Optimal time frequency analysis for biological data - pyBOAT

Gregor Mönke[1], Frieda A. Sorgenfrei[2], Christoph Schmal[3,+], and Adrián E. Granada[4,+]

[1]European Molecular Biology Laboratory, Meyerhofstraße 1, 69117 Heidelberg, Germany
[2]Austrian Centre of Industrial Biotechnology c/o University of Graz, Institute of Chemistry, NAWI Graz, BioTechMed Graz, Heinrichstrasse 28, 8010 Graz, Austria
[3]Institute for Theoretical Biology, Humboldt Universität zu Berlin, Philippstraße 13, 10115 Berlin, Germany
[4]Charité Comprehensive Cancer Center, Charité Universitätsmedizin Berlin, Charitéplatz 1, 10117 Berlin, Germany.
[+]These authors contributed equally to the work.
[*]To whom the correspondence should be addressed:
gregor.moenke@embl.de,
adrian.granada@charite.de

Abstract

Methods for the quantification of rhythmic biological signals have been essential for the discovery of function and design of biological oscillators. Advances in live measurements have allowed recordings of unprecedented resolution revealing a new world of complex heterogeneous oscillations with multiple noisy non-stationary features. However, our understanding of the underlying mechanisms regulating these oscillations has been lagging behind, partially due to the lack of simple tools to reliably quantify these complex non-stationary features. With this challenge in mind, we have developed pyBOAT, a Python-based fully automatic stand-alone software that integrates multiple steps of non-stationary oscillatory time series analysis into an easy-to-use graphical user interface. pyBOAT implements continuous wavelet analysis which is specifically designed to reveal time-dependent features. In this work we illustrate the advantages of our tool by analyzing complex non-stationary time-series profiles. Our approach integrates data-visualization, optimized sinc-filter detrending, amplitude envelope removal and a subsequent continuous-wavelet based time-frequency analysis. Finally, using analytical considerations and numerical simulations we discuss unexpected pitfalls in commonly used smoothing and detrending operations.

1 Introduction

Oscillatory dynamics are ubiquitous in biological systems. From transcriptional to behavioral level these oscillations can range from milliseconds in case of neuronal firing patterns, up to years for the seasonal growth of trees or migration of

birds (Goldbeter et al. [2012], Gwinner [2003], Rohde and Bhalerao [2007]). To gain biological insight from these rhythms, it is often necessary to implement time-series analysis methods to detect and accurately measure key features of the oscillatory signal. Computational methods that enable analysis of periods, amplitudes and phases of rhythmic time series data have been essential to unravel function and design principles of biological clocks (Lauschke et al. [2013], Ono et al. [2017], Soroldoni et al. [2014]). Here we present `pyBOAT`, a framework and software package with a focus on usability and generality of such analysis.

Many time series analysis methods readily available for the practitioner rely on the assumption of stationary oscillatory features, i.e. that oscillation properties such as the period remain stable over time. A plethora of methods based on the assumption of stationarity have been proposed which can be divided into those working in the frequency domain such as Fast Fourier transforms (FFT) or Lomb-Scargle periodograms (Lomb [1976], Ruf [1999]) and those working in the time domain such as autocorrelations (Westermark et al. [2009]), peak picking (Abraham et al. [2018]) or harmonic regressions (Edwards et al. [2010], Halberg et al. [1967], Naitoh et al. [1985], Straume et al. [1991]). In low noise systems with robust and stable oscillations, these stationary methods suffice to reliably characterize oscillatory signals. Recordings of biological oscillations frequently exhibit noisy and time-dependent features such as drifting period, fluctuating amplitude and trend. Animal vocalization (Fitch et al. [2002]), temporal changes in the activatory pathways of somitogenesis (Tsiairis and Aulehla [2016]) or reversible and irreversible labilities of properties in the circadian system due to aging or environmental factors (Pittendrigh and Daan [1974], Scheer et al. [2007]) are typical examples where systematic, often non-linear changes in oscillation periods occur. In such cases, the assumption of stationarity is unclear and often not valid, thus the need to use nonstationary-based methods that capture time-dependent oscillatory features. Recently, the biological data analysis community has developed tools that implement powerful methods tailored to specific steps of time-series analysis such as rhythmicity detection (Hughes et al. [2010], Thaben and Westermark [2014]), de-noising and detrending, and the characterization of nonstationary oscillatory components (Leise [2013], Price et al. [2008]).

To extract time-dependent features of non-stationary oscillatory signals, methods can be broadly divided into those that rely on operations using a moving time-window (e.g. wavelet transform) and those that embeds the whole time series into a phase space representation (e.g. Hilbert transform). These two families are complementary, having application-specific advantages and disadvantages, and in many cases both are able to provide equivalent information about the signal (Quiroga et al. [2002]). Due to the inherent robustness in handling noisy oscillatory data and its interpretability advantages, we implemented at the core of `pyBOAT` a continuous-wavelet-transform approach.

As a software package `pyBOAT` combines multiple steps in the analysis of oscillatory time series in an easy-to-use graphical user interface that requires no prior programming knowledge. With only two user-defined parameters, `pyBOAT` is able to proceed without further intervention with optimized detrending, amplitude envelope removal, spectral analysis, detection of main oscillatory components

(ridge detection), oscillatory parameters readout and visualization plots (Figure 1A). `pyBOAT` is developed under an open-source license, is freely available for download and can be installed on multiple operatings systems.

In the first section of this work we lay out the mathematical foundations at the core of `pyBOAT`. In the subsequent section 3 we describe artifacts generated by the widely used smoothing and detrending techniques and how they are resolved within `pyBOAT`. In section 4 we describe the theory behind spectral readouts in the special case of complex amplitude envelopes. We finalize this manuscript with a short description of the user interface and software capabilities.

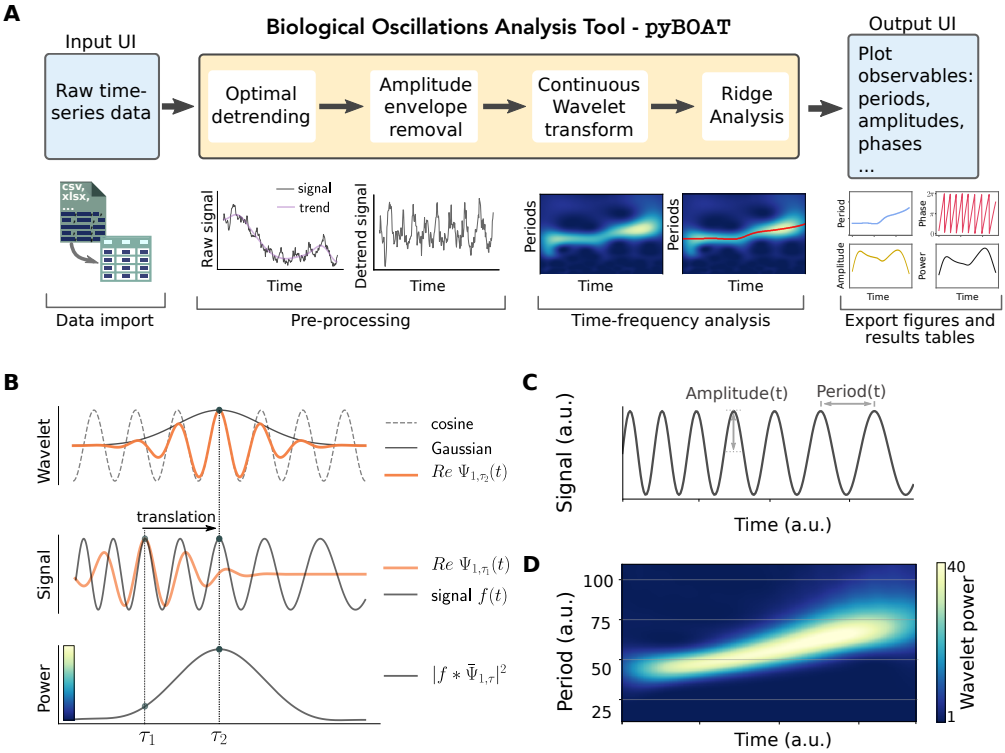

Figure 1: **Overview of the workflow implemented in pyBOAT** A) Schematic showing the proposed and implemented analysis steps. B) *Top panel*: A cosine function (dotted gray line) multiplied by a Gaussian envelope (bold black line) defines the Morlet Wavelet (orange line), here depicted for scale $s = 1$ and localized at $\tau = t_2$. *Middle panel*: The same Morlet Wavelet, translated in time with $\tau = t_1$, is shown together with a sweeping signal whose instantaneous period coincides with the Morlet of scale $s = 1$ exactly at τ_2. *Bottom panel*: Result of the convolution with the sliding Morlet $\Psi_{1,\tau}(t)$ along signal $f(t)$. The power quickly decreases away from τ_2. The curve corresponds to one row in the Wavelet power spectrum of panel D). C) Synthetic signal with periods sweeping from $T_1 = 30$s ($f_1 \approx 0.033$Hz) to $T_2 = 70$s ($f_2 \approx 0.014$Hz). D) Wavelet power spectrum shows time-resolved (instantaneous) periods.

2 Basic Wavelet Theory

In this section we aim to lay down the basic principles of wavelet analysis as employed in our signal analysis tool, albeit the more mathematical subtleties are moved to the Appendix.

