectrograms are produced for each sample, a hard low-pass filter of 14 kHz is then applied. This is chosen arbitrarily such that only noise clearly generated by the hi-hats will be analysed. The PSDs are then extracted as a function of time from each spectrogram and averaged over all frequencies as a function of time. An example graph of PSD is shown in Figure 4.

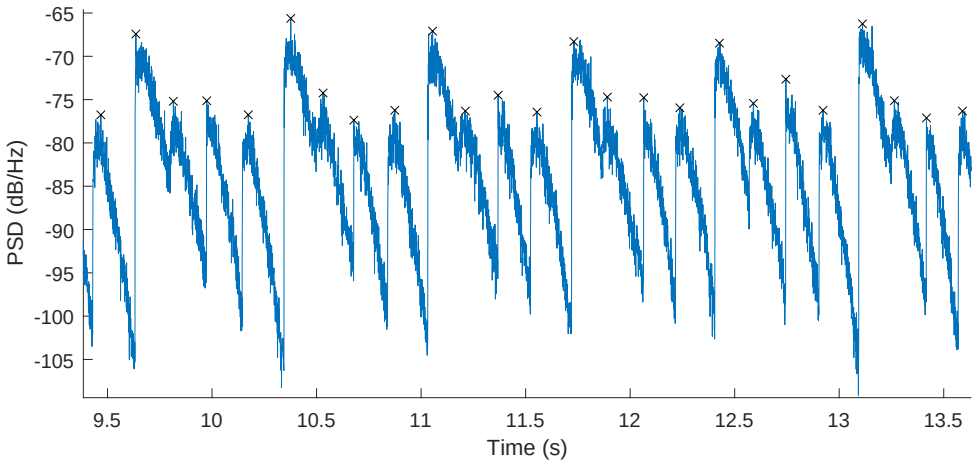


Figure 4: Figure to demonstrate a typical graph of power spectral density (PSD) produced by a musician playing semiquavers on the hi-hats along to "Tom Sawyer", by Rush. The times when the drums are hit are known as onsets. These are marked with a black cross. This differs from previous definitions, which define onsets as the times corresponding to minima in the PSD. This is done as some onsets begin whilst the PSD was still decaying, making detection difficult and erroneous. Examples of this issue can be seen in the third, seventh and eleventh onsets. As the time to move from minima to maxima is the same, onsets were instead defined as maxima in the PSD.

Here, onset times are defined as the times which correspond to maxima in the PSD. This differs to the method employed by Räsänen et al., who defined onset times as the times which corresponded to minima in the PSD[2]. This alternative definition is used as some onsets begin whilst the PSD was still decaying after a previous onset. This results in no well-defined minima in PSD being present even when maxima are clearly visible. This is demonstrated in Figure 4. Using Räsänen et al.'s definition would therefore cause some onsets to be missed, or inaccurately detected. Given the assumption that a drum kit will deform consistently, the time period between maxima and actual minima at each onset will not change. As such, the time difference between each onset, τ, will therefore not change, making this redefinition valid.

To detect onset times, a simple peak detection algorithm is first used to detect the first five peaks in PSD. This is achieved by detecting times where the PSD rose by the arbitrarily chosen value of 100 dB/Hz. The first four of these peaks correspond to the onset times of the four crotchets played before the piece began. The fifth peak corresponds to the time of the first semiquaver played, $t(i = 1)$. Here, i is defined as the number of the onset detected, and $t(i)$ the i^{th} onset time.

A windowing based method is then used to detect the remaining onsets. To find the $i + 1^{th}$ onset, the expected time between onsets, t_{exp}, is first calculated with the equation

$$t_{exp} = 60/bpm/4, \qquad \{1\}$$

where bpm is the number of beats per minute of the piece. As the recording of "Tom Sawyer" used was originally played at 88 beats per minute[25], $t_{exp} = 0.1704s$ for this experiment. t_{exp} is then added to $t(i)$. A window is then formed within a range[2] of ± 34 ms. This creates a small time window in which an onset is expected. The highest peak value of PSD in this time window is then found through simple peak detection and the time recorded as $t(i+1)$. A flow diagram for this code is shown in Figure 5.

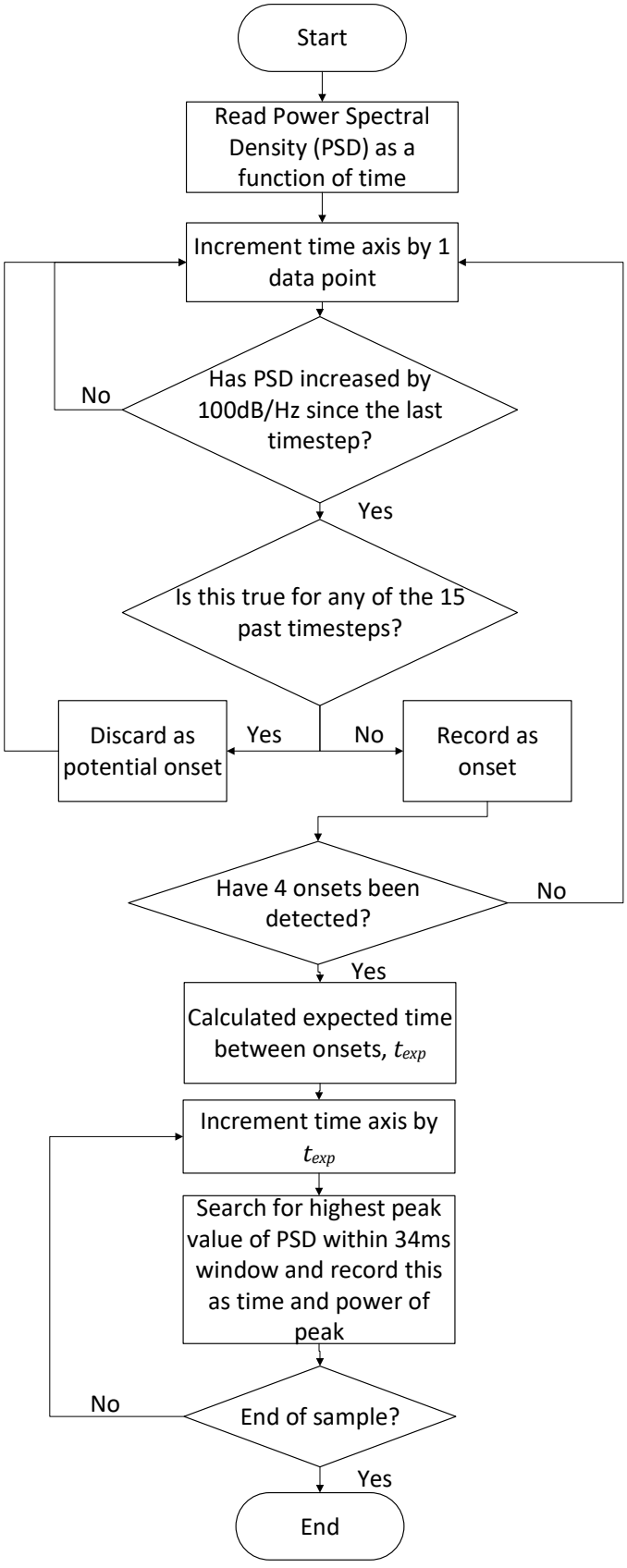


Figure 5: Flowchart to demonstrate the algorithm used to detect onset times when a drum is hit.

Next, the time intervals between onsets, τ, is calculated using the equation

$$\tau(i) = t(i+1) - t(i). \quad \{2\}$$

Erroneous time intervals of $\tau > 0.034$ seconds[2] are then removed, as these time differences correspond to missed onsets. This leaves a final number of N detected onset intervals. The number of intervals expected is calculated by dividing the time in which semiquavers are played by t_{exp}. This is then used with N to calculate the percentage of onsets detected.

2.3 Hurst Exponent & Detrended Fluctuation Analysis

In its simplest form, detrended fluctuation analysis (DFA) measures the self-affinity of a system that evolves over time[16 26 27]. This is then quantified with the Hurst exponent, α. If variations are correlated and fractal, any subsection of this data of any size will look statistically identical to any other subsection of the data. This corresponds to a Hurst exponent of $0.5 < \alpha \leq 1.5$. If the data set is anticorrelated and fractal, $-0.5 \leq \alpha < 0.5$. Alternatively, if the data is uncorrelated, $\alpha = 0.5$. This also corresponds to white noise[8], indicating that long-range correlations are not present and that the system is not fractal. Alternatively, a perfect metronome with no variation in onset times has $\alpha = 0.$[8]. The variations in time when playing music have been shown to be correlated and fractal, with a Hurst exponent of between $0 < \alpha < 0.5$ and $0.5 < \alpha < 1$[19].

DFA can also be performed to various dimensional orders to improve the accuracy of the result[27]. However, in the context of music this has been seen to show qualitatively little difference[2], so only linear detrending is calculated here. The same method can be employed to perform DFA on either the variations in onset times or PSD. However, here we are only interested in the variations in onset times.

To perform DFA on τ, the cumulative fluctuations from the mean, Δy are calculated using the equation

$$\Delta y(i) = \sum_{j=1}^{i} (\tau(j) - \langle\tau\rangle), \quad \{3\}$$

where $\langle\tau\rangle$ is the mean onset interval. This equation is repeated for $1 \leq i \leq N$. It is also possible to use Equation 3 to calculate the drift of τ from the mean, but with $\langle\tau\rangle \to i\langle\tau\rangle$.

