Chapter 4

Analysis of oscillatory time series in the yeast metabolic cycle

To characterise the properties of large sets of time series generated by microfluidics experiments, I sought to develop a pipeline of time series analysis methods.

Short and noisy oscillatory time series are challenging to analyse to give usable characteristics such as period and phase, especially because of the limited information they encode. Microfluidics experiments capture up to 5–10 periods of metabolic cycles, so a Fourier spectrum gives period estimates at low resolution. In addition, there is no standard method of analysing oscillatory time series such as those that arise from the yeast metabolic cycle.

Here, I propose steps for the analysis of datasets of 100–1000 time series related to the yeast metabolic cycle from single cells, using my experimental data as an example case.

Specifically, I address:

1. Data cleaning: choosing data and filtering out long-term trends that may confound analysis.
2. Visualising groups in a dataset: identifying groups within a population of time series based on their similarities. Such a relationship may include groupings or structures within the population of time series.

3. Detection of rhythmicity: determining whether a time series exhibits oscillations.
4. Period estimation: identifying the period of a single time series.
5. Detection of synchrony: identifying whether two types of signal from the same cell are synchronous, and to what extent.

In this chapter, I show that a high-pass Butterworth filter gives control over frequencies when filtering out long-term trends. To discover structure within a dataset, I show that UMAP, a dimension-reduction technique, and modularity clustering, a community detection technique, led to similar groupings. Subsequently, I compared three approaches to rhythmicity detection: a statistical test based on a spectral method, model-fitting and analytically computing a periodogram, and a simple machine learning model. To estimate the period and noise parameters of time series, I then explored the effect of noise on the autocorrelation function of synthetic time series. Finally, I used the cross-correlation function to detect synchrony and to quantify the relationship between two types of oscillators.

4.1 Analysing time series in a biological context

Previous studies have described computational pipelines that included mathematical methods to analyse biological time series.

Zieliński et al. (2022) described a software pipeline (BioDare/BioDare2), catered to circadian rhythm studies, to estimate the period and detect rhythmicity in time series. This pipeline includes choices of methods to detrend and normalise time series, followed by a choice of methods to estimate the period, phase, and

amplitude of the time series. Furthermore, the pipeline also includes statistical tests for the presence of an oscillation in a time series: an implementation of the JTK_CYCLE test (Hughes et al., 2010) along with an empirical derivative (Hutchison et al., 2015).

BioDare2 builds upon Zielinski et al. (2014), which compared and contrasted a set of period-estimation methods — FFT-NLLS, mFourFit, MESA, the Enright periodogram, the Lomb-Scargle periodogram, and spectrum resampling — to conclude with recommendations on time series analysis.

Studies of biological rhythms have used the BioDare pipeline to quantify features of oscillations of fluorescence. These included using FFT-NLLS to calculate the period and amplitude error of fluorescence in mouse brain sections to determine the mechanistic basis of the synchronisation of the suprachiasmatic nucleus (Hamnett et al., 2019). Another study used linear detrending of data followed by FFT-NLLS and spectral resampling to estimate the period and amplitude error of delayed fluorescence of chloroplasts in *Kalanchoë fedtschenkoi* leaves to determine whether phosphorylation of phosphoenolpyruvate affects robustness of circadian rhythms (Boxall et al., 2017).

In addition, Fulcher and Jones (2017) described a software pipeline, termed *hctsa*, that computes over 7700 time series features for input time series. The resulting feature matrix — a row for each time series and a column for each feature — could then be used to identify sets of features that are useful to discriminate between sets of time series or to identify clusters of time series based on their properties. The publication then showed that *hctsa* could be used to distinguish five *Caenorhabditis elegans* strains based on their movement patterns, and to identify clusters in the feature space of time series of *Drosophila melanogaster*

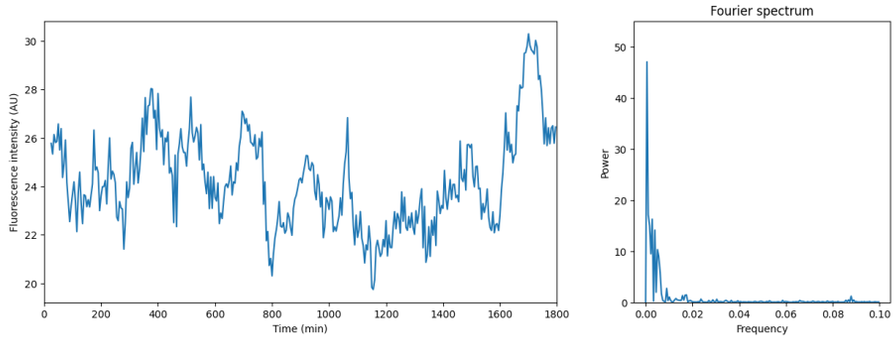
movement patterns which correspond well to experimental groups. To reduce computation time, Lubba et al. (2019) identified 22 features, termed *catch22*, of *hctsa* that performed well in time series classification tasks based on 93 datasets (table 1.1).

Taken together, the two examples of BioDare and *hctsa* demonstrate two approaches to analysis of time series: on one end, relying on mathematical methods, and on the other end, a data science approach to time series classification tasks.

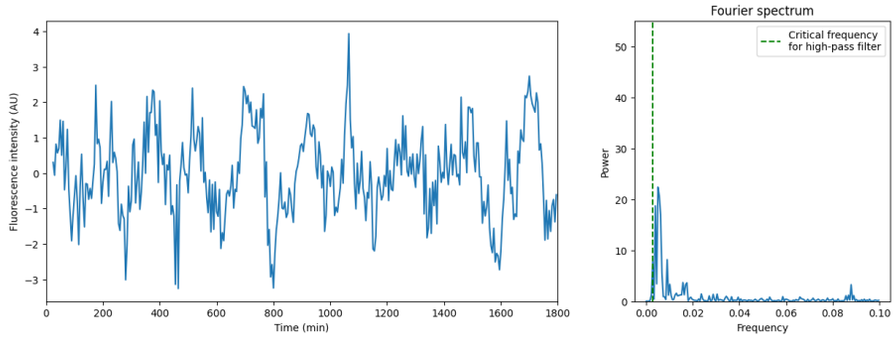
4.2 Data cleaning: filtering out long-term trends

Biological time series often have long-term trends. In my study of the yeast metabolic cycle, such trends include slow, global changes in flavin autofluorescence, which must be removed to uncover the periodic behaviour of flavin autofluorescence that is a component of the metabolic cycle. To determine the detrending method that is most appropriate for my data, I compared a frequency filtering method, a sliding-window detrending method.

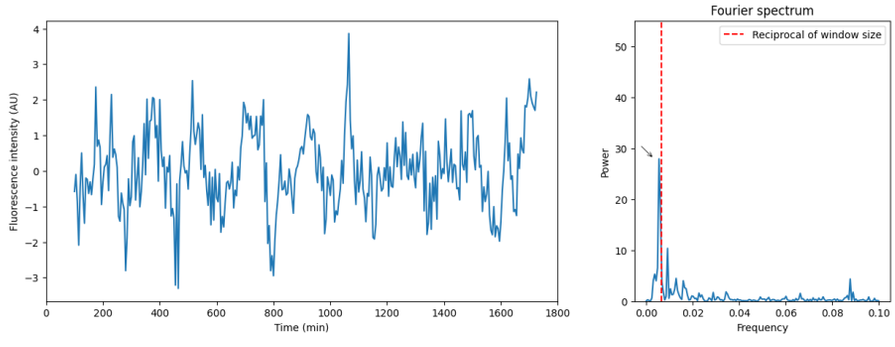
To demonstrate the use of a method that modifies the frequency profile of time series to remove trends, Figs. 4.1a–4.1b show how a time series and its Fourier spectrum changes after the application of a high-pass Butterworth filter with a critical frequency of $2.86 \times 10^{-3}\,\mathrm{min}^{-1}$, corresponding to a period of 350 min. Defining a signal filter offers direct control over frequencies. The critical frequency was based on a reasonable upper limit of periods of the yeast metabolic cycle, from my observations in single-cell microfluidics experiments. Defining this critical frequency is a trade-off between emphasising metabolic oscillations in expected frequency windows and excluding the possibility of long-period metabolic cycles.



(a)



(b)



(c)

Figure 4.1: (Left panels) Time series and (right panels) Fourier spectra corresponding to **(4.1a)** a sample raw time series of flavin autofluorescence, **(4.1b)** the time series processed by a high-pass Butterworth filter with a critical frequency $2.86 \times 10^{-3}\,\mathrm{min}^{-1}$, and **(4.1c)** the time series detrended using a moving average (window size 30). Arrow (↘) indicates artefact.

↳ units?

To show how sliding-window methods adversely affect the frequency profile of time series when used for detrending, I computed the Fourier spectrum of time series detrended using the moving average method. Sliding-window methods are common in detrending biological time series. For example, Cuny et al. (2022) used a moving average: a constant, defined sliding window to smooth time series of yeast cell mass during growth. Fig. 4.1c shows that the moving average method introduced an artefact in the frequency spectrum near the reciprocal of the window size and decreases the number of time points.

4.3 Visualising groups in the dataset

To identify structures in datasets of time series, I implemented UMAP, a dimension reduction method, and modularity clustering, a graph-based clustering method. Such data visualisation methods are important because the structures they show may identify differences between groups that are biologically relevant — for example, subpopulations of oscillations with similar properties. Previous efforts in using computational methods to identify groups in a set of biological time series include using k-means clustering to identify clusters of transcript cycling patterns that correspond to phases of the YMC (Tu et al., 2005), development of a method to cluster featurised multivariate time series based of videos of human motion (Wang et al., 2007), and using signal entropy to featurise fMRI signals followed by modularity clustering to partition the signals into brain regions.

To demonstrate the data visualisation methods, I used time series of flavin autofluorescence oscillations from one experiment with both the wild-type BY4741 strain ($n = 206$) and the mutant *zwf1*Δ strain ($n = 425$). These time series had time points sampled every 5 min in the experiment, for a total of 163 time points. I then defined labels to score whether the time series were oscillatory or not, with 142 of the 206 BY4741 time series classed as oscillatory and 225 of the 425 *zwf1*Δ classed as oscillatory.

4.3.1 UMAP

UMAP (McInnes et al., 2020) is an unsupervised dimension reduction method that can be used to visualise structure in a dataset, and it preserves the distances between points. Specifically, UMAP aims to find a manifold structure of the input observations and compute a low-dimensional embedding that preserves the topological structure of the manifold. This embedding thus serves as coordinates to plot the data onto a low-dimensional space so as to preserve any clustering in the data.

