# Novel methods for distributed acoustic sensing data

Rebecca Elizabeth Wilson, M.Math, M.Res


Submitted for the degree of Doctor of

Philosophy at Lancaster University.

December 2018

STOR-i

excellence with impact

# Abstract

In this thesis, we propose novel methods for analysing nonstationary, multivariate time series, focusing in particular on the problems of classification and imputation within this context. Many existing methods for time series classification are static, in that they assign the entire series to one class and do not allow for temporal dependence with the signal. In the first part of this thesis, we propose a computationally efficient extension of an existing dynamic classification method to the online setting. Dependence within the series is captured by adopting the multivariate locally stationary wavelet (mvLSW) framework and the signal is classified at each time point into one of a number of known classes. We apply the method to multivariate acoustic sensing data in order to detect anomalous regions and evaluate the results against alternative methods in the literature. The second part of this thesis considers imputation in multivariate locally stationary time series containing missing values. We first introduce a method for estimating the local wavelet spectral matrix that can be used in the presence of missingness. We then propose a novel method for imputing missing values that uses the local auto and cross-covariance functions of a mvLSW process to perform one step-ahead forecasting and backcasting. The performance of

this nonstationary imputation approach is then assessed against competitor methods for simulated examples and a case study involving a dataset from a Carbon Capture and Storage facility. The software that implements this imputation scheme is also described, together with examples of the R package functionality.

# Acknowledgements

Firstly, I would like to thank my supervisors Idris Eckley and Matt Nunes for all the advice and guidance that they have provided throughout this project.

I am grateful for the financial support provided by the EPSRC and Shell. I would also like to thank my industrial supervisor Tim Park for the helpful discussions and insights into the project and the data provided.

I would like to thank the staff and students at the STOR-i Doctoral Training Centre for providing a stimulating and enjoyable research environment.

Finally, I would like to thank my family for their encouragement and support over the years, this would not have been possible without you.

# Declaration

I declare that the work in this thesis has been done by myself and has not been submitted elsewhere for the award of any other degree.

Chapter 3 has been accepted for publication as Wilson, R. E., Eckley, I. A., Nunes, M. A. and Park, T. (2018). Dynamic detection of anomalous regions within distributed acoustic sensing data streams using locally stationary wavelet time series. *Data Mining and Knowledge Discovery.*

Chapter 4: Wilson, R. E., Eckley, I. A., Nunes, M. A. and Park, T. (2018). A wavelet-based approach for imputation in highdimensional series. *In preparation.*

Chapter 5: Wilson, R. E., Grose, D., Eckley, I. A., Nunes, M. A. and Park, T. (2018). mvLSWimpute: An R package for imputation in multivariate locally stationary time series. *In preparation for submission to the R Journal.*

Rebecca Elizabeth Wilson

Word count: 44424

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

In recent years, there has been a rapid increase in high-resolution sensors collecting large amounts of data over short periods of time. One example is the high frequency acoustic sensing data generated within the oil and gas sector, often sampled at a rate of 10kHz per second across a range of depths within an oil well. This data collection results in large multivariate time series that can be difficult to model and analyse for a number of reasons. For example, it can be hard to capture complex inter-variable relationships or the series may be nonstationary in nature. In particular, the second-order structure may change over time. Wavelets, a form of localised basis functions, offer one possible approach to solving this problem. Their localisation in both time and scale allows for more efficient modelling of data containing locally changing behaviour or discontinuities (Dahlhaus, 2012).

Within the existing wavelet literature, Nason et al. (2000) introduced the locally stationary wavelet processes as a way of accurately modelling univariate time series exhibiting smoothly varying second-order structure. In the multivariate setting, mod-

elling individual channels of the series using the LSW framework can be deficient as relationships between variables are not taken into account. To this end, Park et al. (2014) developed the multivariate locally stationary wavelet framework which accurately captures the dependence structure between components of a multivariate time series. A review of the properties of wavelets and their transforms is provided in Chapter 2, along with a discussion of time series modelling and the wavelet-based approaches discussed above.

One of the main challenges associated with industrial sensing technology is that the recordings can be corrupted with periods of anomalous behaviour which have a negative impact on any secondary analysis such as forecasting or performance modelling. In Chapter 3, we introduce a computationally efficient extension of the dynamic classification method of Park et al. (2018) to the online setting. This permits real-time classification of a multivariate time series with changing class membership over time. We conclude the chapter by assessing the performance of the proposed classifier using simulated time series and demonstrating how it can be used to detect anomalous regions within multivariate acoustic sensing data. Details of the software that implements the online dynamic classification method can be found in Appendix A.

An additional issue with sensor recordings is that they can be severely affected by technical faults in the recording equipment which can induce missingness within the signal. Further analysis of such time series such as autocovariance and spectral estimation often rely on the data being complete, therefore accurately replacing missing values with appropriate estimates is an important task. Chapter 4 develops a novel imputation method based on the multivariate locally stationary wavelet framework

that can be used to estimate missing values within a multivariate locally stationary time series. We extend the wavelet forecasting approach of Fryzlewicz and Ombao (2009) to the multivariate setting and combine this with a backcasting step to improve accuracy. We again assess the performance of the proposed method against competitor methods through simulated examples and a dataset arising from a Carbon Capture and Storage facility. The software that implements the imputation method is available in the R package **mvLSWimpute**, details of the package and a demonstration of it's functionality can be found in Chapter 5.

Finally, we conclude this thesis with a summary of the main contributions and a discussion of avenues for future research in Chapter 6.

# Chapter 2

# Literature Review

This chapter reviews the literature on wavelets and time series modelling that are required for the work presented in the other chapters of this thesis. The first part of the chapter provides an introduction to wavelets and an overview of some popular transforms. We begin by briefly summarising key aspects of Fourier analysis, pointing out shortcomings of the approach and scenarios in which it will provide poor decomposition results. Wavelets are introduced in Section 2.1.2 through the concept of a multiresolution analysis and some popular wavelet families are summarised in Section 2.1.3. A variety of wavelet transforms are discussed in Sections 2.1.4 and 2.1.5 including the discrete wavelet transform, the non-decimated wavelet transform and wavelet packet transforms.

In the second part of the chapter, we review some approaches for modelling time series. Section 2.2.1 covers methods for modelling stationary time series in both the time and frequency domain. Some approaches for modelling nonstationary time series through locally stationary representations are discussed in Section 2.2.2 includ-

ing Locally Stationary Fourier processes (Dahlhaus, 1997) and the Smooth Localized Exponential model (Ombao et al., 2002). An alternative method for modelling nonstationary time series using wavelets as the basis functions is then discussed. The (univariate) Locally Stationary Wavelet processes (Nason et al., 2000) are described in detail in Section 2.3.1 before the Multivariate Locally Stationary Wavelet framework (Park et al., 2014) is discussed in Section 2.3.2. In the final part of the chapter, various approaches for the classification of nonstationary time series are discussed including the SLEX-based method of Huang et al. (2004) and the wavelet-based approaches of Fryzlewicz and Ombao (2009), Krzemieniewska et al. (2014) and Park et al. (2018).

## 2.1 Introduction to wavelets and their transforms

### 2.1.1 Fourier Analysis

Within classical Fourier analysis, a periodic square integrable function $f \in L^2([-\pi, \pi])$, can be represented in terms of the Fourier basis $\{\exp(inx)\}_{n=-\infty}^{\infty}$ in the following way

$$f(x) = \sum_{-\infty}^{\infty} c_n \exp(inx), \tag{2.1.1}$$

where the Fourier coefficients are computed by

$$c_m = (2\pi)^{-1} \int_{-\pi}^{\pi} f(x) \exp(-imx)\, dx. \tag{2.1.2}$$

Since $\exp(inx) = \cos(nx) + i\sin(nx)$, the Fourier series in equation (2.1.1) can be regarded as the expansion of $f$ in terms of sine and cosine functions.

One of the main drawbacks associated with Fourier series decomposition is that it cannot efficiently deal with discontinuities in a signal due to the nature of the basis functions $\{\cos(nx), \sin(nx)\}_{n\in\mathbb{Z}}$. Specifically, these functions are smooth and have infinite support meaning that every basis sine and cosine function will interact with a discontinuity which will influence the coefficients of the Fourier transform. An example of this can be seen in Figure 2.1.1, the Fourier series representation of the function near the discontinuities is poor and Gibbs effects can clearly be seen.

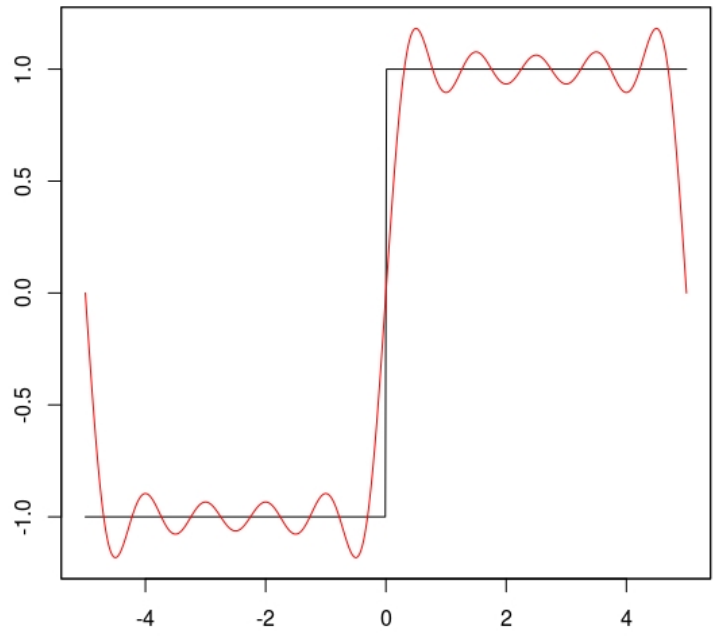



Figure 2.1.1: Example of the Gibbs effect

A further drawback of the Fourier approach is that, whilst it allows us to obtain information on the frequency components present in a signal or the time at which frequencies occur, it does not give information on both. For a more thorough review

of Fourier theory, see Walker (1986) or Percival and Walden (2006).

In order to efficiently model functions containing discontinuities, alternative basis functions are required that capture local features more accurately. Specifically, we require basis functions that are compactly supported. Wavelet basis functions possess this quality, as well as allowing for time-scale decomposition of a signal. For this reason, wavelets are often seen as a desirable alternative to Fourier approaches in situations where discontinuities or nonstationarity are present in a signal. In the next section, we introduce wavelets through the idea of a multiresolution analysis before discussing some of the commonly used wavelet basis functions.

### 2.1.2 Multiresolution Analysis (MRA)

A multiresolution analysis (MRA), first introduced in Meyer (1986) and Mallat (1989a), is a sequence of closed subspaces $\{V_j\}_{j\in\mathbb{Z}}$ in $\mathbb{L}_2(\mathbb{R})$ such that the following containment holds

$$\ldots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \ldots \tag{2.1.3}$$

In order to be considered a MRA, this sequence of closed subspaces must also satisfy some additional conditions given by

1. $\overline{\bigcup_{j\in\mathbb{Z}} V_j} = \mathbb{L}_2(\mathbb{R})$.
2. $\bigcap_{j\in\mathbb{Z}} V_j = \{\mathbf{0}\}$.
3. $f(2^j x) \in V_j \iff f(x) \in V_0$.
4. There exists a scaling function $\phi(x) \in V_0$ such that any element of $V_0$ can be

expressed as a linear combination of translated copies of $\phi(x)$. As such, the set $\{\phi(x-k)\}_{k\in\mathbb{Z}}$ is an orthonormal basis of $V_0$.

The scaling function $\phi(x)$ is also known as the father wavelet. As a consequence of Conditions 3 and 4, we can see that the set

$$\{\phi_{j,k}(x) = 2^{\frac{j}{2}}\phi(2^j x - k)\}_{k\in\mathbb{Z}}$$

is a basis of $V_j$ for some fixed $j$.

A wavelet basis is characterised by two functions, the father wavelet as defined above and a mother wavelet function. If we have subspaces that satisfy the conditions for a MRA then we can define the mother wavelet. First we must define the subspace $W_j$ which is the orthogonal complement of $V_j$ in $V_{j+1}$. In other words,

$$V_{j+1} = V_j \oplus W_j.$$

Due to this relation, we can write any function $g(x) \in V_{j+1}$ uniquely as a linear combination of elements of $V_j$ and $W_j$. I.e. $g(x) = v_j(x) + w_j(x)$ where $v_j(x) \in V_j$ and $w_j(x) \in W_j$.

If we have a sequence of subspaces that satisfy the conditions for a MRA then there exists an orthonormal basis for $\mathbb{L}_2(\mathbb{R})$ given by

$$\{\psi_{j,k}(x) = 2^{\frac{j}{2}}\psi(2^j x - k)\}_{j,k\in\mathbb{Z}},$$

such that $\{2^{\frac{j}{2}}\psi(2^j x - k)\}_{k\in\mathbb{Z}}$ is an orthonormal basis of $W_j$ for fixed $j$. As a consequence of this we can define the mother wavelet, $\psi(x) = \psi_{0,0}(x)$.

A key benefit of the MRA approach is that it provides us with the framework to easily derive some of the important equations relating to wavelet theory, for example the dilation equations. We know that the father wavelet $\phi(x) \in V_0$ and, by the

definition of MRA, specifically relation (2.1.3), we obtain the containment $V_0 \subset V_1$. As a consequence of this, we can conclude that $\phi(x) \in V_1$. Therefore we can represent $\phi(x)$ as a linear combination of functions from $V_1$, which has an orthonormal basis given by $\{\phi_{1,k}(x) = \sqrt{2}\phi(2x-k)\}_{k\in\mathbb{Z}}$. From this we can obtain the dilation equation

$$\phi(x) = \sum_{k\in\mathbb{Z}} h_k \phi_{1,k}(x) = \sum_{k\in\mathbb{Z}} h_k \sqrt{2}\phi(2x-k) \tag{2.1.4}$$

where $\mathcal{H} = \{h_k\}_{k\in\mathbb{Z}}$ is the low-pass filter. In a similar way we can obtain the corresponding dilation equation for $\psi(x)$

$$\psi(x) = \sum_{k\in\mathbb{Z}} g_k \psi_{1,k}(x) = \sum_{k\in\mathbb{Z}} g_k \sqrt{2}\psi(2x-k) \tag{2.1.5}$$

where $\mathcal{G} = \{g_k\}_{k\in\mathbb{Z}}$ is the high-pass filter.

In addition, we can use MRA to show that the following links exist between the father and mother wavelets

$$\psi(x) = \sum_{k\in\mathbb{Z}} g_k \sqrt{2}\phi(2x-k), \tag{2.1.6}$$

$$\phi(x) = \sum_{k\in\mathbb{Z}} h_k \sqrt{2}\psi(2x-k). \tag{2.1.7}$$

For example in the case of equation (2.1.6), we know that $\psi(x) = \psi_{0,0}(x) \in W_0$ and by the definition of the space $W_0$ we obtain the containment $W_0 \subset V_1$. As a consequence, $\psi(x) \in V_1$ can be written as a linear combination of the basis functions for $V_1$. Hence $\psi(x)$ has the representation given by equation (2.1.6).

The quadrature mirror filter relation provides a formula that links the low and high-pass filters, this is given by

$$g_n = (-1)^n h_{1-n}. \tag{2.1.8}$$

Therefore, providing we have full knowledge of one of the filters then we can easily find the other using equation (2.1.8).

### 2.1.3 Wavelet Families

We now discuss some of the most commonly used wavelet families. Specifically, we focus within this thesis on the Daubechies wavelets (Daubechies, 1988). The Daubechies wavelets are split into two families; the "extremal phase" and "least asymmetric" wavelets. More details on these wavelets can be found in Daubechies (1992) or Vidakovic (1999). Within these families, the mother and father wavelet functions exhibit varying levels of smoothness depending on the number of vanishing moments $N$. Figure 2.1.2 shows the Daubechies extremal phase mother wavelet functions for different numbers of vanishing moments. Following Meyer (1992), a mother wavelet $\psi(x)$ of order $N$ must satisfy the following properties:

1. $\psi(x) \in L^\infty(\mathbb{R})$. If $N > 1$ then $\dfrac{\mathrm{d}^n}{\mathrm{d}x^n}\psi(x) \in L^\infty(\mathbb{R})$ for all $n \leq N$.