Next, the N values of τ are split up into windows of equal size, s, along the i axis of τ. This forms a total of N/s windows. If there are not enough data points at the end of the i axis to form a complete window, these points are discarded. This leaves an integer number of windows. A linear equation is then fitted to each window, and the fitted amplitudes of τ, y_s, recorded. A sample result for $s = 100$ is demonstrated in Figure 6. This is repeated for integer s between $5 \leq s \leq 150$. A minimum window size of $s = 5$ is required as window sizes of $s < 5$ are too small with which to measure the self-similarity of. Similarly, at $s > 150$ there are not enough data points to form a sufficient number of windows.

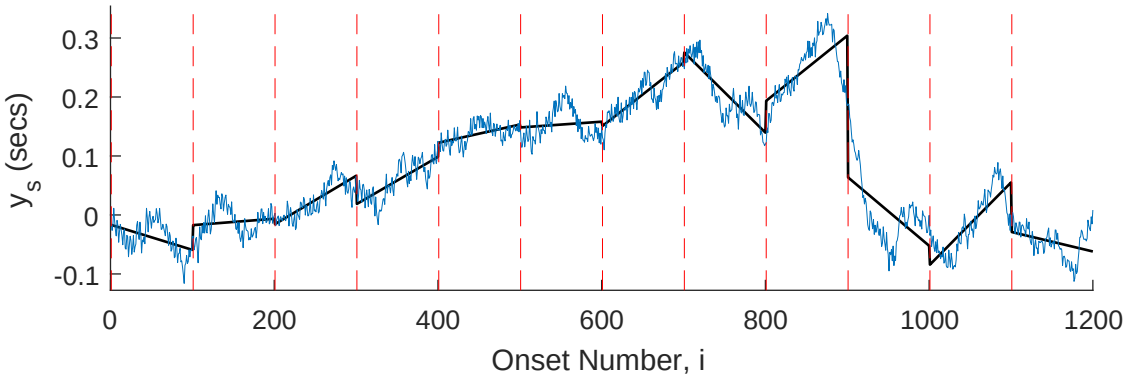


Figure 6: Figure to demonstrate the first step of the DFA algorithm. For an individual sample of data, the cumulative sum of onset differences is plotted in blue. An integer number of windows are then created along the i axis of the sample. Here, edges of windows of size s=100 are shown in red vertical dashes. A linear fit, shown in black is then taken for each window. This is repeated for windows of size s=5 to s=150. The values of the y axis for this black line are then recorded as ys(i).

From here, the root-mean-square fluctuations for each window, F_k, are calculated with the equation

$$F_k(s) = \sqrt{\frac{1}{s}\sum_{i=ks+1}^{ks+s}[y(i) - y_s(i)]^2}, \qquad \{4\}$$

where k is an integer. This equation is then repeated for integer k between $0 \leq k \leq \frac{N}{s} - 1$ and $5 \leq s \leq 150$. This will produce N/s values of $F_k(s)$ for each value of k.

Finally, a mean is taken along the k axis for each of these N/s elements, $F_k(s)$, with the equation

$$F(s) = \langle F_k(s) \rangle, \qquad \{5\}$$

where $F(s)$ relates the self-similarity of the windows as a function of window size. A flow diagram of this process is shown in Figure 7. With the above algorithms, a sample can be fully analysed in under ten seconds.

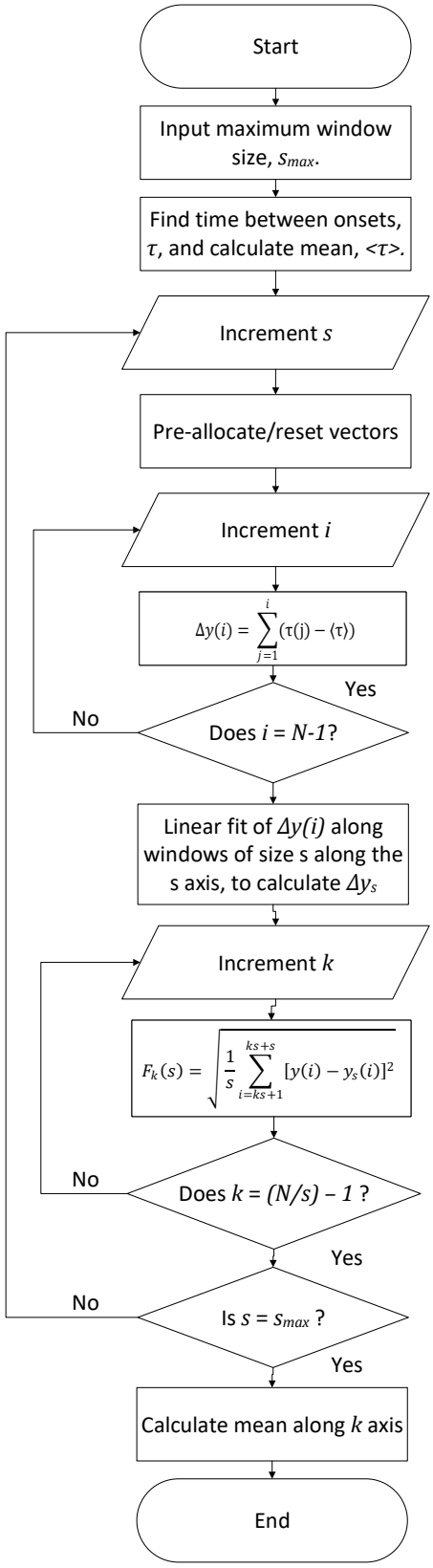
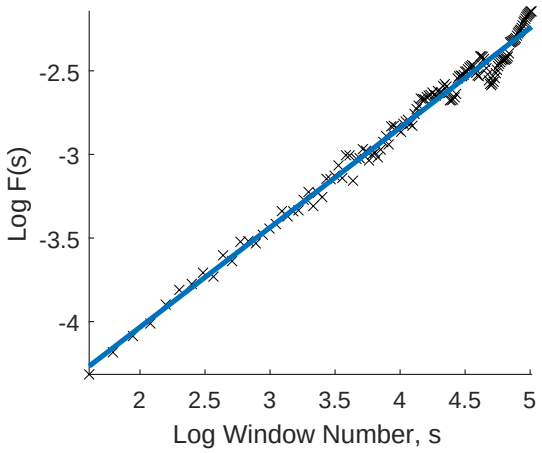
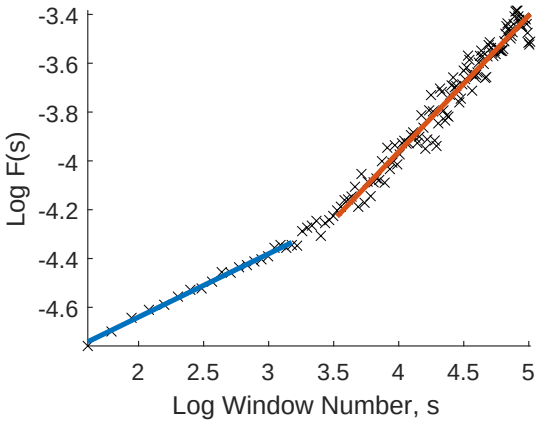


Figure 7: Flowchart to demonstrate the algorithm used to perform detrended fluctuation (DFA) on the differences in times when a drum is hit. The output, $F(s)$ can be plotted as a function of the window size input into the algorithm, s. The gradient of the logs of these values will be equal to the Hurst exponent, α, which quantifies the self-similarity of a system.

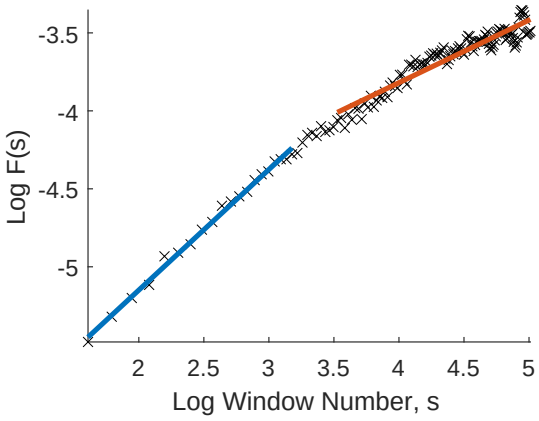
As[2] $F(s) \propto s^{\alpha}$, α is calculated by fitting a straight line to a graph of $log(F(s))$ against $log(s)$. To automatically classify graphs as straight or double-barrelled, the gradients between $5 \leq s \leq 20$ and $30 \leq s \leq 150$ are first calculated through linear curve fitting. This range is selected as double-barrelled graphs consistently change gradient in the range $20 < s < 30$. From here, errors due to curve fitting are added to the shallowest gradient and subtracted from the steepest one, and the difference taken. If this difference is smaller than 0.1, a graph is classified as straight. This is chosen arbitrarily to allow for consistent results free from human bias. However, if the difference is greater than 0.1, the graph is classified as double-barrelled. This is the same behaviour as seen during Räsänens analysis of "I Keep Forgettin'". double-barrelled graphs are then classified as forward double-barrelled if the first gradient is smaller than the second. If the first gradient is greater than the second, they are otherwise classified as reverse double-barrelled. Examples of each of these graphs are demonstrated in Figure 8.