To evaluate whether UMAP was able to discover a structure within the BY4741 & *zwf1*Δ dataset that corresponded to meaningful divisions, I featurised the time series with *catch22*, then used UMAP to compute two-dimensional embeddings. Fig. 4.2a demonstrates that UMAP suggested a small group of non-oscillatory time series that differed markedly from the rest ($*$ in figure), and a larger group that was more similar to oscillatory time series ($**$ in figure). In addition, Fig. 4.2b demonstrates that UMAP suggested that the BY4741 time series were more similar to each other. In contrast, *zwf1*Δ occupied more positions within the

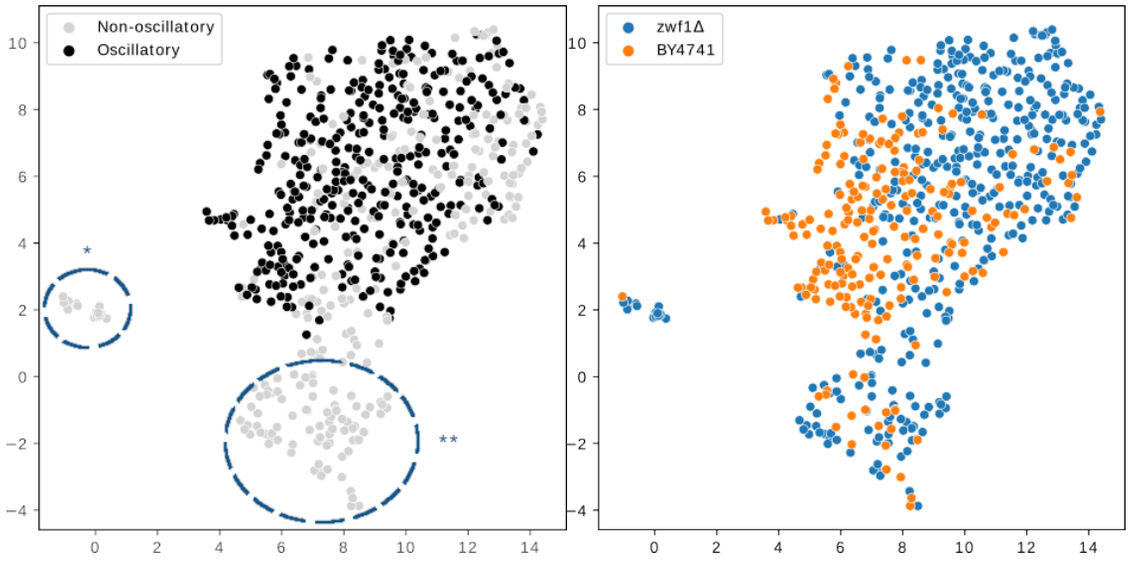


Figure 4.2: UMAP embedding ($n = 5$, min_dist $= 0.5$, $d = 2$, Euclidean distance as the metric) of a dataset of time series featurised using *catch22*. Each node represents a time series, coloured either by **(4.2a)** whether each is oscillatory or not, human-labelled ($*$ and $**$ indicating two groups of non-oscillatory nodes of interest), or by **(4.2b)** strain ('BY4741' or '*zwf1*Δ').

embedding space. These embeddings agreed with my observation that time series from the *zwf1*Δ strain had a larger variety of shapes and oscillation quality than the BY4741 strain. Thus, UMAP may have potential to separate oscillatory and non-oscillatory time series, or time series of different shapes.

To improve the visualisation, I performed a grid search of the n and min_dist UMAP hyperparameters (Appendix 1.2) to find the best combination. Fig. 4.3 suggests that $50 \leq n \leq 150$ and $0.25 \leq$ min_dist ≤ 1 resulted in a good separation between the BY4741 and *zwf1*Δ nodes. In addition, non-oscillatory time series was consistently displayed into groups separate from the rest as the hyperparameters were varied.

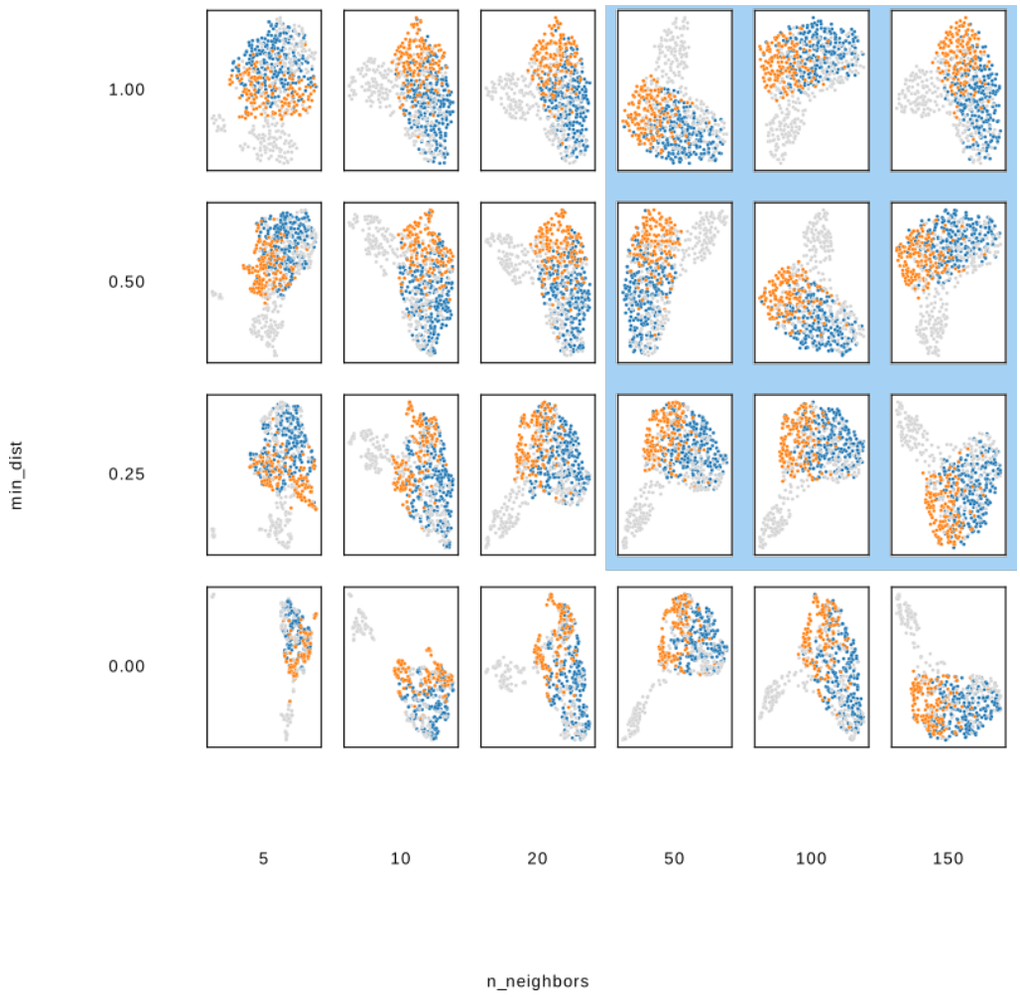


Figure 4.3: Grid search of UMAP hyperparameters: number of neighbours along the horizontal axis and minimum distance along the vertical axis. Data points are coloured according to category: grey indicates non-oscillatory time series, orange indicates oscillatory time series from BY4741 cells, and blue indicates oscillatory time series from *zwf1*Δ cells.

add a colour legend to figure?

4.3.2 Graph-based clustering

Modularity clustering is a mathematical method that partitions a graph into clusters to optimise a 'modularity' value, defined so that the method finds a trade-off between maximising the connections within a cluster and minimising the connections between clusters (Newman, 2006). This optimisation problem is computationally difficult, so approximations such as the Louvain algorithm are needed for large networks (Blondel et al., 2008), with the Leiden algorithm (Traag et al., 2019) subsequently developed to ensure that communities are well-connected and to provide an optimum number of communities.

To assess the performance of a graph-based clustering method in identifying clusters in time series data, I represented a dataset of time series as a graph before using modularity clustering to identify clusters (Fig. 4.4). Fig. 4.5a shows that construction of a pruned graph based on similarities between time series highlights two groups of non-oscillatory time series, similar to the UMAP representation (Fig. 4.2). Furthermore, Fig. 4.5b suggests that these two groups belong to separate communities, and shows that modularity clustering was able to further show communities among the oscillatory time series. However, such communities did not divide cleanly along the division between BY4741 and *zwf1*Δ cells, suggesting that time series features alone were not able to divide these two strains. This agreed with my observation that some oscillatory zwf1Δ time series resembled BY4741 time series.

In sum, the general agreement between UMAP and modularity clustering shows that the BY4741 & *zwf1*Δ dataset had internal structure defined by rhythmicity of time series.

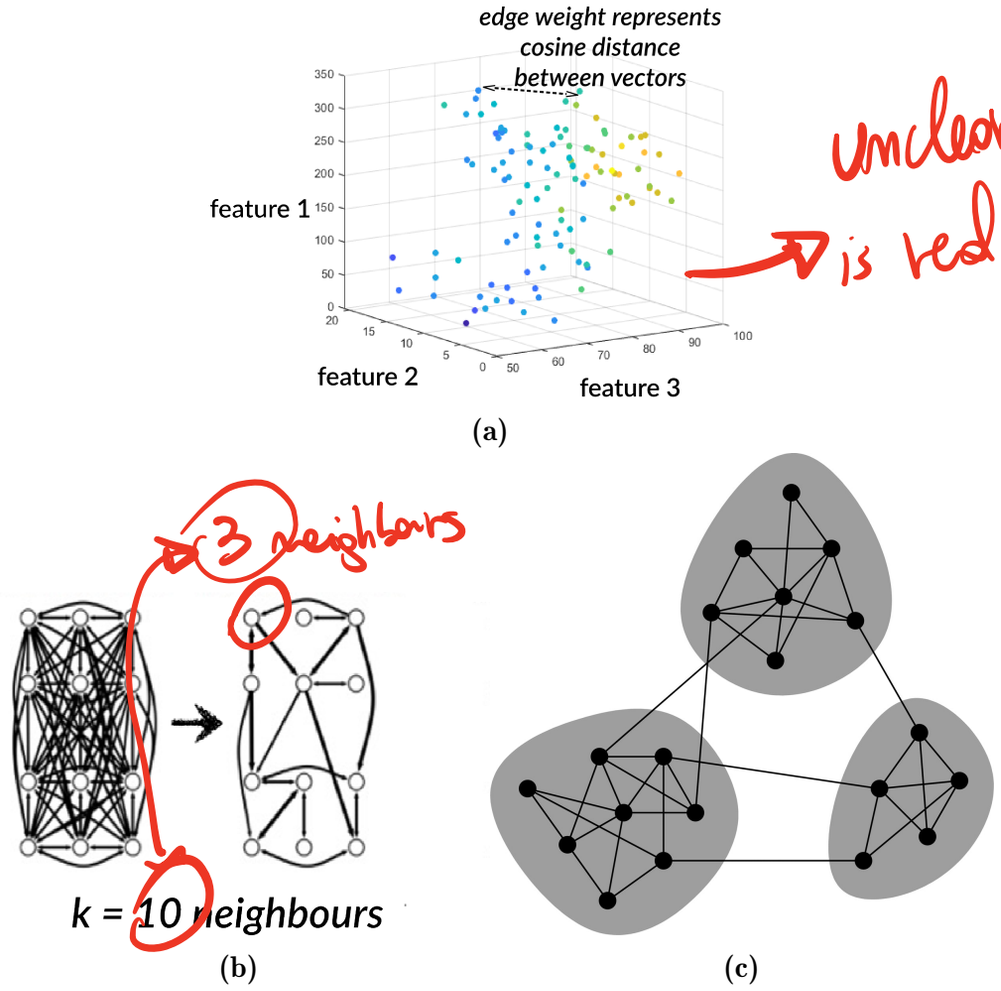


Figure 4.4: Process of preparing a dataset of time series for modularity clustering. **(4.4a)** Constructing a graph representation: each time series was represented as a vector of *catch22* features in 22-dimensional space. The cosine distances between each pair of vector was computed, and became the edge weights of a complete graph with each time series as a node. **(4.4b)** Pruning the complete graph, by deleting edges, so that each node had at least 10 neighbours. **(4.4c)** Modularity clustering, using the Leiden algorithm (Traag et al., 2019), to identify communities. 4.4c adapted from Newman (2006).