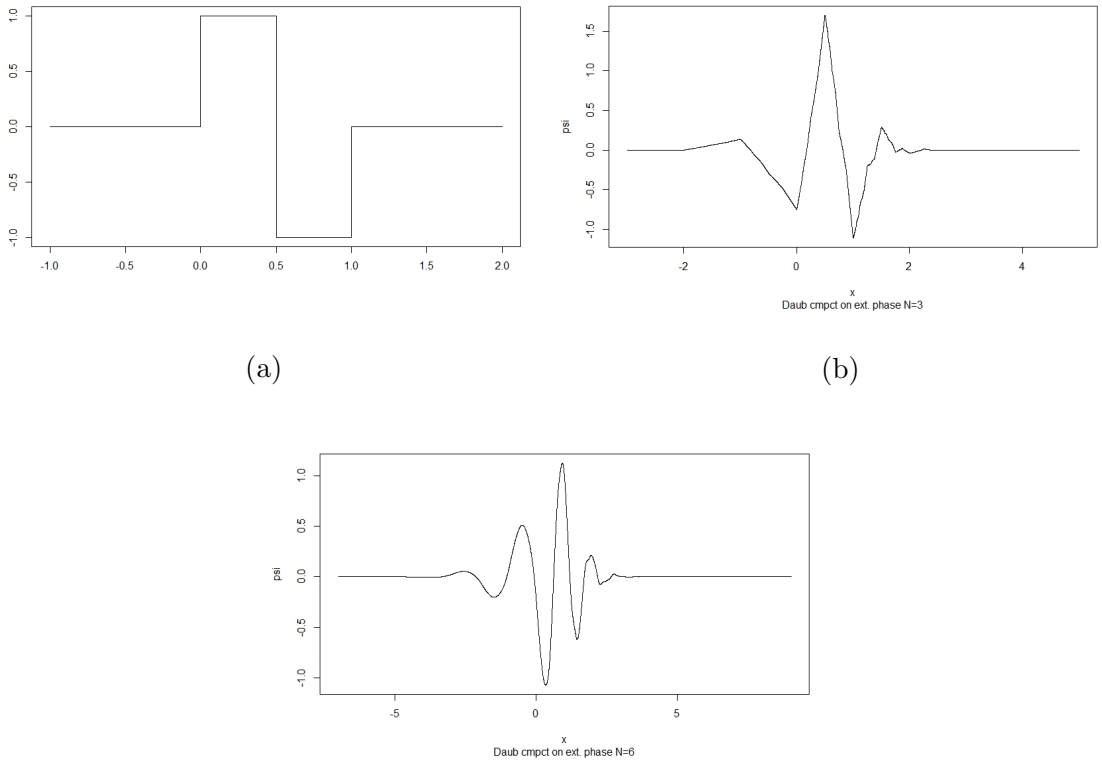
2. $\psi(x)$ and all its derivatives up to order $N$ decrease rapidly as $x \to \pm\infty$.

3. For all $k \in \{0, \ldots, N\}$,
$$\int_{-\infty}^{\infty} x^k \psi(x) \mathrm{d}x = 0 \tag{2.1.9}$$

4. The collection $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ forms an orthonormal basis of $L^2(\mathbb{R})$, the $\psi_{j,k}$ being constructed from the mother wavelet using the identity
$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k). \tag{2.1.10}$$

Here $j$ relates to the dilation (known as the scale), whilst $k$ relates to the translation (known as the location).

The second property ensures that the wavelet has compact support whilst the third, often referred to as the *vanishing moments* property, ensures that wavelets can produce sparse representations of functions that contain a finite number of discontinuities.




(a) (b)

(c)

Figure 2.1.2: Daubechies Extremal Phase mother wavelets for $N = 1, 3$ and 6.

It can be shown that any function $f(x) \in L^2(\mathbb{R})$ can be represented as a linear combination of a smooth component and scaled and translated copies of the mother

wavelet

$$f(x) = \sum_{k} c_{0,k}\phi_{0,k}(x) + \sum_{j=1}^{\infty}\sum_{k=-\infty}^{\infty} d_{j,k}\psi_{j,k}(x) \tag{2.1.11}$$

where $\{d_{j,k}\}$ are the set of wavelet detail coefficients which give information on the local oscillatory behaviour of the function at scale $j$ and location $k$.

### 2.1.4 Discrete Wavelet Transform (DWT)

In the previous section we introduced some of the most common wavelet families that are used in analysis. These wavelets are the building blocks for a variety of methods including wavelet transforms. In this section we will consider one such transform, specifically the Discrete Wavelet Transform (DWT). The DWT provides a time-scale representation of a signal in which the information contained in the signal is encoded in a set of smooth and detail coefficients.

The dilation equations defined in equations (2.1.4) and (2.1.5) can be used to derive the following relations which allow us to find the father and mother wavelets at progressively coarser scales using the high and low-pass filters

$$\phi_{j-1,k}(x) = \sum_{n\in\mathbb{Z}} h_{n-2k}\phi_{j,n}(x), \tag{2.1.12}$$

$$\psi_{j-1,k}(x) = \sum_{n\in\mathbb{Z}} g_{n-2k}\phi_{j,n}(x). \tag{2.1.13}$$

The Discrete Wavelet Transform, proposed in Mallat (1989a,b), takes a regularly spaced signal of dyadic length $2^J$ (for some $J > 0$) and decomposes it into a set of smooth and detail coefficients which encapsulate the information contained in the signal at different scales/resolutions. Equations (2.1.12) and (2.1.13) can be used to derive the equations that calculate the smooth and detail coefficients at each level of

the transform. The number of coefficients produced by the transform changes at each step of the decomposition with more coefficients at finer scales and fewer at coarser scales.

The smooth coefficients, $c_{j,k}$, are associated with the father wavelet functions, $\phi_{j,k}(x)$, and the detail coefficients, $d_{j,k}$, are associated with the mother wavelet functions $\psi_{j,k}(x)$. The smooth coefficients at scale $2^j$ are given by

$$c_{j,k} = \int_{-\infty}^{\infty} f(x) 2^{\frac{j}{2}} \phi(2^j x - k) \mathrm{d}x = \int_{-\infty}^{\infty} f(x) \phi_{j,k}(x) \mathrm{d}x = \langle f(x), \phi_{j,k}(x) \rangle. \quad (2.1.14)$$

In a similar way, the detail coefficients at scale $2^j$ can be found using

$$d_{j,k} = \int_{-\infty}^{\infty} f(x) 2^{\frac{j}{2}} \psi(2^j x - k) \mathrm{d}x = \int_{-\infty}^{\infty} f(x) \psi_{j,k}(x) \mathrm{d}x = \langle f(x), \psi_{j,k}(x) \rangle. \quad (2.1.15)$$

Using ideas motivated by the MRA framework, it is possible to define equations to compute the smooth and detail coefficients at coarser scales in the following way

$$c_{j-1,l} = \sum_k h_{k-2l} c_{j,k}, \quad (2.1.16)$$

$$d_{j-1,l} = \sum_k g_{k-2l} c_{j,k}, \quad (2.1.17)$$

where $\{h_k\}$ and $\{g_k\}$ are the low and high-pass filter coefficients. The DWT takes an initial data vector $\mathbf{x} = \{x_0, x_1, \ldots, x_{T-1}\}$ of length $T = 2^J$ and splits it into several smooth and detail vectors. Let $\mathbf{c}^J = \{c_{J,k}\}_k = \mathbf{x}$ be the initial data vector. Setting $j = J$ in equation (2.1.16) and (2.1.17) allows us to compute the smooth ($c_{J-1,k}$) and detail ($d_{J-1,k}$) coefficients at the finest scale. The set of smooth coefficients are then filtered again using equations (2.1.16) and (2.1.17) to obtain coefficients at the next coarsest scale. This process is repeated until the coarsest scale is reached, the final

set of coefficients are then given by

$$(\mathbf{c}_0, \mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{J-1}). \tag{2.1.18}$$

Note that the DWT is a decimated transform, for each level $j \in \{1, \ldots, J\}$ the transform calculates coefficients for locations $k \in \{0, \ldots, 2^{J-j} - 1\}$ which means that the number of coefficients decreases for coarser levels.

The support of the wavelet filters used to decompose the series can sometimes extend beyond the range of the data. In order to deal with these boundary problems, Nason and Silverman (1994) proposed a number of solutions including

- **Symmetry** - The sequence is reflected at the endpoints to extend the original length of the sequence, so that it has the form $(y_0, \ldots, y_{T-1}, y_{T-2}, y_{T-3}, \ldots)$.
- **Periodic** - Assume the sequence is periodic on the range of the data, so that $y_{k+T} = y_{k-T} = y_k$ for $k = 0, \ldots, T-1$.
- **Zero padding** - The sequence is padded with zeroes outside the range of the data, $(y_0, \ldots, y_{T-1}, 0, 0, \ldots)$.

Other solutions include specifically designed wavelets that always remain on the domain of the original data, see Cohen et al. (1993) for a description of wavelets on the interval.

Instead of using the MRA framework, we can think of the DWT in an alternative way that involves a filtering and decimation step, as described in Nason and Silverman (1995). First we define the even decimation operator $\mathcal{D}_0$ that takes every even-indexed

element in a sequence. This has the form

$$(\mathcal{D}_0 x)_l = x_{2l} \tag{2.1.19}$$

for the sequence $\{x_l\}$. Let $\mathcal{H}$ and $\mathcal{G}$ represent the low and high-pass convolution operators, defined by

$$(\mathcal{H}x)_k = \sum_n h_{n-k} x_n, \tag{2.1.20}$$

$$(\mathcal{G}x)_k = \sum_n g_{n-k} x_n. \tag{2.1.21}$$

As above, let $c_J = \{c_{J,k}\}_k = \{x_0, x_1, \ldots, x_{2^J-1}\}$ be the initial data vector. Applying the DWT on this vector using equations (2.1.16) and (2.1.17) can be written in the following way using operator notation

$$c_{j-1} = \mathcal{D}_0 \mathcal{H} c_j, \tag{2.1.22}$$

$$d_{j-1} = \mathcal{D}_0 \mathcal{G} c_j, \tag{2.1.23}$$

for $j = J, \ldots, 1$. Following Nason and Silverman (1995), the DWT smooth and detail coefficients at level $j$ given the original data $c_J$ can be written as

$$d_j = \mathcal{D}_0 \mathcal{G} (\mathcal{D}_0 \mathcal{H})^{J-j-1} c_J, \tag{2.1.24}$$

$$c_j = (\mathcal{D}_0 \mathcal{H})^{J-j} c_J, \tag{2.1.25}$$

for $j = 0, 1, \ldots, J-1$.

A useful property of the DWT is that it is an orthogonal transform and can therefore be inverted to obtain the initial data vector. Given the smooth and detail coefficients at the coarsest level, Mallat (1989b) showed that the inversion relation is

given by

$$c_{j,k} = \sum_l h_{k-2l} c_{j-1,l} + \sum_l g_{k-2l} d_{j-1,l}. \tag{2.1.26}$$

The original series can be recovered exactly using equation (2.1.26) given the smooth coefficient at the coarsest scale $\mathbf{c}_0$ and the detail coefficients $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{J-1}$.

One of the major weaknesses of the Discrete Wavelet Transform is that it is not translation-invariant, this means that if we take the DWT of an input vector and then shift this input vector by some amount and again perform the transform then we will obtain different results. This is not an attractive quality for a wavelet transform to possess since it means that analysis using this transform will be affected by the choice of origin. An example of this can be seen in Figure 2.1.3; shifting the original data by one to the left results in completely different wavelet coefficients. The **wavethresh** R package (Nason, 2016) was used to apply the wavelet transform and generate the plots of the wavelet decomposition coefficients. The Non-decimated Wavelet Transform (NDWT) was developed as a solution to this problem and will be outlined in the next section.

### 2.1.5 Non-decimated Wavelet Transform (NDWT)

Nason and Silverman (1995) introduced the Non-decimated Wavelet Transform and outlined the key differences between this method and that of the Discrete Wavelet Transform. As with the DWT, this transform can only be used when the input data is regularly spaced and of dyadic length. The first step of the NDWT is to take the input data vector and filter with the high and low-pass filters. However, no decimation step

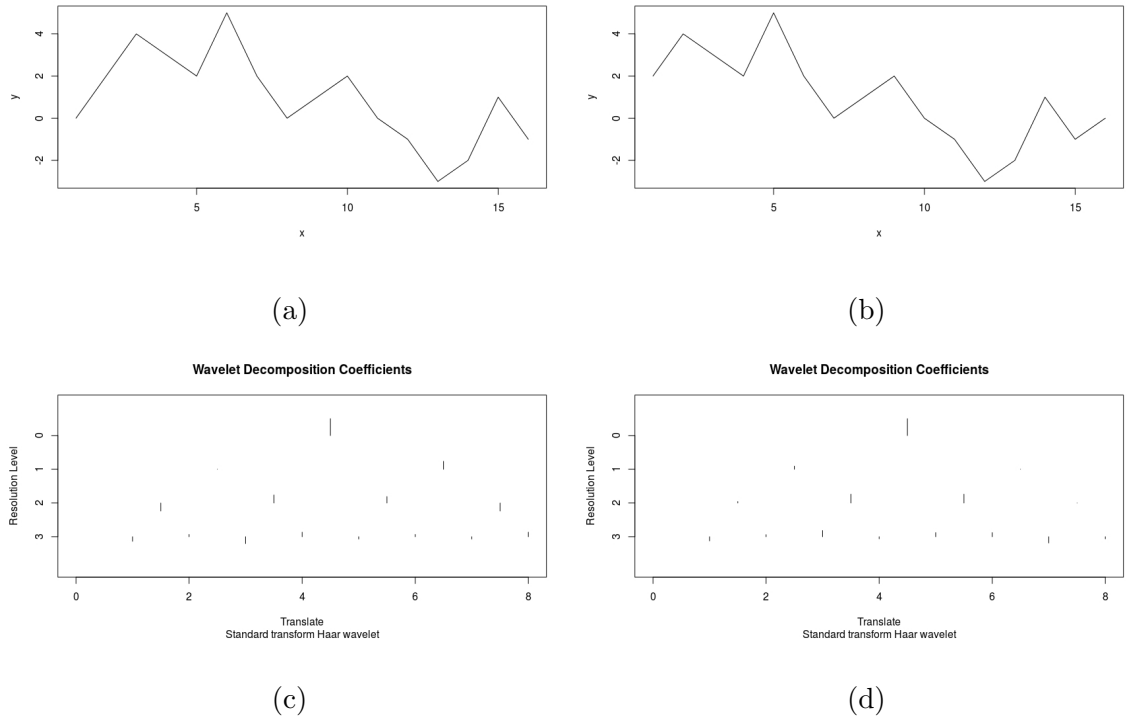



Figure 2.1.3: Example of the lack of translation-invariance of the DWT; (a) shows the original data whilst (b) shows the data rotated by a unit shift. Figures (c) and (d) depict the Haar DWT for the original and shifted data respectively.

is carried out which means that the number of smooth and detail coefficients does not change at each level of the process.

At each step of the NDWT we have to pad the high and low pass filters with zeroes. Let $\mathcal{H} = \{h_k\}_{k\in\mathbb{Z}}$ be the initial low-pass filter and let $\mathcal{G} = \{g_k\}_{k\in\mathbb{Z}}$ be the high-pass filter. Following the notation of Nason and Silverman (1995), we define the filters $\mathcal{H}^{[r]}$ and $\mathcal{G}^{[r]}$ which consist of the elements

$$\begin{aligned} h^{[r]}_{2^r j} &= h_j, & g^{[r]}_{2^r j} &= g_j, \\ h^{[r]}_{k} &= 0, & g^{[r]}_{k} &= 0, \quad \text{if } k \text{ is not a multiple of } 2^r. \end{aligned} \tag{2.1.27}$$

The filter $\mathcal{H}^{[r]}$ can be thought of as the filter which is obtained after a zero is added

between every adjacent element of the filter $\mathcal{H}^{[r-1]}$ which is used at the previous step of the algorithm. As in Section 2.1.4, let $\mathbf{c}^J = \{c_{J,k}\}_k$ be the initial data vector. The smooth and detail coefficients at scale $j-1$ are given by

$$\mathbf{c}^{(j-1)} = \mathcal{H}^{[J-j]}\mathbf{c}^{(j)}, \tag{2.1.28}$$

$$\mathbf{d}^{(j-1)} = \mathcal{G}^{[J-j]}\mathbf{c}^{(j)}, \tag{2.1.29}$$

where $\mathbf{c}^{(j)}$ is the vector of smooth coefficients at scale $j$ and $\mathbf{d}^{(j)}$ is the vector of detail coefficients. The operator $\mathcal{H}^{[r]}$ can be thought of as the convolution operator with the filter $\mathbf{h}^{(r)}$.

One of the major strengths of the NDWT, in comparison to the DWT, is that it includes extra information from the data at coarser scales, since no decimation step is carried out. Any information contained in the odd-indexed elements is included in the transform and not discarded, therefore each level of the transform will have coefficients for all locations $k \in \{0, \ldots, T-1\}$. In addition to this, the NDWT is translation-invariant and so is not affected by the choice of origin of the input data.

An example of the NDWT applied to a time series can be found in Figure 2.1.4, where the time series considered are the same as those from Figure 2.1.3. The translation-invariance property of the NDWT can be seen from Figures 2.1.4(c) and 2.1.4(d), a unit shift in the data has resulted in a shift in the wavelet coefficients at each level. It can also be seen that the NDWT provides more information than the DWT as we have $T = 2^J$ wavelet coefficients at each level in this case rather than the decreasing number shown in Figures 2.1.3(c) and 2.1.3(d). However it must be noted that the transform is overcomplete and therefore is not orthogonal. As a

consequence of this, the NDWT does not have a unique inverse and so the algorithm is not easily invertible. A number of approaches have been developed for inverting the NDWT including the Average Basis method of Coifman and Donoho (1995) or using the algorithm of Coifman and Wickerhauser (1992) to select a best basis and reconstructing the signal from this.