(a) Straight Graph



(b) Forward Double-Barrelled Graph



(c) Reverse Double-Barrelled Graph

Figure 8: Figure to demonstrate the variety of graphs produced by the DFA algorithm on the variation in times when a closed hi-hat was hit. Graph (a) is considered to be straight, as it has a single line of best fit, shown in blue. (b) is considered to be "forward" double-barrelled, as it can be fit with two straight lines. The first line, shown in blue, has a smaller gradient than that of second red line. (c) is the exact opposite to (b), and is classified as being "reverse" double-barrelled. To automatically classify graphs, two lines are curve fit to the dataset, and associated errors taken. One error is added to the shallowest line, and the other added to the steepest line. The difference between these numbers is then calculated. If it is greater than 0.1, the graph is reverse double-barrelled. If it is less than 0.1, a graph is forward double-barrelled. All other graphs are considered to be straight.

3 Results & Discussion

3.1 Reproducing Double-Barrelled Results

To determine if the double-barrelled result produced by Räsänen et al. can be reproduced, the number of samples producing double-barrelled and straight graphs are recorded. This is then investigated as a function of both playing experience and method. Single handed and double handed methods are also combined and compared against the metronomic method. This allows it to be seen if the act of grooving with music affects the production of double-barrelled graphs.

3.1.1 Experience

Firstly, the relationship between experience and the shape of graphs is explored. Metronomic data is excluded as it involves unrealistic performances. Musicians are split up into one of four groups based on their experience. These groups are for years of experience, E, between $0 \leq E \leq 5$, $5 < E \leq 10$, $10 < E \leq 15$, and $E > 15$. There are 8,4,5 and 5 musicians in each category, respectively. The percentages of samples producing straight or double-barrelled results are then recorded. These results are shown in Table 1.

Table 1: Table to investigate the production of double-barrelled graphs produced by the DFA algorithm as a function of playing experience. DFA is performed on the variation between onset times of a closed hi-hat. There are 44 samples, based off the results from 22 musicians playing semiquavers along to "Tom Sawyer" by Rush. The shape of graphs produced are classified as straight, forward double-barrelled or reverse double-barrelled. Data is then filtered by grouping musicians by the number of years they have been playing drums for. It is found that a majority of results were straight, indicating that the double-barrelled result produced by Esa Räsänen et al. is not universal and that experience does not affect the production of double-barrelled graphs.

	Straight	Forward Double-Barrelled	Reverse Double-Barrelled
$0 \leq E \leq 5$	87.50%	12.50%	0.00%
$5 < E \leq 10$	87.50%	0.00%	12.50%
$10 < E \leq 15$	60.00%	10.00%	30.00%
$E > 15$	100.00%	0.00%	0.00%

Here, it can be clearly seen that an overwhelming number of samples are straight, irrespective of experience. There are also a similar number of forward and reverse double-barrelled results in each case. It is also difficult to see any sort of trend in the number of these graphs between experience groups. However, there is a far smaller majority of straight graphs in the range $5 < E \leq 10$. This could be explained by the limited data set used, with only 5 musicians being sampled within this range. However, there is still a significant majority of graphs being straight in all cases. It can therefore be concluded the appearance of double-barrelled graphs probably does not depend upon experience.

3.1.2 Technique & Groove

Next, the production of double-barrelled graphs is analysed as a function of playing technique. These results are shown in Table 2.

Table 2: Table to investigate the production of double-barrelled graphs produced by the DFA algorithm. DFA is performed on the variation between onset times of a closed hi-hat. There are 66 samples, based off 22 musicians playing semiquavers along to "Tom Sawyer" by Rush. Each musician performs three times. One performance is with two hands, one is with one hand, and the last is in as metronomic and unaccented a fashion as possible. DFA is then performed on the variation of onset times. The shape of graphs produced are classified as straight, forward double-barrelled or reverse double-barrelled. Data is then filtered by the technique used. It is found that a majority of results are straight, indicating that the double-barrelled result produced by Räsänen et al. is not universal and that technique does not affect the production of double-barrelled graphs.

	Straight	Forward Double-Barrelled	Reverse Double-Barrelled
Single Handed	86.36%	9.11%	4.53%
Double Handed	81.81%	4.53%	13.66%
Metronomic	86.36%	4.53%	9.11%

Once again, the graphs produced are consistently and overwhelmingly straight. In all cases, over 80% of results are straight. Averaging over all three methods, it can be seen that 84.84% of graphs are straight. There are also similar percentages of samples producing forward and reverse double-barrelled results. These results are more definitive than when comparing experiences. This is potentially as there are more samples of data in each group, 22, instead of approximately 8. These results are also more consistent. As such, it is likely that the direction of double-barrelled graphs is not related to the specific technique used to play.

Lastly, the presence of a groove is investigated. By combining the results for single and double handed methods from Table 2, it can be seen that 84.04% are straight when musicians are grooving, and 86.36% are straight when not. Moreover, only 13.64% of graphs are forward double-barrelled and only 6.82% are reverse double-barrelled when grooving. This is very similar to the numbers of forward and reverse bias double-barrelled graphs when there is no groove. As of the similarity of the data, groove is also unlikely to influence the production of double-barrelled graphs.

3.1.3 Discussion

Considering everything, it is unlikely that the double-barrelled nature of the Hurst exponent in music proposed by Räsänen et al. is universal. This is as the vast majority of graphs produced are straight and there are similar numbers of forward and reverse bias double-barrelled graphs. These observations are not influenced by technique, the presence of a groove, or experience.

However, the musicians used are of nowhere near the quality of Porcaro. Double-barrelled results may simply be a unique hallmark of his talent or of innate talent in general. Their existence may also be unique to given pieces of music, such as "I Keep Forgettin'". Alternatively, the link between drumming and α may even be polynomial.

One probable explanation for the existence of double-barrelled structures in Räsänen et al.'s result involves errors in onset time detection. During this experiment, onset times are found to be accurate to within $\pm$0.005 s. When given our automated method and clean samples, it is unlikely that the 0.001 s of precision estimated by Räsänen et al. is reasonable. $\pm$0.01 s would be a sensible approximation instead. After inserting white noise of $\pm$0.1s into the onset times found by Räsänen et al., the graph of $F(s)$ against s produced becomes much straighter. This is shown in Figure 9. α also becomes similar to our results, at $\alpha = 0.52 \pm 0.06$.

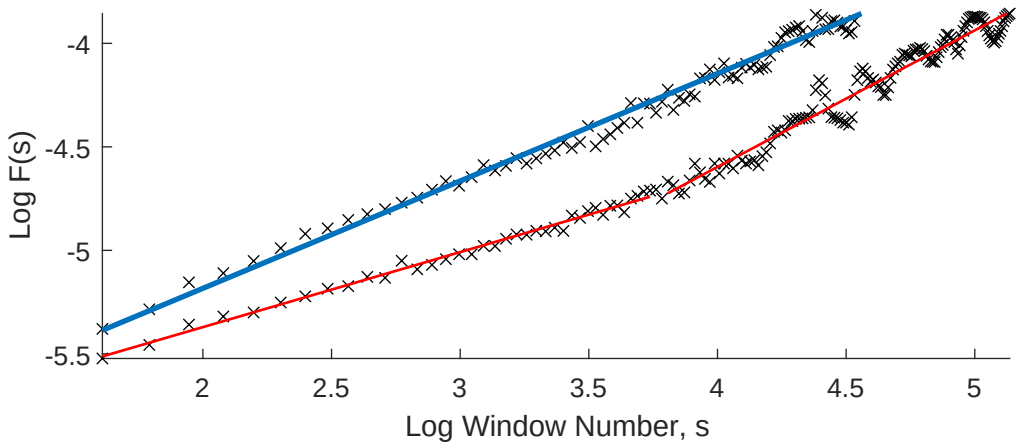


Figure 9: Figure to demonstrate the effect of errors on the data produced by Räsänen et al.. The onset times when the hi-hat was hit was originally thought to be accurate to within ± 0.001s, and therefore called irrelevant. This produced a graph with two gradients of $\alpha = 0.31$ and $\alpha = 0.72$, where α is the Hurst exponent. This double-barrelled graph is shown as the two red lines. Instead, we apply white noise of maximum amplitude 0.01s to these onset times, and propagate it through the DFA algorithm used to produce this graph. The resultant data, shown in blue, is visibly much straighter, and has $\alpha = 0.52 \pm 0.06$. This corresponds to drumming not being a fractal process.

The semi-manual process employed by Räsänen et al. to detect onsets was also ineffective. Only 76% of the total onsets were detected in "I Keep Forgettin'". This is far inferior to our combination of clean audio samples and a fully automated detection algorithm, which consistently detects over 95% of potential onsets. As such, over 200 extra onsets are detected during a typical two minute recording. This extra data may be responsible for straightening the graphs, as more windows can be formed to compare against. The relationship between α and the number of onsets detected would therefore be worth investigating to confirm this.

Another potential explanation for the double-barrelled graphs seen regards the DFA algorithm itself. DFA has been shown to introduce unexpected structures into the graphs of $F(s)$ against s used to calculate α[28][29]. This flaw holds particularly true for window sizes below s=200 and shorter data sets below 500 points of data[28]. As such, a small number of results may produce double-barrelled graphs not because of a universality in music, but instead because of a limitation of the DFA algorithm used. This could be investigated in future by attempting to reproduce double-barrelled graphs with alternative algorithms such as Modified DFA[30], Centred Moving Average DFA[31] or Multifractal DFA[32].

3.2 Is Drumming Fractal?

To determine if drumming is fractal and has long-range correlations, α is calculated for each sample. In all cases, samples producing double-barrelled results are excluded for consistency. As before, these values are then investigated as a function of experience, technique, and groove. In each case, the number of samples with values of α within one standard deviation of error from $\alpha = 0.5$ is then recorded. This was as $\alpha = 0.5$ corresponds to non-fractal white noise.