Introduction to the Wavelet transform

The classic approach to do frequency analysis of periodic signals is the well-known Fourier analysis. Its working principle is the decomposition of a signal $f(t)$ into Sines and Cosines, known as basis functions. These harmonic components have no *localization in time* but are sharply *localized in frequency*: each harmonic component carries exactly one frequency effective everywhere in time. Thus, the straighforward Fourier analysis underperforms in cases of time-dependent oscillatory features, such as when the period of the oscillation changes in time (Figure 1C). The goal behind wavelets is to reach an optimal compromise between time and frequency localization (Gabor [1946]). Gabor introduced a Gaussian modulated harmonic component, also known as *Morlet wavelet*:

$$\Psi(t) = \pi^{-1/4}\, e^{-t^2/2}\, e^{i\omega_0 t} \tag{1}$$

$$= \pi^{-1/4}\, e^{-t^2/2}\, \left[cos(\omega_0 t) + i\, sin(\omega_0 t)\right] \tag{2}$$

The basis harmonic functions for time-frequency analysis are then generated from the *mother wavelet* by *scaling* and *translation*:

$$\Psi_{s,\tau}(t) = s^{-1/2}\, \Psi\left(\frac{t-\tau}{s}\right) \tag{3}$$

Varying the time localization τ slides the wavelet left and right on the time axis. Scale s changes the *center frequency* of the Morlet wavelet according to $\omega_{center}(s) = \omega_0/s$ (see also Appendix equation (8)). Higher scales therefore generate wavelets with lower center frequency. The Gaussian envelope suppresses the harmonic component with frequency ω_{center} farther away from τ, therewith localizing the wavelet in time (Figure 1B top panel). This frequency $\omega_{center}(s)$ is conventionally taken as the Fourier equivalent (or pseudo-) frequency of a Morlet wavelet with scale s. It is noteworthy to state that wavelets are in general not as sharply localized in frequency as their harmonic counterparts (Figure S1). This is a trade-off imposed by the uncertainty principle to gain localization in time (Gröchenig [2013]).

The wavelet transform of a signal $f(t)$ is given by the following integral expression:

$$\mathcal{W}[f](\tau, s) = \int_{-\infty}^{\infty} f(t)\, s^{-1/2}\, \bar{\Psi}\left(\frac{t-\tau}{s}\right) dt \tag{4}$$

$$= f * \bar{\Psi}(\tau, s) \tag{5}$$

For a fixed scale, this equation has the form of a convolution as denoted by

the '$*$' operator, where $\bar{\Psi}$ denotes the complex conjugate of Ψ. For an intuitive understanding it is helpful to consider above expression as the cross-correlation between the signal and the wavelet of scale s (or center frequency $\omega_{center}(s)$). The translation variable τ slides the wavelet along the signal. Since the wavelet decays fastly away from τ, only the instantaneous correlation of the wavelet with center frequency ω_{center} and the signal around τ significantly contributes to the integral (Figure 1B middle and lower panel). By using an array of wavelets with different frequencies (or periods), this allows to scan for multiple frequencies in the signal in a time-resolved manner.

The result of the transform: $\mathcal{W} : f(t) \rightarrow \widetilde{f}(t, \omega)$ is a complex valued function of two variables, frequency ω and time localization τ. In the following, we implicitly convert scales to frequency via the corresponding central frequencies $\omega_{center}(s)$ of the Morlet wavelets. To obtain a physically meaningful quantity, one defines the *wavelet power spectrum*:

$$P_f^{\mathcal{W}}(\tau, \omega) = \frac{|\widetilde{f}(\tau, \omega)|^2}{\sigma^2} \tag{6}$$

We adopted the normalization with the variance of the signal σ^2 from Torrence and Compo [1998] as it allows for a natural and statistical interpretation of the wavelet power. By stacking the transformations in a frequency order, one constructs a two-dimensional time-frequency representation of the signal, where the power itself is usually color coded (Figure 1D), using a dense set of frequencies to scan for approximates of the *continuous Wavelet transform*.

Wavelet Power Interpretation

It is important to note that the *time averaged* wavelet power spectrum

$$\int_{-\infty}^{+\infty} P_f^{\mathcal{W}}(\tau, \omega) d\tau \rightarrow P_f^{\mathcal{F}}(\omega) \tag{7}$$

is an unbiased estimator for the true Fourier power spectrum $P_f^{\mathcal{F}}$ of a signal f (Percival [1995]).

This allows to directly compare Fourier power spectra to the wavelet power. White noise is the simplest noise process which may serve as a null hypothesis. Normalized by variance white noise has a flat mean Fourier power of one for all frequencies. Hence, also the variance normalized Wavelet power of one corresponds to the mean expected power for white noise: $P_{WN}^{\mathcal{F}}(\omega) = 1$ (Figure 2C). This serves as a universal unit to compare different empirical power spectra.

Extending these arguments to random fluctuations of the Fourier spectrum allows for the calculation of confidence intervals on wavelet spectra. If a background spectrum $P_0^{\mathcal{F}}(\omega)$ is available, the confidence power levels can be easily calculated as:

$$C(\omega) = \frac{1}{2} P_0^{\mathcal{F}}(\omega) \chi^2 \tag{8}$$

Assuming normality for the distribution of the complex Fourier components of the background spectrum, one can derive that the power itself is chi-square dis-

tributed (Chatfield [1995]). Thus, picking a desired confidence (e.g. $\chi^2(95\%) \approx 6$) gives the scaling factor for the background spectrum. Only Wavelet powers greater than this confidence level are then considered to indicate oscillations with the chosen confidence. The interested reader may find more details in section 4 of Torrence and Compo [1998].

A wavelet power of $C = 3$ corresponds to the 95% confidence interval in case of white noise (Figure 2B), which is frequency independent. For the practitioner, this should be considered the absolute minimum power, required to report 'oscillations' in a signal. It should be noted that especially for biological time series, due to correlations present also in non-oscillatory signals, white noise often is a poor choice for a null model (see also supplementary information A.3). A possible solution is to estimate the background spectrum from the data itself, this however is beyond the scope of this work.

3 Optimal Filtering - Do's and Dont's

A biological recording can be decomposed into components of interest and those elements which blur and challenge their analysis, most commonly noise and trends. Various techniques for *smoothing* and *detrending* have been developed to deal with these issues. Often overlooked is the fact that both, *smoothing* and *detrending* operations can introduce spectral biases, i.e. attenuation and amplification of certain frequencies. In this section we lay out a mathematical framework to understand and compare the effects of these two operations, showing examples of the potential pitfalls and at the same time providing a practical guide to avoid these issues. Finally, we discuss how `pyBOAT` minimizes most of these common artifacts.

Smoothing

The operation which removes the fast, high-frequency (low period) components of a signal is colloquially called *smoothing*. This is most commonly done as a sliding time window operation (convolution). In general terms we can refer to a window function $w(t)$ such that the smoothed signal is given as:

$$f_S(t) = f(t) * w(t). \tag{9}$$

By employing the convolution theorem, it turns out that the Fourier transformation of the smoothed signal

$$\widehat{f_S}(\omega) = \mathcal{F}[f(t) * w(t)] \tag{10}$$

$$= \widehat{f}(\omega)\ \widehat{w}(\omega) \tag{11}$$

is simply given by the product of the individual Fourier transforms.

It follows that the Fourier power spectrum of the smoothed signal reads as:

$$P^{\mathcal{F}}_{f_S}(\omega) = \frac{|\widehat{f}(\omega)|^2 \, |\widehat{w}(\omega)|^2}{\sigma^2_{f_S}} \tag{12}$$

$$= \frac{\sigma^2_f}{\sigma^2_{f_S}} \, |\widehat{w}(\omega)|^2 \, P^{\mathcal{F}}_f(\omega) \, . \tag{13}$$

Applying a few steps of Fourier algebra shows that the original power spectrum $P^{\mathcal{F}}_f$ gets modified by the *low pass response* of the window function $|\widehat{w}|^2$ scaled by the ratio of variances $\sigma^2_f/\sigma^2_{f_S}$. Also without resorting to mathematical formulas, smoothing and its effect on the time-frequency analysis can be easily grasped visually. A broad class of filtering methods falls into the category of *convolutional filtering*, meaning that there is some operation in a sliding window done to the data, e.g. for *moving average* and *LOESS* or *Savitzky-Golay* filtering (Savitzky and Golay [1964]). Moving average filter is a widely used smoothing technique, defined simply by a box-shaped window that slides in the time domain. In Figure 2 we summarize the spurious effects that this filter can have on noisy biological signals. White noise, commonly used as descriptor for fluctuations in biological systems, is a random signal with no dominant period. The lack of dominant period can be seen from a raw white noise signal (Figure 2A) and more clearly from the almost flat landscape on the power spectrum (Figure 2B). Applying to raw white noise signal a moving average filter of size 5-times the signal's sampling interval ($M = 5\Delta t$) leads to a smoothed noise signal (Figure 2C) that now has multiple dominant periods, as seen by the emergence of high power islands in Figure 2D. Comparing the original spectrum (Figure 2B) with the white noise smoothed spectrum (Figure 2D), it becomes evident that smoothing introduces a strong increase in Wavelet power for longer periods. In other words, smoothing perturbs the original signal by creating multiple high-power islands of long periods, also referred as spurious oscillations. To better capture the statistics behind these smoothing-induced spurious oscillations, it is best to look at the time averaged Wavelet spectrum. Figure 2E shows the mean expected power after smoothing white noise with a moving average filter. A zone of attenuated small periods become visible, the sloppy stopband for periods around $\lesssim 7\Delta t$. These are the fast, high-frequency components which get removed from the signal. However, for larger periods $\gtrsim 10$ the Wavelet power gets amplified up to 5-fold. It is this gain, given by $\sigma^2_f/\sigma^2_{f_S}$, which leads to spurious results in the analysis.

As stated before, variance normalized white noise has a mean power of 1 for all frequencies or periods ($P^{\mathcal{F}}_{WN}(\omega) = 1$). This allows to use a straightforward numerical method to estimate a filter response $|\widehat{w}(\omega)|^2$, i.e. applying the smoothing operation to simulated white noise and time averaging the Wavelet spectra. This Monte Carlo approach works for every (also non-convolutional) smoothing method. Results for the Savitzky-Golay filter applied to white noise signals can be found in Supplementary Figure S2. Convolutional filters will in general produce more gain and hence more spurious oscillations with increasing window sizes in the time domain.