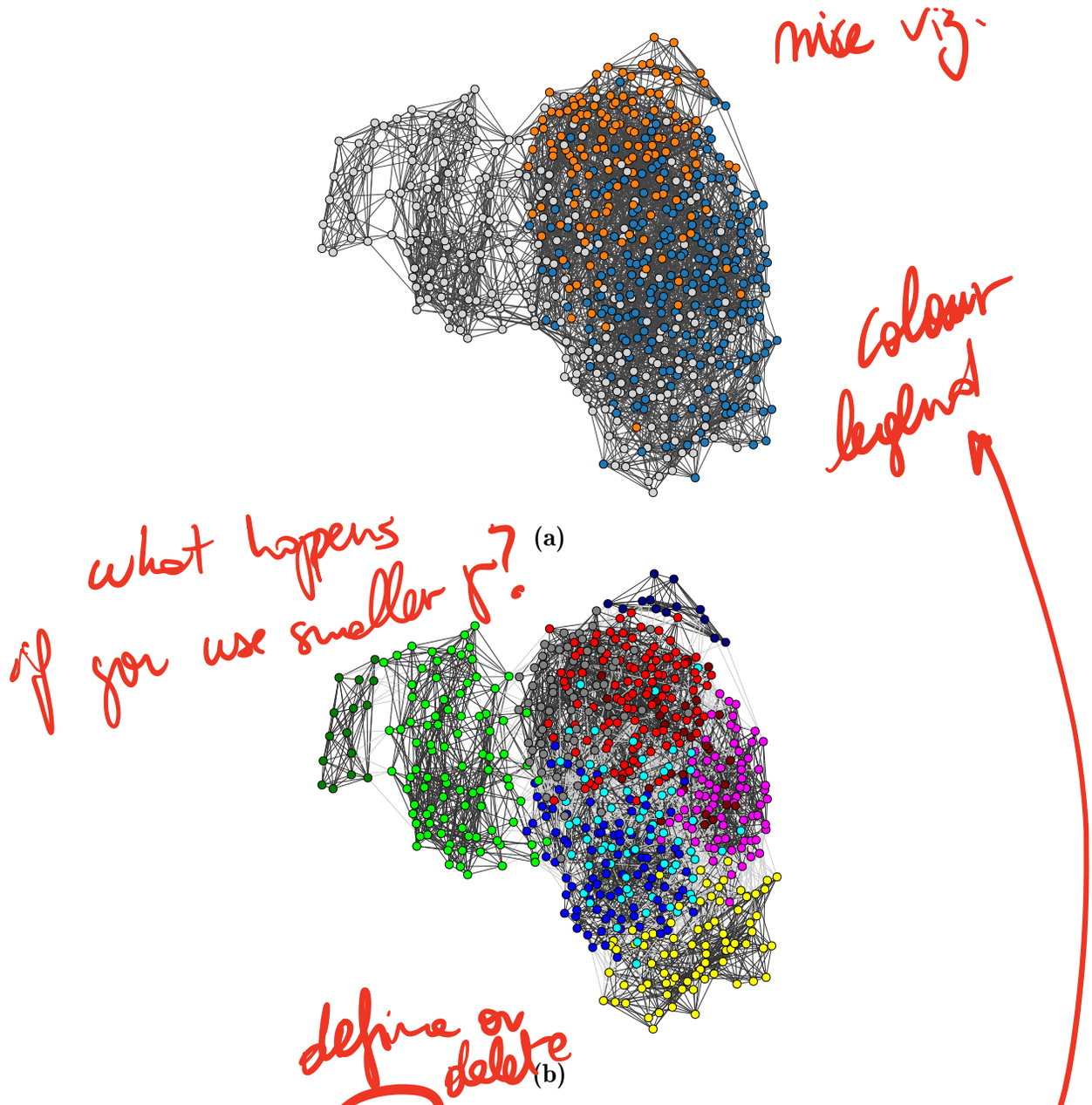


Figure 4.5: Pruned geometric graph of BY4741 and *zwf1*Δ time series from the same experiment. **(4.5a)** Nodes coloured by group: grey, non-oscillatory; blue, oscillatory *zwf1*Δ; orange, oscillatory BY4741. Thickness of edges represent edge weights, scaled by similarity found by cosine distances. **(4.5b)** Nodes coloured by community (out of 10) as optimised by the Leiden algorithm. Edges within a community are in black, while edges between communities are in light grey.

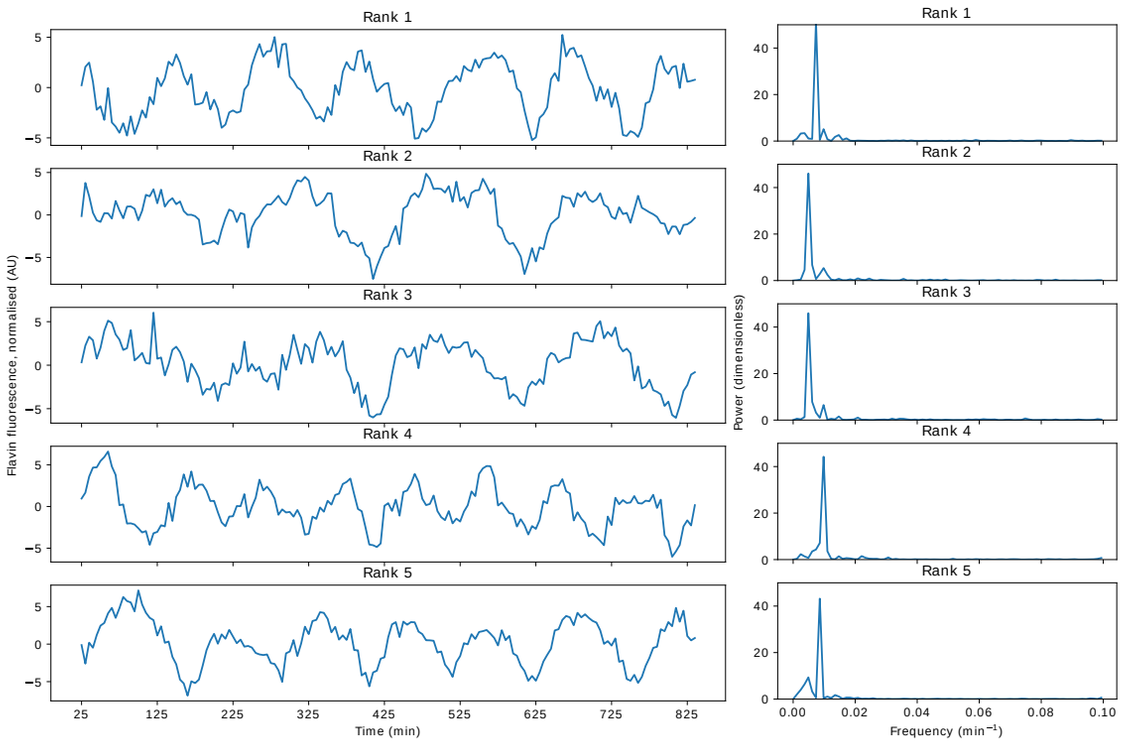
4.4 Detection of rhythmicity

To identify metabolic cycles from flavin autofluorescence signals, it is important to have a systematic method to determine whether a time series is oscillatory. To determine the time series classification method that is most appropriate for my data, I compared a spectral method, a model-fitting method, and a machine learning method.

4.4.1 Rhythmicity detection using spectral methods

In order to classify oscillatory and non-oscillatory time series, I modified a classifier based on a spectral method that included a statistical test (section 2.4.1). This classifier was based on Glynn et al. (2006), which described a method that employed the peak power from the Lomb-Scargle periodogram (Lomb, 1976) to rank time series by the quality of oscillation and to perform a statistical test to determine whether a time series is oscillatory or non-oscillatory (Scargle, 1982).

Figs. 4.6–4.7 suggest that the best- and worst-ranked time series by the quality of their oscillatory signals conformed to subjective judgements of quality. Lowest-ranked time series resembled white noise and led to periodograms with power equally spread across all frequencies, thus bringing down the height of the highest peak. Conversely, highest-ranked time series resembled sinusoids and therefore led to periodograms with a strong power corresponding to the frequency of the sinusoid that would model the time series. However, some time series with visually irregular oscillations (Fig. 4.6; ranks 2, 3) were given higher ranks than those with more regular oscillations, thus calling into question the reliability of the ranking method.



(a) **(b)**

Figure 4.6: (4.6a) Best five time series in the *zwf1*Δ dataset and **(4.6b)** their periodograms, ranked by the quality of oscillation based on the maximum power in the periodogram (Glynn et al., 2006).

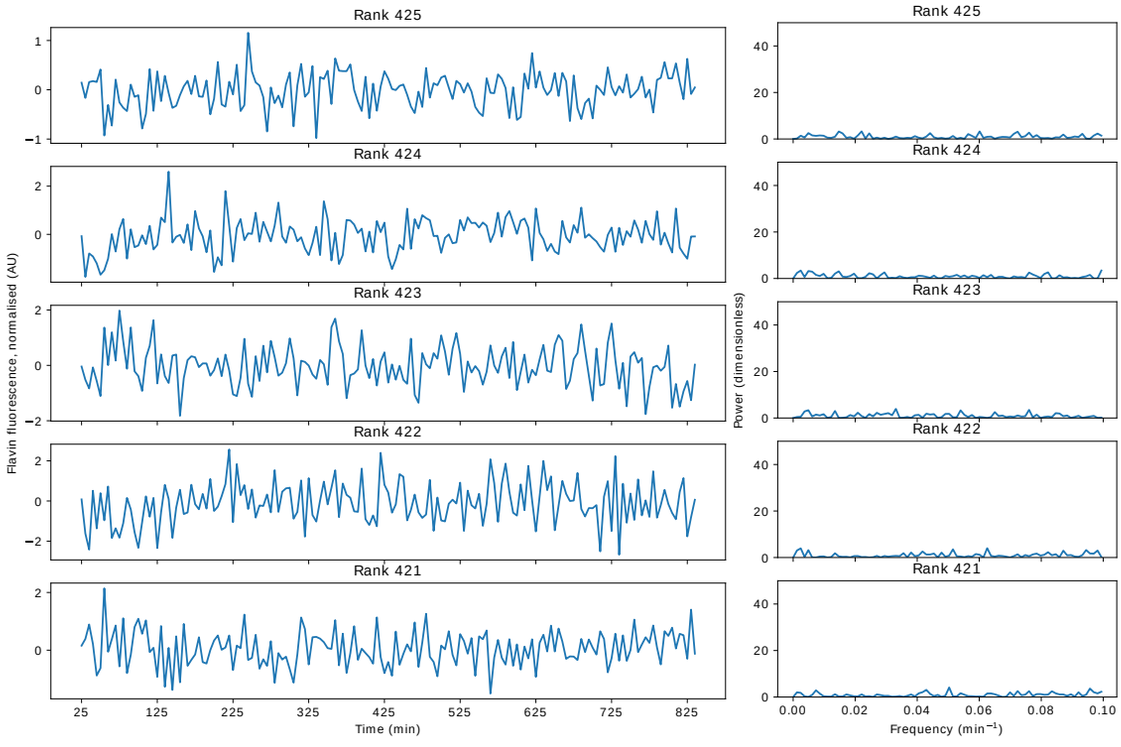


Figure 4.7: (4.7a) Worst five time series in the *zwf1*Δ dataset and **(4.7b)** their periodograms, ranked by the quality of oscillation based on the maximum power in the periodogram (Glynn et al., 2006).