3.2.1 Experience

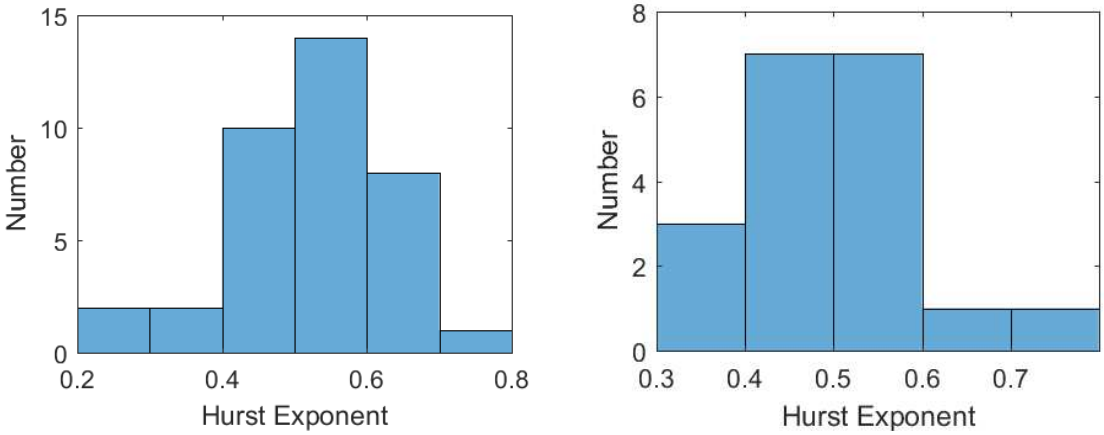
First, α is investigated as a function of experience. As before, metronomic results were excluded. Experience ranges are also identical to those used previously; $0 \leq E \leq 5$, $5 < E \leq 10$, $10 < E \leq 15$, and $E > 15$. This results in 50.00%, 50.00%, 30.00% and 70.00% of results, respectively, being within one standard deviation of 0.5. Although these results differ, there are not enough experience ranges from which to see a trend. The low sample number may also potentially explain the 40% spread of results. Also, the spread of α across each experience ranges has standard deviations of 0.11, 0.07, 0.11 and 0.08, respectively.

As these metrics are similar and show no obvious trend between experience ranges, α might not depend on the quality of a musician. Instead, α may be intrinsic to an individual musician. This would suggest that fractal patterns are not due to subconscious actions, but instead due to the physical aspect of performing said action. This could be explored by giving each musician an opportunity to play multiple pieces of music multiple times.

3.2.2 Technique & Groove

Similar results are also found when investigating techniques. For both the double and single handed methods, 50.00% of results are found to be within one standard deviation of error from $\alpha = 0.5$. Similarly, 54.50% of results are within one standard deviation for the metronomic technique. As these results are similar, it is unlikely that method has an effect on the value of the Hurst exponent.

Finally, this similarity is still seen when comparing results with and without a groove. Although many results are centred around $\alpha = 0.5$, there is a large distribution of results. This is shown in Figure 10. For the combined single and double handed techniques, α has a range of 0.44, a standard deviation of 0.10, and a mean of 0.52. This is highly similar to the metronomic result when no groove is present. For this technique, α is found to have a range of 0.34, a standard deviation of 0.09, and a mean of 0.51. The large size of the range and standard deviation during the same song implies that α is also not specific to an individual piece of music.



(a) Single & Double Handed (With Groove) **(b)** Metronomic (Without Groove)

Figure 10: Figure to demonstrate the distribution of Hurst exponents, α, produced during the experiment. 22 musicians are recorded as they play semiquavers on the hi-hats of a drum kit. This is repeated with both one and two handed methods and allowing them to groove to the music. This is also repeated in as metronomic a fashion as possible. DFA is then performed on the variations in times when the hi-hats were hit. The distribution of values of α calculated are shown when grooving (a), and not grooving (b).

3.2.3 Discussion

Combining all of the above, it is difficult to confirm if drumming is a fractal process. Interestingly, this contrasts with prior work, which is of the view that drumming is indeed a fractal process. Drumming may not be fractal as many results are within one standard deviation of $\alpha = 0.5$. However, drumming may be fractal as there were wide distributions of α found.

However, the large number of results being within one standard deviation of $\alpha = 0.5$ could be explained by errors produced by DFA. At higher values of s, the graphs of $F(s)$ against s used to calculate α would appear to oscillate, as demonstrated in Figure 11. This is likely due to a flaw in the DFA algorithm, which has been shown to introduce unexpected structures into the graphs of $F(s)$ against s used to calculate α[28 29]. When oscillations were present, this resulted in large errors when curve fitting for α of up to approximately α=0.07. As it could only be approximated, it was not considered when calculating final values. This oscillatory data could also not be cut off for several reasons. Firstly, oscillations began to appear at a consistent window size. Also, some graphs appeared to change to or from being double-barrelled if oscillations were removed due to the decrease in data set size. As such, removing data manually would therefore introduce human bias. An algorithmic method of detecting and removing this data would be a better solution. The total size of the data set was also limited, making it difficult to observe distributions.

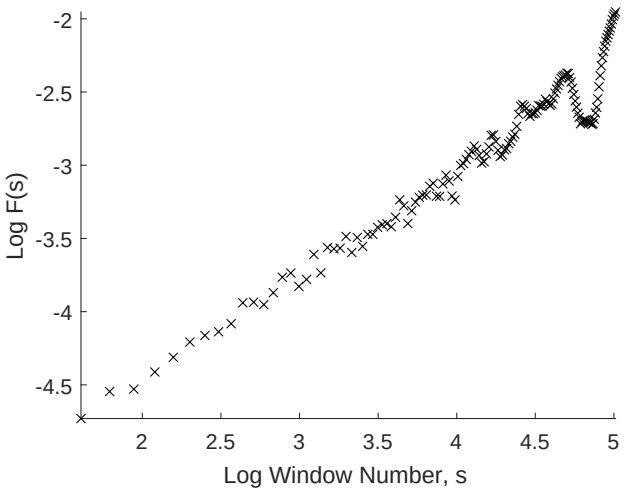


Figure 11: Figure to demonstrate oscillations produced by the DFA algorithm. At higher window sizes, s, the graphs begin to oscillate, producing a large error. These can not be removed manually as the oscillations begin to appear at different values of s for different samples. This is problematic as manually picking samples would cause a human bias to be introduced when determining if these graphs are straight or instead have two linear gradients present.

3.3 Quantification of Errors & Other Limitations of Approach

Two of the main sources of error in this experiment involve those of curve fitting and inaccuracy when determining onsets. This inaccuracy occurs because drumsticks and hi-hats deform slightly when hit, so sound is not produced instantaneously. This means that there are no perfectly sharp peaks in PSD from which to determine onsets. A slight curve near the maxima in PSD is seen instead. An example of this is shown in Figure 12.

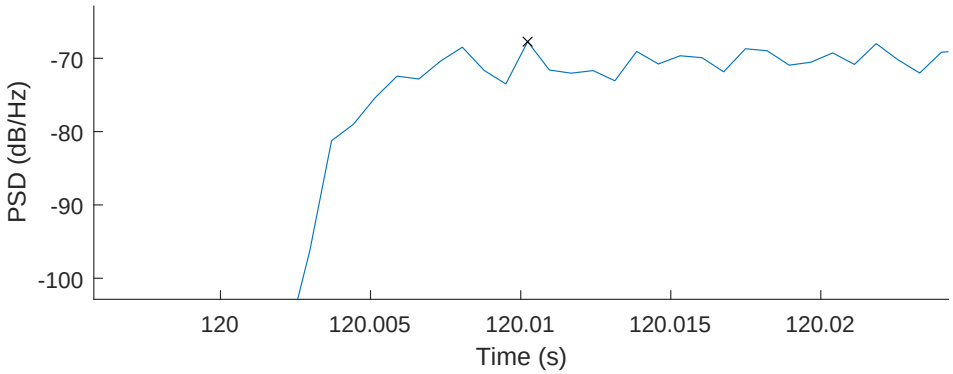


Figure 12: Figure to demonstrate the error produced when detecting onset times when a hi-hat is hit. An onset time is detected as the time where the power spectral density (PSD) peaks, and is shown as a black cross. Normally, it should be defined as the time when the power stops rising. However, when observing the PSD, shown in blue, it is difficult to automatically determine when the PSD stops rising. Manually identifying each of the 45606 onsets detected in this project is impractical. Instead, the range in which the power may stop rising is taken, and used as the error on onset time detection. This was estimated to be ± 0.005 s.

Although this could be somewhat accounted for through manual analysis of all 45,606 detected onsets, this is impractical. Instead, an approximation is made by manually analysing the time onset error of a limited number of onsets. This results in the error of $t(i)$ being estimated to be ± 0.005s.

Despite it being possible to algebraically propagate this error through the DFA algorithm, this was not attempted here. Instead, white noise is used to estimate the error on a due to imprecise onset detection. This is as DFA quantifies the self-similarity of a data set, and white noise has a statistically constant degree of self-similarity. As such, the error on $t(i)$ is first added in quadrature with itself to give an error on τ, $\delta\tau$. White noise of amplitude $\pm\delta\tau/2$ is then added to τ. As the insertion of noise produces slightly varying results, DFA is then performed 50 times with different combinations of random noise added to the same audio sample. The standard deviation of each of the 50 values of α is then taken and added in quadrature with the average error from each of the 50 curve fits. This results in an error on a due to onset detection of ± 0.02. This is found to be the same for multiple data samples. To produce the final error on α for each sample, 0.02 is added in quadrature with the error produced when curve fitting for α without white noise.