If smoothing, even with a rather small window ($M = 5\Delta t$), already potentially introduces false positive oscillations, what does that mean for practical time-frequency analysis? For Wavelet analysis the answer is plain and clear: smoothing is simply not needed at all. A close inspection of the unaltered white noise Wavelet spectrum shown in Figure 2B, shows the same structures for higher periods as in the spectrum of the smoothed signal (Figure 2D). The big difference is, that even though these random apparent oscillations get picked up by the Wavelets, their low power directly indicates their low significance. As Wavelet analysis (see previous section) is based on convolutions, it already has power preserving *smoothing built in*. As illustration, we show in Figure 2F a raw noisy signal with lengthening period (noisy chirp) and the corresponding power spectrum (Figure 2F lower panel). Without any smoothing the main periodic signal can be clearly identified in the power spectrum. Thus, Wavelet analysis does not require smoothing for the detection of oscillations in very noisy signals. For all other spectrum analysis methods which rely on explicit smoothing, characteristics of the background noise and the signal to noise ratio are crucial to avoid detecting spurious oscillations. These are usually both quantities not readily available *a priori* or in practice.

Detrending

Complementary to smoothing, an operation which removes the slow, low frequency components of a signal is generally called detrending. Strong trends can dominate a signal by effectively carrying most of the variance and power. There are at least two broad classes of detrending techniques: parametric fitting and convolution based. Both aim to estimate the trend as a function over time to be subtracted from the original signal. A parametric fit always is the best choice, if the deterministic processes leading to the trend are known and well understood. An example is the so called photobleaching encountered in time-lapse fluorescent imaging experiments, here an exponential trend can often be well fitted to the data based on first principle deliberations (Song et al. [1995]). However, there are often other slow processes, like cell viability or cells drifting in and out of focus, which usually can't be readily described parametrically. For all these cases convolutional detrending with a window function $w(t)$ is a good option and can be written as:

$$f_d(t) = f(t) - f(t) * w(t). \tag{14}$$

The trend here is nothing less than the smoothed original signal, i.e. $f(t) * w(t)$. However with the signal itself falling into the stop-band of the low-pass filter, with the aim to not capture and subtract any signal components.

Using basic algebra in the frequency domain we obtain an expression relating

the window $w(t)$ to the power spectrum of the original signal $f(t)$:

$$P^{\mathcal{F}}_{f_D}(\omega) = \frac{|\widehat{f}(\omega)|^2 \ (1 - |\widehat{w}(\omega)|)^2}{\sigma^2_{f_D}} \tag{15}$$

$$= \frac{\sigma^2_f}{\sigma^2_{f_D}} \ (1 - |\widehat{w}(\omega)|)^2 \ P^{\mathcal{F}}_f(\omega) \tag{16}$$

As in the case of smoothing, the so called *high-pass response* of the window function is given by $(1-|\widehat{w}|)^2$ and scaled with the ratio of variances $\sigma^2_f/\sigma^2_{f_D}$. In strong contrast to smoothing, there is no overall gain in power in the range of the periods, passing through the filter (called *passband*). There is however, in case of moving average and other time-domain filters (see also Figure S2), no simple passband region. Instead, there are *rippling artifacts* in the frequency domain, meaning some periods getting amplified to up to 150% and others attenuated by up to 25%. To showcase why this can be problematic, we constructed a synthetic chirp signal sweeping through a range of periods $T_1 - T_2$, however, this time modified by a linear and an oscillatory trend (Figure 3A). The oscillatory component of the trend was chosen for clarity with a specific time scale given by its period T_{trend}, which is three times the longest period found in the chirp signal. Strongly depending on the specific window size chosen for the moving average filter, there are various effects on both, the time and frequency domain (shaded area in Figure 3B) such as the introduction of amplitude envelopes and/or incomplete trend removal (Figure 3C). A larger window size is better to reduce the effect of the ripples inside the passband. However, the filter decay (*roll-off*) towards larger periods becomes very slow. That in turn means that trends can not be fully eliminated. Smaller window sizes perform better in detrending, but their passband can be dominated by ripples (see also supplementary Figure S3). In practice, sticking to filters originally designed for the time-domain without having oscillatory signals in mind, can easily lead to biased results of a time-frequency analysis. However, given the moderate gains of the detrending filter response, there is a much smaller chance to mistakenly detect spurious oscillations compared to the case of smoothing.

The Sinc filter

The Sinc filter, also known as the optimal filter in the frequency domain, is a function with a constant value of one in the passband. In other words, frequencies which pass through are neither amplified nor attenuated. Accordingly, this filter also should be constantly zero in the stop-band, the frequencies (or periods) which should be filtered out. This optimal low-pass response can be formulated in the frequency domain simply as:

$$\widehat{w}_{opt}(\omega) = \begin{cases} 1, & \text{if } \omega \leq \omega_c \\ 0, & \text{if } \omega > \omega_c \end{cases}$$

Here ω_c is the *cut off frequency*, an infinitely sharp transition band dividing the frequency range into pass- and stop-band. It is effectively a box in the

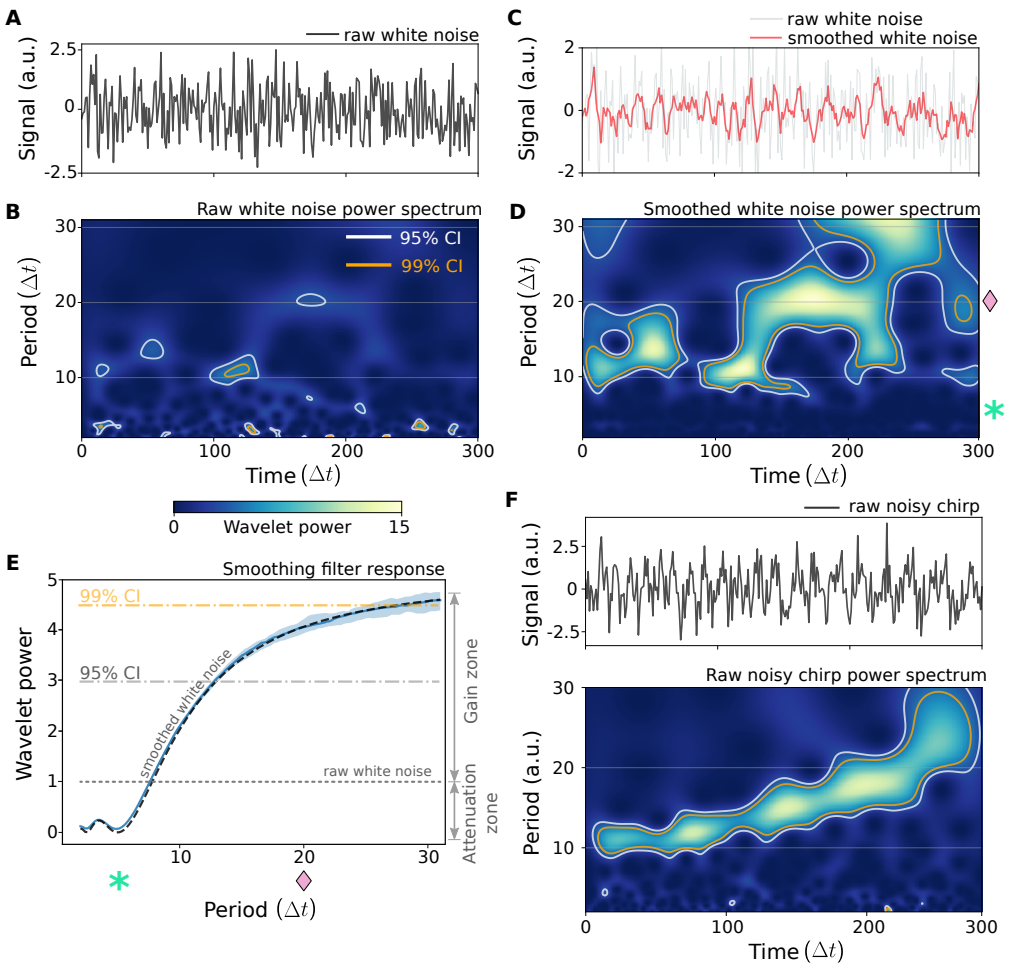

Figure 2: **Smoothing inevitably introduces signal artefacts** A) Raw non-smoothed white noise time-series. B) Wavelet power spectrum of the raw white noise signal shown in A. 95% (white line) and and 99% (orange line) confidence contour levels are indicated. C) Moving average smoothed white noise signal (*red line*) from panel A. D) Wavelet power spectrum of the smoothed signal shown in C. Note the highly significant islands of high power ($P_f^{\mathcal{W}} \approx 15$) for periods greater than around 10 sampling intervals, the largest one is in the $T \approx 20\Delta t$ band, marked by (♦). The (∗) marks the end of the effective stop-band region of the filter. E) Attenuations and gain effects of smoothing with a moving average filter, (quartiles in blue). Low periods (high frequencies) are suppressed while higher periods are amplified up to 5-fold in average power. The *dashed black line* denotes the theoretically calculated Fourier spectrum of the moving average filter, scaled with $1/\sigma_{f_S}^2 \approx 5$, coinciding with the numerical results (blue shaded area). The gray dotted line depicts the mean power of a white noise signal. F) *Top:* Noisy slowing oscillation (chirp) time-series with a signal-to-noise ratio of 1. *Bottom:* Corresponding wavelet power spectrum, obtained from the raw unsmoothed signal as processed by `pyBOAT`. In panel E, the averaged wavelet power distribution was calculated using 50 time series with 25.000 sample points each and smoothed with a window size $M = 5\Delta t$.