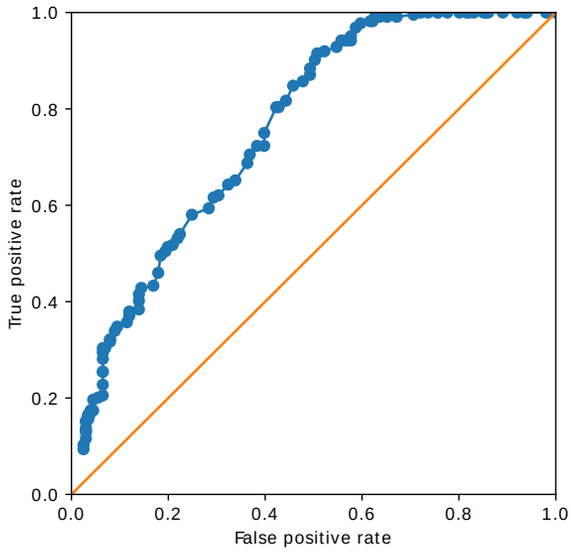


Figure 4.8: ROC curve of classifier based on Glynn et al. (2006) as the false discovery rate was varied, trained on the *zwf1*Δ dataset.

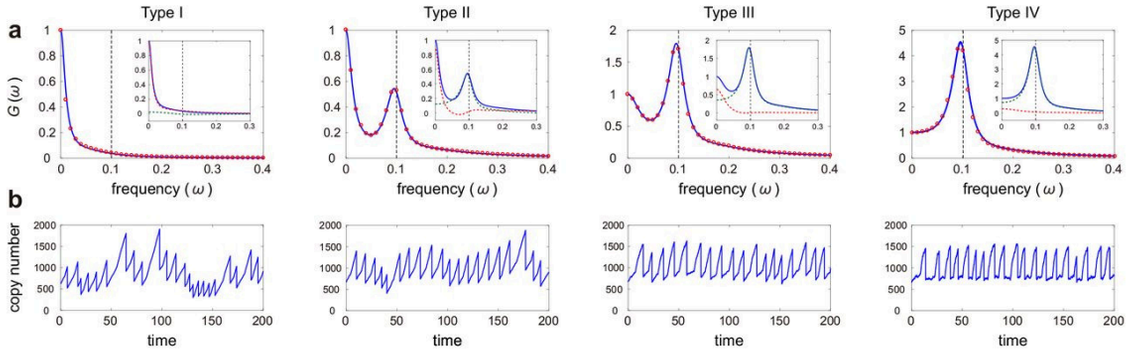
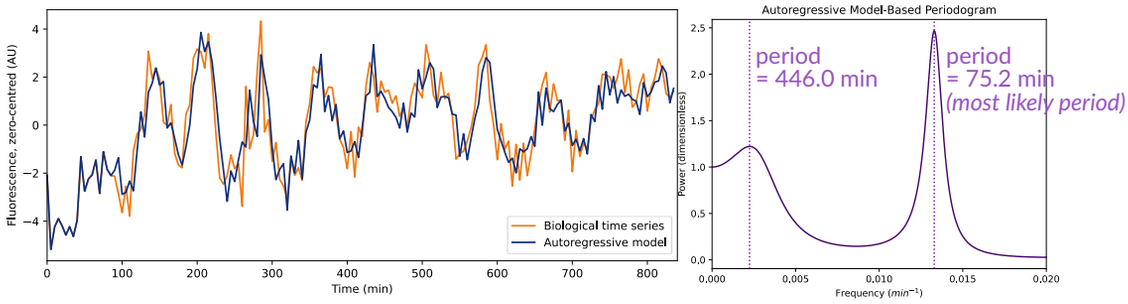


Figure 4.9: Power spectra (a) analytically derived from fitting an autoregressive model to time series (b) can be divided into four types. Type I lacks a local maximum and is denoted as lacking oscillations. Figure adapted from Jia and Grima (2020).

The method described by Glynn et al. (2006) also describes a false discovery rate that is used to control the proportion of oscillations that are oscillatory. To assess the performance of this method as a classifier for rhythmicity detection, Fig. 4.8 shows the receiver operating characteristic (ROC) curve, created as the false discovery rate was varied. The area under the ROC curve ($A = 0.762$) suggests that the classifier performed modestly well, especially for a large ($n = 425$) dataset of time series with a large variety of quality of oscillations.

4.4.2 Rhythmicity detection using model fitting

To assess the performance of a time series classification method based on the autoregressive model, I implemented the method described by Jia and Grima (2020), used to characterise synthetic time series of stochastic, oscillatory gene expression in a dividing cell. In the implementation of the autoregressive model used by Jia and Grima (2020), each data point was expressed as a linear combination of a number of data points that precede it, and model parameters led to an analytical solution for the periodogram, thus giving an advantage over the low-resolution Fourier spectrum (section 2.4.2). The resulting power spectra fell into four categories, one of which corresponded to a lack of oscillations, characterised



(a) **(b)**

Figure 4.10: (4.10a) Sample time series (orange), with a fitted autoregressive model (blue) of order 18 computed according to Jia and Grima (2020). **(4.10b)** Periodogram defined based on parameters of the autoregressive model.

		Human-defined labels		
		Positive	Negative	Total
Predicted by AR model	Positive	141	83	224
	Negative	124	77	201
	Total	265	160	425

Table 4.1: Confusion matrix to evaluate the performance of using the autoregressive model (Jia and Grima, 2020) to detect rhythmicity in the *zwf1*Δ dataset.

by an absence of a local maximum in the power spectrum (Fig. 4.9). This method thus allows computing the frequency of the oscillation from the location of the peak in the periodogram and quality of the oscillation from the height of the peak.

Fig. 4.10 shows that the autoregressive model was able to correctly identify a time series as oscillatory at a period of 75.2 min, as evidenced by the location of a peak in the periodogram that the model predicted. To assess the performance of the use of the autoregressive model for rhythmicity detection across a dataset, I extended this method across the *zwf1*Δ dataset, treating Type I power spectra (Fig. 4.9) as non-oscillatory. The confusion matrix (table 4.1) suggests that the method leads to poor accuracy ($a = 0.513$), as it classed a large proportion of time series as non-oscillatory.

4.4.3 Rhythmicity detection using machine learning

As an alternative to the mathematical methods previously discussed, I trained a support vector classifier to classify oscillatory and non-oscillatory time series from the *zwf1*Δ cells (appendix 1.3).

To ensure that the dynamic ranges of the fluorescence signals do not affect rhythmicity detection, I normalised each time series $x_i = x_{i,1}, x_{i,2}, \ldots, x_{i,j}, \ldots x_{i,n}$ to produce a processed time series $z_i = z_{i,1}, z_{i,2}, \ldots, z_{i,j}, \ldots z_{i,n}$ as follows:

$$z_{i,j} = \frac{x_{i,j} - \mu_i}{\sigma_i} \tag{4.1}$$

where μ_i is the mean value of x_i, and σ_i is the standard deviation of x_i. As a result, each normalised time series z_i has a mean of 0 and a standard deviation of 1. From this input data, 75% of the time series formed the training set.

To determine the most effective way to featurise the data, I computed the precision and recall of support vector classifiers trained on data featurised using different methods. All support vector classifiers were trained using a radial bias kernel, a kernel coefficient $\gamma = 1/N$, where N is the number of features, and a regularisation parameter $C = 10$. Fig. 4.11 suggests that featurisation using *catch22* and the Fourier spectrum gave comparably high performances, as evidenced by high precision and recall, and with a low degree of overfitting, as evidenced by a small variation of both metrics across the rounds of cross-validation.

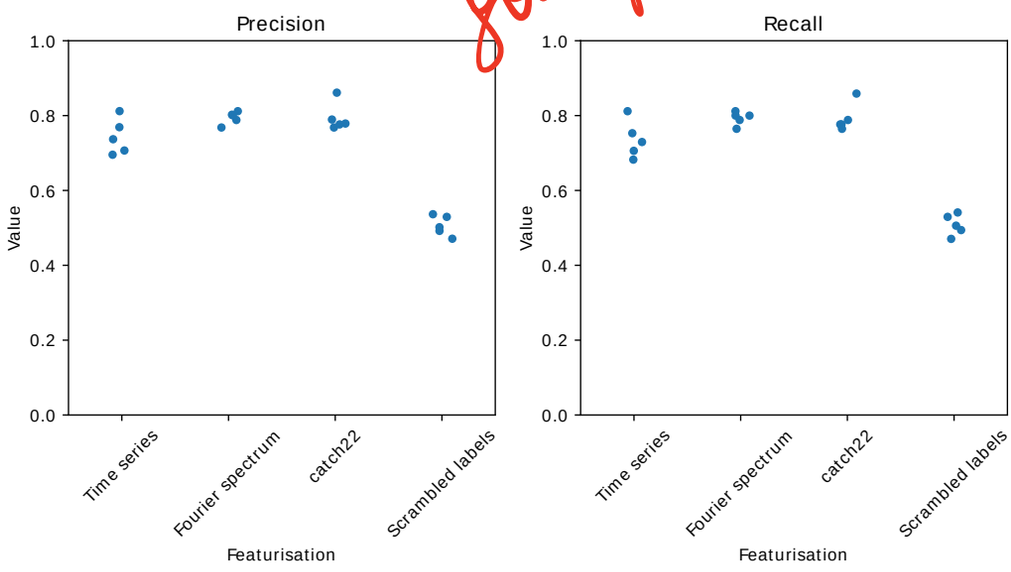
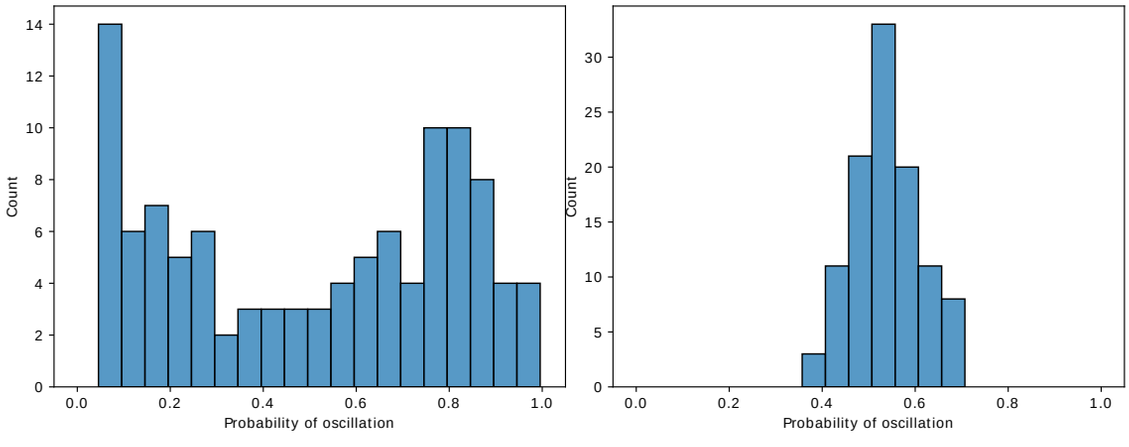


Figure 4.11: (Left) Precision and (right) recall from five-fold cross-validation of support vector classifiers trained using different featurisation methods: using the time points as features, using the power values in the Fourier spectrum as features, and using *catch22* features. As a control, the oscillatory and non-oscillatory labels were randomly assigned to the time series and the time points were used as features.



(a) **(b)**

Figure 4.12: (4.12a) Histogram of probabilities of whether a time series in the test data set is classified as oscillatory by the SVC (featurisation with *catch22*, $\gamma = 1/22$, $C = 10$), and as a control, **(4.12b)** with labels randomly re-assorted to time series.