Another large issue with the experiment regards quantifying experience. The number of years a musician has played is not a perfect metric with which to measure talent. It is not possible to measure or quantify innate natural talent, and all musicians will have different rehearsal and performance schedules of varying calibre and effectiveness. To improve upon this, musicians could be sorted by the number of public performances they have given. This is as it could be assumed that better musicians would be given the opportunity to give a greater number of performances.

Moreover, many of the single handed results for less talented musicians are problematic because such musicians will frequently break down and stop. Musicians with 0-5 years of experience produce an average of 384 onsets, whereas those with over 15 years produce an average of 698 onsets. This is even slightly greater than the average number of onsets detected, 691. These musicians often report that they did not possess the dexterity to groove along to the piece with one hand. They therefore prefer playing with two hands. The more talented musicians, however, feel that they could get a better groove by playing one handed. This is the reason as given by Jeff Porcaro.

Many musicians are also unfamiliar with the music. If a musician had not heard "Tom Sawyer" before, they would not immediately be able to groove along to it. This would make the results in which a groove is present similar to those in which a groove is not present. With a piece more simplistic and pop-like, such as "I Keep Forgettin," it would be easier for these musicians to pick up the groove. Musicians could also be asked to listen to the piece of music before being asked to play it.

Other issues involved musicians being constrained to a range of onset variations. As seen in Figure 6, musicians drift from the expected time between onsets. The piece was therefore not recorded with a metronome. However, by playing in time with the recording instead of truly playing for themselves, the drift of onsets from the mean becomes suppressed. This alters the groove of each individual. As such, results may have the same issues seen when playing with a metronome. In future, a drummer could be asked to play along to a piece of music with the drum track removed. Alternatively, a feedback loop could also be developed to speed up or slow down a metronomically recorded piece of music in response to a musician playing along to it.

4 Conclusion

In conclusion, the double-barrelled graph produced by Räsänen et al. is highly unlikely to be universal amongst music and the hi-hats. This is largely because 84.84% of the graphs produced with 66 samples were straight, as shown by summing Table 2. The existence of double-barrelled structures could be explained by the DFA algorithm [28 29]. The errors in onset detection may also explain their production, as demonstrated in Figure 9.

It is also found that experience, technique, and the presence of a groove are unlikely to affect the production of double-barrelled graphs. This is mainly as no strong trends were observed amongst different techniques and experience ranges. Results were also highly similar; only experiences between 10 and 15 years had under 80% of graphs being straight. However, in this case a majority of 59.91% of graphs were still straight. This discrepancy may therefore be due to the limited data set. These results are shown in Tables 1 and 2.

As such, it is highly possible that double-barrelled results are either due to innate talent, or the result of a flaw with DFA. In future, the production of double-barrelled results could be investigated for highly talented musicians playing together and alone. Also, DFA could qualitatively be shown to produce this graph structure by calculating α using alternative algorithms.

Nonetheless, this result is in disagreement with the work done by Räsänen et al.. This means that generating drum patterns with two Hurst exponents may not be necessary to generate more pleasing sound music.

Finally, it is also difficult to confirm if drumming is a fractal process. Drumming may not be a fractal process, as over 50% of results were within one standard deviation of $\alpha = 0.5$. However, as shown in Figure 11, oscillations produce a large error of up to $\alpha = 0.07$, and may have caused many results to fall within one standard deviation of $\alpha = 0.5$. Moreover, the large range of results, 0.44, and large standard deviation of this spread, 0.10, mean that many samples did in fact exhibit fractal qualities. These results are partly demonstrated in Figure 10.

Interestingly, this suggestion that drumming may not be a fractal process disagrees with prior work. There is also no sufficient explanation why previous studies have found that people overwhelmingly prefer drum patterns generated with long-range correlations [9 10]. It is entirely possible that it is not the fractal nature of human actions that makes drumming pleasant, but instead another phenomenon that is sometimes produced when producing fractal sounds. More research is therefore needed before considering the implementation of α into commercial music generation software.

However, before any of these conclusions can be made with certainty, a much larger data set is required. Musicians should have been given a chance to rehearse and familiarise themselves with the piece. Other songs should also be investigated. It is also important to verify the quality of the DFA algorithm used, as DFA has been shown to introduce some of the structures observed here.

5 References

[1] J Clarke R Voss. 1/f noise in music and speech. *Acoustical Society of America Journal*, 63:258–263, 1978. doi:10.1121/1.381721.

[2] Esa Räsänen, Otto Pulkkinen, Tuomas Virtanen, Manfred Zollner, and Holger Hennig. Fluctuations of hi-hat timing and dynamics in a virtuoso drum track of a popular music recording. *PloS one*, 10(6):e0127902, 2015. doi:10.1371/journal.pone.0127902.

[3] Richard P Taylor, Adam P Micolich, and David Jonas. Fractal analysis of pollock's drip paintings. *Nature*, 399(6735):422, 1999. doi:10.1038/20833.

[4] Jooeun Ahn and Neville Hogan. Long-range correlations in stride intervals may emerge from non-chaotic walking dynamics. *PLoS ONE*, 8(9):e73239, 2013. doi:10.1371/journal.pone.0073239.

[5] Sunkyu Yu, Xianji Piao, Jiho Hong, and Namkyoo Park. Bloch-like waves in random-walk potentials based on supersymmetry. *Nature Communications*, 6:8269, 2015. doi:10.1038/ncomms9269.

[6] Gert J. van Tonder. Recovery of visual structure in illustrated japanese gardens. *Pattern Recognition Letters*, 28(6):728–739, 2007. doi:10.1016/j.patrec.2006.08.001.

[7] Harold Edwin Hurst. Long-term storage capacity of reservoirs. *Trans. Amer. Soc. Civil Eng.*, 116:770–808, 1951.

[8] D. Gilden, T Thornton, and M. Mallon. 1/f noise in human cognition. *Science*, 267 (5205):1837–1839, 1995. doi:10.1126/science.7892611.

[9] Holger Hennig, Ragnar Fleischmann, and Theo Geisel. Musical rhythms: The science of being slightly off. *Physics Today*, 65(7):64–65, 2012. doi:10.1063/pt.3.1650.

[10] Holger Hennig, Ragnar Fleischmann, Anneke Fredebohm, York Hagmayer, Jan Nagler, Annette Witt, Fabian J. Theis, and Theo Geisel. The nature and perception of fluctuations in human musical rhythms. *PLoS ONE*, 6(10):e26457, 2011. doi:10.1371/journal.pone.0026457.

[11] Wosuk Ro and Younghun Kwon. 1/f noise analysis of songs in various genre of music. *Chaos, Solitons & Fractals*, 42(4):2305–2311, 2009. doi:10.1016/j.chaos.2009.03.129.

[12] Bill Manaris, Dallas Vaughan, Christopher Wagner, Juan Romero, and Robert B. Davis. Evolutionary music and the zipf-mandelbrot law: Developing fitness functions for pleasant music. In *Lecture Notes in Computer Science*, pages 522–534. Springer Nature, 2003. doi:10.1007/3-540-36605-9_48.

[13] D. J. Levitin, P. Chordia, and V. Menon. Musical rhythm spectra from bach to joplin obey a 1/f power law. *Proceedings of the National Academy of Sciences*, 109(10):3716–3720, 2012. doi:10.1073/pnas.1113828109.

[14] Archi Banerjee, Shankha Sanyal, Tarit Guhathakurata, Ranjan Sengupta, and Dipak Ghosh. Categorization of stringed instruments with multifractal detrended fluctuation analysis. *arXiv*, 2016.

[15] Michael McDonald. *I keep forgettin' (every time you're near)*. Warner Bros, 1982.

[16] C.-K. Peng, S. V. Buldyrev, S. Havlin, M. Simons, H. E. Stanley, and A. L. Goldberger. Mosaic organization of DNA nucleotides. *Physical Review E*, 49(2):1685–1689, 1994. doi:10.1103/physreve.49.1685.

[17] Jeff Porcaro. *Instructional DVD for Drums*. Quantum Leap, 1986.

[18] H. Hennig. Synchronization in human musical rhythms and mutually interacting complex systems. *Proceedings of the National Academy of Sciences*, 111(36):12974–12979, 2014. doi:10.1073/pnas.1324142111.

[19] Zhi-Yuan Su and Tzuyin Wu. Music walk, fractal geometry in music. *Physica A: Statistical Mechanics and its Applications*, 380:418–428, 2007. doi:10.1016/j.physa.2007.02.079.

[20] Rolf Bader. *Nonlinearities and Synchronization in Musical Acoustics and Music Psychology*. Springer Nature, 2013. doi:10.1007/978-3-642-36098-5.

[21] Rush. *Tom Sawyer*. Mercury, 1981.

[22] Ingo R. Titze. *Principles of Voice Production*. Prentice Hall, 1994. ISBN 978-0-13-717893-3.

[23] Ronald J. Baken and Robert F. Orlikoff. *Clinical Measurement of Speech & Voice*. SINGULAR PUB GROUP, 1999. ISBN 1565938690.

[24] Richard A. Altes. Detection, estimation, and classification with spectrograms. *The Journal of the Acoustical Society of America*, 67(4):1232–1246, 1980. doi:10.1121/1.384165.