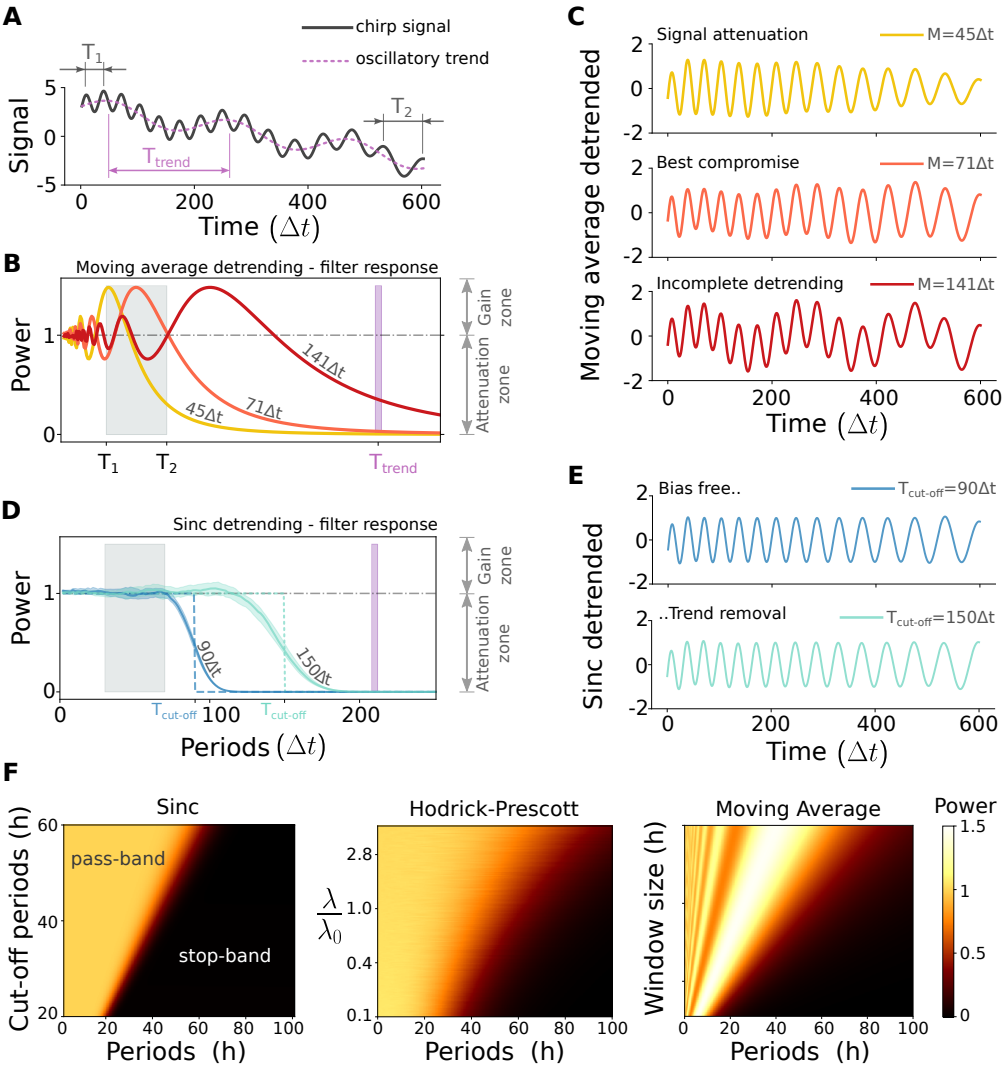

Figure 3: **Detrending: moving average vs. optimal sinc** (A) Synthetic signal sweeping through the periods $T_1 = 30\Delta t$ to $T_2 = 70\Delta t$ within 600 sampling intervals (Δt). This signal has an oscillatory and a linear trend: $\sin(2\pi/T_{trend}\ t) - 0.0066t + 1.5$, with $T_{trend} = 210\Delta t$. (B) Detrending (or high-pass) response of the moving average filter with three different window sizes. The period range of the signal and the timescale of the trend are marked by shaded grey and violet regions respectively. (C) Time-domain effects of the moving average filters shown in B. A decreasing amplitude trend for $M = 45\Delta t$ (*top*), a best compromise case $M = 71\Delta t$, and an incomplete trend removal for $M = 141\Delta t$ (*bottom*) can be observed. (D) Detrending response of the optimal sinc filter for two different cut-off periods as indicated. (E) Time-domain result for the sinc filters shown in C, indicating no amplitude effects while the trend is removed. (F) Parameter-dependent performance test of three common detrending filters, namely the sinc (*left*), Hodrick-Prescott (*middle*) and moving average (*right*), respectively. The sinc filter provides the clearest separation between pass- and stop-band. The reference value for the Hodrick-Prescott filter $\lambda_0 = 0.05 * \left(\frac{T}{\Delta t}\right)^4$ for $T = 24\Delta t$ is taken from Ravn and Uhlig [2002].

frequency domain (dashed lines in Figure 3D). Note that the optimal high-pass or detrending response simply and exactly swaps the pass- and stop-band. In the time domain via the inverse Fourier transform, this can be written as:

$$w_{opt}(t) = \frac{\sin(\omega_c t)}{\pi t}.$$

This function is known as the *sinc* function and hence the name *sinc filter*. An alternative name used in electrical engineering is brick-wall filter. In practice, this optimal filter has a nonzero roll-off as shown for two different cut-off periods ($T_c = \frac{2\pi}{\omega_c}$) in Figure 3D. The sinc function mathematically requires the signal to be of infinite length. Therefore, every practical implementation implements *windowed sinc filters* (Smith et al. [1997]), see also supplementary information S3) about possible implementations. Strikingly still, there are no ripples or other artifacts in the frequency-response of the windowed sinc filter. And hence also the 'real world' version allows for a bias free time-frequency analysis. As shown in Figure 3E, the original signal can be exactly recovered via detrending. To showcase the performance of the sinc filter, we numerically compared its performance against two other common methods, the Hodrick-Prescott and moving average filter (Figure 3F). The stop- and passband separation of the sinc filter clearly is the best, although the Hodrick-Prescott filter with a parameterization as given by Ravn and Uhlig [2002] also gives acceptable results (see also supplementary Figure S5). The moving average is generally inadvisable, due to its amplification right at the start of the passband.

In addition to its advantages in practical signal analysis, the Sinc filter also allows to analytically calculate the gains from filtering pure noise (see also supplementary information A.3). The gain, and therefore the probability to detect spurious oscillations, introduced from smoothing is typically much larger compared to detrending. However, if a lot of energy of the noise is concentrated in the slow low frequency bands, also detrending with small cut-off periods alone can yield substantial gains (see Figure S6 and Figure S7). Importantly, when using the Sinc filter, the background spectrum of the noise will always be uniformly scaled by a constant factor in the pass-band. There is no mixing of attenuation and amplification as for time-domain filters like moving average (Figure 3B and C). If the spectrum of the noise can be estimated, or an empirical background spectrum is available, the theory presented in A.3 allows to directly calculate the correct confidence intervals.

4 Readout - Along the Ridge

Wavelet power spectra are by itself already a valuable output of a signal analysis and give a direct visual representation of the relative weight of periods (frequencies) present in the signal of interest in a time dependent manner. However, especially in the context of biological rhythms, there often is the well justified assumption that there is only one main oscillatory component present [1]. The

[1] This is also the underlying assumption of the Hilbert transform, the main component being the so called 'analytical signal' (Hussain and Boashash [2002]).

extraction of the instantaneous period, amplitude and phase of that component is of prime interest for the practitioner. In this section we show how to obtain these important features using wavelet transforms as implemented in `pyBOAT`.

Tracking the main oscillatory components

From the perspective of Wavelet power spectra, such main oscillatory components are characterized by concentrated and time-connected regions of high power. *Wavelet ridges* are a means to trace these regions on the time-period plane. For the vast majority of practical applications a simple maximum ridge extraction is sufficient. This maximum ridge can be defined as:

$$r(t_k) = \left[\max(P_f^{\mathcal{W}}(t_0, T), ..., \max(P_f^{\mathcal{W}}(t_n, T))\right], \tag{17}$$

with n being the number of sample points in the signal. Thus, the ridge $r(t_k)$ maps every time point t_k to a row of the power spectrum, and therewith to a specific instantaneous period T_k (Figure 4C and D). Evaluating the power spectrum along a ridge, gives a time series of powers: $P_f^{\mathcal{W}}(t_k, r(t_k))$. Setting a power threshold value is recommended to avoid evaluating the ridge in regions of the spectrum where the noise dominates (In Figure 4C threshold is set to 5). Alternatively to simple maximum ridge detection, more elaborated strategies for ridge extraction have been proposed (Carmona et al. [1995]).

Amplitude normalization

A problem often encountered when dealing with biological data is a general time-dependent amplitude envelope (Figure 4A). Under our wavelet approach, the power spectrum is normalized with the overall variance of the signal. Consequently, regions with low signal amplitudes but robust oscillations are nevertheless represented as very low power blurring them with the spectral floor (Figure 4C). This leads to the impractical situation, where even a noise free signal with an amplitude decay will show very low power at the end (Figure 4E,F and S8), defeating its statistical purpose. A practical solution in this case is to estimate an amplitude envelope and subsequently normalize the signal with this envelope (Figure 4A and B). We specifically show here non-periodic envelopes, estimated by a sliding window (see also methods). After normalization, lower amplitudes are no longer penalized and an effective power-thresholding of the ridge is possible (Figure 4D and F).

A limitation of convolutional methods, including Wavelet-based approaches, are *edge effects*. At the edges of the signal, the Wavelets only partially overlap with the signal leading to a so-called *cone of influence* (COI) (Figure 4C and D). Even though the periods are still close to the actual values, phases and especially the power should not be trusted inside the COI (see discussion and supplementary Figure S9).