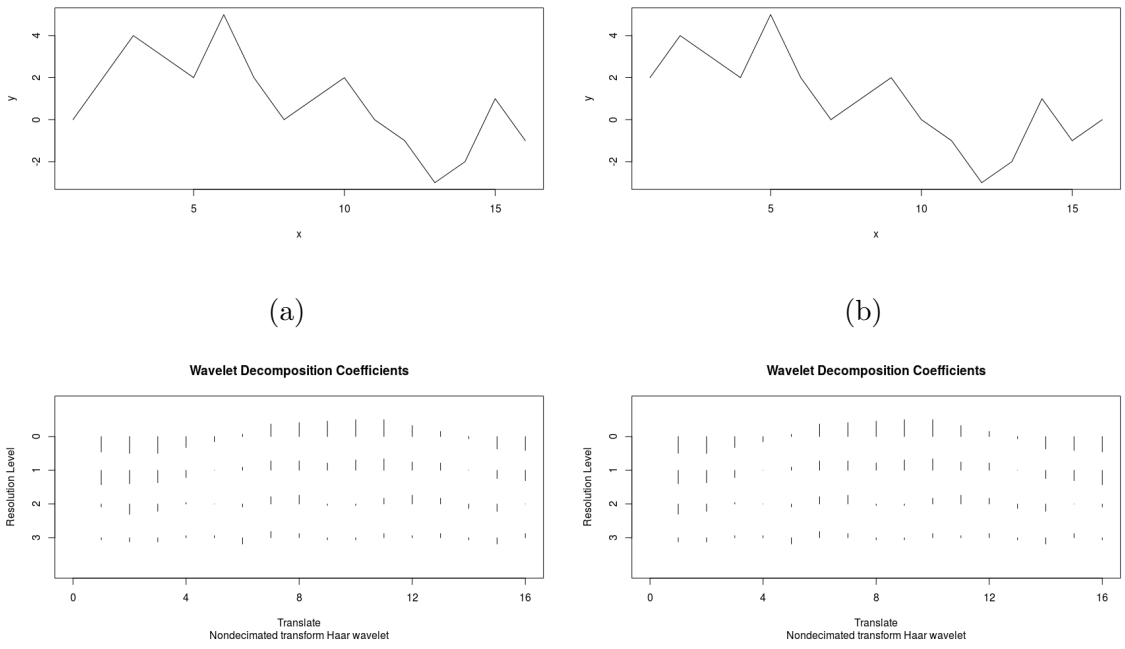



Figure 2.1.4: Example of the translation-invariance of the NDWT; (a) shows the original data whilst (b) shows the data rotated by a unit shift. Figures (c) and (d) depict the Haar NDWT for the original and shifted data respectively.

In addition to the DWT and NDWT, there exists a range of other wavelet transforms within the literature; we include a brief discussion of a few examples for completeness. One such example is the Maximal Overlap Discrete Wavelet Transform (MODWT) which has the same translation-invariance property of the NDWT whilst

relaxing the restriction that the input data must be dyadic length, see Percival and Walden (2006) for a complete description. Another example is the wavelet packet transform of Coifman and Wickerhauser (1992). This is a generalization of the discrete transforms discussed above in which the low and high-pass filters are applied to both the smooth and detail coefficients and these are then carried forward to the next level of the transform. As with the NDWT, if all of the wavelet packets are included in the decomposition then it can be seen that this is an overcomplete transform. However Coifman and Wickerhauser (1992) proposed the best basis algorithm to select which packets are best for representing the series and forming an orthogonal transform. Finally, the lifting scheme, introduced in Sweldens (1996, 1998), can be used to produce a multi-scale decomposition of signals observed on an irregularly spaced grid or containing missing data in a computationally efficient manner.

## 2.2 Time Series Analysis

We now turn our attention to some key time series analysis concepts relevant for this thesis. A time series is a collection of observations of a process made sequentially through time. These measurements can either be taken continuously through time or at discrete, successive time points. Within this thesis, we focus on discrete time series $\{x_t\}_{t\in\mathbb{N}}$. A comprehensive overview of time series analysis concepts can be found in Shumway and Stoffer (2000) and Chatfield (2004). Below we briefly review the concept of time series stationarity before concentrating on popular time series models for both the stationary and nonstationary setting.

One of the most widely covered topics within time series analysis is that of stationarity, in particular stationary time series modelling. Informally, one can consider a stationary time series to be one whose statistical properties do not change over time. More specifically, following Shumway and Stoffer (2000), a time series is said to be strictly stationary if the probabilistic behaviour of every subset of values

$$\{x_{t_1}, x_{t_2}, \ldots, x_{t_k}\}$$

is the same as that of the time shifted subset

$$\{x_{t_1+h}, x_{t_2+h}, \ldots, x_{t_k+h}\}.$$

This can be expressed as,

$$P(x_{t_1} \leq c_1, \ldots, x_{t_k} \leq c_k) = P(x_{t_1+h} \leq c_1, \ldots, x_{t_k+h} \leq c_k)$$

for all $k = 1, 2, \ldots$, all time points $t_1, t_2, \ldots, t_k$, all time shifts $h = 0, \pm 1, \pm 2, \ldots$, and all constants $c_1, c_2, \ldots, c_k$.

In practice, these strict assumptions of stationarity can often be too restrictive and so we consider an alternative; that of second order stationarity. A process is said to be second order stationary if it has constant mean and the covariance between observations depends only on the lag between them and not on time, in other words

$$E[x_t] = \mu, \qquad \text{and} \qquad \gamma_t(\tau) = \text{cov}(x_t, x_{t+\tau}) = \kappa_\tau.$$

For the remainder of this thesis, the use of the term stationary time series will refer to second order stationary time series. In addition, we assume that all of the time series considered throughout this thesis are zero-mean processes. The problem of

modelling stationary time series and efficiently capturing their dependence structure is an important one, we will now review some popular time series models in Section 2.2.1.

### 2.2.1 Stationary time series modelling

There are many existing stationary time series models within the literature that incorporate some degree of changing dependence; some focus on modelling the behaviour of the series in the time domain whereas others concentrate on the frequency domain.

#### Stationary processes in time domain

Popular approaches to modelling univariate stationary processes in the time domain include the moving average processes and the autoregressive processes, introduced in Yule (1927). A moving average process of order $q$, denoted MA($q$), is a linear combination of the current innovation term and the $q$ most recent innovations. The $t^{\text{th}}$ observation of the process takes the form

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q} \tag{2.2.1}$$

where $\epsilon_t$ are iid white noise processes with variance $\sigma^2$ for all $t \in \mathbb{N}$ and $\{\theta_t\}_{t\in\{1,\ldots,q\}}$ is the set of MA coefficients.

In a similar way, an autoregressive process of order $p$ is a linear combination of the current innovation term and the $p$ most recent observations. The $t^{\text{th}}$ observation of the process takes the form

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + \epsilon_t \tag{2.2.2}$$

where again $\epsilon_t$ are iid white noise processes with variance $\sigma^2$ and the $\phi_i$ are the model parameters with $\phi_p \neq 0$ for an order $p$ process.

Whittle (1951) combined the MA and AR processes to form the autoregressive moving average (ARMA) processes. The $t^{\text{th}}$ observation of an ARMA$(p, q)$ process is given by

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q} \tag{2.2.3}$$

where $p$ and $q$ are the AR and MA orders respectively and $\epsilon$ is white noise with variance $\sigma^2$.

For a stationary time series, the autocorrelation and partial autocorrelation functions can be used to estimate the values of $p$ and $q$ to be used in the ARMA model. The Box-Jenkins methodology is commonly used to fit the stationary time series model and estimate the AR and MA model parameters, further details on this method can be found in Chatfield (2004) and Box et al. (2015).

All of the models described above are univariate and could be applied to individual components of a multivariate time series however this would not account for any dependencies *between* the components. For this reason, multivariate extensions of the above models have been proposed in the literature. By way of introduction, we briefly outline the vector autoregressive moving average processes. Interested readers can find further details in Tiao and Tsay (1989) and Lütkepohl (2007).

Similar to the ARMA$(p, q)$ process defined in equation (2.2.3), a vector autoregressive moving average (VARMA) model can be used to capture multivariate dependence where the current observation is a linear combination of past realisations and inno-

vations. The $t^{\text{th}}$ observation of an m-dimensional VARMA$(p, q)$ process $\mathbf{X}_t$ takes the form

$$\mathbf{X}_t = A_1\mathbf{X}_{t-1} + \ldots + A_p\mathbf{X}_{t-p} + \mathbf{Z}_t + M_1\mathbf{Z}_{t-1} + \ldots + M_q\mathbf{Z}_{t-q} \tag{2.2.4}$$

where $\{A_l\}_{l\in 1,\ldots,p}$ and $\{M_l\}_{l\in 1,\ldots,q}$ are $(m \times m)$ matrices of AR and MA coefficients respectively which capture the temporal dependence of the multivariate time series. In this case, $\mathbf{Z}_t$ is a $(m \times 1)$ innovation vector with mean 0 and covariance matrix $\Sigma$.

**Stationary processes in frequency domain**

Within the frequency domain, the Fourier basis has been used to construct representations of stationary univariate processes, see Priestley (1981) for a comprehensive description.

A zero-mean stationary stochastic process $X_t$ can be represented in the following way within the Fourier setting

$$X_t = \int_{-\pi}^{\pi} A(\omega)\exp(i\omega t)\mathrm{d}\xi(\omega) \qquad \text{for } t \in \mathbb{Z} \text{ and } \omega \in [-\pi, \pi]. \tag{2.2.5}$$

Here, $A(\omega)$ is the amplitude of the process, $\exp(i\omega t)$ is the oscillation and $\mathrm{d}\xi(\omega)$ is an orthonormal increments process. The spectrum of a process provides information on the power of the process at a given frequency. More precisely, for a time series of length $T$, the spectrum at frequency $\omega$ of a stationary process $X$ is given by

$$f_X(\omega) = T \sum_{\tau=-\infty}^{\infty} \gamma(\tau)\, \exp(-i\omega\tau T) \tag{2.2.6}$$

where $\gamma(\tau) = \mathrm{cov}(X_t, X_{t+\tau})$. The covariance of the process can also be expressed in

terms of the spectrum in the following way

$$\gamma(\tau) = \frac{1}{T} \int f_X(\omega) \exp(i\omega\tau T) \mathrm{d}\omega. \tag{2.2.7}$$

### 2.2.2 Nonstationary time series modelling

In practice, many of the time series observed from industrial and other applications are nonstationary. I.e. their second order structure evolves over time. When the stationarity assumptions no longer hold, the previously described models should not be applied in their standard forms because poor estimates of the model parameters will be produced and the nonstationarity will not be accurately captured. Consequently, further thought is required to model the changing second order behaviour. One way to achieve this would be to adapt the ARMA framework to allow the model coefficients to change over time, see Subba Rao (1970) and Grenier (1983) for details on how to estimate these coefficients. In this section, we review some more recent approaches to modelling nonstationary time series using a variety of basis functions, including the Locally Stationary Fourier processes and the Smooth Localized Exponential (SLEX) model.

#### Locally Stationary Fourier processes

The problem of modelling nonstationary behaviour using a Fourier representation dates back to the work of Priestley (1965), through the concept of an evolutionary spectra. We describe here the Locally Stationary Fourier (LSF) processes as detailed in Dahlhaus (1997). Within this LSF framework, a zero-mean nonstationary series

$\{X_{t,T}\}_{t=0}^{T-1}$ can be represented as follows

$$X_{t,T} = \int_{-\pi}^{\pi} A_{t,T}^{0}(\omega)\exp(i\omega t)\mathrm{d}\xi(\omega) \tag{2.2.8}$$

where $\{\exp(i\omega t)\}_{\omega}$ is a set of harmonics and $A_{t,T}^{0}(\omega)$ is the transfer function. A number of smoothness conditions are imposed on $A_{t,T}^{0}(\omega)$ to ensure that the amplitude is not too irregular over time. Further details on these conditions can be found in Definition 2.1 of Dahlhaus (1997). The representation in equation (2.2.8) assumes that the series is locally stationary, that is, if we observed the series at sufficiently small intervals then it would appear stationary.

Within the LSF framework, Dahlhaus (1997) introduced the concept of rescaled time. In other words, rather than observing a time series and the amplitude over a grid $t \in \{1, \ldots, T\}$, instead observe them on the interval $u = t/T \in [0, 1]$. In this setting, increasing the number of observations $T$ corresponds to observing the series on a finer grid. This mathematical construct is important, as it enables us to establish theoretical guarantees about the behaviour of LSF estimates.

Dahlhaus (2000) presents the multivariate generalization of the univariate local stationarity described above. It is shown that there exists a locally stationary representation analogous to that in equation (2.2.8) for Gaussian multivariate series and time-varying MA($\infty$) processes. For a $d$-dimensional series, the univariate transfer function is replaced with a $(d \times d)$ transfer function matrix $A^0$ where smoothness conditions are imposed on each entry $A_{ab}^{0}$ to control the evolution of the amplitude (for $a, b = 1, \ldots, d$). A complete description of these conditions can be found in Definition 2.1 of Dahlhaus (2000).

## Smooth Localized Exponential (SLEX) model

The LSF framework is by no means the only representation of a locally stationary time series. For example, Ombao et al. (2002) build on the Locally Stationary Fourier model of Dahlhaus (1997) to provide an alternative locally stationary representation of a nonstationary process. Within the time series literature, windowed or tapered Fourier exponentials have traditionally been used to model and analyse nonstationary processes (Daubechies, 1992). These functions take the form

$$\phi_F(u) = \Psi(u)\exp(i2\pi\omega u) \tag{2.2.9}$$

where $\Psi$ is a taper with compact support and $\omega \in (-\frac{1}{2}, \frac{1}{2}]$. The main drawback associated with the windowed Fourier exponentials is that, whilst they are localized in time, they are not necessarily orthogonal (Wickerhauser, 1994). Ombao et al. (2002) replace the set of Fourier complex exponential basis functions within the Locally Stationary Fourier representation of equation (2.2.8) by the Smooth Localized Complex Exponential (SLEX) basis functions in order to overcome this problem. Their approach preserves the orthogonality of the basis functions through the use of a projection operator, rather than applying a single taper. This projection operator has the same effect as applying two smooth and compactly supported windows to the Fourier basis functions.

The nonstationary time series is segmented dyadically into stationary blocks, where neighbouring blocks are allowed to overlap by some small amount $\epsilon$. A SLEX basis vector defined on block $S$ with support $\{\alpha_0 - \epsilon, \ldots, \alpha_0, \ldots, \alpha_1 - 1, \alpha_1 - 1 + \epsilon\}$

has elements $\{\phi_{S,\omega_k}(t)\}$ where

$$\begin{aligned}\phi_{S,\omega_k}(t) &= \phi_{\omega_k}\left(\frac{t-\alpha_0}{|S|}\right) \\ &= \Psi_+\left(\frac{t-\alpha_0}{|S|}\right)\exp(i2\pi\omega_k(t-\alpha_0)) + \Psi_-\left(\frac{t-\alpha_0}{|S|}\right)\exp(-i2\pi\omega_k(t-\alpha_0))\end{aligned} \tag{2.2.10}$$

where $\omega_k = \dfrac{k}{|S|}$ and $k = -\dfrac{|S|}{2}+1,\ldots,\dfrac{|S|}{2}$. A complete description of the locally stationary SLEX model can be found in Ombao et al. (2002), including a description of how to select the best SLEX basis for the particular time series.

A multivariate extension of the SLEX model was introduced by Ombao et al. (2005), in which the scalar transfer functions used within the locally stationary univariate representation are again replaced by $(d \times d)$ transfer function matrices defined on particular blocks. In addition, the cost function used to determine the best SLEX model for the time series is modified to take into account auto and cross-correlation information from all channels.

## 2.3 Wavelets in Time series

The various methods for modelling nonstationary time series through locally stationary representations, discussed in Section 2.2.2, suffer from a number of drawbacks. For example, the windowed Fourier exponentials used to model nonstationary time series need to be carefully selected to accurately reflect the behaviour of the series. An automatic approach for determining these windows based on the observed time series has been developed but can be computationally intensive to implement. On

the other hand, whilst the SLEX model is computationally efficient, the requirement that the stationary blocks that segment the time series must be of dyadic length can be restrictive. In this section, we discuss an alternative model that is built on non-decimated wavelets, called the Locally Stationary Wavelet model. We then discuss the multivariate generalization, the Multivariate Locally Stationary Wavelet model, which is used within the time series classification and imputation methods outlined in Chapter 3 and 4 respectively.

### 2.3.1 Locally Stationary Wavelet model

The Locally Stationary Wavelet (LSW) processes, introduced by Nason et al. (2000), can be used to model a nonstationary time series with changing second order structure. The model has as its building blocks the discrete non-decimated wavelets which are compactly supported and localized in time and scale. A LSW process $\{X_{t,T}\}_{t=0,\ldots,T-1}$, $T = 2^J \geq 1$ is formally defined as follows

$$X_{t,T} = \sum_{j=1}^{\infty} \sum_{k} W_j(k/T)\psi_{jk}(t)\xi_{jk}. \tag{2.3.1}$$

Here, $\{\psi_{jk}(t)\}_{jk}$ is a collection of discrete non-decimated wavelets, $\xi_{jk}$ is a random orthonormal increments sequence and $\{W_j(k/T)\}$ is a set of amplitudes. Consequently, the time series is modelled as the sum of a collection of non-decimated wavelets $\{\psi_{jk}(t)\}_{jk}$ (oscillations) which have random amplitudes $\{W_j(k/T)\xi_{jk}\}$.

The quantities in equation (2.3.1) must satisfy a number of assumptions in order to ensure that the LSW process has zero mean, does not oscillate too rapidly and the random orthonormal increments sequence is uncorrelated. Following Nason et al.

(2000), these assumptions are

1. $E(\xi_{jk}) = 0$ for all $j$ and $k$.

2. $\text{cov}(\xi_{jk}, \xi_{lm}) = \delta_{jl}\delta_{km}$, where $\delta_{jk}$ is the Kronecker delta.