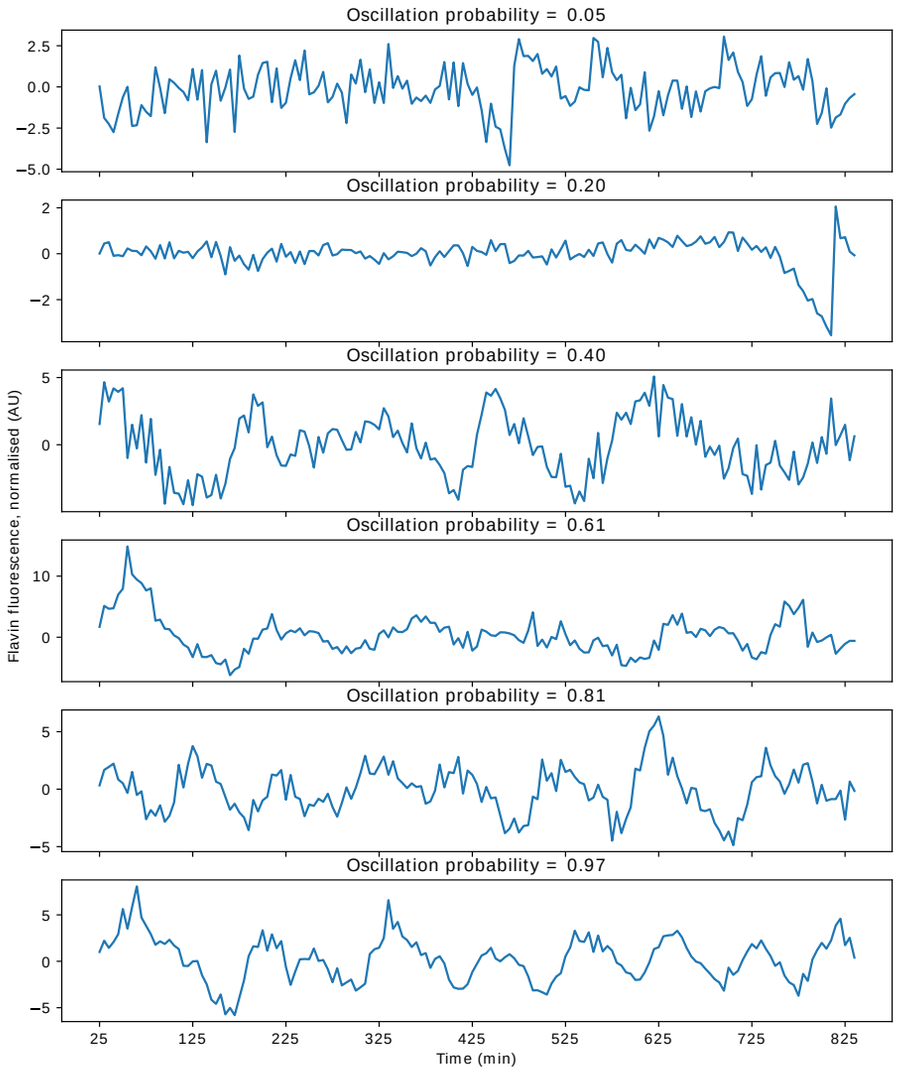


Figure 4.13: Sample time series arranged by probability that each is oscillatory, as predicted by the support vector classifier.

To predict the probability that each time series was oscillatory, I extended the support vector classifier for this purpose using Platt scaling (Platt, 1999). Fig. 4.12a suggests that the classifier performed well in discriminating between the two classes, as evidenced by the U-shaped histogram of probabilities, in contrast to the control (Fig. 4.12b). In addition, these probabilities can also serve as a score to rank time series by oscillation quality (Fig. 4.13).

4.5 Period estimation using the autocorrelation function

Estimating the period of oscillatory time series is important as it provides a quantitative measure of how yeast metabolic cycles respond to genetic nutrient perturbations. To show that the autocorrelation function can be used to estimate the period and noise properties of both symmetric and asymmetric oscillations, I adapted the autocorrelation function as used by Pietsch et al. (2023) (section 2.4.3). To calibrate the method, I generated synthetic oscillations — sinusoids and the FitzHugh-Nagumo oscillator (FitzHugh, 1961) — to investigate the effect of their properties on the autocorrelation function (section 2.4.4). Subsequently, I applied the autocorrelation function to characterise experimentally-recorded time series.

4.5.1 Synthetic data

Harmonic oscillator: effect of noise parameters

To compare the effect of Gaussian noise and Gillespie noise on the autocorrelation function, I computed the autocorrelation functions from a population of sinusoids with either type of noise added via elementwise sums.

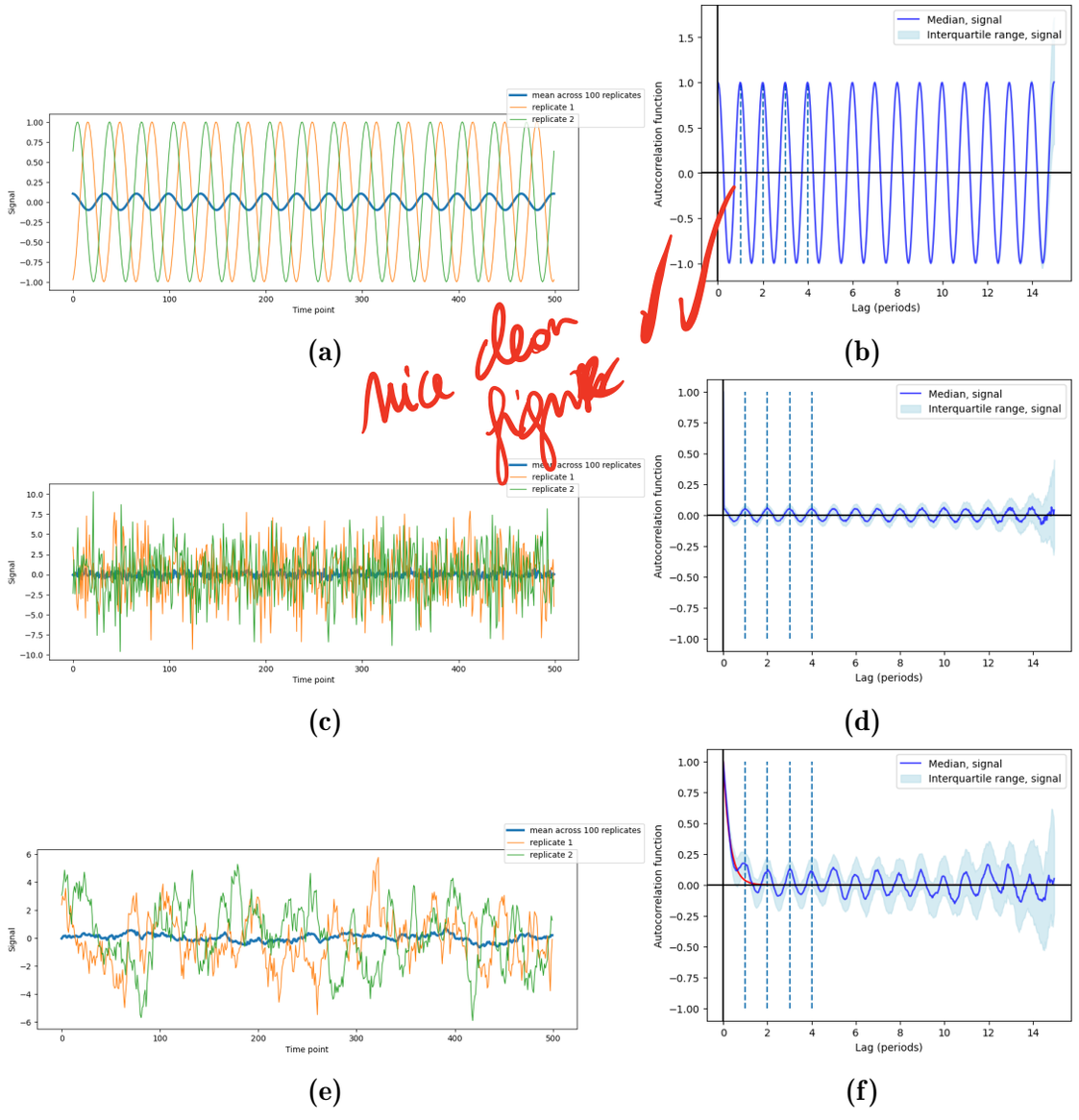


Figure 4.14: **(4.14a)** Sample sinusoids without noise, and **(4.14b)** its autocorrelation function. **(4.14c)** Sample sinusoids with Gaussian noise defined by drawing samples from $\mathcal{N}(0, \sigma^2 = 3)$, and **(4.14d)** its autocorrelation function. **(4.14e)** Sample sinusoids of with Gillespie noise ($k_0 = 5$ and $d_0 = 0.05$), and **(4.14f)** its autocorrelation function. Red line is defined by $y = \mathrm{e}^{-2d_0 T}$, where T represents the lag in units of period of the sinusoids. For each case, the frequency of the sinusoids was 0.03, and there were 100 repeats, randomly out-of-phase.

Fig. 4.14b shows that the autocorrelation function computed from a population of out-of-phase sinusoids could be modelled by a cosine with the same period as the sinusoids. Following this, Fig. 4.14d shows that the addition of Gaussian noise preserved the point $(0, 1)$, but the amplitude of the cosine that models the autocorrelation function was decreased. Furthermore, the variation of the autocorrelation function among time series at long lags was increased, as evidenced by the interquartile range, because less data was used to compute the autocorrelation function at longer lags.

Gillespie noise is based on the birth-death process, and its two parameters control noise parameters (section 2.4.4). Specifically, given a birth rate k_0 and a death rate d_0, the noise has a standard deviation of noise amplitude $A = \sqrt{k_0/d_0}$ and noise timescale $\tau = 1/d_0$. Fig. 4.14f shows that when Gillespie noise was added to the sinusoids, the medium autocorrelation followed the exponential decay function $y = \mathrm{e}^{-2d_0T}$, where T represents lag. In addition, the locations of the peaks of the autocorrelation function were preserved. The observation thus suggests that the death rate d_0 parameter of Gillespie noise controlled the shape of the autocorrelation function.

To quantify the effect of the noise timescale on the shape of the autocorrelation function, I varied the death rate parameter d_0 when generating Gillespie noise. Fig. 4.15 shows that a higher death rate decreased the decay timescale of the autocorrelation function (Fig. 4.15b), while a lower death rate introduced long-term trends in the simulated signals (Fig. 4.15c). A lower death rate also increased the variation between autocorrelation functions between replicates (Fig. 4.15d).