Once the trace of consecutive Wavelet power maxima has been determined and thresholded, evaluating the transform along it yields the instantaneous envelope amplitude, normalized amplitude and phases (see Figure 4E,F and G). To obtain

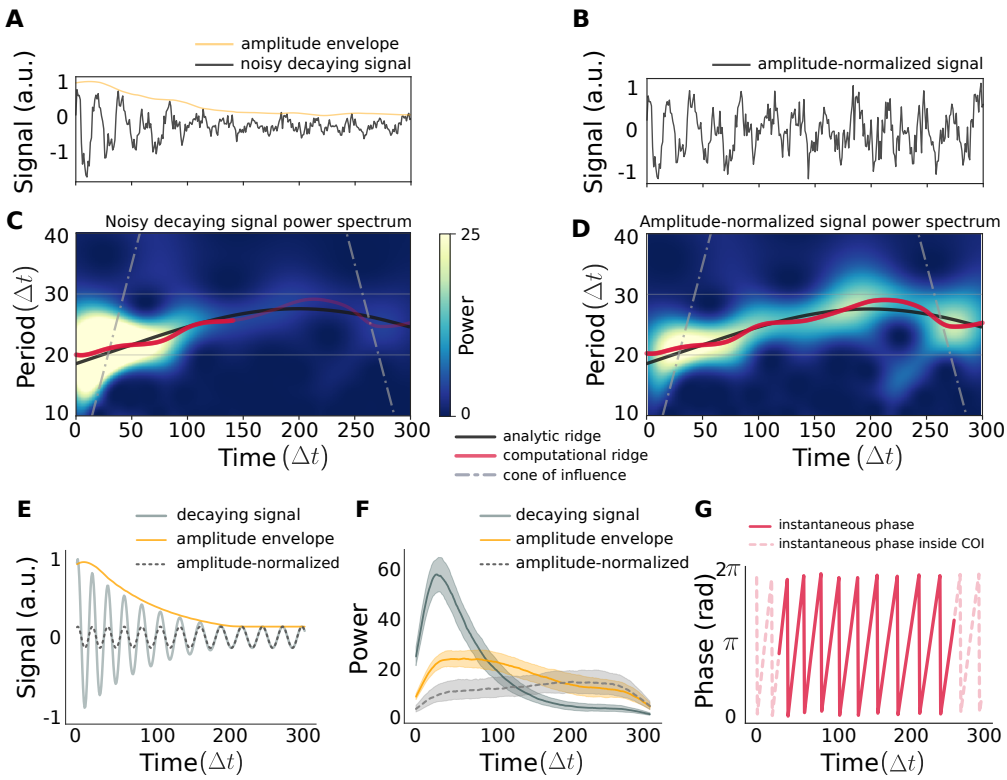

Figure 4: **Ridge detection and amplitude normalization** (A) A noisy synthetic signal with a decaying amplitude envelope (orange line). (B) Signal from panel A after amplitude normalization. (C) Wavelet power spectrum of the signal in panel A. The decaying amplitudes lead to very low powers towards the end of the signal. The extracted maximum power ridge (red line) reaches the lower threshold of 5 in the middle part of the signal (indicated by the light red line transition). As a reference, the (expected) analytic ridge is depicted by a black line. The cone of influence is indicated by gray dashed lines. (D) Wavelet power spectrum of the normalized signal in panel B. The oscillations are well captured, indicated by the high power during at all times. Hence, the thresholded maximum ridge traces the whole signal. (E) The deterministic parts of the synthetic signal shown in A. (F) Median and quartiles of the power along non-thresholded ridges of the signals as indicated ($N_{trial} = 50$). Signals with amplitude decay have only very low powers towards the end. The amplitude normalized signals do not have a constant signal-to-noise ratio even after normalization, and therefore still have higher power on average in the beginning as compared to the signal without envelope. (G) Instantaneous phase extracted from the amplitude normalized signal. The COI extent is marked by dashed lines.

the amplitudes, we derive a simple scaling rule for the power time series to be converted into amplitudes (see derivations in A.4).

Applications

After introducing the different time series analysis steps using synthetic data for clarity, in this paragraph, we discuss examples of `pyBOAT` applications to real data. To showcase the versatility of our approach, we chose datasets obtained from different scientific fields.

In Figure 5A we display COVID-19 infections in Italy as reported by the Disease Prevention and Control (ECDC). A Sinc-filter trend identification with a cut-off period of 14 days reveals a steep increase in newly reported infections at the beginning of March and a steady decline after the beginning of April. Subtracting this non-linear trend clearly exposes oscillations with a stable period of one week (Figure 5B, and see Supplementary Figure S10A for power spectrum analysis). Similar findings were recently reported in Ricon-Becker et al. [2020].

The signals shown in Figure 5C show cycles in Hare-Lynx population sizes as inferred from the yearly number of pelts, trapped by the Hudson Bay Company. Data has been taken from Odum and Barrett [1971]. The corresponding power spectra are shown in the supplement (Figure S10B), and reveal a fairly stable 10 year periodicity. After extracting the instantaneous phases with `pyBOAT`, we calculated the time-dependent phase differences as shown in Figure 5B. Interestingly, the phase difference slowly varies between being almost perfectly out of phase and being in phase for a few years around 1885.

The next example signal shows a single-cell trajectory of a U2OS cell carrying a Geminin-CFP fluorescent reporter (Granada et al. [2020]). Geminin is a cell cycle reporter, accumulating in the G2 phase and then steeply declining during mitosis. Applying `pyBOAT` on this non-sinusoidal oscillations reveals the cell-cycle length over time (Figure 5F), showing a slowing down of the cell cycle progression for this cell. Ensemble dynamics for a control and a cisplatin treated population are shown in the Supplementary Figure S10C.

The final example data set is taken from Mönke et al. [2017], here populations of MCF7 cells where treated with different dosages of the DNA damaging agent NCS. This in turn elicits a dose-dependent and heterogeneous p53 response, tracked in the individual cells for each condition (Figure 5G). `pyBOAT` also features readouts of the ensemble dynamics: Figure 5H shows the time-dependent period distribution in each population, Figure 5I the phase coherence over time. The latter is calculated as $R(t) = \left|\sum_j e^{i\phi_j(t)}\right|$. It ranges from zero to one and is a classical measure of synchronicity in an ensemble of oscillators Kuramoto [1984]. The strongly stimulated cells (400ng NCS) show stable oscillations with a period of around 240min, and retain more phase coherent after an initial drop in synchronicity. The medium stimulated cells (100ng NCS) start to slow down on average already after the first pulse, both the spread of the period distribution and the low phase coherence indicate a much more heterogeneous response. Two individual cells and their wavelet analysis are shown in Supplementary Figure S10D.