3. There exists a Lipschitz continuous function $W_j(z)$ for each $j \geq 1, z \in (0,1)$ which satisfies the following properties:

   (a) $\sum_j |W_j(z)|^2 < \infty$ uniformly in $z \in (0,1)$,

   (b) The Lipschitz constants $L_j$ are uniformly bounded in $j$ and

   $$\sum_j 2^j L_j < \infty.$$

   (c) There exists a sequence of constants $C_j$ such that for each $T$

   $$\sup_k \left| w_{j,k;T} - W_j\left(\frac{k}{T}\right) \right| \leq \frac{C_j}{T}$$

   where for each $j = 1, \ldots, J(T) = \log_2(T)$ the supremum is over $k = 0, 1, \ldots, T-1$, and where $\{C_j\}$ fulfils

   $$\sum_j C_j < \infty.$$

The first assumption ensures that the LSW process has mean zero and the second ensures that the orthonormal increments sequence is uncorrelated. The third assumption controls how the amplitudes change over time, with the amplitudes $w_{j,k:T}$ behaving similarly to some transfer function $W_j(z)$, which in turn is controlled so that it does not oscillate too rapidly.

With the basic model framework now in place, we proceed to consider some of the key measures associated with the LSW framework. The first of these is the

evolutionary wavelet spectrum (EWS). The EWS provides information about the power of the process at scale $j$ and location $z$ where $z = \frac{k}{T}$ is the rescaled time. This has the form

$$S_j(z) = |W_j(z)|^2 \tag{2.3.2}$$

for scale $j$ and rescaled time $z$. As in the Fourier setting, Nason et al. (2000) show that the autocovariance of function of a series can be written in terms of the EWS, similarly to equation (2.2.7). The local autocovariance (LACV) function gives information on the covariance of an LSW process around a location $z = k/T \in (0, 1)$. The LACV can be written as

$$c(z, \tau) = \text{cov}(X_{[zT]}, X_{[zT]-\tau}) = \sum_{j=1}^{\infty} S_j(z)\Psi_j(\tau) \tag{2.3.3}$$

where the autocorrelation wavelets $\Psi_j(\tau)$ are given by

$$\Psi_j(\tau) = \sum_k \psi_{jk}(0)\psi_{jk}(\tau). \tag{2.3.4}$$

**Estimation of the EWS**

The first step towards estimating the EWS is to calculate the empirical wavelet coefficients of $X_{t,T}$, which are defined to be

$$d_{j,k} = \sum_{t=0}^{T-1} X_{t,T}\psi_{jk}(t). \tag{2.3.5}$$

The raw wavelet periodogram of a LSW process $X_{t,T}$ is then given by

$$I_{j,k} = |d_{j,k}|^2. \tag{2.3.6}$$

Nason et al. (2000) show that the (asymptotic) mean and variance of the raw wavelet periodogram have the form

$$\mathbb{E}[I_{j,k}] = \sum_{l} A_{j,l} S_l(z) + \mathcal{O}(T^{-1}), \tag{2.3.7}$$

$$\text{var}[I_{j,k}] = 2\Big\{ \sum_{l} A_{j,l} S_l(z) \Big\}^2 + \mathcal{O}\left(\frac{2^j}{T}\right). \tag{2.3.8}$$

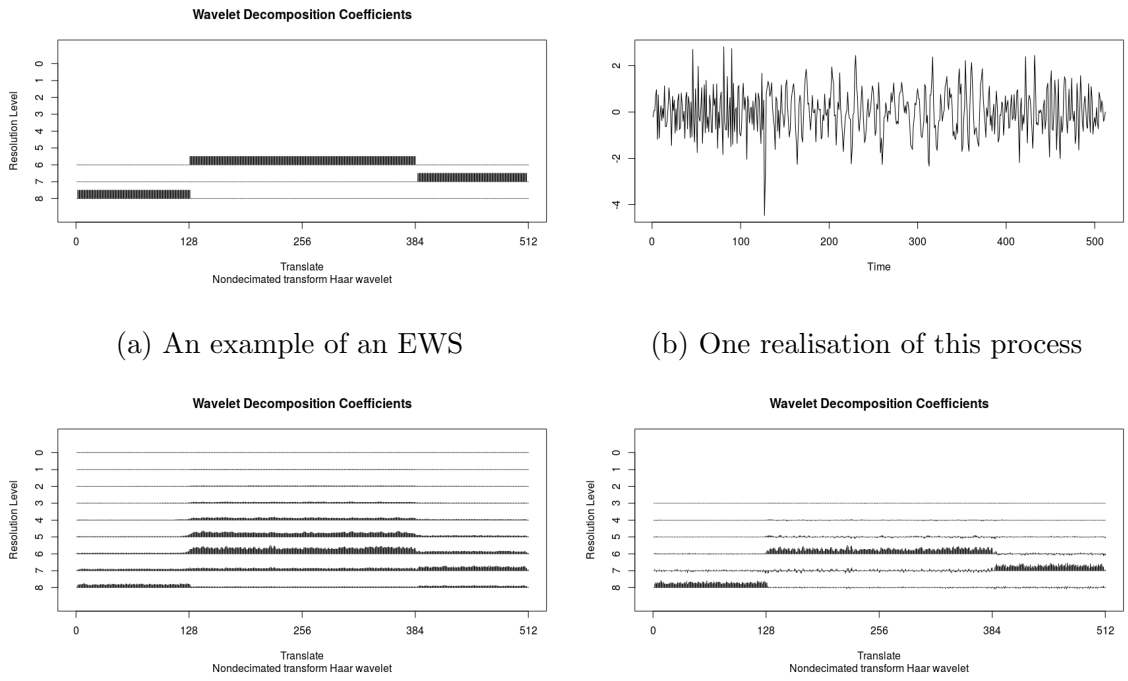
Note that $A_{j,l}$ refers to the $(j, l)$ entry of the inner product matrix $A$ of the autocorrelation wavelets, defined by

$$A_{j\ell} = \langle \Psi_j, \Psi_\ell \rangle = \sum_{\tau} \Psi_j(\tau) \Psi_\ell(\tau) \tag{2.3.9}$$

where $\Psi$ are the autocorrelation wavelets as defined in equation (2.3.4).

It can be seen from equation (2.3.7) and (2.3.8) that the raw wavelet periodogram is an asymptotically biased and inconsistent estimator of the EWS. Nason et al. (2000) propose smoothing the raw wavelet periodogram before correcting by $A^{-1}$ to obtain a consistent and unbiased estimator. Smoothing the raw wavelet periodogram can be done in a number of ways, Nason et al. (2000) use the translation-invariant denoising of Coifman and Donoho (1995) but alternative methods can be found in Fryzlewicz and Nason (2006) and Van Bellegem and Von Sachs (2008).

We demonstrate the effect of correcting the raw wavelet periodogram through an example. We consider an EWS containing varying power at finer scales, as in Figure 2.3.1(a). A realisation of this process can be seen in Figure 2.3.1(b), the effect of the changing power regimes in the EWS can clearly be seen in the simulated series. We simulate 100 realisations from the EWS shown in Figure 2.3.1(a) before calculating the raw and corrected wavelet periodogram for each realisation. Figure 2.3.1(c) shows




(a) An example of an EWS

(b) One realisation of this process

(c) Mean of 100 raw periodograms

(d) Mean of 100 corrected periodograms

Figure 2.3.1: Example showing the effect of correcting the raw wavelet periodograms.

the mean of the 100 raw wavelet periodograms, from this we can see the bias in the raw periodogram as power has leaked into surrounding scales. Correcting the raw wavelet periodogram removes most of this bias, Figure 2.3.1(d) shows the mean of the 100 corrected wavelet periodograms. We can see that most of the power leakage has disappeared and the corrected periodogram is much closer to the true EWS.

### 2.3.2 Multivariate Locally Stationary Wavelet framework

The Multivariate Locally Stationary Wavelet (mvLSW) framework, developed by Park et al. (2014), is a multivariate extension of the LSW processes that can be used to model multivariate time series with an evolving dependence structure between

components of the series. This framework allows each individual component of the time series to exhibit nonstationary behaviour, which is captured by the model in addition to the cross-dependence structure between components. A similar model is developed in Cho and Fryzlewicz (2015) however this is restricted to the context of changepoint detection of piecewise stationary signals.

Suppose that we have a $P$-variate input vector of the form $\mathbf{X}_{t,T} = \{X_{t,T}^{(1)}, X_{t,T}^{(2)}, \ldots, X_{t,T}^{(P)}\}^\top$. Each element $X_{t,T}^{(i)}$ (for $i = 1, 2, \ldots, P$) is a univariate component of the signal, known as a channel. The transfer function, $W_j(k/T)$, from the univariate case is replaced by a $P \times P$ transfer matrix of functions, denoted $\mathbf{V}_j(k/T)$. The random orthonormal increments sequence, $\xi_{jk}$, is replaced by a set of random vectors of the form $\{\mathbf{z}_{j,k}\} = \left\{[z_{j,k}^{(1)}, z_{j,k}^{(2)}, \ldots, z_{j,k}^{(P)}]^\top\right\}$.

Following Park et al. (2014), a $P$-variate locally stationary wavelet process $\{\mathbf{X}_{t,T}\}_{t=0,1,\ldots,T-1}$, $T = 2^J \geq 1$ has the following representation

$$\mathbf{X}_{t,T} = \sum_{j=1}^{\infty} \sum_{k} \mathbf{V}_j(k/T)\psi_{j,t-k}\mathbf{z}_{j,k}, \tag{2.3.10}$$

where $\mathbf{V}_j(k/T)$ is the transfer function matrix and $\{\psi_{j,t-k}\}_{j,k}$ is a set of discrete non-decimated wavelets. There are a number of conditions required of the quantities in equation (2.3.10) analogous to those for the univariate LSW case. Specifically, the random vectors, $\mathbf{z}_{j,k}$, are uncorrelated, have mean vector $\mathbf{0}$ and variance-covariance matrix equal to the $P \times P$ identity matrix. The lower-triangular transfer function matrix must be made up of Lipschitz continuous functions with Lipschitz constants $L_j$ that satisfy

$$\sum_j 2^j L_j^{(p,q)} < \infty.$$

The time-dependent transfer function matrix, $\mathbf{V}_j(k/T)$, encapsulates both the behaviour of the individual channels and the cross-channel dependence over time. The spectral structure of the multivariate time series is important as this provides information on the time-scale decomposition of power within the series; the transfer function matrix is key to determining this structure. In particular, given a multivariate LSW signal, $\mathbf{X}_{t,T}$, with transfer function matrix, $\mathbf{V}_j(k/T)$, the local wavelet spectral (LWS) matrix is given by

$$\mathbf{S}_j(z) = \mathbf{V}_j(z)\mathbf{V}_j(z)^\top \tag{2.3.11}$$

where $j$ is the scale and $z = \frac{k}{T}$ is the rescaled time. As in the univariate setting, the local auto and cross-covariance of a multivariate LSW series can be expressed in terms of the LWS matrix. Following Park et al. (2014), let $c^{(p,p)}(z,\tau)$ denote the local autocovariance of channel $p$ at lag $\tau$ and let $c^{(p,q)}(z,\tau)$ denote the local cross-covariance between channels $p$ and $q$. The local auto and cross-covariance functions are then defined to be

$$c^{(p,p)}(z,\tau) = \sum_{j=1}^{\infty} S_j^{(p,p)}(z)\Psi_j(\tau), \tag{2.3.12}$$

$$c^{(p,q)}(z,\tau) = \sum_{j=1}^{\infty} S_j^{(p,q)}(z)\Psi_j(\tau). \tag{2.3.13}$$

Here $\Psi$ are the discrete autocorrelation wavelets.

The LWS matrix is the multivariate analogue of the Evolutionary Wavelet Spectrum, defined in equation (2.3.2), and gives a measure of the local contribution to both the variance of the channels and the cross-covariance between channels made at a particular scale, $j$, and time, $z$. The diagonal elements of the matrix contain the spectra of the individual channels whereas the off-diagonal elements consist of

the cross-spectra between channels. Using the notation of Park et al. (2014), we denote the spectra of the individual channel $p$ by $\mathbf{S}_j^{(p,p)}$ and the cross-spectra between channels $p$ and $q$ as $\mathbf{S}_j^{(p,q)}$.

A key strength of the mvLSW framework is that it provides a way to model time-varying dependence structure between channels. The concept of wavelet coherence was originally introduced in the bivariate setting in Sanderson et al. (2010), Park et al. (2014) extend this to the multivariate setting using the mvLSW framework. The wavelet coherence is defined to be the linear relationship between two channels, including any indirect links between them that depend on other channels of the signal. The wavelet coherence matrix, $\rho_j(z)$, at scale, $j$, and rescaled time, $z$, can be defined in terms of the local wavelet spectral matrix and has the form

$$\boldsymbol{\rho}_j(z) = \mathbf{D}_j(z)\mathbf{S}_j(z)\mathbf{D}_j(z), \tag{2.3.14}$$

where, $\mathbf{S}_j(z)$, is the local wavelet spectral matrix and $\mathbf{D}_j(z)$ is a diagonal matrix with elements $\mathbf{S}_j^{(p,p)}(z)^{(-1/2)}$. Individual elements of the coherence matrix have the form

$$\rho_j^{(p,q)}(z) = \frac{S_j^{(p,q)}(z)}{\sqrt{S_j^{(p,p)}(z)S_j^{(q,q)}(z)}}, \tag{2.3.15}$$

where $\rho_j^{(p,q)}(z)$ is the coherence between channel $p$ and $q$ for scale $j$, at rescaled time $z \in (0,1)$. The coherence takes a value between $-1$ and 1 where a value of $\pm 1$ shows a strong positive/negative linear relationship between channels at that scale and time. On the other hand, a value close to 0 indicates that there is very little linear dependence between channels.

Strong coherence between two channels may not necessarily mean that they are directly linked, this dependence could be driven through a third channel. To overcome

this, Park et al. (2014) introduce the concept of wavelet partial coherence which is a measure of the coherence between two channels after removing the effects of all other channels. The wavelet partial coherence matrix at scale, $j$, and rescaled time, $z$, is defined to be

$$\mathbf{\Gamma}_j(z) = -\mathbf{H}_j(z)\mathbf{G}_j(z)\mathbf{H}_j(z) \tag{2.3.16}$$

where $\mathbf{G}_j(z) = \mathbf{S}_j(z)^{-1}$ and $\mathbf{H}_j(z)$ is a diagonal matrix with entries $G_j^{(p,p)}(z)^{-1/2}$.

**Estimation of the LWS**

We can estimate the spectral properties of a multivariate signal by first calculating the empirical wavelet coefficients for each channel of the signal, and then using this to obtain the raw wavelet periodogram. More formally, the empirical wavelet coefficient vector, $\mathbf{d}_{j,k} = \left\{d_{j,k}^{(1)}, d_{j,k}^{(2)}, \ldots, d_{j,k}^{(P)}\right\}^\top$, is given by

$$\mathbf{d}_{j,k} = \sum_{t=0}^{T-1} \mathbf{X}_t \psi_{j,k}(t). \tag{2.3.17}$$

This can then be used to obtain the raw wavelet periodogram matrix, $\mathbf{I}_{j,k}$, which has the following form

$$\mathbf{I}_{j,k} = \mathbf{d}_{j,k}\mathbf{d}_{j,k}^\top. \tag{2.3.18}$$

As in the univariate case, Park et al. (2014) demonstrate that the raw wavelet periodogram matrix is an asymptotically biased and inconsistent estimator of the

LWS. In particular, they show that

$$\mathbb{E}[\mathbf{I}_{j,k}] = \sum_{l=1}^{J} A_{j,l}\mathbf{S}_l(k/T) + \mathcal{O}(T^{-1}), \tag{2.3.19}$$

$$\begin{aligned}\mathrm{var}(I_{j,k}^{(p,q)}) = &\sum_{l=1}^{J} A_{j,l}S_l^{(p,p)}(k/T)\sum_{l=1}^{J} A_{j,l}S_l^{(q,q)}(k/T) \\ &+ \left(\sum_{l=1}^{J} A_{j,l}S_l^{(p,q)}(k/T)\right)^2 + \mathcal{O}(2^{2j}/T).\end{aligned} \tag{2.3.20}$$

To account for this, the raw wavelet periodogram must be smoothed and corrected in some way. Park et al. (2014) propose smoothing the periodogram using a rectangular kernel smoother with window length $2M + 1$ to obtain the estimator

$$\tilde{\mathbf{I}}_{j,k} = \frac{1}{2M+1}\sum_{m=-M}^{M} \mathbf{I}_{j,k+m}. \tag{2.3.21}$$

The bias of the smoothed wavelet periodogram $\tilde{\mathbf{I}}$ can be corrected using the inverse of the inner product matrix in order to give a consistent and unbiased estimate of the LWS matrix $\hat{\mathbf{S}}$ with entries given by

$$\widehat{S}_{j,k} = \sum_{l=1}^{J} A_{jl}^{-1}\tilde{I}_{l,k}. \tag{2.3.22}$$

The wavelet coherence matrix can then easily be estimated by substituting $\hat{\mathbf{S}}$ into equation (2.3.14).

## 2.4 Classification of nonstationary time series

In recent years, the problem of classifying time series has been covered widely in existing literature. See Bagnall et al. (2017) for a comprehensive review. In this thesis, we restrict our attention to the classification of nonstationary time series. Below, we

briefly review some approaches to do this both in the univariate and multivariate contexts.

### 2.4.1 Static classification

We first consider static classification approaches in which an entire nonstationary signal is assigned to one particular class. In this situation, the class membership of the test signal is not permitted to vary over time. We will summarise a number of classification methods which are based on the nonstationary models described in Section 2.2.2 including the SLEX-based method of Huang et al. (2004) and the LSW-based approaches of Fryzlewicz and Ombao (2009) and Krzemieniewska et al. (2014).