[25] Miguel Alonso, Gal Richard, and Bertrand David. Tempo estimation for audio recordings. *Journal of New Music Research*, 36(1):17–25, 2007. doi:10.1080/09298210701653260.

[26] Jan W Kantelhardt, Eva Koscielny-Bunde, Henio H.A Rego, Shlomo Havlin, and Armin Bunde. Detecting long-range correlations with detrended fluctuation analysis. *Physica A: Statistical Mechanics and its Applications*, 295(3-4):441–454, 2001. doi:10.1016/s0378-4371(01)00144-3.

[27] Espen A. F. Ihlen. Introduction to multifractal detrended fluctuation analysis in matlab. *Frontiers in Physiology*, 3, 2012. doi:10.3389/fphys.2012.00141.

[28] R. M. Bryce and K. B. Sprague. Revisiting detrended fluctuation analysis. *Scientific Reports*, 2, 2012. doi:10.1038/srep00315.

[29] Amir Bashan, Ronny Bartsch, Jan W. Kantelhardt, and Shlomo Havlin. Comparison of detrending methods for fluctuation analysis. *Physica A: Statistical Mechanics and its Applications*, 387(21):5080–5090, 2008. doi:10.1016/j.physa.2008.04.023.

[30] Xi-Yuan Qian, Gao-Feng Gu, and Wei-Xing Zhou. Modified detrended fluctuation analysis based on empirical mode decomposition for the characterization of anti-persistent processes. *Physica A: Statistical Mechanics and its Applications*, 390(23-24):4388–4395, 2011. doi:10.1016/j.physa.2011.07.008.

[31] Jose Alvarez-Ramirez, Eduardo Rodriguez, and Juan Carlos Echeverría. Detrending fluctuation analysis based on moving average filtering. *Physica A: Statistical Mechanics and its Applications*, 354:199–219, 2005. doi:10.1016/j.physa.2005.02.020.

[32] Jan W. Kantelhardt, Stephan A. Zschiegner, Eva Koscielny-Bunde, Shlomo Havlin, Armin Bunde, and H.Eugene Stanley. Multifractal detrended fluctuation analysis of nonstationary time series. *Physica A: Statistical Mechanics and its Applications*, 316(1-4):87–114, 2002. doi:10.1016/s0378-4371(02)01383-3.