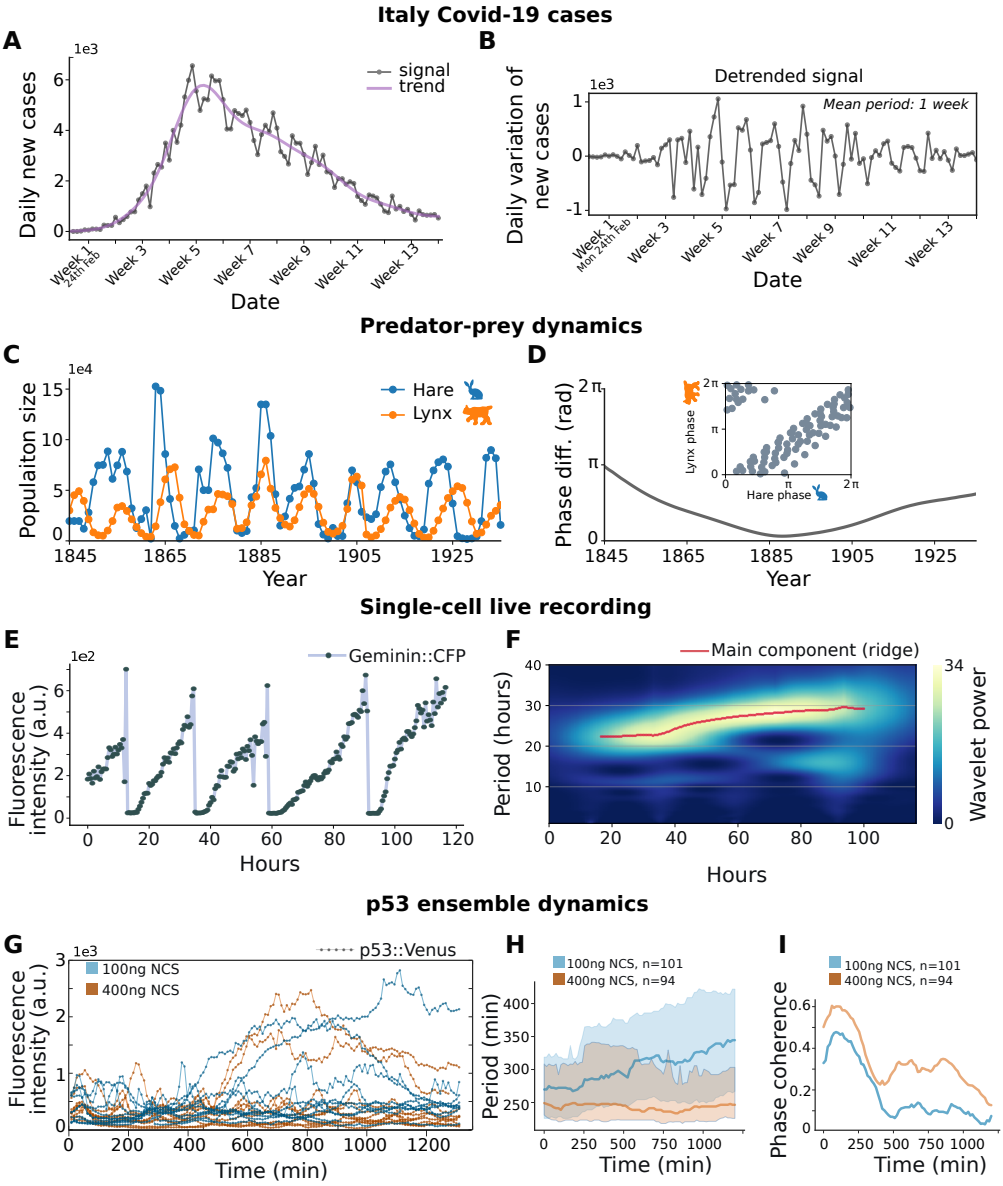

Figure 5: **Time-dependent analysis of oscillations using** `pyBOAT` A) Daily confirmed COVID-19 infections in Italy (gray line; dots represent single data points) from 19th Feb to May 25th 2020. Non-linear baseline trend as determined by the Sinc-filter, using a cut-off period of 14 days (purple line). B) Daily oscillatory variation of Covid-19 new cases. Plot in B) was obtained by removing the trend from the data on panel A. C) Population number of snowshoe hares (blue) and lynx (orange). D) Difference between the instantaneous (i.e. time-dependent) phases of the hare and lynx oscillations as extracted from the wavelet-spectra ridges. In the inset, instantaneous phases of the hare and the lynx are plotted against each other. E) Single-cell trajectory of a U2OS cell carrying a Geminin-CFP fluorescent reporter. F) Wavelet analysis reveals a strong main periodic component (ridge in red). G) Single-cell trajectories of MCF7 cells carrying a p53-venus fluorescent reporter under low 100ng NCS and high 400ng NCS DNA damage. 10 randomly selected traces by condition. H) Median and quartiles of the detected periods over time. I) Phase-coherence over time as a measure for synchronicity for each condition.

Graphical interface

The extraction of period, amplitude and phase is the final step of our proposed analysis workflow, which is outlined in the screen-captures of Figure 6. The user interface is separated into several sections. First, the 'DataViewer' allows to visualize the individual raw signals, the trend determined by the Sinc filter, the detrended time series and the amplitude envelope. Once satisfactory parameters have been found, the actual Wavelet transform together with the ridge are shown in the 'Wavelet Spectrum' window. After ridge extraction, it is possible to plot the instantaneous observables in a 'Readout' window. Each plot produced from the interface can be panned, zoomed in and saved separately if needed.

Once the fidelity of the analysis has been checked for individual signals, it is also possible to run the entire analysis as a 'batch process' for all signals imported. One aggregated result which we found quite useful is to determine the 'rhythmicity' of a population by creating a histogram of time-averaged powers of the individual ridges. A classification of signals into 'oscillatory' and 'non-oscillatory' based on this distribution, e.g. by using classical thresholds (Otsu [1979]) is a potential application. Examples of the provided ensemble readouts are shown in Figure 5H and I and Supplementary Figure S10C.

Finally, `pyBOAT` also features a synthetic signal generator, allowing to quickly explore its capabilities also without having a suitable dataset at hand. A synthetic signal can be composed of up to two different chirps, and AR1-noise and an exponential envelope can be added to simulate possible challenges often present in real data (see also Material and Methods and Supplementary Figure S11).

Installation and user guidelines of `pyBOAT` can be found in the github repository.

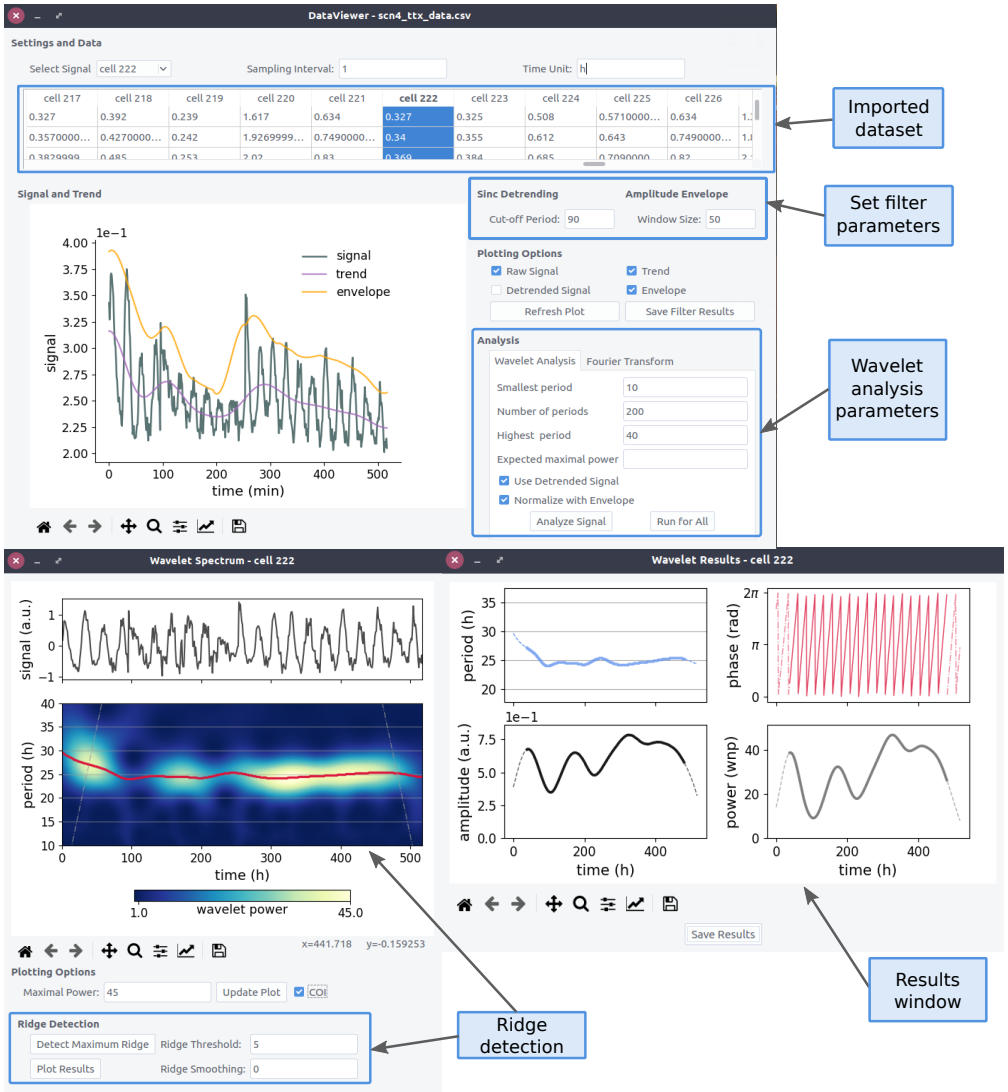

Figure 6: **Screen captures of the elements of `pyBOAT` in action** The signal gets detrended with a cut-off period of 90h, and an amplitude envelope is estimated via a window of size 50h. See labels and main text for further explanations. The example trajectory displays a circadian rhythm of 24h and is taken from the data set published in Abel et al. [2016].

5 Discussion

Recordings of biological oscillatory signals can be conceptualized as an aggregate of multiple components, those coming from the underlying system of interest and additional confounding factors such as noise, modulations and trends that can disguise the underlying oscillations. In cases of variable period with noisy amplitude modulation and non-stationary trends the detection and analysis of oscillatory processes is a non-trivial endeavour. Here we introduced `pyBOAT`,

a novel software package that uses a statistically rigorous method to handle non-stationary rhythmic data. `pyBOAT` integrates pre- and post-processing steps without making *a priori* assumptions about the sources of noise and periodicity of the underlying oscillations. We showed how the signal processing steps of smoothing, detrending, amplitude envelope removal, signal detection and spectral analysis can be resolved by our hands-off standalone software (Figure 1 and 5).

Artifacts introduced by the time series analysis methods itself are a common problem that inadvertently disturbs results of the time-frequency analysis of periodic components (Wilden et al. [1998]). Here we first analyzed the effects of data-smoothing on a rhythmic noisy signal and showed how common smoothing approaches disturb the original recordings by introducing non-linear attenuations and gains to the signal (Figures 2, S6 and S7). These gains easily lead to spurious oscillations that were not present in the original raw data. These artifacts have been characterized since long for the commonly used moving-average smoothing method, known as the Slutzky-Yule effect (Slutzky [1937]). Using an analytical framework, we describe the smoothing process as a filter operation in frequency domain. This allows us to quantify and directly compare the effects of diverse smoothing methods by means of response curves. Importantly, we show here how any choice of smoothing unavoidably transforms the original signal in a non-trivial manner. One potential reason for its prevalence is that practitioners often implement a smoothing algorithm without quantitatively comparing the spectral components before versus after smoothing. `pyBOAT` avoids this problem by implementing a wavelet-based approach that *per se* evades the need to smooth the signal.

Another source of artifacts are detrending operations. Thus, we next studied the spectral effects that signal detrending has on rhythmic components. Our analytical and numerical approaches allowed us to compare the spectral effects of different detrending methods in terms of their response curves (see Figure 3). Our results show that detrending also introduces non-trivial boosts and attenuations to the oscillatory components of the signal, strongly depending on the background noise (Figures S6 and S7). In general there is no universal approach and optimally a detrending model is based on information about the sources generating the trend. In cases without prior information to formulate a parametric detrending in the time domain, we suggest that the safest method is the convolution based *sinc filter* , as it is an "ideal" (step-function) filter in the frequency domain (Figures 3C and S3). Furthermore we compared the performance of the *sinc filter* with two other commonly applied methods to remove non-linear trends in data (Figure 3F), i.e. the moving average (Díez-Noguera [2013]) and Hodrick-Prescott (Myung et al. [2012], Schmal et al. [2018], St. John and Doyle [2015]) filter.