The classification method introduced in Huang et al. (2004) uses the SLEX model to classify an unseen test signal using information from a set of (univariate) training signals of known class membership. The first step of the algorithm involves determining the best basis representation of each training signal from the SLEX library before estimating the SLEX periodograms based on these chosen representations. These periodograms are then averaged over the signals known to belong to each class to obtain estimates of the spectral information. An unseen time series is then assigned to a class, $\Pi_c$, if the Kullback-Liebler divergence between its estimated spectrum and the spectrum of $\Pi_c$ is smaller than that between the estimated spectrum and the spectra of any other class.

Whilst the approach of Huang et al. (2004) is computationally efficient and allows for accurate classification of (an entire) nonstationary time series, it suffers from the constraint of dyadic segmentation due to the use of the SLEX model. Fryzlewicz and

Ombao (2009) proposed an LSW-based alternative that overcomes this restriction. Again, the first step of this approach involves calculating the empirical wavelet spectrum, $L_{j,k} = \sum_i (A^{-1})_{i,j} I_{i,k}$, where $I_{i,k}$ is defined as in equation (2.3.6). The empirical wavelet spectra are then averaged over the training signals known to belong to class $c$ in order to estimate the evolutionary wavelet spectrum, denoted $\hat{S}_j^{(c)}(z)$. The discriminating set, $\mathcal{M}$, is then determined by choosing a specified proportion of the timescale indices $(j, k)$ that maximise the divergence index

$$\Delta(j,k) = \sum_{g=1}^{G} \left[ \hat{S}_j^{(g)}(k/T) - \frac{1}{G} \sum_{h=1}^{G} \hat{S}_j^{(h)}(k/T) \right]^2 . \tag{2.4.1}$$

In order to classify a time series, Fryzlewicz and Ombao (2009) first estimate its empirical wavelet spectrum, $L_{j,k}$, and then calculate the following squared distances for each class $g$

$$D_g = \sum_{(j,k)\in\mathcal{M}} (L_{j,k} - \hat{S}_j^{(g)}(k/T))^2. \tag{2.4.2}$$

The time series is then assigned to the class $g$ that minimises the above distance measure.

Whilst this method works well when each of the class generators produce spectra with equal variability across realizations, when this is not the case choosing the most divergent coefficients may not be the best approach for classification. Krzemieniewska et al. (2014) proposed an extension to the work of Fryzlewicz and Ombao (2009) that accounts for within-class variation between signals. As in Fryzlewicz and Ombao (2009), the empirical wavelet spectra for each class is estimated by averaging the periodograms for the time series replications known to belong to that class. Krzemieniewska et al. (2014) then estimate the variability at different scales and locations $\sigma_{j,k}^2$

in the following way

$$\sigma^2_{j,k} = \frac{1}{\sum_{g=1}^{c} N_g}\left(\sum_{g=1}^{c}\sum_{n=1}^{N_g}(L^{g,n}_{j,k} - \hat{S}^{(g)}_j(k/T))^2\right). \tag{2.4.3}$$

The subset of discriminating coefficients, $\mathcal{M}$, is chosen by selecting pairs $(j,k)$ that maximise the alternative divergence index

$$\tilde{\Delta}(j,k) = \Delta(j,k)/\sigma^2_{j,k}. \tag{2.4.4}$$

Here, $\Delta(j,k)$ is defined in equation (2.4.1). The modified distance for an observed signal is given by

$$\tilde{D}_g = \sum_{(j,k)\in\mathcal{M}} \frac{\left(L_{j,k} - \hat{S}^{(g)}_j(k/T)\right)^2}{\sigma^2_{j,k}}. \tag{2.4.5}$$

The signal is then assigned to the class $g$ for which $\tilde{D}_g$ is smallest.

### 2.4.2 Dynamic classification

In practice, classifying a signal entirely into one class is not appropriate for many real systems where the nonstationarity of the series may be due to class switching. In this case, methods that allow the class membership of the signal to change over time are preferable. Park et al. (2018) introduce a dynamic classification approach that is based on the mvLSW model, using information on the within and cross-channel dependence to classify a multivariate, nonstationary signal. Given a set of multivariate training signals of known class membership, denoted $\{Y^{(i)}_t\}$ for $i \in \{1, 2, \ldots, N_i\}$, the LWS matrix for each one can be estimated using the raw wavelet periodogram as in equation (2.3.22). The wavelet coherence matrix for each training signal, $\rho_{j,k;Y^{(t)}}$, can be estimated by substituting the LWS matrix into equation (2.3.14). Park et al.

(2018) apply a Fisher z transform to the coherence estimates in order to ensure that they can be approximated by a Gaussian distribution. For a class $c$, the transformed coherence $\xi_j^{(c)}$ is given by

$$\xi_j^{(c)} = \tanh^{-1}\rho_j^{(c)}. \tag{2.4.6}$$

The mean and variance of the transformed coherence for class $c$ can be estimated using the transformed coherence for the training signals that are known to belong to that particular class. In a similar way to the static classification method of Krzemieniewska et al. (2014), these mean and variance estimates are used to determine the subset of coefficients that show the largest difference between the classes in terms of the transformed coherence. In the multivariate setting, the subset $\mathcal{M}$ consists of the scale and channel indices, $(j, p, q)$, for $p < q$ that maximise the discrepancy measure $\Delta_j^{(p,q)}$ which is given by

$$\Delta_j^{(p,q)} = \sum_{c=1}^{N_c} \sum_{g=c+1}^{N_c} \left| \frac{\xi_j^{(p,q)(c)} - \xi_j^{(p,q)(g)}}{\sqrt{\text{var}(\xi_j^{(p,q)(c)}) + \text{var}(\xi_j^{(p,q)(g)})}} \right|. \tag{2.4.7}$$

As the class membership of the observed signal is allowed to change over time, this approach estimates the probability that the signal belongs to a particular class at a given time. Park et al. (2018) first estimate the transformed coherence for the signal $X_t$, denoted $\hat{\xi}_{j,k;X}$. Bayes' theorem is then used to estimate the probability that the signal belongs to class $c$ at a particular time given prior information in the following way

$$\Pr\left[C(k) = c|\hat{\xi}_{j,k;X}\right] \propto \Pr\left[C(k) = c\right] \mathcal{L}(\hat{\xi}_{j,k;X}|\xi_j(k/T) = \xi_j^{(c)} \quad \forall\ j) \tag{2.4.8}$$

where $\mathcal{L}(\theta|x)$ is the likelihood, $C(k)$ represents the class assignment of the signal at rescaled time $k$ and $\Pr\left[C(k) = c\right]$ is the prior probability that the signal is in class $c$

at rescaled time $k$. This is typically assigned the probability of $\frac{1}{N_c}$ provided that no initial information is known.

# Chapter 3

# Dynamic detection of anomalous regions within distributed acoustic sensing data streams using locally stationary wavelet time series

## 3.1 Introduction

The ability to accurately analyse geoscience data at, or close to, real time is becoming increasingly important. For example, within the oil and gas sector this need can arise as a consequence of (i) the sheer volume of data now being collected and (ii) operational considerations. It is this setting that we consider in this article, seeking to enable the rapid identification of certain anomalous features within Distributed Acoustic Sensing data obtained from an oil producing facility. Specifically, we seek to

build on recent work within the nonstationary time series community to develop an approach that permits the online monitoring of these complex signals.

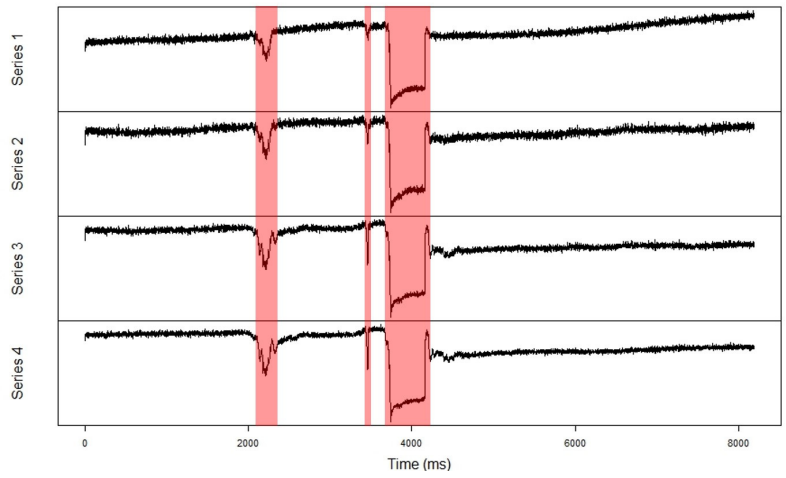
The technology used to generate the data considered in this article, Distributed Acoustic Sensing (DAS), involves the use of a fibre-optic cable as a sensor in which the entire length of the fibre is used to measure acoustic or thermal disturbances. DAS originates from the defence industry where it is commonly used in security and border monitoring (Owen et al., 2012). Recently, the technology has been applied within the oil and gas industry, for example in pipeline monitoring and management (Williams, 2012; Mateeva et al., 2014). The use of DAS to monitor production volumes and composition within a well requires the installation of a fibre-optic cable along the length of the well combined with an interrogator unit on the surface (Paleja et al., 2015). This unit sends light pulses down the cable and processes the back-scattered light. The installation of such technology has become popular as it is can be a cost effective way to obtain continuous, real-time and high-resolution information.

When monitoring the behaviour of wells it is important to be able to detect unusual occurrences, including potential corruptions of the data. Striping is one particular form of corruption that can have a particularly deleterious effect, rendering data potentially unusable in a specific time region. Stripes are characterised by sudden, and distinctive changes in the structure of the signal over time, see Mateeva et al. (2014) and Ellmauthaler et al. (2017) for examples. These features can be present simultaneously across all channels or only apparent across a subset of channels, for example from the surface to a set depth within the well. Crucially, the occurrence of stripes simultaneously at different locations indicates that these features are not

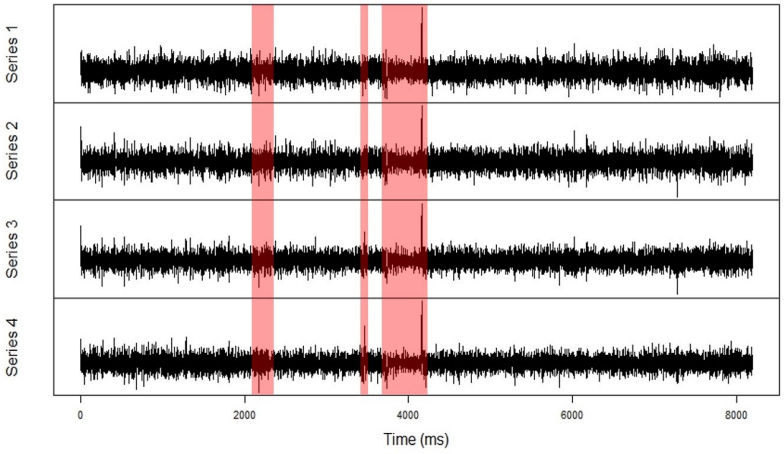
physical. Instead stripes can occur for a number of reasons, including a disturbance of the fibre-optic cable near the unit, or problems with the electronics due to the high sampling rate.

Visually, stripes can manifest themselves in a variety ways. Some are visually obvious within the DAS data, such as the stripe that occurs at around 4000ms in Figure 3.1.1(a). Other occurrences can be more subtle, and therefore more challenging to detect. For example, the stripe could be a change in the second-order structure. Critically such features can make it difficult to carry out further analysis of the data, such as flow rate analysis. For this reason, there is significant interest in being able to detect regions of striping as soon as they occur, so that they can be removed whilst keeping as much of the original signal intact as possible. It is this challenge of dynamically detecting striping regions that motivates the work presented in this article.

There exist a variety of techniques for the classification of time series in the statistical and machine learning literature. An exhaustive review is beyond the scope of this article, but popular classification methods include hidden Markov models (HMM) (see e.g. Rabiner (1989); Ephraim and Merhav (2002); Cappé et al. (2009)); support vector machines (Cortes and Vapnik, 1995; Muller et al., 2001; Kampouraki et al., 2009); Gaussian mixture models (McLachlan and Peel, 2004; Povinelli et al., 2004; Kersten, 2014); nearest neighbour classifiers (Zhang et al., 2004; Wei and Keogh, 2006) and multiscale methods (Chan and Fu, 1999; Mörchen, 2003; Aykroyd et al., 2016) to name but a few. More recent contributions for large-scale (online) classification include the MOA machine learning framework (Bifet et al., 2010; Read et al., 2012). For

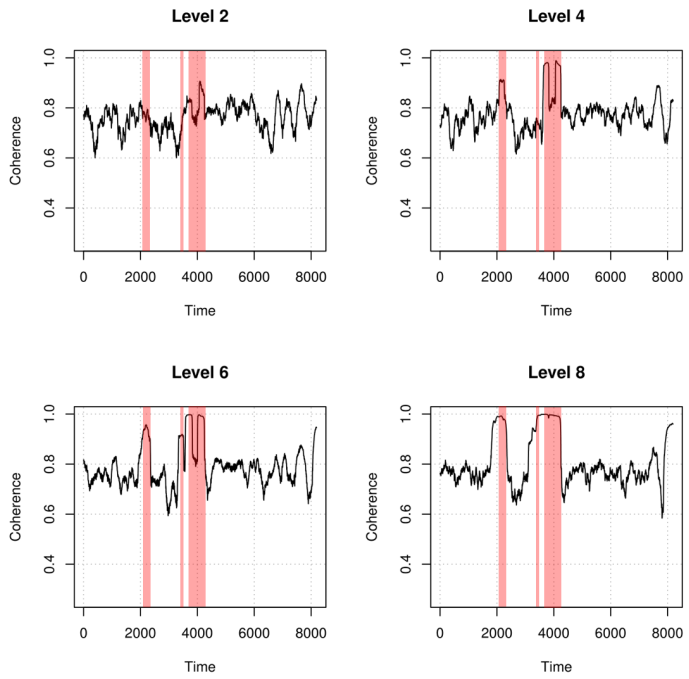



(a)

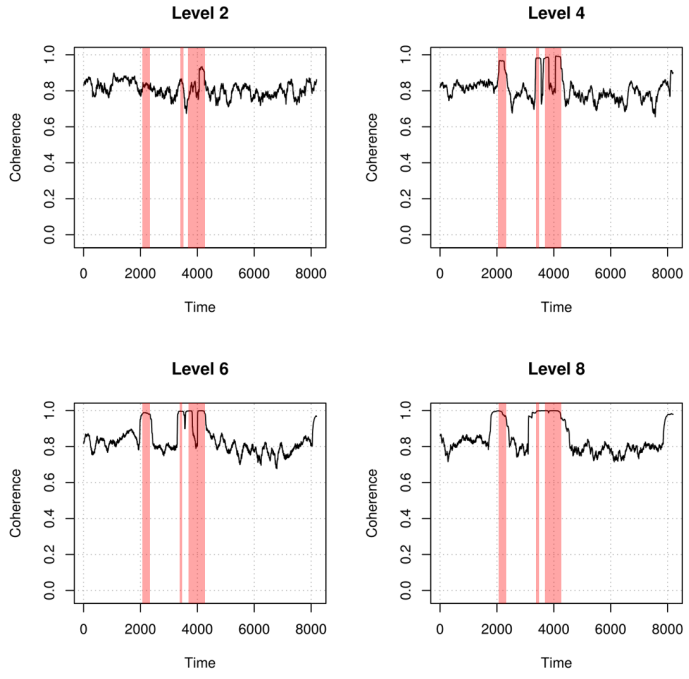



(b)

Figure 3.1.1: Time series plots of DAS amplitude at four different well depths over the same time period: (a) original series; (b) detrended series. The highlighted regions in (a) indicate three examples of striping.




(a)




(b)

Figure 3.1.2: Hidden time-varying coherence structure of the DAS series in Figure 3.1.1 at selected wavelet scales (resolutions): (a) coherence between series 1 and 2; (b): coherence between series 3 and 4.

a recent overview of classification in the time series context, see for example Fu (2011). Dependent on the application being considered, one might adopt various modelling choices. For example, some classifiers have distinct advantages, such as simplicity of implementation, speed or suitability for massive online applications. However many, such as GMM or SVM-based approaches, do not explicitly allow temporal dependence or are limited to a narrow class of series structure (HMMs), which is seen as crucial to classification of time series in the majority of realistic settings (see e.g. Bifet et al. (2013)). Complex hidden dependence structure is typical of the DAS data studied in this article (see Figure 3.1.2).

Our approach to the dynamic stripe identification problem builds on recent work within the time series literature. Wavelet approaches to modelling time series have become very popular in recent years, principally because of their ability to provide time-localised measures of the spectral content inherent within many contemporary data (e.g. Killick et al. (2013); Nam et al. (2015); Chau and von Sachs (2016); Nason et al. (2017)). This *locally stationary* modelling paradigm is flexible enough to represent a wide range of nonstationary behaviour and has also been extended to enable the modelling and estimation of multivariate nonstationary time series structures (e.g. Sanderson et al. (2010) and Park et al. (2014)). Typically these settings assume that the data have already been collected, and are available for offline analyses.

The novel contribution in this article is to employ the mvLSW modelling framework of Park et al. (2014) to represent the DAS data, using a moving window approach, thereby extending previous work to the online dynamic classification setting. This modelling framework allows us to classify multivariate time series with complex

dependencies both *within* and *between* channels of the series, including those which exhibit visually subtle changes in behaviour over time. Reusing data calculations allows us to also produce a computationally efficient nondecimated wavelet transform in the online setting.