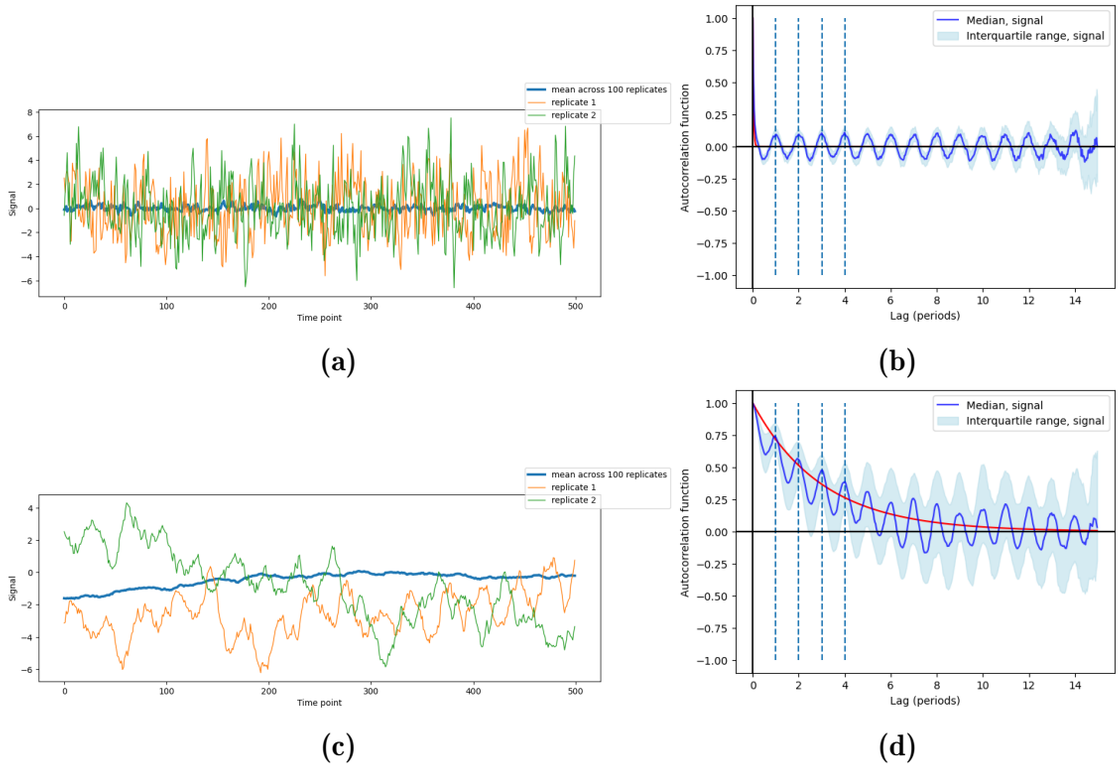


Figure 4.15: (4.15a) Sample sinusoids with Gillespie noise ($k_0 = 5$ and $d_0 = 0.5$), and **(4.15b)** its autocorrelation function. **(4.15c)** Sample sinusoids with Gillespie noise ($k_0 = 5$ and $d_0 = 0.005$), and **(4.15d)** its autocorrelation function. Red lines are defined by $y = \mathrm{e}^{-2d_0T}$, where T represents the lag in units of period of the sinusoids. For each case, the frequency of the sinusoids was 0.03, and there were 100 repeats, randomly out-of-phase.

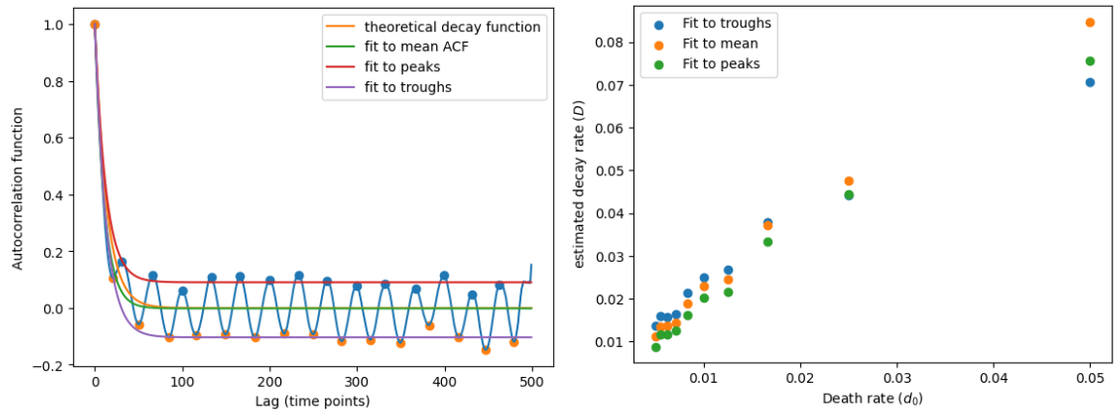


Figure 4.16: (4.16a) Fitting exponential decay functions to estimate d_0 from the autocorrelation function. **(4.16b)** The relationship between d_0 and the decay rate D found from fitting exponential decay functions to the mean autocorrelation function, the peaks, and the troughs of this mean function. Here, k_0 was held constant at 5.

To show how d_0 can be estimated from the autocorrelation function, I fitted exponential decay functions to the mean autocorrelation, the peaks of the mean function, and the troughs of the mean functions using non-linear least squares fitting (Fig. 4.16a). The functions were of the form:

$$y = (1 - C)\mathrm{e}^{-DT} + C \tag{4.2}$$

where y represents the vertical axis (autocorrelation axis), T represents the lag multiples of the period of the sinusoid oscillator, and with C and D being variable parameters whose values are to be determined.

Fig. 4.16b suggests that the decay rates D of the autocorrelation function increases linearly with d_0. In other words, the death rate d_0 controls the decay rate D of the autocorrelation function. Conversely, if D can be estimated from the autocorrelation function, then the noise timescale of the time series can be estimated.

To quantify the effect of the noise amplitude on the autocorrelation function, I varied the birth rate parameter k_0 when generating Gillespie noise. Fig. 4.17 shows that a higher birth rate increased the amplitude of noise (Fig. 4.17a) and increased the variation between replicate autocorrelation functions (Fig. 4.17b), while the opposite was true for a higher birth rate (Figs. 4.17c–4.17d).

To show how the noise amplitude can be estimated from the autocorrelation function, I fit exponential decay functions (Eq. 4.2) as in Fig. 4.16a. Fig. 4.18 suggests that the y-displacements C of the exponential fits to peaks and troughs converged to 0 as k_0/d_0 increased, showing that the amplitude of the oscillations in the autocorrelation function decreases as the noise amplitude increases. In other

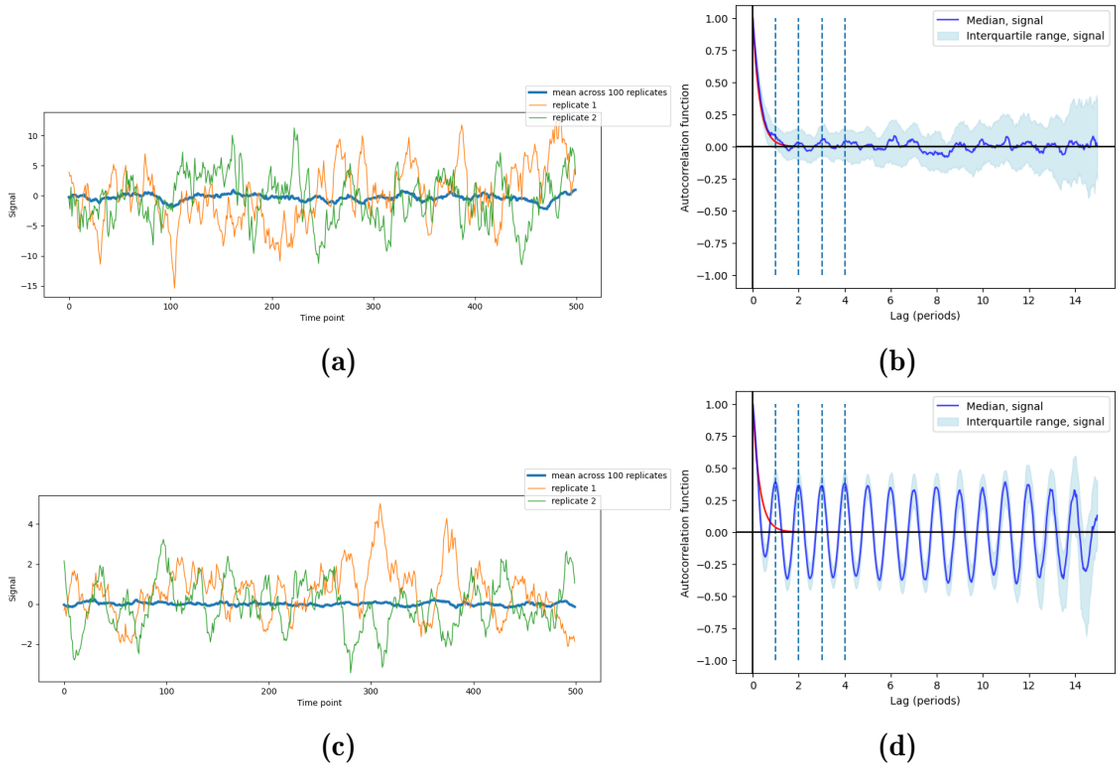


Figure 4.17: (4.17a) Sample sinusoids with Gillespie noise ($k_0 = 25$ and $d_0 = 0.05$), and **(4.17b)** its autocorrelation function. **(4.17c)** Sample sinusoids with Gillespie noise ($k_0 = 1$ and $d_0 = 0.05$), and **(4.17d)** its autocorrelation function. Red lines are defined by $y = \mathrm{e}^{-2d_0T}$, where T represents the lag in units of period of the sinusoids. For each case, the frequency of the sinusoids was 0.03, and there were 100 repeats, randomly out-of-phase.

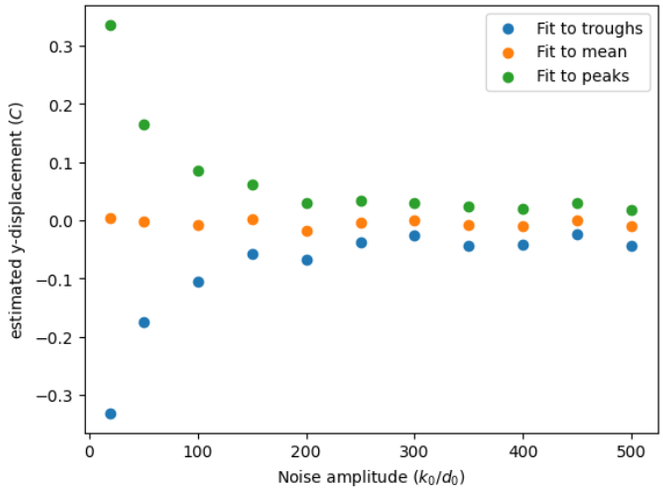
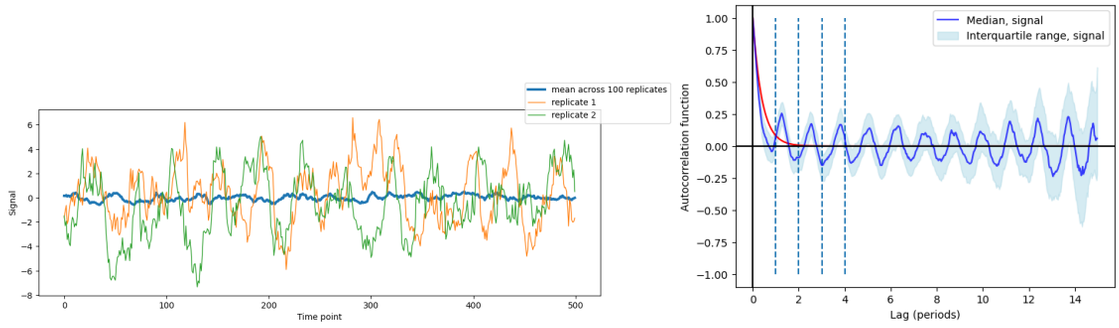


Figure 4.18: The relationship between the noise amplitude and the y-displacement C found from fitting exponential decay functions to the mean autocorrelation function, the peaks, and the troughs of this mean function. Here, d_0 was held constant at 0.05.



(a) **(b)**

Figure 4.19: (4.19a) Sample FitzHugh-Nagumo oscillators ($RI_{\text{ext}} = 0.4$, $\tau = 12.5$, $a = 0.7$, $b = 0.82$) with Gillespie noise ($k_0 = 5$ and $d_0 = 0.05$), and **(4.19b)** its autocorrelation function. Red line is defined by $y = \mathrm{e}^{-2d_0T}$, where T represents the lag in units of period of the sinusoids. There were 100 repeats, randomly out-of-phase.

words, the birth rate d_0 controls the y-displacement parameter C of the autocorrelation function, which is a proxy for the function's amplitude. Conversely, if C can be estimated from the autocorrelation function, then the noise amplitude of the time series can be estimated.

4.5.2 FitzHugh-Nagumo oscillator: effect of oscillator shape

To test whether Gillespie noise parameters can be estimated from the autocorrelation function computed from an asymmetric oscillation, I repeated the exponential fitting in section 4.5.1 on the FitzHugh-Nagumo oscillator.