In addition to smoothing and detrending, amplitude-normalization by means of the amplitude envelope removal is another commonly used data processing step that `pyBOAT` is able to perform. Here we further show how that for decaying signals amplitude normalization grants that the main oscillatory component of interest can be properly identified in the power spectrum (Figure 4A to D). This main component is identified by a ridge-tracking approach that can be then used

to extract instantaneous signal parameters such as amplitudes, power and phase (Figure 4E to G).

Rhythmic time series can be categorized into those showing stationary oscillatory properties and the non-stationary ones where periods, amplitudes and phases change over time. Many currently available tools for the analysis of biological rhythms rely on methods aimed at stationary oscillatory data, using either a standalone software environment such as `BRASS` (Edwards et al. [2010], Locke et al. [2005]), `ChronoStar` (Klemz et al. [2017]) and `CIRCADA` (Cenek et al. [2020]) or an online interfaces such as `BioDare` (Moore et al. [2014], Zielinski et al. [2014]).

Continuous wavelet analysis allows to reveal non-stationary period, amplitude and phase dynamics and to identify multiple frequency components across different scales within a single oscillatory signal (Leise [2013], Leise et al. [2012], Rojas et al. [2019]) and is thus complementary to approaches that are designed to analyze stationary data. In contrast to the `R`-based `waveclock` package (Price et al. [2008]), `pyBOAT` can be operated as a standalone software tool that requires no prior programming knowledge as it can be fully operated using its graphical user interface (GUI). An integrated batch processing option allows the analysis of large data sets within a few "clicks". For the programming interested user, `pyBOAT` can also be easily scripted without using the GUI, making it simple to integrate it into individual analysis pipelines. `pyBOAT` also distinguishes itself from other wavelet-based packages (e.g. Harang et al. [2012]) by adding a robust *sinc filter*-based detrending and a statistically rigorous framework, providing the interpretation of results by statistical confidence considerations.

`pyBOAT` is not specifically designed to analyze oscillations in high-throughput "omics" data. For such sake, specialized algorithms such as ARSER (Yang and Su [2010]), JTK-cycle (Hughes et al. [2010]), MetaCycle (Wu et al. [2016]) or RAIN (Thaben and Westermark [2014]) are more appropriate. Its analysis reveals basic oscillatory properties such as the time-dependent (instantaneous) rhythmicity, period, amplitude and phase but is not aimed at more specific statistical tests such as, e.g., tests for differential rhythmicity as implemented in DODR (Thaben and Westermark, 2016). The continous-wavelet analysis underlying `pyBOAT` requires equidistant time series sampling with no gaps. Methods such as Lomb-Scargle periodograms or harmonic regressions are more robust or even specifically designed with respect to unevenly-sampled data (Lomb [1976], Ruf [1999]). Being beyond the scope of this manuscript, it will be interesting in future work to integrate the ability to analyze unevenly sampled data into the pyBOAT software, either by the imputation of missing values (e.g. by linear interpolation) or the usage of wavelet functions specifically designed for this purpose (Thiebaut and Roques [2005]).

`pyBOAT` is fast, easy too use and statistically robust analysis routine designed to complement existing methods and advance the efficient time series analysis of biological rhythms research. In order to make it publicly available, `pyBOAT` is a free and open-source, multi-platform software based on the popular Python (Van Rossum and Drake [2009]) programming language. It can be downloaded using the following link : https://github.com/tensionhead/pyBOAT, and is available on the Anaconda distribution (via the *conda-forge* channel).

6 Methods

Software

`pyBOAT` is written in the Python programming language (Van Rossum and Drake [2009]). It makes extensive use of Python's core scientific libraries *numpy* and *scipy* (Virtanen et al. [2020]) for the numerics. Additionally we use *matplotlib* (Hunter [2007] for visualization, and *pandas* (McKinney [2010]) for data management. `pyBOAT` is released under the open source GPL-3.0 license, and its code is freely available from https://github.com/tensionhead/pyBOAT. The readme on this repository contains further information and installation instructions. `pyBOAT` is also hosted on the popular Anaconda distribution, as part of the *conda-forge* community https://conda-forge.org/.

Amplitude envelope

To estimate the amplitude envelope in the time domain, we employ a moving window of size L and determine the minimum and maximum of the signal inside the window for each time point t. The amplitude at that time point is then given by $a(t) = \frac{1}{2}(max(t) - min(t))$. This works very well for envelopes with no periodic components, like an exponential decay. However, this simple method is not suited for oscillatory amplitude modulations. It is also recommended to sinc-detrend the signal before estimating the amplitude envelope. Note that L should always be larger then the maximal expected period in the signal, as otherwise the signal itself gets distorted.

Synthetic signals

A noisy chirp signal can be written as: $f(t_i) = a\ cos(\phi(t_i)) + d\ x(t_i)$, where $\phi(t_i)$ is the instantaneous phase and the $x(t_i)$ are samples from a stationary stochastic process (the background noise). The increments of the t_i are the sampling interval: $t_{i+1} - t_i = \Delta t$, with $i = 0, 1, ..., N$ samples. Starting from a linear sweep through angular frequencies: $\omega(0) = \omega_1$ and $\omega(t_N) = \omega_2$, we have $\omega(t) = \frac{\omega_2 - \omega_1}{t_N} t + \omega_1$. The instantaneous phase is then given by

$$\phi(t) = \int_0^t \omega(t')dt' = \frac{1}{2t_N}(\omega_2 - \omega_1)t^2 + \omega_1 t.$$

Sampling N times from a gaussian distribution with standard deviation equal to one corresponds to gaussian white noise $\xi(t_i)$. With $x(t_i) = \xi(t_i)$ the signal to noise ration (SNR) then is a^2/d^2.

A realization of an AR1 process can be simulated by a simple generative procedure: the inital $x(t_0)$ is a sample from the standard normal distribution. Then the next sample is given by: $x(t_i) = \alpha x(t_{i-1}) + \xi(t_i)$, with $\alpha < 1$.

Simulating pink noise is less straightforward, and we use the Python package *colorednoise* from https://pypi.org/project/colorednoise for the simulations. Its implementation is based on Timmer and Koenig [1995].

Acknowledgments

We gratefully thank Bharath Ananthasubramaniam, Hanspeter Herzel, Pedro Pablo Rojas and Shaon Chakrabarti for fruitful discussions and comments on the manuscript. We further thank Jelle Scholtalbers and the GBCS unit at the EMBL in Heidelberg for technical support. We thank members of the Aulehla and Leptin labs for comments, support and helpful advice.

Funding

Gregor Mönke's research was supported by the EMBL Interdisciplinary Postdoc Programme (EIPOD) under Marie Słodowska-Curie Actions COFUND grant number 664726. Christoph Schmal acknowledges support from the Deutsche Forschungsgemeinschaft (DFG) through grant number SCH3362/2-1. Adrian E. Granada's research was supported by the German Federal Ministry for Education and Research (BMBF) through the Junior Network in Systems Medicine, under the auspices of the e:Med Programme (grant 01ZX1917C).

References

Abel, J. H., Meeker, K., Granados-Fuentes, D., John, P. C. S., Wang, T. J., Bales, B. B., Doyle, F. J., Herzog, E. D., and Petzold, L. R. (2016). Functional network inference of the suprachiasmatic nucleus. Proceedings of the National Academy of Sciences, 113(16):4512–4517.

Abraham, U., Schlichting, J. K., Kramer, A., and Herzel, H. (2018). Quantitative analysis of circadian single cell oscillations in response to temperature. PLOS ONE, 13(1):1–15.

Carmona, R., Hwang, W. L., and Torrésani, B. (1995). Identification of chirps with continuous wavelet transform. In Wavelets and Statistics, pages 95–108. Springer.

Cenek, L., Klindziuk, L., Lopez, C., McCartney, E., Burgos, B. M., Tir, S., Harrington, M. E., and Leise, T. L. (2020). Circada: Shiny apps for exploration of experimental and synthetic circadian time series with an educational emphasis. Journal of Biological Rhythms, 35(2):214–222.

Chatfield, C. (1995). Problem solving: a statistician's guide. CRC Press.

Díez-Noguera, A. (2013). Methods for serial analysis of long time series in the study of biological rhythms. J Circadian Rhythms, 1(11):7.

Edwards, K. D., Akman, O. E., Knox, K., Lumsden, P. J., Thomson, A. W., Brown, P. E., Pokhilko, A., Kozma-Bognar, L., Nagy, F., Rand, D. A., and Millar, A. J. (2010). Quantitative analysis of regulatory flexibility under changing environmental conditions. Molecular Systems Biology, 6(1):424.

Fitch, W. T., Neubauer, J., and Herzel, H. (2002). Calls out of chaos: the adaptive significance of nonlinear phenomena in mammalian vocal production. Animal Behaviour, 63(3):407 – 418.

Gabor, D. (1946). Theory of communication. part 1: The analysis of information. Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering, 93(26):429–441.

Goldbeter, A., Gerard, C., Gonze, D., Leloup, J.-C., and Dupont, G. (2012). Systems biology of cellular rhythms. FEBS Letters, 586(18):2955–2965.

Granada, A. E., Jiménez, A., Stewart-Ornstein, J., Blüthgen, N., Reber, S., Jambhekar, A., and Lahav, G. (2020). The effects of proliferation status and cell cycle phase on the responses of single cells to chemotherapy. Molecular Biology of the Cell, 31(8):845–857.

Gröchenig, K. (2013). Foundations of time-frequency analysis. Springer Science & Business Media.

Gwinner, E. (2003). Circannual rhythms in birds. Current Opinion in Neurobiology, 13(6):770 – 778.