This work is organised as follows. In Section 3.2, we describe the proposed online classification method. Section 3.3 contains a simulation study evaluating the performance of the proposed classifier using synthetic data, further justifying the use of time-varying coherence as a feature for classification. A case study using an acoustic sensing dataset is then described in Section 3.4, where we discuss the utility of the proposed classifier as a stripe detection method. Finally, Section 3.5 includes some concluding remarks.

## 3.2 Online dynamic classification of multivariate series

In order to adapt the existing dynamic classification method outlined in Section 2.4.2 to an online setting, we make use of a moving window approach. The use of such a window encapsulates the constraint in many data streaming applications that there is only limited data storage and memory with which to perform analysis.

Our online dynamic classification technique proceeds as follows. For a window of length $w = 2^J$ the first step of our algorithm is to calculate the set of discriminative indices as defined in equation (2.4.7) using a set of training signals of length $w$. For reasons of efficiency, the discriminative indices are used in the classification step

for each window of the data. Although window-specific indices could be used, in our experience, updating the set of discriminative indices for each window increases computational complexity without providing significant accuracy improvement. The dynamic classification method described in Section 2.4.2 is applied to the first window of data to obtain the probability that the signal belongs to a particular class for the time points in the window.

Upon arrival of a new data point, the window then shifts by one, and the data under analysis consists of the old data together with the new data point, but we also lose the first data point contained in the previous window. The online wavelet transform is then used to efficiently update the wavelet coefficients and the transformed coherence estimate for the new window. Using the information previously calculated from the training signals, we can then obtain the probability that the signal belongs to a particular class for the time points contained in the new window. The algorithm continues by repeatedly moving the window for each new data point and estimating the probability of each data point belonging to a class until we reach the end of the data stream.

During our classification algorithm, we obtain multiple estimates for the probability that a signal belongs to a particular class (at each time point) from the different windows into which a data point falls. For example, for a time series of length $T$ analysed with a moving window of length $w < T$, we obtain $w$ estimates for the probabilities of an individual time point $t$ belonging to a given class $c$, which we denote $p_{t,i}^{(c)}$ for window $i$.

A question that arises as a result of the iterative approach is how to combine

the estimates from different windows to obtain an overall probability that the time point belongs to a particular class, and hence classify the signal. In what follows, for computational simplicity we use a simple average, but other more sophisticated combination methods could be used. In other words, our final probability estimates are given by

$$p_t^{(c)} = \frac{1}{w}\sum_{i=1}^{w} p_{t,i}^{(c)} \qquad \text{for } t = 1, 2, \ldots, T. \tag{3.2.1}$$

In some applications, an overall classification of the signal is required rather than probability estimates. In this case, the class $c$ that has the largest probability $p_t^{(c)}$ is assigned to the time point $t$ for all $t \in \{1, 2, \ldots, T\}$.

A summary of our method for estimating the probability that a given multivariate signal belongs to a particular class $c$ at a particular time is given in Algorithm 1.

### 3.2.1 Edge Effects

One of the main issues related to the online dynamic classification is the edge effects generated by the windowing procedure. These can result in sudden peaks in the probability of the signal belonging to a particular class at the edges of the window. For example, we consider a time-varying vector autoregressive moving average process with two different classes defined by the following coefficient matrices

$$\text{Class 1}: \ \mathbf{X}_t = \begin{pmatrix} -0.3 & -0.2 & 0.3 \\ -0.2 & -0.3 & 0.1 \\ 0.3 & 0.1 & 0.2 \end{pmatrix} \mathbf{X}_{t-1} + \mathbf{Z}_t + \begin{pmatrix} 1 & -0.6 & 0.3 \\ -0.6 & 1 & -0.3 \\ 0.3 & -0.3 & 1 \end{pmatrix} \mathbf{Z}_{t-1}$$

$$\text{Class 2}: \ \mathbf{X}_t = \begin{pmatrix} 0.4 & 0.1 & -0.2 \\ 0.1 & 0.3 & -0.3 \\ -0.2 & -0.3 & -0.2 \end{pmatrix} \mathbf{X}_{t-1} + \mathbf{Z}_t + \begin{pmatrix} 1 & 0.8 & 0.4 \\ 0.8 & 1 & 0.1 \\ 0.4 & 0.1 & 1 \end{pmatrix} \mathbf{Z}_{t-1}$$

**Algorithm 1** Online dynamic classification: Finding the average probability that a multivariate signal belongs to a particular class over time.

1: Let $X$ be a $P$-variate signal of length $T$ that we wish to classify using a moving window of length $w$.

2: Calculate the set of discriminative indices using a set of P-variate training signals of length $w$, whose class assignments are known.

3: Apply dynamic classification method to the first window of data $X[\ , 1 : w]$ to obtain the probability that the signal belongs to a particular class $c$ for the time points in the window, denoted $p_{t,1}^{(c)}$ for $t = 1, 2, \ldots, w$.

4: Iterate for $i$ in 2 to $T - w + 1$

(a) Apply the online wavelet transform to the new window of data $X[\ , i : i+w-1]$ to update the wavelet coefficients.

(b) Update the transformed coherence at the set of discriminative indices using the wavelet coefficients calculated in the previous step.

(c) Apply dynamic classification method to obtain the probability that the signal belongs to a particular class for window $i$, denoted $p_{t,i}^{(c)}$ for $t = i, i+1, \ldots, i+w-1$.

5: Average probability estimates for each window using (4) to obtain the final probability that the signal $X$ belongs to a particular class $c$ over time, $p_t^{(c)}$ for $t = 1, 2, \ldots, T$.

where $\mathbf{Z}_t$ and $\mathbf{Z}_{t-1}$ are zero-mean multivariate normal realisations, distributed with class-dependent covariances

$$\Sigma_1 = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{pmatrix}, \qquad \Sigma_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

We simulate a time series of length 600 from this process, where the signal is in class 1 for the first 300 timepoints and then switches to class 2 for the remainder of the series. An example realisation from this process can be found in Figure 3.2.1.

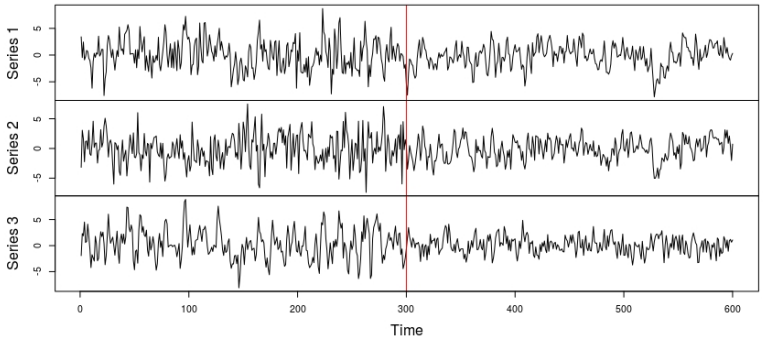
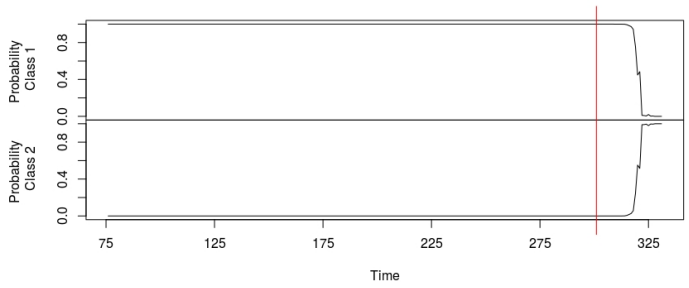
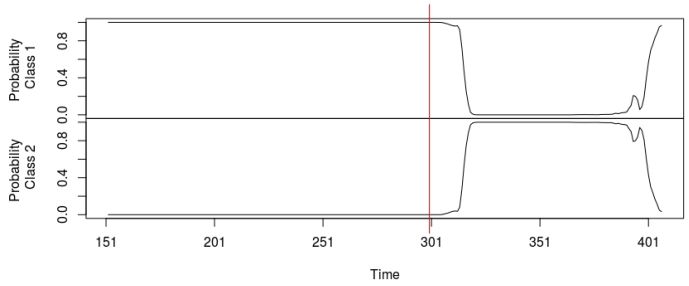



Figure 3.2.1: Realisation of a two class time-varying vector autoregressive moving average process, where a change in class takes place at time 300.

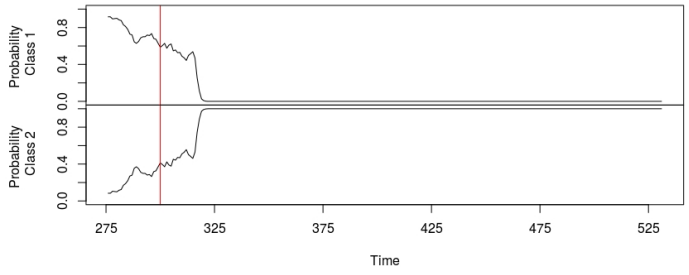
After applying the online dynamic classification method to this process with a window length of 256, we generate the probabilities that the signal belongs to each class for the individual windows. Figure 3.2.2 shows some examples of the features and edge effects that can arise as part of the classification process. One thing that can be seen that from Figure 3.2.2 is that, whilst the class change within the signal takes place at time 300, it can take around 15-20 timepoints for the probabilities to change.




(a) Probability of belonging to each class for window 75.



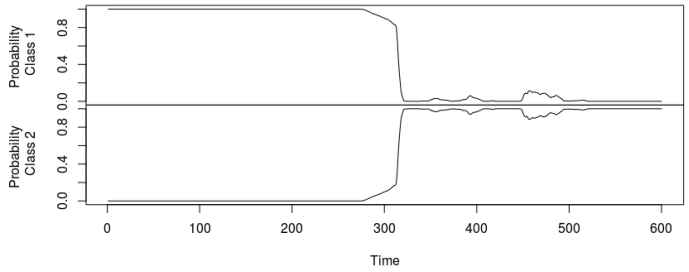

(b) Probability of belonging to each class for window 151.



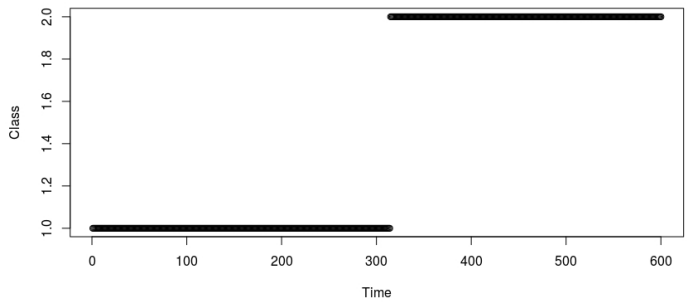

(c) Probability of belonging to each class for window 275.

Figure 3.2.2: Probability of belonging to each class for various windows of data, the true class change at time 300 is marked in red.

This can result in some small areas of misclassification around a class change, as there is a small offset of time before the online dynamic classification method reports a change in class. In addition, an edge effect can be seen in Figure 3.2.2(b), the probabilities of belonging to each class contain a sudden unexpected change towards the end of the window. These edge effects are present within a small proportion of windows across the analysis but this has a negligible effect on the average probabilities that the signal is in a particular class over time and the corresponding class memberships, see Figure 3.2.3.




(a) Average probability that the signal belongs to each class over time.




(b) Class membership of the signal over time.

Figure 3.2.3: Average probabilities and class membership of the series over time.

However in situations where these edge effects are more prevalent, averaging the prob-

ability estimates at each time point using equation (3.2.1) may not be the best option. Alternative methods for obtaining an overall probability estimate could include taking the median of the quantities in equation (3.2.1) or using a weighted average with more weight given to the estimates arising from the centre of a window rather than the edges.

Currently a time point within a series can only be classified after the probability estimates have been obtained from each window that covers that particular point. Often the most reliable probability estimates are generated from the centre of the sliding window which means that potentially a point could be classified using $\frac{w}{2}$ probablity estimates rather than the $w$ needed currently. An avenue for future research would be to investigate this further and determine if points within a time series could be accurately classified before the sliding window ends.

## 3.3 Synthetic Data Examples

We now turn to assess the performance of our proposed online dynamic classification approach. To this end, a simulation study is designed to test the ability of this wavelet-based appproach to classify data streams exhibiting various characteristics. More specifically, the study consists of three different scenarios. These scenarios are chosen to mimic signals arising in practice:

Scenario 1: Signal of length 1024, short time segments of length 100 between changes in class, nine class changes in total.

Scenario 2: Signal of length 1024, alternating long/short segments of length 300 and

100 between changes, five class changes in total.

Scenario 3: Signal of length 2048, long segments of length 300 between changes, six class changes in total.

For all scenarios, the generated series randomly switch classes between time segments. A window length of 256 is used when implementing the online dynamic classification method and the training data consists of 10 signals, some of which contain changes in class. The R packages **wavethresh** (Nason, 2016) and **mvLSW** (Taylor et al., 2017) are used to calculate the wavelet coefficients and transformed coherence that are used in the online dynamic classification.

Long segments of length 300 between class changes are chosen to ensure that there is a maximum of one class change in each dynamic classification window. In the situation where the class changes are reasonably far apart, we expect the online dynamic classification algorithm to classify the signal well. As a contrast, short segments of length 100 are also chosen to demonstrate some potential limitations of the method. In particular, when the signal contains multiple class changes that are close together, there is a possibility that our approach will misclassify the signals.

For each scenario, we consider a number of examples of generating processes for the classes in the multivariate series. The first example we examine consists of three classes where each class is defined by a trivariate normal signal with mean $\mu = (0, 0, 0)$ and differing cross-channel dependence structure. More specifically, the classes are

defined by the three covariance matrices

$$\Sigma^{(1)} = \begin{pmatrix} 1 & 0 & 0.3 \\ 0 & 1 & 0.7 \\ 0.3 & 0.7 & 1 \end{pmatrix}, \Sigma^{(2)} = \begin{pmatrix} 1 & 0.6 & 0.1 \\ 0.6 & 1 & -0.4 \\ 0.1 & -0.4 & 1 \end{pmatrix} \text{ and } \Sigma^{(3)} = \begin{pmatrix} 1 & -0.5 & -0.2 \\ -0.5 & 1 & 0.1 \\ -0.2 & 0.1 & 1 \end{pmatrix}.$$

Example simulated data for this process using the different class switching scenarios above are shown in Figure 4.4.1(a).

To investigate the potential of our proposed approach further, we studied an example with a time-varying moving average (VMA) process, with three classes defined by the following coefficient matrices:

$$\text{Class 1:} \quad \mathbf{X}_t = \mathbf{Z}_t + \begin{pmatrix} 1 & 0 & 0.6 \\ 0 & 1 & 0.3 \\ 0.6 & 0.3 & 1 \end{pmatrix} \mathbf{Z}_{t-1} + \begin{pmatrix} 1 & 0.2 & 0.9 \\ 0.2 & 1 & 0.5 \\ 0.9 & 0.5 & 1 \end{pmatrix} \mathbf{Z}_{t-2}$$

$$\text{Class 2:} \quad \mathbf{X}_t = \mathbf{Z}_t + \begin{pmatrix} 1 & -0.7 & -0.3 \\ -0.7 & 1 & 0.4 \\ -0.3 & 0.4 & 1 \end{pmatrix} \mathbf{Z}_{t-1} + \begin{pmatrix} 1 & 0.9 & -0.3 \\ 0.9 & 1 & 0 \\ -0.3 & 0 & 1 \end{pmatrix} \mathbf{Z}_{t-2},$$

$$\text{Class 3:} \quad \mathbf{X}_t = \mathbf{Z}_t + \begin{pmatrix} 1 & -0.4 & 0.2 \\ -0.4 & 1 & -0.6 \\ 0.2 & -0.6 & 1 \end{pmatrix} \mathbf{Z}_{t-1} + \begin{pmatrix} 1 & 0.1 & -0.5 \\ 0.1 & 1 & -0.3 \\ -0.5 & -0.3 & 1 \end{pmatrix} \mathbf{Z}_{t-2},$$

where $\mathbf{Z}_t$, $\mathbf{Z}_{t-1}$ and $\mathbf{Z}_{t-2}$ are IID multivariate Gaussian white noise (see Figure 4.4.1(b)).