A Code

A.1 Main Code

```
% Oliver Gordon & Dominic Coy, 2017
% Is Drumming Fractal? - Main Code

%% Setup/Settings
clear variables;
close all;

% Settings
sample_start = 1;                                   % Sample to start from
sample_end = 22;                                    % Sample to end with
technique = 'Single Handed';                        % Double Handed OR
                                                    % Single Handed OR
                                                    % Metronomic

bpm = 88;                                           % BPM of piece

filter_order = 100;                                 % FIR filter order
max_pass_freq = 16000;                              % Max bandpass freq
min_pass_freq = 600;                                % Min bandpass freq
hard_freq = 14000;                                  % Max hardpass freq

basic_thresh_peak_no = 100;                         % Simple onset drift
basic_thresh_peak = 15;                             % Simple onset amp

thresh_onset = 0.034;                               % Max onset drift
thresh_amp = 0.1;                                   % Max onset amp
s_min = 5;                                          % Min window size
s_max = 150;                                        % Max window size

db1_end = 20;                                       % Max window size H1
db2_start = 30;                                     % Min window size H2
db_thresh = 0.1;                                    % Straight/Barrelled?

db = 1;
exp_onset = 60/bpm/4;

% Clear sound if sound still playing
if exist('player','var') == 1
    stop(player)
end

% Ensure 2016b or newer because of syntax change
if verLessThan ('matlab','9.1.0.441655') ~= 0
    error('Circshift changed syntax after 2016a. Use 2016b or newer.')
end

% Read in Rasenen data for checking |Onset Times|Amplitudes|
% all_data = dlmread('Docs\Check File.txt');

% Set up matrix to store exponents. Rows = samples, Columns = double
% barrelled test (11 if forward db, 12 if reverse, 13 if straight),
% exponent 1, error, exponent 2, error, exponent straight, error
```

```
final_result = NaN(7,(sample_end-sample_start+1));

% Set up matrices to store onsets and power variation. Columns = sample no.
final_onset_difs = NaN((sample_end-sample_start+1),3000);
final_power_difs = NaN((sample_end-sample_start+1),3000);

% Repeat entire process for all samples
for sample_no = sample_start:sample_end
%% Read In Files

% Read in raw audio file and make mono
Sample_Name = [technique,' ',num2str(sample_no)];
[audio_data,audio_freq] = audioread(['Samples\',Sample_Name,'.wav']);
[~,channels] = size(audio_data);
if channels == 2
    audio_data = (audio_data(:,1) + audio_data(:,2)) / 2;
end

%% Pre-Processing

% 100th order FIR bandpass for 0.6-16kHz to remove low and high noise
[b,a] = fir1(filter_order,[min_pass_freq/(audio_freq/2),...
    max_pass_freq/(audio_freq/2)],'bandpass');
audio_data_FIR = filter(b, a, audio_data);

% Calculate spectogram
[~,spec_freq,spec_time,power] = ...
    spectrogram(audio_data_FIR, 64, [], [], audio_freq, 'yaxis');

% Hard lowpass for <14kHz so only well defined power changes will be used
% to calculate onset times
hard_ind = round((hard_freq/22500)*length(spec_freq));
F_hard = spec_freq((1:hard_ind),:);

% Calculate corresponding power
power = 10*log10(abs(power)+eps);
power = power(1:hard_ind,:);
power_tot = mean(power);

%% Onset Times - Basic Algorithm

% Define time axes
time = 1/audio_freq : 1/audio_freq : length(audio_data)/audio_freq;
time_power = linspace(1/audio_freq,length(audio_data)/audio_freq, ...
    length(power_tot));

Npeaksmax = round(length(audio_data_FIR)/audio_freq/60*bpm*4);

% Find locations of peaks & delete peaks incorrectly appearing during onset
    locs = find(power_tot - (circshift(power_tot,-1)) < -basic_thresh_peak)+1;
    peak_power = power_tot(locs);

% Calculate corresponding times between peaks
onset_time = time_power(locs);
onset_dif = diff(onset_time);
mean_onset_dif = mean(onset_dif);

%% Onset Times - Stronger Algorithm
```

```
111
112 % Calculate array number corresponding to window time
113 exp_onset_samples = round(exp_onset/(time_power(2)-time_power(1)));
114 window_onset_samples = round(thresh_onset/(time_power(2)-time_power(1)));
115
116 % Find first peak using basic algorithm to begin more advanced peak
117 % detection - code runs in <0.02 seconds, so don't bother adapting for just
118 % first peak
119 window_mid = locs(5);
120
121 % Set up loop
122 window_no = 1;
123 locs1 = NaN(1,Npeaksmax);
124 locs1(1) = window_mid;
125
126 while window_mid < length(time_power)-window_onset_samples
127     % Define window range to search in
128     window_range = window_mid-window_onset_samples ...
129         : window_mid+window_onset_samples;
130
131     % Detect potential onset peaks and record the highest, or carry on if
132     % none found
133     [~,peak_loc] = findpeaks(power_tot(window_range),'sortstr','descend');
134     if isempty(peak_loc) == 0
135         locs1(window_no) = window_mid-window_onset_samples+peak_loc(1)-1;
136     end
137
138     % Move to next windowing region
139     window_mid = locs1(window_no)+exp_onset_samples;
140     window_no = window_no+1;
141 end
142
143 % Calculate time and amplitude of peaks for each time axis
144 locs1 = locs1(1:window_no-2);
145 peak_power1 = power_tot(locs1);
146 onset_time1 = time_power(locs1);
147 locs2 = round(onset_time1./(time(2)-time(1)));
148 onset_time2 = time(locs2-(filter_order/2));
149 onset_amp2 = audio_data(locs2-(filter_order/2));
150
151 % Calculate difference between onset times, and remove %erroneous results
152 %delta_t = diff(all_data(:,1))';              % Test onset differences
153 %delta_p = diff(all_data(:,2))';
154 %delta_t = step(dsp.ColoredNoise(...          % Noise of chosen B.
155 %     'InverseFrequencyPower',0.5,...        % a = (B+1)/2
156 %    'SamplesPerFrame',5000))';
157 %delta_t = exp_onset.*ones(5000,1);           % Metronome
158
159 delta_t = diff(onset_time1);                 % Actual data
160 delta_p = diff(peak_power1);
161
162 % Remove errenous onset differences
163 delta_t(exp_onset-delta_t>thresh_onset | ...
164     exp_onset-delta_t<-thresh_onset)=[];
165 delta_p(delta_t(exp_onset-delta_t>thresh_onset | ...
166     exp_onset-delta_t<-thresh_onset)) = [];
167
168 % Add random offset for error
```

```
%random_offset = (0.01+0.01).*rand(1,length(delta_t)) - 0.01;
%delta_t = delta_t+random_offset;

mean_t = mean(delta_t);

% Store power difference and onset time difference
final_onset_difs(sample_no,1:length(delta_t)) = delta_t;
final_power_difs(sample_no,1:length(delta_p)) = delta_p;

%% DFA

% Pre-allocate array of F_k
F_k_s = zeros(floor(length(delta_t)/s_max),s_max);

for s = s_min:s_max

    % Pre-allocate/reset inner loops
    delta_y = zeros(1,length(delta_t));
    fitrange = 1:s:length(delta_y)-s;
    delta_y_s = zeros(1,fitrange(end)+s-1);
    F_k_substep = zeros(1,length(s));

    % Sum onset difference over i from i=1 to i=N data points
    for i = 1:length(delta_t)
        % Sum over j from j=1 to j=i
        delta_y(i) = sum(delta_t(1:i)-mean_t);
    end

    % Omit data when not enough present to form new window of size s
    delta_y = delta_y(1:length(delta_y_s));

    % Perform linear fits for onset differences in windows of size s
    for fitx = 1:s:length(delta_y)
        polyco = polyfit_s(1:s , delta_y(fitx:fitx+s-1) , 1);
        delta_y_s(fitx:fitx+s-1) = ...
            (polyco(1).*(1:s))+polyco(2);
    end

    % Calculate RMS fluctuations of linear detrending
    for k_range = 0:((length(delta_y_s)/s)-1)
        sumstart = (k_range*s)+1;
        sumend = (k_range*s)+s;

        F_k_substep(k_range+1) = sum((delta_y(sumstart:sumend) - ...
            delta_y_s(sumstart:sumend)).^2);
    end

    F_k_s(1:length(F_k_substep),s) = sqrt((1/s).*F_k_substep);
end

% Calculate mean (ignoring leftover values = 0) and define axis for cftool
F_s = (sum(F_k_s,1)./sum(F_k_s~=0,1));
s_axis = (1:s);

%% Find Hurst Exponent

% Remove NaN data and configure axis for use in curve fit
F_s(isnan(F_s))=[];
```

```
227 F_s = log(F_s);
228 s_axis = log(s_axis(end-length(F_s)+1:end));
229
230 % Curve fit first line of window sizes to first order
231 [pcoeff1,pstruct1] = polyfit((s_axis(1:db1_end)),(F_s(1:db1_end)),1);
232 [yfit1,perror1] = polyval(pcoeff1,log(s_axis(1:db1_end)),pstruct1);
233 perror1 = mean(perror1);
234
235 % Curve fit second line of window sizes to first order
236 [pcoeff2,pstruct2] = polyfit((s_axis(db2_start:end)),(F_s(db2_start:end)),1);
237 [yfit2,perror2] = polyval(pcoeff2,log(s_axis(db2_start:end)),pstruct2);
238 perror2 = mean(perror2);
239
240 % See if double barrelled by checking gradients+errors against threshold
241 % and store results
242 if (pcoeff2(1)-perror2) - (pcoeff1(1)+perror1) > db_thresh
243     % Forward bias double barrel
244
245     % Print values
246     disp('Forward Double Barrelled')
247     disp(['Hurst Exponent 1 = ',num2str(pcoeff1(1)),...
248         char(177),num2str(sqrt(perror1.^2+0.02^2))])
249     disp(['Hurst Exponent 2 = ',num2str(pcoeff2(1)),...
250         char(177),num2str(sqrt(perror1.^2+0.02^2))])
251
252     % Store values
253     final_result(1,sample_no) = 11;
254     final_result([2,4],sample_no) = [pcoeff1(1);pcoeff2(1)];
255     final_result([3,5],sample_no) = [perror1;perror2];
256 elseif (pcoeff1(1)-perror1) - (pcoeff2(1)+perror2) > db_thresh
257     % Reverse bias double barrel
258
259     % Print values
260     disp('Reverse Double Barrelled')
261     disp(['Hurst Exponent 1 = ',num2str(pcoeff1(1)),...
262         char(177),num2str(perror1)])
263     disp(['Hurst Exponent 2 = ',num2str(pcoeff2(1)),...
264         char(177),num2str(perror2)])
265
266     % Store values
267     final_result(1,sample_no) = 12;
268     final_result([2,4],sample_no) = [pcoeff1(1);pcoeff2(1)];
269     final_result([3,5],sample_no) = [perror1;perror2];
270 else
271     % Straight line
272     disp('Straight')
273
274     % Fit straight line
275     [pcoeff3,pstruct3] = polyfit((s_axis),(F_s),1);
276     [yfit3,perror3] = polyval(pcoeff2,log(s_axis),pstruct2);
277     perror3 = mean(perror3);
278     disp(['Hurst Exponent = ',num2str(pcoeff3(1)),...
279         char(177),num2str(perror3)])
280
281     % Store value
282     final_result(1,sample_no) = 13;
283     final_result([6,7],sample_no) = [pcoeff3(1);perror3];
284     db=0;
```

```
285 end
286
287 %% Plotting
288
289 % Clear all axis
290 arrayfun(@cla,findall(0,'type','axes'))
291
292 % ------
293 % Figure 1
294 % ------
295 % Plot graph of gradient Hurst Coefficient
296 Fig1 = figure(1);
297 hold on
298 plot(s_axis,F_s,'kx')
299
300 % Plot Hurst exponents
301 if db == 0
302     plot(s_axis,((s_axis.*pcoeff3(1))+pcoeff3(2)))
303     db = 1;
304 else
305     plot(s_axis(1:db1_end),...
306         ((s_axis(1:db1_end).*pcoeff1(1))+pcoeff1(2)))
307     plot(s_axis(db2_start:end),...
308         ((s_axis(db2_start:end).*pcoeff2(1))+pcoeff2(2)))
309 end
310 hold off
311 xlim([min(s_axis),max(s_axis)])
312 ylim([min(F_s),max(F_s)])
313 xlabel('Log Window Number, s')
314 ylabel('Log F(s)')
315 title(['Graph to Calculate Hurst Exponent, ',Sample_Name])
316 saveas(gcf,['Figures\',technique,' ',num2str(sample_no),...
317 'Hurst Exponent'],'png')
318
319 % ------
320 % Figure 2
321 % ------
322 % Plot spectogram
323 Fig2 = figure(2);
324 imagesc(spec_time, F_hard, power);
325 image_spec = gca;
326 image_spec.YDir = 'normal';
327 xlabel('Time (s)')
328 ylabel('Frequency (Hz)')
329 title(['Spectogram, ',Sample_Name])
330 cbar = colorbar;
331 cbar.Label.String = 'Power/Freq (dB/Hz)';
332 axis tight;
333 %saveas(gcf,['Figures\',technique,' ',num2str(sample_no),...
334 %'Spectogram'],'epsc')
335
336 % ------
337 % Figure 3
338 % ------
339 % Plot original waveform
340 Fig3 = figure(3);
341 plot_orig = subplot(3,1,1);
342 hold on
```

```
plot(time,audio_data);
plot(onset_time2,onset_amp2,'kx');
xlabel('Time (s)');
ylabel('Amplitude');
xlim([0,time(end)]);
hold off

% Plot filtered waveform
plot_FIR = subplot(3,1,2);
plot(time,audio_data_FIR);
xlabel('Time (s)');
ylabel('Amplitude');
xlim([0,time(end)]);

% Plot power
plot_power = subplot(3,1,3);
hold on
plot(time_power,power_tot);
plot(onset_time1,peak_power1,'kx');
xlabel('Time (s)');
ylabel('PSD (dB/Hz)');
xlim([0,time(end)]);
hold off
%saveas(gcf,['Figures\',technique,' ',num2str(sample_no),...
%'Waveforms'],'epsc')

% ------
% Figure 4
% ------
% Plot onset time difference
Fig4 = figure(4);
hold on
plot_delta_y = subplot(2,1,1);
xlim([0,length(delta_y)]);

% Plot windowed onset time difference fits
hold on
plot(delta_y_s,'k-','LineWidth',0.05)
plot(delta_y)
xlabel('Onset Number')
ylabel('Drift From Mean (s)')
xlim([0,length(delta_y)]);

% Plot vertical lines of width window length
for linedif = 1:s:length(delta_y)
    ax = gca;
    line([linedif,linedif],ax.YLim,...
        'Color','r','LineStyle','--','LineWidth',0.05)
end

% Plot onset time variation
plot_onsetdif = subplot(2,1,2);
hold all
plot(exp_onset-delta_t);
xlabel('Onset Number');
ylabel('Onset Variation (s)');
title(['Onset Time Variation, ',Sample_Name])
%saveas(gcf,['Figures\',technique,' ',num2str(sample_no),' Drift'],'epsc')
```

```
401
402 %% "Fun" Stuff
403
404 % Link axes tile figure windows (Created by Peter J. Acklam, 1998)
405 linkaxes([plot_orig,plot_FIR,plot_power,image_spec],'x')
406 linkaxes([plot_delta_y,plot_onsetdif],'x')
407 tilefigs
408
409 % Output time /percentage of onsets found.
410 fprintf(['Theoretical onset difference = ',num2str(60/bpm/4),' secs. \n'])
411 fprintf(['Calculated onset difference = ',num2str(mean_t),' secs.\n'])
412 disp([num2str(length(locs1)/(Npeaksmax-4)*100),'% of peaks found'])
413
414 end
415
416 % Play
417 %player = audioplayer(audio_data, 1*audio_freq);
418 %player = audioplayer(audio_data_FIR, 1*audio_freq);
419 %play(player);
420
421 % Save final exponents and differences to file
422 dlmwrite(['Docs\',num2str(technique),' Exponents.txt'],final_result)
423 dlmwrite(['Docs\',num2str(technique),' Variation.txt'],...
424     [final_onset_difs;final_power_difs])
425
426 beep
427 disp('Done!')
```