Halberg, F., Tong, Y. L., and Johnson, E. A. (1967). Circadian System Phase — An Aspect of Temporal Morphology; Procedures and Illustrative Examples, pages 20–48. Springer Berlin Heidelberg, Berlin, Heidelberg.

Harang, R., Bonnet, G., and Petzold, L. R. (2012). Wavos: a matlab toolkit for wavelet analysis and visualization of oscillatory systems. BMC research notes, 5(1):163.

Hughes, M. E., Hogenesch, J. B., and Kornacker, K. (2010). Jtk_cycle: An efficient nonparametric algorithm for detecting rhythmic components in genome-scale data sets. Journal of Biological Rhythms, 25(5):372–380.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. Computing in science & engineering, 9(3):90–95.

Hussain, Z. M. and Boashash, B. (2002). Hilbert transformer and time delay: statistical comparison in the presence of gaussian noise. IEEE transactions on signal processing, 50(3):501–508.

Klemz, R., Reischl, S., Wallach, T., Witte, N., Jürchott, K., Klemz, S., Lang, V., Lorenzen, S., Knauer, M., Heidenreich, S., Xu, M., Ripperger, J. A., Schupp, M., Stanewsky, R., and Kramer, A. (2017). Reciprocal regulation of carbon monoxide metabolism and the circadian clock. Nature Structural & Molecular Biology, 24(1):15–22.

Kuramoto, Y. (1984). Chemical turbulence. In Chemical Oscillations, Waves, and Turbulence, pages 111–140. Springer.

Lauschke, V. M., Tsiairis, C. D., François, P., and Aulehla, A. (2013). Scaling of embryonic patterning based on phase-gradient encoding. Nature, 493(7430):101–105.

Leise, T. L. (2013). Wavelet analysis of circadian and ultradian behavioral rhythms. Journal of circadian rhythms, 11(1):5.

Leise, T. L., Wang, C. W., Gitis, P. J., and Welsh, D. K. (2012). Persistent cell-autonomous circadian oscillations in fibroblasts revealed by six-week single-cell imaging of per2::luc bioluminescence. PLOS ONE, 7(3):1–10.

Locke, J. C. W., Southern, M. M., Kozma-BognÃ¡r, L., Hibberd, V., Brown, P. E., Turner, M. S., and Millar, A. J. (2005). Extension of a genetic network model by iterative experimentation and mathematical analysis. Molecular Systems Biology, 1(1):2005.0013.

Lomb, N. R. (1976). Least-squares frequency analysis of unequally spaced data. Astrophysics and space science, 39(2):447–462.

McKinney, W. (2010). Mckinney, wes. "data structures for statistical computing in python. Proceedings of the 9th Python in Science Conference, vol. 445, pp. 51-56.

Mönke, G., Cristiano, E., Finzel, A., Friedrich, D., Herzel, H., Falcke, M., and Loewer, A. (2017). Excitability in the p53 network mediates robust signaling with tunable activation thresholds in single cells. Scientific reports, 7:46571.

Moore, A., Zielinski, T., and Millar, A. (2014). Online period estimation and determination of rhythmicity in circadian data, using the biodare data infrastructure. Methods in molecular biology (Clifton, N.J.), 1158:13–44.

Myung, J., Hong, S., Hatanaka, F., Nakajima, Y., Schutter, E. D., and Takumi, T. (2012). Period Coding of Bmal1 Oscillators in the Suprachiasmatic Nucleus. The Journal of Neuroscience, 32(26):8900–8918.

Naitoh, P., Englund, C. E., and Ryman, D. H. (1985). Circadian rhythms determined by cosine curve fitting: analysis of continuous work and sleep-loss data. Behavior Research Methods, Instruments, & Computers, 17(6):630–641.

Odum, E. P. and Barrett, G. W. (1971). Fundamentals of ecology, volume 3. Saunders Philadelphia.

Ono, D., Honma, S., Nakajima, Y., Kuroda, S., Enoki, R., and Honma, K.-i. (2017). Dissociation of per1 and bmal1 circadian rhythms in the suprachiasmatic nucleus in parallel with behavioral outputs. Proceedings of the National Academy of Sciences, 114(18):E3699–E3708.

Otsu, N. (1979). A threshold selection method from gray-level histograms. IEEE transactions on systems, man, and cybernetics, 9(1):62–66.

Percival, D. P. (1995). On estimation of the wavelet variance. Biometrika, 82(3):619–631.

Pittendrigh, C. S. and Daan, S. (1974). Circadian oscillations in rodents: A systematic increase of their frequency with age. Science, 186(4163):548–550.

Price, T. S., Baggs, J. E., Curtis, A. M., FitzGerald, G. A., and Hogenesch, J. B. (2008). WAVECLOCK: wavelet analysis of circadian oscillation. Bioinformatics, 24(23):2794–2795.

Quiroga, R. Q., Kraskov, A., Kreuz, T., and Grassberger, P. (2002). Performance of different synchronization measures in real data: a case study on electroencephalographic signals. Physical Review E, 65(4):041903.

Ravn, M. O. and Uhlig, H. (2002). On adjusting the hodrick-prescott filter for the frequency of observations. Review of economics and statistics, 84(2):371–376.

Ricon-Becker, I., Tarrasch, R., Blinder, P., and Ben-Eliyahu, S. (2020). A seven-day cycle in covid-19 infection and mortality rates: Are inter-generational social interactions on the weekends killing susceptible people? medRxiv.

Rohde, A. and Bhalerao, R. P. (2007). Plant dormancy in the perennial context. Trends in Plant Science, 12(5):217 – 223.

Rojas, P., Plath, J. A., Gestrich, J., Ananthasubramaniam, B., Garcia, M. E., Herzel, H., and Stengl, M. (2019). Beyond spikes: Multiscale computational analysis of in vivo long-term recordings in the cockroach circadian clock. Network Neuroscience, 3(4):944–968.

Ruf, T. (1999). The lomb-scargle periodogram in biological rhythm research: Analysis of incomplete and unequally spaced time-series. Biological Rhythm Research, 30(2):178–201.

Savitzky, A. and Golay, M. J. (1964). Smoothing and differentiation of data by simplified least squares procedures. Analytical chemistry, 36(8):1627–1639.

Scheer, F. A. J. L., Wright, Jr., K. P., Kronauer, R. E., and Czeisler, C. A. (2007). Plasticity of the intrinsic period of the human circadian timing system. PLOS ONE, 2(8):1–7.

Schmal, C., Herzog, E. D., and Herzel, H. (2018). Measuring relative coupling strength in circadian systems. Journal of Biological Rhythms, 33(1):84–98.

Slutzky, E. (1937). The summation of random causes as the source of cyclic processes. Econometrica, 5(2):105–146.

Smith, S. W. et al. (1997). The scientist and engineer's guide to digital signal processing.

Song, L., Hennink, E., Young, I. T., and Tanke, H. J. (1995). Photobleaching kinetics of fluorescein in quantitative fluorescence microscopy. Biophysical journal, 68(6):2588–2600.

Soroldoni, D., Jörg, D. J., Morelli, L. G., Richmond, D. L., Schindelin, J., Jülicher, F., and Oates, A. C. (2014). A doppler effect in embryonic pattern formation. Science, 345(6193):222–225.

St. John, P. C. and Doyle, F. J. (2015). Quantifying Stochastic Noise in Cultured Circadian Reporter Cells. PLoS Computational Biology, 11(11).

Straume, M., Frasier-Cadoret, S. G., and Johnson, M. L. (1991). Least squares analysis of fluorescence data. In Lakowicz, J., editor, Topics in Fluorescence Spectros- copy, Volume 2: Plenum, pages 117 – 240.

Thaben, P. F. and Westermark, P. O. (2014). Detecting rhythms in time series with rain. Journal of Biological Rhythms, 29(6):391–400.

Thaben, P. F. and Westermark, P. O. (2016). Differential rhythmicity: detecting altered rhythmicity in biological data. Bioinformatics, 32(18):2800–2808.

Thiebaut, C. and Roques, S. (2005). Time-Scale and Time-Frequency Analyses of Irregularly Sampled Astronomical Time Series. EURASIP Journal on Advances in Signal Processing, 2005(15):852587.

Timmer, J. and Koenig, M. (1995). On generating power law noise. Astronomy and Astrophysics, 300:707.

Torrence, C. and Compo, G. P. (1998). A practical guide to wavelet analysis. Bulletin of the American Meteorological society, 79(1):61–78.

Tsiairis, C. and Aulehla, A. (2016). Self-organization of embryonic genetic oscillators into spatiotemporal wave patterns. Cell, 164(4):656 – 667.

Van Rossum, G. and Drake, F. L. (2009). Python 3 Reference Manual. CreateSpace, Scotts Valley, CA.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E. W., Vand erPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and Contributors, S. . . (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17:261–272.

Westermark, P. O., Welsh, D. K., Okamura, H., and Herzel, H. (2009). Quantification of circadian rhythms in single cells. PLoS computational biology, 5(11).

Wilden, I., Herzel, H., Peters, G., and Tembrock, G. (1998). Subharmonics, biphonation, and deterministic chaos in mammal vocalization. Bioacoustics, 9(3):171–196.

Wu, G., Anafi, R. C., Hughes, M. E., Kornacker, K., and Hogenesch, J. B. (2016). MetaCycle: an integrated R package to evaluate periodicity in large scale data. Bioinformatics, 32(21):3351–3353.

Yang, R. and Su, Z. (2010). Analyzing circadian expression data by harmonic regression based on autoregressive spectral estimation. Bioinformatics, 26(12):i168–i174.

Zielinski, T., Moore, A. M., Troup, E., Halliday, K. J., and Millar, A. J. (2014). Strengths and limitations of period estimation methods for circadian data. PLOS ONE, 9(5):1–26.