The third example we consider is a vector autoregressive process with intra- and cross-channel changes in dependence between each class (see Figure 4.4.1(c)). The

three classes in the example are defined by

$$\text{Class 1:}\quad \mathbf{X}_t = \begin{pmatrix} 0.2 & 0.3 & 0 \\ 0.3 & 0.5 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{X}_{t-1} + \begin{pmatrix} 0.6 & -0.1 & 0 \\ -0.1 & -0.3 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{X}_{t-2} + \epsilon_1,$$

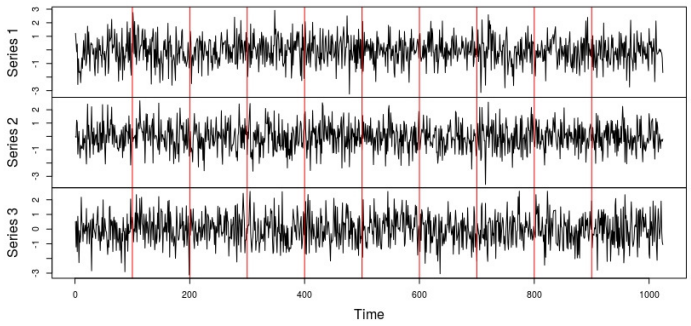
$$\text{Class 2:}\quad \mathbf{X}_t = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0.4 & -0.4 \\ 0 & -0.4 & 0.4 \end{pmatrix} \mathbf{X}_{t-1} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & -0.6 & 0.2 \\ 0 & 0.2 & 0.3 \end{pmatrix} \mathbf{X}_{t-2} + \epsilon_2,$$

$$\text{Class 3:}\quad \mathbf{X}_t = \begin{pmatrix} -0.1 & 0 & 0.4 \\ 0 & 0 & 0 \\ 0.4 & 0 & -0.5 \end{pmatrix} \mathbf{X}_{t-1} + \begin{pmatrix} 0.2 & 0 & -0.2 \\ 0 & 0 & 0 \\ -0.2 & 0 & -0.3 \end{pmatrix} \mathbf{X}_{t-2} + \epsilon_3,$$

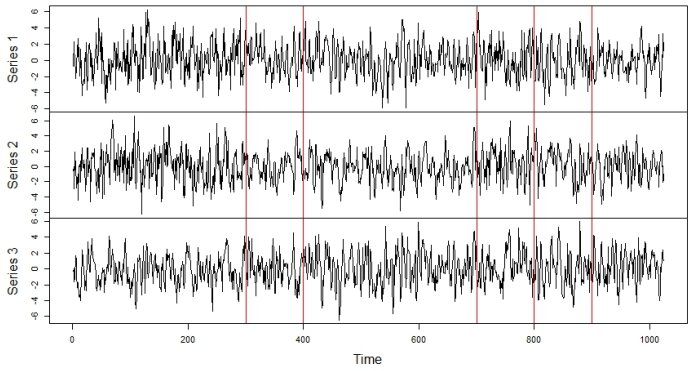
where the noise vectors $\epsilon_i$ are zero-mean multivariate normal realisations, distributed with covariances

$$\Sigma_{\epsilon_1} = \begin{pmatrix} 3 & 0.3 & 0.9 \\ 0.3 & 3 & 1.4 \\ 0.9 & 1.4 & 3 \end{pmatrix}, \; \Sigma_{\epsilon_2} = \begin{pmatrix} 2 & 1.3 & 0.4 \\ 1.3 & 1.8 & 0.3 \\ 0.4 & 0.3 & 2 \end{pmatrix}, \; \Sigma_{\epsilon_3} = \begin{pmatrix} 5 & 3.3 & 2.5 \\ 3.3 & 4.5 & 2.8 \\ 2.5 & 2.8 & 3.5 \end{pmatrix}.$$

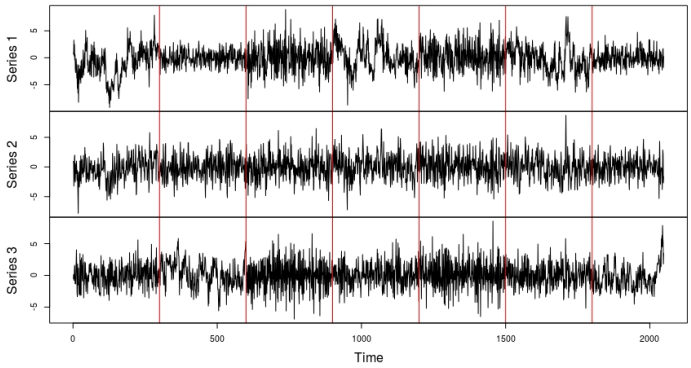
**Competitor methods.** In the simulation study, we compare our proposed method with a number of alternative classification techniques. Firstly we consider a Hidden Markov Model (HMM) approach – a probabalistic model of the joint distribution of observed variables, together with their "hidden" states (in this setting, classes). Such methods have previously been used for classification in the literature, see for example




(a) Scenario 1, multivariate Gaussian series




(b) Scenario 2, vector moving average series




(c) Scenario 3, vector autoregressive series

Figure 3.3.1: Example realisations of generating processes for the different scenarios used in the simulation study. (a) Short segments of length 100 between class changes; (b) Alternating long/short segments of length 300 and 100 between changes; (c) Long segments of length 300 between changes.

Ainsleigh et al. (2002). In this model, it is assumed that (i) the observed data at a particular time is independent of all other variables, given its class and (ii) given the previous class, the class at a time is independent of all other variables (i.e. the changes in class are Markovian). This means that we assume that the probability of changing class does not depend on time or previous class membership, which can be an unrealistic assumption to make in practice. Furthermore, HMMs can be computationally intensive to implement especially in multiclass settings, requiring procedures such as the EM algorithm for tractable model fitting, see e.g. Cappé et al. (2009). An introduction to HMMs and their applications can be found in Zucchini and MacDonald (2009).

A sequential HMM approach is applied to both the full test signal and its transformed coherence at the set of discriminative indices, using the *R* package **HMM** (Himmelmann, 2010). In both cases, the model is initialized to have equal state probabilities, and then trained using the initial data. When a new data point arrives, the probabilities of belonging to each state are computed. This process of increasing the number of data points and computing the probabilities is repeated until we reach the end of the signal. As with the online dynamic classification approach, multiple estimates for the probability of belonging to a state at a particular time point are obtained. This is because each time a data point falls within a window, probabilities associated to the time point belonging to a particular class are calculated. For each time point, the estimates are averaged and the overall classification of the signal is then defined to be the most likely state at each point. We also considered a third variant of the sequential HMM approach that was applied to each window of the data

used in the online dynamic classification and the corresponding transformed coherence. However this produced poor results so we omit them from the comparisons below.

To demonstrate the importance of accounting for the dependence structure within the test series, we also apply a support vector machine (SVM) classifier to the series, available in the *R* package **e1071** (Meyer et al., 2015), as well as the mixture modelling approach from the **mclust** *R* package (Fraley et al., 2017) (denoted *GMM*). These methods do not explicitly allow temporal dependence in the classification rules, and so we would expect them to perform poorly in cases where this dependence features in the test series. Specifically, we used a radial basis kernel for the SVM classifier. The GMM approach implemented allows for potentially different numbers of mixture components and covariance structures for each class, with the number of components chosen with the Bayesian information criterion (BIC). Similar to the HMM method described above, we show results on the SVM and GMM methods applied to the transformed coherence measure – the results for the techniques on the raw series performed poorly and so they aren't reported in the tables. In addition, we compare our method to the Naïve Bayes (NB) classifier in the **RMOA** (Wijfells, 2014) suite of online methods (again using the transformed coherence). This latter technique uses a Bayesian classification rule similar to that in (2.4.8), and hence provides a useful comparison to our proposed use of time-varying wavelet coherence in a Bayesian rule. We also investigated the performance of several of the ensemble classification techniques implemented in the **RMOA** package, however their performance was similar to the NB classifier so we omit these results for brevity.

**Training procedure details.** The training data for both the online dynamic classification and the sequential HMM approaches consists of ten signals of length 256. Of the ten signals, we simulate two each from Class 1, 2 and 3 and the remaining four signals contain a mixture of all three. For the competitor methods that are applied to the transformed coherence measure, the training data has a slightly different form. In this case, the training signals are simulated with class memberships as defined above but the approaches are trained on the transformed coherences of these signals at the set of discriminative indices rather than the raw data. For the different scenarios and generating processes considered, in practice we find that the subset of most discriminative indices tends to consist of the finest scales, i.e. scales $1-3$, but that all channel indices appear to be important.

For each of the scenarios, 100 replications of the test signals are simulated and three different classification evaluation measures are considered. In particular, the number of class changes detected is recorded along with the V-measure (Rosenberg and Hirschberg, 2007) and the true positive detection rate, defined to be the proportion of each signal that is correctly classified. A change is detected if the signal switches class and this change lasts for longer than four time points.

The V-measure assesses the quality of a certain segmentation (given the truth) and is measured on the $[0, 1]$ scale where a value of 1 represents perfect segmentation. Following Rosenberg and Hirschberg (2007), assume $N$ is the number of datapoints within a data set, $C = \{c_i, i = 1, \ldots, n\}$ is a set of classes and $K = \{k_i, 1, \ldots, m\}$ is a set of clusters. Let $A = \{a_{ij}\}$ where $a_{ij}$ is the number of datapoints that are members

of class $c_i$ and elements of cluster $k_j$. The homogeneity $h$ is then defined to be

$$h = \begin{cases} 1 & \text{if } H(C,K) = 0, \\ 1 - \frac{H(C|K)}{H(C)} & \text{otherwise} \end{cases} \tag{3.3.1}$$

where

$$H(C|K) = -\sum_{k=1}^{|K|}\sum_{c=1}^{|C|}\frac{a_{ck}}{N}\log\frac{a_{ck}}{\sum_{c=1}^{|C|}a_{ck}},$$
$$H(C) = -\sum_{c=1}^{|C|}\frac{\sum_{k=1}^{|K|}a_{ck}}{n}\log\frac{\sum_{k=1}^{|K|}a_{ck}}{n}.$$

Similarly, the completeness $c$ is defined as

$$c = \begin{cases} 1 & \text{if } H(K,C) = 0, \\ 1 - \frac{H(K|C)}{H(K)} & \text{otherwise} \end{cases} \tag{3.3.2}$$

where

$$H(K|C) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{a_{ck}}{N}\log\frac{a_{ck}}{\sum_{k=1}^{|K|}a_{ck}},$$
$$H(K) = -\sum_{k=1}^{|K|}\frac{\sum_{c=1}^{|C|}a_{ck}}{n}\log\frac{\sum_{c=1}^{|C|}a_{ck}}{n}.$$

The V-measure is then calculated as the harmonic mean of homogeneity and completeness

$$V = \frac{(1+\beta)hc}{\beta h + c} \tag{3.3.3}$$

where $\beta$ is a user-set parameter that determines whether completeness or homogeneity is weighted more strongly in the calculation. Throughout the simulation study, we set the parameter $\beta$ to 1.

The classification results for the different examples described above can be seen in Tables 3.3.1 – 3.3.3. Sequential HMM denotes the results for the full test signal and

Embedded HMM denotes the results for the transformed coherence; similar descriptors are used for the SVM, GMM and NB classifiers applied to the transformed coherence of the raw data. We remind the reader that these classification methods performed very poorly on the original series, and so are not reported in the tables. In each case, we have recorded the average number of changes detected, V-measure and true positive rate (described above) over the 100 replications; the numbers within the brackets represent the standard deviation of the corresponding quantities. Recall that the number of true class changes for Scenarios 1, 2 and 3 are nine, five and six respectively.

For the three class multivariate normal example (Table 3.3.1), it can be seen that all three methods overestimate the number of changes detected. The online dynamic classification performs the best in terms of the average number of changes detected, only marginally overestimating the number of changes, and is competitive with other methods in terms of V-measure and average true positive rate. Both the sequential HMM approach and the Naïve Bayes classification rule perform well in this setting according to the V-measure and the average true positive rate. However, we note here that the improvement over our proposed method is minimal considering the variability in the estimates.

As we introduce dependence into the series, the distinction between our proposed method and its competitors becomes more marked. For the moving average process (Table 3.3.2), the performance of the online dynamic classification method improves as we increase the length of the segments between class changes; on the other hand, the sequential HMM procedure (as with the other competitors applied to the original

Table 3.3.1: Performance of classification procedures over 100 replications of multivariate Gaussian series for different scenarios of class changes, using the evaluation measures described in the text. Numbers in brackets represent the standard deviation of estimation errors. Bold numbers indicate best result.

| | Scenario 1 (nine changes) | Scenario 2 (five changes) | Scenario 3 (six changes) |
|---|---|---|---|
| Method | Average number of changes detected | | |
| Online dynamic classification ($w = 256$) | **9.38 (0.65)** | **5.58 (0.88)** | 6.19 (0.44) |
| Sequential HMM | 10.44 (3.80) | 6.71 (5.65) | 10.96 (13.78) |
| Embedded HMM | 11.90 (3.29) | 8.49 (3.53) | 15.29 (3.74) |
| Embedded SVM | 13.55 (2.34) | 8.50 (2.02) | 8.02 (1.56) |
| Embedded GMM | 17.88 (3.58) | 15.45 (3.10) | 29.16 (5.34) |
| Embedded NB | 12.71 (2.27) | 7.43 (1.50) | **6.16 (0.42)** |
| Method | Average V-measure | | |
| Online dynamic classification ($w = 256$) | 0.89 (0.02) | 0.89 (0.03) | 0.94 (0.01) |
| Sequential HMM | **0.94 (0.05)** | **0.93 (0.09)** | 0.94 (0.09) |
| Embedded HMM | 0.78 (0.05) | 0.74 (0.10) | 0.80 (0.04) |
| Embedded SVM | 0.80 (0.03) | 0.82 (0.04) | 0.91 (0.02) |
| Embedded GMM | 0.81 (0.02) | 0.73 (0.04) | 0.75 (0.03) |
| Embedded NB | 0.82 (0.06) | 0.86 (0.03) | **0.95 (0.01)** |
| Method | Average true positive rate | | |
| Online dynamic classification ($w = 256$) | 0.91 (0.02) | 0.94 (0.02) | **0.97 (0.01)** |
| Sequential HMM | **0.93 (0.11)** | **0.95 (0.11)** | 0.94 (0.13) |
| Embedded HMM | 0.59 (0.09) | 0.64 (0.13) | 0.75 (0.10) |
| Embedded SVM | 0.70 (0.05) | 0.85 (0.05) | 0.95 (0.02) |
| Embedded GMM | 0.57 (0.06) | 0.64 (0.05) | 0.79 (0.05) |
| Embedded NB | 0.75 (0.06) | 0.88 (0.04) | **0.97 (0.01)** |

Table 3.3.2: Performance of classification procedures over 100 replications of vector moving average series for different scenarios of class changes, using the evaluation measures described in the text. Numbers in brackets represent the standard deviation of estimation errors. Bold numbers indicate best result.

| | Scenario 1 (nine changes) | Scenario 2 (five changes) | Scenario 3 (six changes) |
|---|---|---|---|
| Method | Average number of changes detected | | |
| Online dynamic classification ($w = 256$) | **9.82 (1.10)** | **5.78 (0.93)** | **6.59 (0.87)** |
| Sequential HMM | 35.75 (7.38) | 33.75 (8.82) | 77.24 (16.54) |
| Embedded HMM | 10.28 (3.71) | 8.69 (2.74) | 14.57 (4.89) |
| Embedded SVM | 11.90 (1.90) | 5.96 (1.10) | 9.18 (2.28) |
| Embedded GMM | 13.31 (2.89) | 14.21 (3.32) | 18.09 (4.56) |
| Embedded NB | 12.07 (1.63) | 5.87 (0.68) | 11.38 (2.36) |
| Method | Average V-measure | | |
| Online dynamic classification ($w = 256$) | **0.87 (0.02)** | **0.89 (0.03)** | **0.94 (0.01)** |
| Sequential HMM | 0.76 (0.04) | 0.66 (0.05) | 0.64 (0.06) |
| Embedded HMM | 0.75 (0.07) | 0.73 (0.08) | 0.79 (0.05) |
| Embedded SVM | 0.85 (0.02) | 0.88 (0.04) | 0.87 (0.03) |
| Embedded GMM | 0.81 (0.02) | 0.71 (0.03) | 0.81 (0.03) |
| Embedded NB | 0.84 (0.02) | 0.86 (0.03) | 0.87 (0.03) |
| Method | Average true positive rate | | |
| Online dynamic classification ($w = 256$) | **0.89 (0.03)** | **0.93 (0.02)** | **0.97 (0.01)** |
| Sequential HMM | 0.54 (0.15) | 0.57 (0.14) | 0.50 (0.15) |
| Embedded HMM | 0.62 (0.08) | 0.67 (0.11) | 0.68 (0.05) |
| Embedded SVM | 0.83 (0.03) | 0.91 (0.03) | 0.92 (0.03) |
| Embedded GMM | 0.70 (0.05) | 0.62 (0.05) | 0.72 (0.03) |
| Embedded NB | 0.81 (0.03) | 0.92 (0.02) | 0.90 (0.03) |

series) cannot cope with the dependence in the data, drastically overestimating the number of changes in the data.

The online dynamic classification algorithm outperforms the competitors consistently for the autoregressive series, as shown in Table 3.3.3. More specifically, it classifies the changes well in terms of the V-measure and true positive rate, i.e. a low misclassification rate. Provided that the set of training data accurately represents the range of classes present, we would expect the dynamic classification approach to be able to correctly detect both the location of the changes and the classes involved. In contrast, whilst the comparative methods detect the location of class changes well resulting in high V-measure, they can struggle to identify which class the signal belongs to after the class change has occurred, resulting in a lower true positive rate (a higher overall rate of misclassification). This can potentially be a challenge if accurate detection of anomalous areas is important.

In addition, note that in nearly all cases across the examples and scenarios, there is less variability in the evaluation measures using our proposed online dynamic classification (indicated by lower standard deviations). We also note here that the use of the coherence measure improves the performance of all competitor methods, justifying its efficacy as a classification feature in many settings. Crucially, we also found that the online dynamic classification approach was faster than HMM-based methods for longer time series. A more detailed analysis of the computational runtimes of the algorithms can be found in Section 3.7.