Fig. 4.19b shows that when the oscillator had a different shape, the waves in the autocorrelation function changed shape, becoming more pointed. In addition, Fig. 4.20b suggests that fitting an exponential decay function to the autocorrelation function to estimate noise parameters was less reliable, particularly with high noise timescales. As a consequence, estimating the noise timescale τ from the

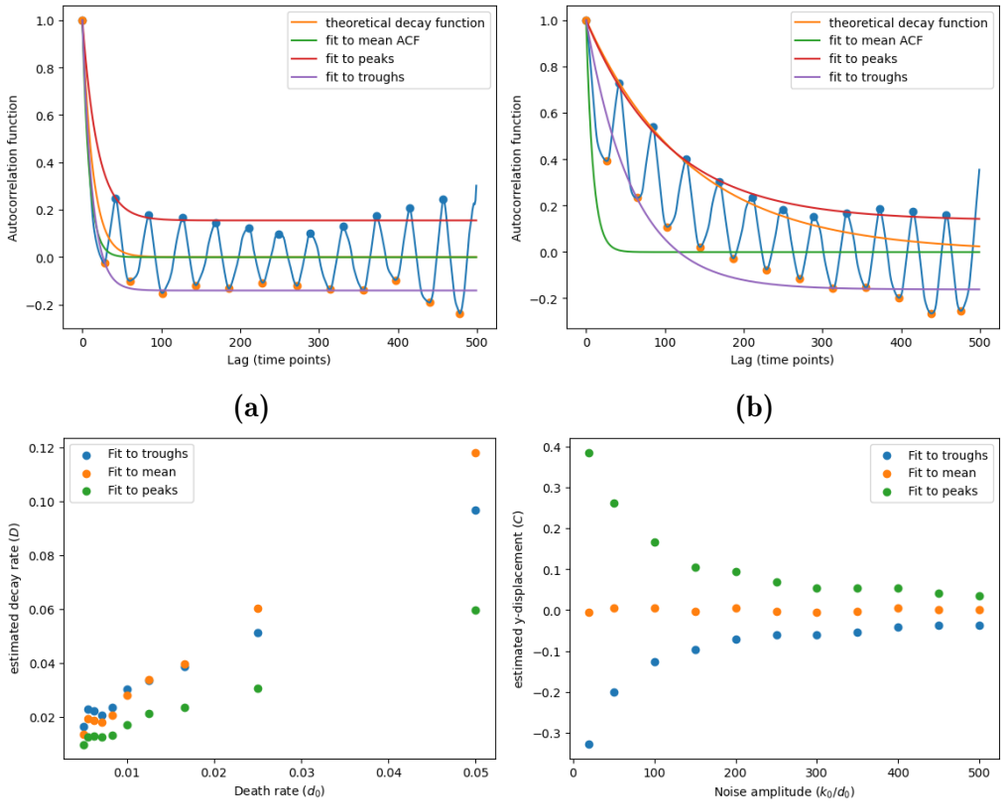


Figure 4.20: (4.20a) Fitting exponential decay functions of the form $y = (1 - C)\mathrm{e}^{-DT} + C$, with C and D as variable parameters, to the autocorrelation function, generated with $k_0 = 5$ and $d_0 = 0.05$, and **(4.20b)** with $k_0 = 5$ and $d_0 = 0.005$. **(4.20c)** The relationship between d_0 and the decay rate D ($k_0 = 5$). **(4.20d)** The relationship between k_0/d_0 and y-displacement C of the exponential fit ($d_0 = 0.05$)

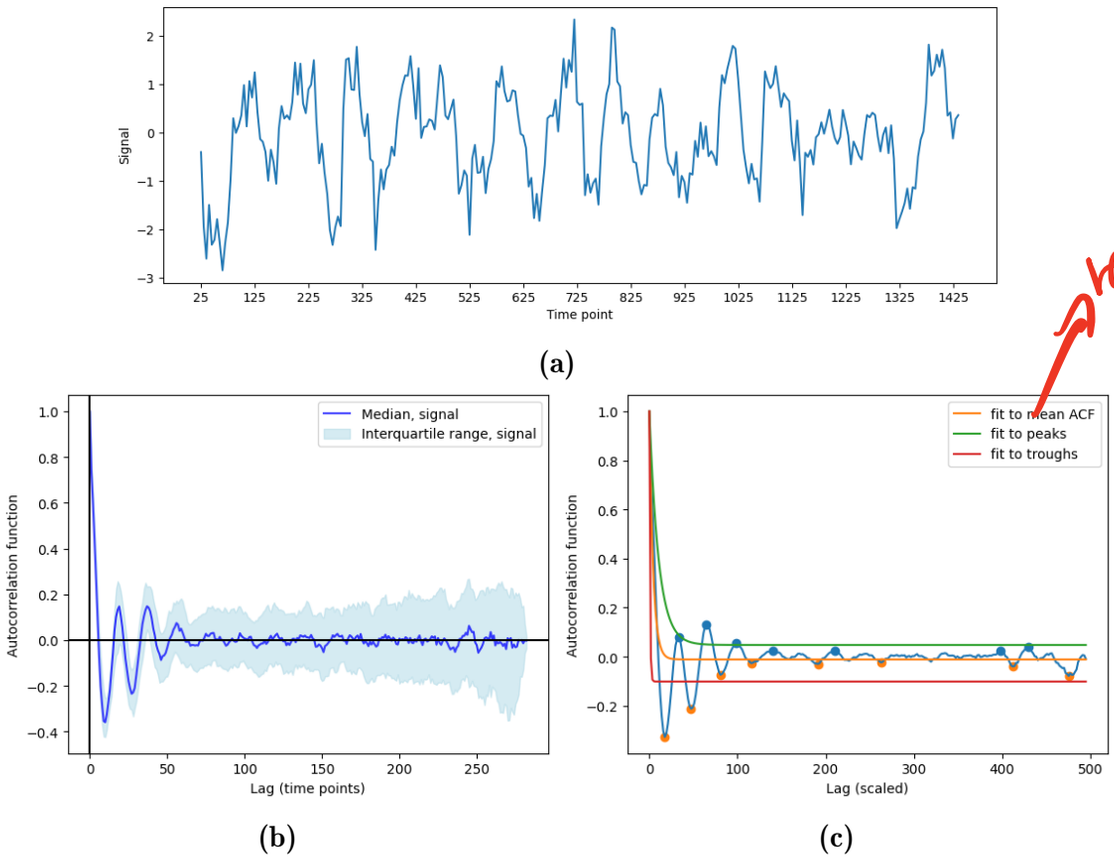


Figure 4.21: (4.21a) Sample time series of flavin autofluorescence. **(4.21b)** Autocorrelation function across a population of time series of flavin autofluorescence. **(4.21b)** Fitting exponential decay functions to determine noise parameters, lag axis scaled to match Fig. 4.16.

decay rate D of the exponential decay function produced a greater range of uncertainty (Fig. 4.20c). However, estimating the noise amplitude A based on the y-displacement C of the exponential decay function remained as reliable as the sinusoid oscillator case (Fig. 4.20d).

4.5.2 Real data

To deduce the period and noise parameters of a experimentally-recorded sinusoid-like signal (Fig. 4.21a), I computed the autocorrelation functions of a population of flavin autofluorescence time series (Fig. 4.21b), and fitted exponential functions to the mean autocorrelation function, scaled to have amplitudes and periods fit

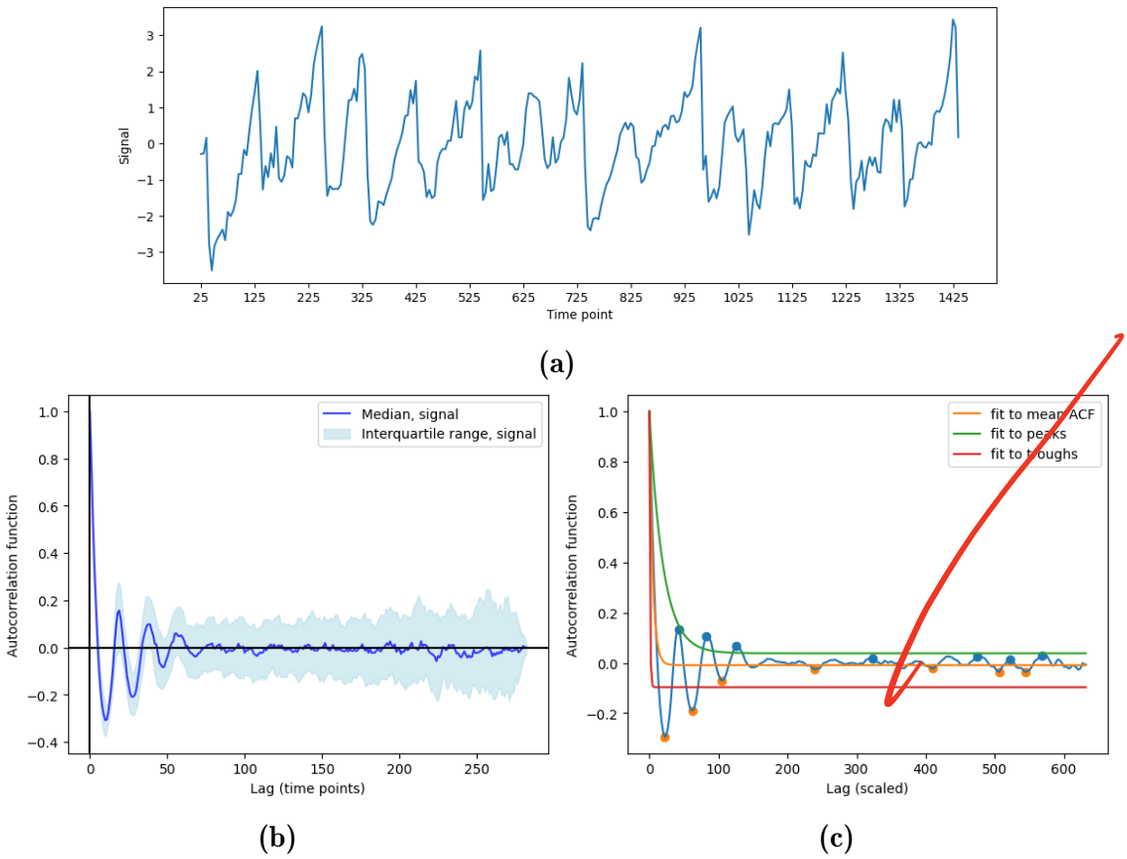


Figure 4.22: (4.22a) Sample time series of histone 2B abundance. **(4.22b)** Autocorrelation function across a population of time series of histone 2B abundance. **(4.22b)** Fitting exponential decay functions to determine noise parameters.

the simulated sinusoids (Fig. 4.21c). The autocorrelation function suggests an average period of 19 time points, corresponding to 95 min, as expected from the nutrient conditions. However, estimation of noise parameters was complicated by the damping in the autocorrelation function, giving a different shape compared to the synthetic data and fewer peaks and troughs for fitting. Nevertheless, the decay rate D of the exponential function fitted to the mean autocorrelation function suggested a noise timescale of 7.16 and the y-displacement C of the exponential function fitted to the peaks of the autocorrelation function suggested a noise amplitude of 161.84.