A.2 Analysis Code

```
% Oliver Gordon & Dominic Coy, 2017
% Is Drumming Fractal? - Analysis Code

clear variables
close all

%% Importing
technique_all = {'Double Handed','Single Handed','Metronomic'};
no_samples = 22;
no_techniques = 2;

% Preallocate storage arrays
variation_all = NaN(no_samples*2,3000*no_techniques);
exponents_all = NaN(7,no_samples*no_techniques);

% Read in stored values
experience_all = xlsread('Docs\People.xlsx');
for technique = 1:2

    % Read in experiences
    experience = repmat(experience_all(:,4),no_techniques,1);

    % Read in variations and separate out
    variation_all(:,((((technique-1)*3000)+1):(technique*3000))) = ...
        dlmread(['Docs\',char(technique_all(technique)),' Variation.txt']);
    variation_onset = variation_all(1:no_samples,:);
    variation_power = variation_all(no_samples+1:end,:);

    % Read in exponents and separate out
    exponents_all(:,((((technique-1)*no_samples)+1):...
        (technique*no_samples))) = ...
        dlmread(['Docs\',char(technique_all(technique)),' Exponents.txt']);
    exponents_class = exponents_all(1,:);
    exponents_double = exponents_all([2,4],:);
    exponents_double_err = sqrt((exponents_all([3,5],:).^2)+(0.02^2));
    exponents_single = exponents_all(6,:);
    exponents_single_err = sqrt((exponents_all(7,:)).^2+(0.02^2));

end
%% Analysis

% Find index of values that are forward db, reverse db, straight
index_forward = find(exponents_class == 11);
index_backward = find(exponents_class == 12);
index_straight = find(exponents_class == 13);

% Find index of values in each experience range
index_0_5 = find(experience<=5);
index_5_10 = find(experience>5 & experience<=10);
index_10_15 = find(experience>10 & experience<=15);
index_15_max = find(experience>15);

% Find percentage of results forward, reverse, straight for experience
% ranges
perc_forward_0_5 = sum(sum(index_0_5 == index_forward))/...
```

```
    (no_samples.*no_techniques)*100;
perc_forward_5_10 = sum(sum(index_5_10 == index_forward))/...
    (no_samples.*no_techniques)*100;
perc_forward_10_15 = sum(sum(index_10_15 == index_forward))/...
    (no_samples.*no_techniques)*100;
perc_forward_15_end = sum(sum(index_15_max == index_forward))/...
    (no_samples.*no_techniques)*100;

perc_backward_0_5 = sum(sum(index_0_5 == index_backward))/...
    (no_samples.*no_techniques)*100;
perc_backward_5_10 = sum(sum(index_5_10 == index_backward))/...
    (no_samples.*no_techniques)*100;
perc_backward_10_15 = sum(sum(index_10_15 == index_backward))/...
    (no_samples.*no_techniques)*100;
perc_backward_15_end = sum(sum(index_15_max == index_backward))/...
    (no_samples.*no_techniques)*100;

perc_straight_0_5 = sum(sum(index_0_5 == index_straight))/...
    (no_samples.*no_techniques)*100;
perc_straight_5_10 = sum(sum(index_5_10 == index_straight))/...
    (no_samples.*no_techniques)*100;
perc_straight_10_15 = sum(sum(index_10_15 == index_straight))/...
    (no_samples.*no_techniques)*100;
perc_straight_15_end = sum(sum(index_15_max == index_straight))/...
    (no_samples.*no_techniques)*100;

% Store this as matrix
perc_age = [perc_straight_0_5, perc_forward_0_5, perc_backward_0_5 ; ...
    perc_straight_5_10, perc_forward_5_10, perc_backward_5_10 ; ...
    perc_straight_10_15, perc_forward_10_15, perc_backward_10_15 ; ...
    perc_straight_15_end, perc_forward_15_end, perc_backward_15_end];

% Find percentage of results that are forward, reverse, straight
perc_straight_tot = sum(perc_age(:,1));
perc_forward_tot = sum(perc_age(:,2));
perc_backward_tot = sum(perc_age(:,3));

% Find percentage of results outside 1 s.d. of 0.5 for straight
mean = 0.5;
ins_sing = exponents_single((exponents_single > ...
    mean-exponents_single_err) & ...
    (exponents_single < mean+exponents_single_err));
index_ins_sing = find(ismember(exponents_single,ins_sing));

% Find percentage of straight
% results outside 1 s.d of 0.5, for different experiences
ins_0_5 = length(find(ismember(index_ins_sing,index_0_5)))...
    /length(index_0_5)*100;
ins_5_10 = length(find(ismember(index_ins_sing,index_5_10)))...
    /length(index_5_10)*100;
ins_10_15 = length(find(ismember(index_ins_sing,index_10_15)))...
    /length(index_10_15)*100;
ins_15_end = length(find(ismember(index_ins_sing,index_15_max)))...
    /length(index_15_max)*100;

% Find total percentage of straight results outside 1 s.d.
disp([num2str((length(ins_sing))/...
    no_samples/no_techniques*100), '% of results within 1 s.d of '...,
```

```
    num2str(mean)]);
disp([num2str(ins_0_5),'% of 0-5 years within 1 s.d. of ',...
    num2str(mean)])
disp([num2str(ins_5_10),'% of 5-10 years within 1 s.d. of ',...
    num2str(mean)])
disp([num2str(ins_10_15),'% of 10-15 years within 1 s.d. of ',...
    num2str(mean)])
disp([num2str(ins_15_end),'% of 15+ years within 1 s.d. of ',...
    num2str(mean)])

% Find standard deviation of straight results for different experiences
disp(['std of 0-5 group = ',...
    num2str(nanstd(exponents_single(index_0_5)))]);
disp(['std of 5-10 group = ',...
    num2str(nanstd(exponents_single(index_5_10)))]);
disp(['std of 10-15 group = ',...
    num2str(nanstd(exponents_single(index_10_15)))]);
disp(['std of 15+ group = ',...
    num2str(nanstd(exponents_single(index_15_max)))]);

% Find means of onsets and power for both axis
mean_variation_onset = nanmean(variation_onset);
mean_variation_power = nanmean(variation_power);
row_mean_variation_onset = nanmean(variation_onset,2);
row_mean_variation_power = nanmean(variation_power,2);

% Find drift away from mean drift for both power and onset
bpm = 88;
exp_onset = 60/bpm/4;

drift_mean_onset = mean_variation_onset - exp_onset;
drift_mean_power = mean_variation_power - exp_onset;

%% Plotting

% ------
% Figure 1
% ------
% Plot drift from mean onset time
figure(1)
plot(cumsum(drift_mean_onset(~isnan(drift_mean_onset))));
%set(gca,'fontsize',16)
xlabel('Onset Number');
ylabel('Drift From Mean (s)');
%title('Graph to Show the Drift Away From the Mean for Onset time');

% ------
% Figure 2
% ------
% Plot drift from mean onset power
figure(2)
plot(cumsum(drift_mean_power(~isnan(drift_mean_power))));
set(gca,'fontsize',16)
xlabel('Onset Number');
ylabel('Drift Away From the Mean');
title('Graph to Show the Drift Away From the Mean for Onset Power');

% ------
```

```
% Figure 3
% ------
% Plot histogram of hurst exponent, with different subfigures for different
% experience ranges
figure(3);
max_val = max(experience);
experience_onset = [experience, exponents_all(6, :)'];
filterLims(experience_onset, 0, 5, 1);
filterLims(experience_onset, 6, 10, 2) ;
filterLims(experience_onset, 11, 15, 3);
filterLims(experience_onset, 16, max_val, 4) ;

% -------------
% Figures 4 - 8
% -------------
nbins = 15;
sample_no = 1:14;

% Histogram of hurst exponent
figure(4)
histogram(exponents_single)
xlabel('Hurst Exponent');
ylabel('Number');
% title('Histogram to Show Hurst Exponents');
% set(gca,'fontsize',16)
disp(['std = ', num2str(nanstd(exponents_single))])
disp(['range = ', num2str(max(exponents_single)-min(exponents_single))])
disp(['mean = ', num2str(nanmean(exponents_single))])

% Histogram of all onset time variation
figure(5)
histogram(variation_onset(sample_no,:));
xlabel('Onset Variation (s)');
ylabel('Number');
title('Histogram to Show Onset Time Variations');
set(gca,'fontsize',16)

% Histogram of all onset power variation
figure(6)
histogram(variation_power(sample_no,:));
xlabel('Power Variation');
ylabel('Number');
title('Histogram to Show Power Variations');
xlim([min(min(variation_power(sample_no,:))), ...
    max(max(variation_power(sample_no,:)))])
set(gca,'fontsize',16)

% Histogram of mean onset time variation
figure(7)
histogram(row_mean_variation_onset,nbins);
xlabel('Mean Onset Variation (s)');
ylabel('Number');
title('Histogram to Show the Mean Onset Variations');
set(gca,'fontsize',16)

% Histogram of mean power variation
figure(8)
histogram(row_mean_variation_power,nbins);
```

```
xlabel('Mean Power Variation');
ylabel('Number');
title('Histogram to Show the Mean Power Variations');
set(gca,'fontsize',16)
```

```
function out = filterLims(in, lower, upper, subfignum)
% Oliver Gordon & Dominic Coy, 2017
% Is Drumming Fractal? - Function to plot subfigure of histograms
out = in;

% Delete data outside of age range specified
out(~ and(lower <= out(:, 1), out(:, 1) <= upper) , :)= [];

% Plot
subplot(2, 2, subfignum);
nbins = upper-lower+1;
histogram(out(:, 2),nbins);
xlabel('Hurst Exponent');
ylabel('Number');
title(['Hurst Exponents Between ',num2str(lower),...
    ' and ', num2str(upper), ' years ']);
set(gca,'fontsize',16)
end
```