Similarly, to deduce the period and noise parameters of a experimentally-recorded asymmetric signal (Fig. 4.22a), I computed the autocorrelation functions of a population of histone 2B abundance time series (Fig. 4.22b), and fitted exponential functions to the mean autocorrelation function, scaled to have amplitudes and periods fit the simulated FitzHugh-Nagumo oscillators (Fig. 4.22c). The autocorrelation function also suggests an average period of 19 time points, corresponding to 95 min. As was the case for the flavin autofluorescence time series, the damping in the autocorrelation function complicated estimation of noise parameters; though the decay rate D of the exponential function fitted to the mean autocorrelation function suggested a noise timescale of 11.19 and the y-displacement C of the exponential function fitted to the peaks of the autocorrelation function suggested a noise amplitude of 292.41. The differences of these noise parameters relative to the flavin autofluorescence time series suggest different noise properties, which can be explained by the different fluorescence channels and exposure times used to generate each type of signal.

4.6 Detection of synchrony

To test a method to detect the synchrony and quantify the temporal lag between two types of oscillations, I computed the cross-correlation functions of a population of sinusoid and FitzHugh-Nagumo oscillators. Cross-correlation has been used to investigate the relationship between the expression levels of two genes in a model feed-forward loop (Dunlop et al., 2008), and to investigate the relationship between instantaneous growth rate and the expression of *lac* genes of enzymes in central metabolism across a population of *E. coli* cells (Kiviet et al., 2014).

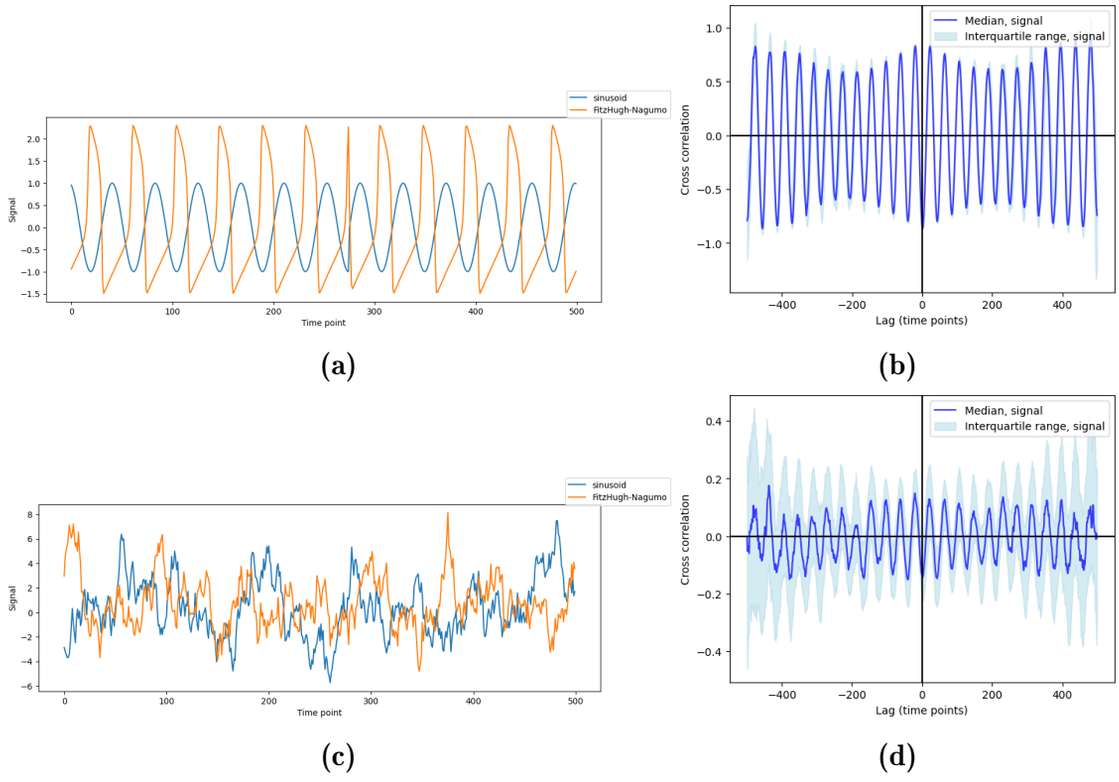
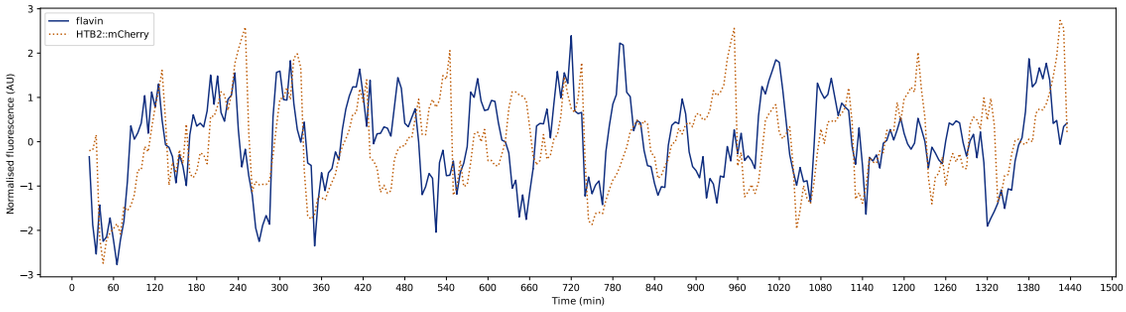


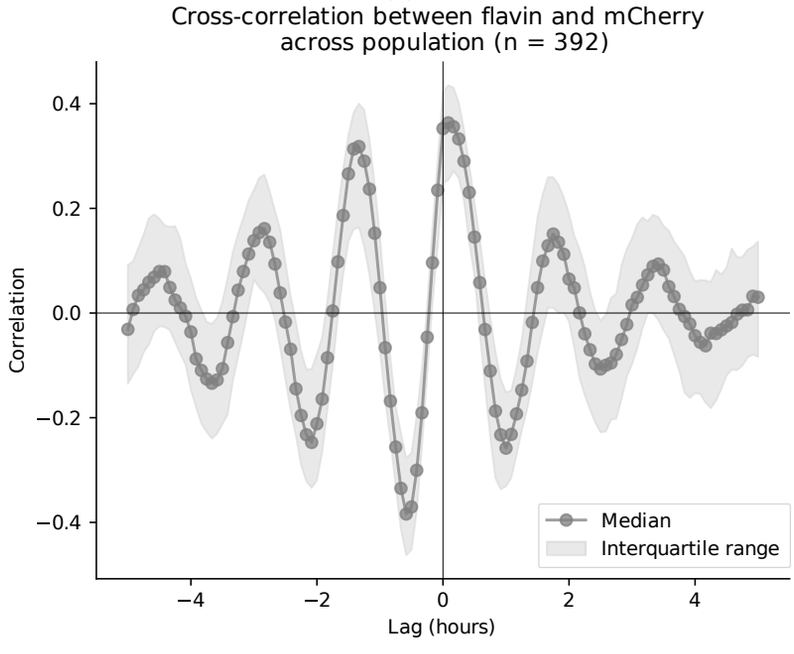
Figure 4.23: (4.23a) (Blue) Sample sinusoid ($f = 0.0235$) and (orange) FitzHugh-Nagumo oscillator ($RI_{\text{ext}} = 0.4$, $\tau = 12.5$, $a = 0.7$, $b = 0.82$) of the same frequency and without noise, and **(4.23b)** the cross-correlation function of the FitzHugh-Nagumo oscillators with respect to the sinusoids. **(4.23c)** (Blue) Sample sinusoid and (orange) FitzHugh-Nagumo oscillator with same parameters as 4.23a, but with Gillespie noise ($d_0 = 0.05$, $k_0 = 5$), and **(4.23d)** the cross-correlation function of the FitzHugh-Nagumo oscillators with respect to the sinusoids. For each case, there were 400 repeats, randomly out-of-phase.

4.6.1 Synthetic data

Fig. 4.23b shows that the cross-correlation function identifies that the sinusoids, on average, peaked 20 time points before the FitzHugh-Nagumo oscillators, close to the actual value of 20.75 time points. This shift was evidenced by the position of the peak of the cross-correlation function closest to the vertical axis. The cross-



(a)



(b)

Figure 4.24: (4.24a) Sample time series of flavin autofluorescence (blue) and histone 2B abundance (orange). **(4.24b)** Cross-correlation function between the flavin autofluorescence time series and the histone 2B abundance time series.

correlation function further showed that synchrony between the two oscillators was maintained along the entire time series, across all time series. Furthermore, Fig. 4.23d suggests that even with strong Gillespie noise, the lag between the two oscillators could still be deduced from the cross-correlation function.

4.6.2 Real data

To show how the cross-correlation function can be used to quantify the synchrony between flavin autofluorescence oscillations and HTB2::mCherry levels in a population of cells, Fig. 4.24 displays a sample pair of time series (Fig. 4.24a) and the cross-correlation function from the population of cells (Fig. 4.24b). The cross-correlation function suggests that the histone 2B oscillations peaked after the flavin autofluorescence oscillations by an average of 5 min.

4.7 Discussion

This chapter discusses methods to filter long-term trends in time series data, to visualise structures within a dataset of time series, to detect rhythmicity in a time series, to estimate period and noise parameters, and to detect the synchrony between two time series.

My results suggest that using a high-pass Butterworth filter to filter out long-term trends in time series data gives better control over the frequency profile of the time series than moving-average methods, which is often used to detrend time series from biological oscillators. Such results highlight that a degree of caution is needed to choose methods for a crucial step in data analysis.

My exploration of UMAP and modularity clustering suggests that both methods were useful to discovering structure within time series, particularly in discriminating between oscillatory and non-oscillatory time series. These methods further indicated sub-groups of time series that may have similar properties, such as shape or oscillation quality, which may correspond to sub-populations of metabolic cycle-producing cells in a culture. The consistency between the two methods strongly suggest that such groups in the dataset are meaningful.

Subsequently, my exploration of three approaches to detect rhythmicity — deriving a statistical test of a power spectrum, deriving a periodogram from an autoregressive model, and a support vector classifier — highlights the difficulty of rhythmicity detection in noisy biological time series. The spectral method described by (Glynn et al., 2006) had a modest performance. The autoregressive model was able to identify the most likely period in some time series, but otherwise classified most time series as non-oscillatory, and lacked a tuning parameter. Finally, the support vector classifier suggests that a simple machine learning model could be adapted for rhythmicity detection, subject to a good feature set and training data.

Ultimately, rhythmicity detection requires supplying a threshold in some form, be it a range of frequencies in which oscillations are expected (Zielinski et al., 2014), a parameter that controls the proportion of time series detected as oscillatory, or training labels. This is because there is no way to objectively specify a failure rate for a rhythmicity detection method as there is no independent method to estimate rhythmicity (Zielinski et al., 2014).

My observations concerning the autocorrelation function confirms its use for estimating the period of an oscillatory time series, as used previously by Papagiannakis et al. (2017). As periodicity-estimation methods have a limited ability to estimate the period of short, noisy time series owing to little input data, combining several such methods can be useful to produce a picture of the periodicity of oscillatory time series. For example, Potvin-Trottier et al. (2016) combines the autocorrelation function and the Fourier transform to study the changes in the periodicity of a modified model of the repressilator.

Furthermore, I showed that the autocorrelation function may be used to estimate parameters to describe noise, assuming that the noise can be modelled by the birth-death process. However, further work, such as synthetic time series generated from a wider variety of parameters or additional estimation methods are likely required for adequate estimation of noise parameters from real data. In addition, it is possible that other types of noise better describe the noise from real data, and such types of noise may lead to different effects on the autocorrelation function.

Finally, my results show that the cross-correlation function, as used by Dunlop et al. (2008), Kiviet et al. (2014), and Pietsch et al. (2023), can be used to detect synchrony between two sets of time series and to quantify the temporal relationship between the time series, even if the time series are very noisy.

Taken together, the analysis methods discussed in this chapter can form the basis of a powerful data analysis pipeline to analyse large datasets of potentially oscillatory biological time series.