Table 3.3.3: Performance of classification procedures over 100 replications of vector autoregressive series for different scenarios of class changes, using the evaluation measures described in the text. Numbers in brackets represent the standard deviation of estimation errors. Bold numbers indicate best result.

| | Scenario 1 (nine changes) | Scenario 2 (five changes) | Scenario 3 (six changes) |
|---|---|---|---|
| Method | Average number of changes detected | | |
| Online dynamic classification ($w = 256$) | **9.79 (0.96)** | **5.36 (0.66)** | **6.64 (0.87)** |
| Sequential HMM | 12.03 (5.49) | 9.30 (4.46) | 17.27 (6.44) |
| Embedded HMM | 13.11 (2.70) | 10.44 (3.16) | 17.83 (6.60) |
| Embedded SVM | 12.80 (3.14) | 7.59 (2.03) | 12.37 (2.78) |
| Embedded GMM | 16.82 (3.04) | 6.32 (2.75) | 19.64 (4.40) |
| Embedded NB | 14.64 (3.13) | 6.87 (1.40) | 11.91 (2.62) |
| Method | Average V-measure | | |
| Online dynamic classification ($w = 256$) | **0.87 (0.02)** | **0.89 (0.03)** | **0.92 (0.02)** |
| Sequential HMM | 0.81 (0.08) | 0.73 (0.09) | 0.75 (0.07) |
| Embedded HMM | 0.79 (0.03) | 0.71 (0.08) | 0.74 (0.06) |
| Embedded SVM | 0.78 (0.03) | 0.83 (0.05) | 0.84 (0.03) |
| Embedded GMM | 0.75 (0.02) | 0.61 (0.07) | 0.73 (0.04) |
| Embedded NB | 0.78 (0.06) | 0.84 (0.04) | 0.84 (0.03) |
| Method | Average true positive rate | | |
| Online dynamic classification ($w = 256$) | **0.89 (0.02)** | **0.95 (0.02)** | **0.96 (0.01)** |
| Sequential HMM | 0.70 (0.11) | 0.69 (0.09) | 0.65 (0.11) |
| Embedded HMM | 0.59 (0.09) | 0.60 (0.10) | 0.59 (0.10) |
| Embedded SVM | 0.65 (0.05) | 0.89 (0.04) | 0.89 (0.04) |
| Embedded GMM | 0.51 (0.04) | 0.45 (0.04) | 0.51 (0.05) |
| Embedded NB | 0.64 (0.06) | 0.89 (0.03) | 0.89 (0.03) |

**Choice of window length** Within this simulation study, a window length of 256 is chosen when implementing the online dynamic classification method for all scenarios. Determining the best window length to use involves a trade-off between accuracy and speed; longer windows give more accurate results but also take more time to implement. Consider the three class trivariate normal process used within the simulation study, an example of which can be seen in Figure 4.4.1(a). For each of the scenarios considered in the study, we again simulate 100 replications of the test signal and apply the online dynamic classification method with differing window lengths of 128, 256 and 512. In this case, we record both the usual accuracy measures and the average time taken, the results can be found in Table 3.3.4. Whilst a window length of 128 takes the shortest amount of time to run, it overestimates the number of changes detected compared to the longer windows. For the windows of length 256 and 512, the average number of changes detected, V-measure and true positive rate are very similar. However the average time taken to run the algorithm with a window length of 512 is much longer than that for a window length of 256, especially in the case of Scenario 3. For this reason, we choose a window length of 256 for the simulations as any accuracy benefits from using a longer window will be outweighed by the increase in the time taken to implement the method in this case.

## 3.4 Case Study

In the previous section we considered the efficacy of our approach against tried and tested examples. We now turn to consider an application arising from our collabora-

Table 3.3.4: Performance of online dynamic classification algorithm with differing window lengths over 100 replications of multivariate Gaussian series for different scenarios of class changes, using the evaluation measures described in the text. Numbers in brackets represent the standard deviation of estimation errors. Bold numbers indicate best result.

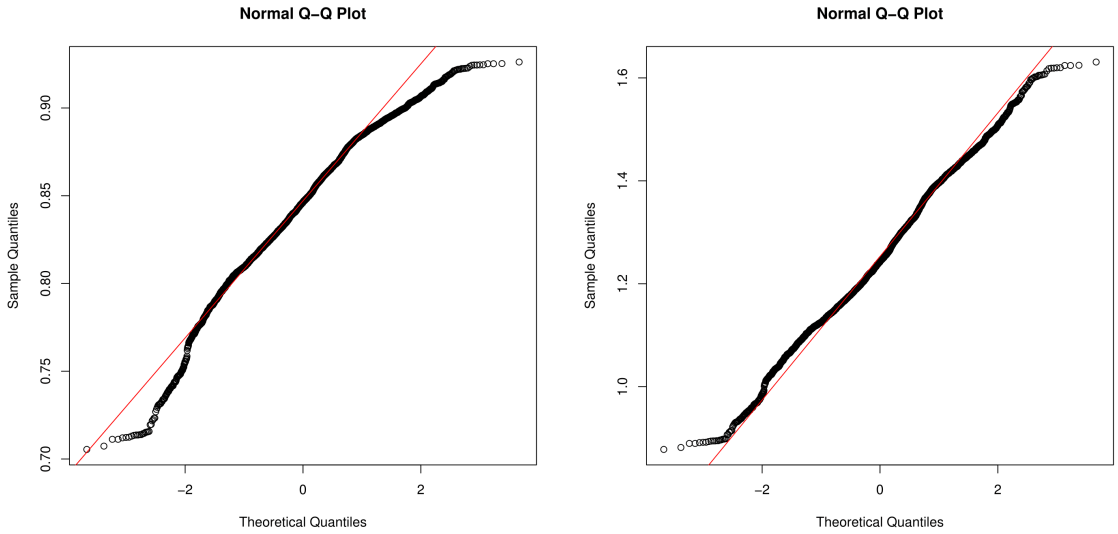
| | Scenario 1 (nine changes) | Scenario 2 (five changes) | Scenario 3 (six changes) |
|---|---|---|---|
| Method | Average number of changes detected | | |
| Online dynamic classification ($w = 128$) | 11.28 (1.29) | 13.22 (3.77) | 10.28 (2.79) |
| Online dynamic classification ($w = 256$) | 10.54 (1.37) | **5.80 (1.28)** | **6.56 (0.88)** |
| Online dynamic classification ($w = 512$) | **9.68 (0.96)** | 6.12 (1.26) | 6.86 (0.86) |
| Method | Average V-measure | | |
| Online dynamic classification ($w = 128$) | 0.87 (0.02) | 0.79 (0.04) | 0.92 (0.02) |
| Online dynamic classification ($w = 256$) | 0.88 (0.02) | **0.90 (0.03)** | **0.95 (0.01)** |
| Online dynamic classification ($w = 512$) | **0.89 (0.02)** | **0.90 (0.02)** | 0.94 (0.02) |
| Method | Average true positive rate | | |
| Online dynamic classification ($w = 128$) | 0.88 (0.02) | 0.87 (0.05) | 0.96 (0.01) |
| Online dynamic classification ($w = 256$) | 0.90 (0.02) | **0.95 (0.02)** | **0.98 (0.01)** |
| Online dynamic classification ($w = 512$) | **0.92 (0.02)** | **0.95 (0.02)** | 0.97 (0.01) |
| Method | Average time taken (seconds) | | |
| Online dynamic classification ($w = 128$) | 56.25 (5.58) | 55.68 (7.19) | 114.14 (12.31) |
| Online dynamic classification ($w = 256$) | 91.65 (3.52) | 100.22 (9.10) | 202.54 (14.27) |
| Online dynamic classification ($w = 512$) | 134.43 (9.82) | 130.08 (15.71) | 367.86 (24.66) |

tion with researchers working in the oil and gas industry.

The general philosophy is to apply our online dynamic classification method to acoustic sensing data provided by an industrial collaborator, with the aim of detecting striping within these signals. The training data consists of ten quadvariate signals of length 4096 obtained from a subsampled version of an acoustic sensing dataset. The class assignments for each of the training signals have been decided by an industrial expert. The test signal is obtained from the same dataset and is a quadvariate signal of length 8192, unseen in the training signals. The test series exhibits autocorrelation as well as dependence between series (see Figure 3.1.2). Due to the zero-mean assumption of the mvLSW model, in practice we detrend the series before analysis by taking first order differences of each component series, see Figure 3.1.1(b). We apply the online dynamic classification approach with a moving window of length 4096 to the test signal. Note that, we considered moving windows of length 1024, 2048, and 4096 within this analysis but in this case we select a window of length 4096 due to the accuracy benefits that a longer window can provide.

It is important to check that the Fisher-z transformed coherence estimates can be approximated by a Gaussian distribution. In order to validate this assumption, we generated Q-Q plots for both the original and the transformed coherence for different windows of data across the signal and for different channel combinations. Figure 3.4.1 includes one such example, it can be seen that the distribution of the transformed coherence in this case more closely follows a Gaussian distribution than that of the original coherence. Within this work, we only considered scenarios where the transformed coherence could be suitably approximated by a Gaussian distribution however

this assumption may not always hold.




(a) Original coherence (b) Transformed coherence

Figure 3.4.1: Q-Q plots for the original and transformed coherence between channels 2 and 3 for the first window of the data.

Based on the results from Section 3.3, for comparison the sequential HMM method is applied to the full test signal with the first 400 data points used to train a two-state model. We also apply the sequential HMM approach to the transformed coherence of the test signal, again training a two state model using the first 400 data points. Two-state models have been applied to demonstrate our belief that the acoustic sensing data contains areas of stable behaviour and striping.

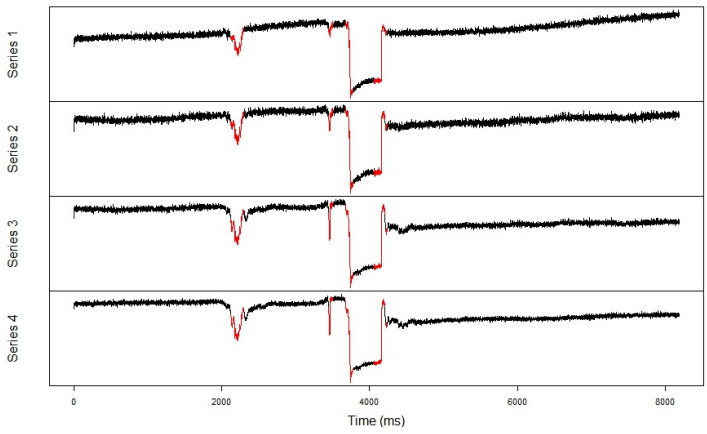
In this case the true class membership of the test signal is unknown, therefore we compare the results visually. One of the main issues with visually classifying striping features in this way is that the results can be subjective. For example, consider the acoustic sensing data in Figure 3.4.2 around timepoint 4000. Observed over a

sufficiently small interval, the data around timepoint 4000 could be considered to be normal behaviour rather than striping. It is these potential differences in labelling that can lead to misclassification.

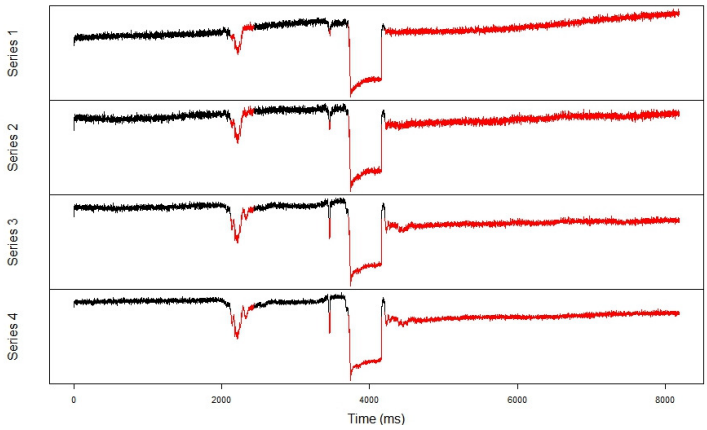
The classification results for each of the methods are found in Figure 3.4.2; areas of the test signal for which a change in class is detected are shown in red. It can be seen that the online dynamic classification method performs best in that it detects the stripes in the test signal with only minimal areas of misclassification when compared to the expert's judgement. In contrast, applying the sequential HMM approach to the transformed coherence of the signal also results in a change of class being detected at the stripes but the class changes take place over a longer period than we would expect. Finally, applying the sequential HMM method to the full signal results in the stripes being detected but the end of the test signal being misclassified.

Recalling that the overall aim of this analysis has been to detect sudden regions of interest within (multivariate) acoustic sensing signals, as accurately as possible whilst minimising the number of falsely detected points – then the results look very positive. Specifically the classification results obtained by the online dynamic classification method compare very favourably. It is interesting to note that in these examples, coarser scales (i.e. scales 6-11) appear to play a key role in the classification.

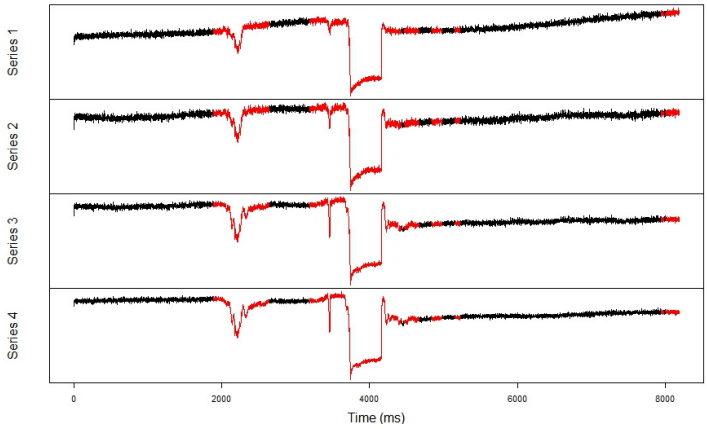
When compared with a subjective analysis of the data as displayed in Figure 3.1.1 we see that, each method correctly assigns 'non-stripe' regions with the following (correct) classification proportions; 0.944 for online dynamic classification, 0.792 for embedded HMM and 0.477 for sequential HMM.




(a) Online dynamic classification




(b) Sequential HMM




(c) Embedded HMM

Figure 3.4.2: Classification results obtained from applying online dynamic classification, Sequential HMM and Embedded HMM approaches to acoustic sensing data, areas of the signal for which a change in class is detected are shown in red.

## 3.5 Concluding remarks

In this article, we introduced an online dynamic classification method that can be used to detect changes in class within a data stream. We demonstrated the efficacy of the method using simulated data examples and an acoustic sensing dataset from an oil producing facility. The case study shows that our approach can be successfully used to detect anomalous periods in acoustic data, resulting in fewer areas of misclassification compared to more traditional classification methods, such as Markov Model approaches. Moreover, we have found that the use of a coherence measure in classification improves the performance of these methods.

In practice, we have found that a parsimonious choice of window is required: as with other moving window approaches, too short a window, and the results are not satisfactory; too long a window increases the computational time and potentially produces edge effects. We leave the challenge of automatically choosing window length as an avenue for future research. In addition, we have observed that, as with other competitor methods, our approach classifies well when the distance between class changes is comparable to the window length but can struggle when we have shorter segments between changes.

Future work may consider the problem of detecting stripes that are characterised by more gradual changes in their properties. In practice, these features may be less obvious and we might wish to not just detect but also classify the type of stripe present in the acoustic sensing data. Our method could potentially be used to do this, provided that our training data represents the range of stripes that we wish to

classify.

## 3.6 Online nondecimated wavelet transforms

In this section, we give more detail on the online wavelet transform that forms part of the online dynamic classification method outlined in Section 3.2. Assume that we have data of the form $(x_0, x_1, \ldots, x_T)$ and we wish to apply the online wavelet transform with window length $w = 2^J$ for $w < T$. The first step is to apply the Haar NDWT to the first window of the data $\mathbf{c}^J = \{x_0, x_1, \ldots, x_{w-1}\}$ using the following equations

$$\mathbf{c}^{(j-1)} = \mathcal{H}^{[J-j]}\mathbf{c}^{(j)}$$

$$\mathbf{d}^{(j-1)} = \mathcal{G}^{[J-j]}\mathbf{c}^{(j)}$$

for $j = 1, \ldots, J$. Following the notation of Nason and Silverman (1995), the filters $\mathcal{H}^{[r]}$ and $\mathcal{G}^{[r]}$ are defined by the following equations

$$h^{[r]}_{2^r j} = h_j \qquad g^{[r]}_{2^r j} = g_j$$

$$h^{[r]}_k = 0 \qquad g^{[r]}_k = 0 \quad \text{if } k \text{ is not a multiple of } 2^r.$$

For the Haar wavelet filters, $\mathcal{H}^{[0]}$ consists of the elements $h_0 = h_1 = \frac{1}{\sqrt{2}}$ and $\mathcal{G}^{[0]}$ has the form $g_0 = \frac{1}{\sqrt{2}}, g_1 = -\frac{1}{\sqrt{2}}$. As a result of applying the Haar NDWT, we obtain the smooth and detail coefficients at each level of the transform,
$(\mathbf{c}^J, \mathbf{c}^{J-1}, \ldots, \mathbf{c}^1, \mathbf{c}^0, \mathbf{d}^{J-1}, \ldots, \mathbf{d}^1, \mathbf{d}^0)$. In this case, we focus solely on the Haar NDWT as this allows us to make the largest reduction in the number of calculations required to update the wavelet coefficients for each window whilst also providing good classifi-

cation results. However, the method could be generalised to use other wavelet filters rather than Haar.

The next step is to shift the window along by one so that the new data vector is of the form $\{x_1, \ldots, x_w\}$. Instead of reapplying the transform to this new window of data, the wavelet coefficients can be calculated using the information obtained from the transform of the previous window. To do this, the elements of the smooth and detail coefficients are shifted and the coefficients that change as a result of gaining the new data point $x_w$ and losing $x_0$ are recomputed.

Let $\mathbf{c}^{S,J} = \{c_{J,1}, \ldots, c_{J,w-1}, c_{J,0}\}$ denote the shifted version of the vector $\mathbf{c}^J$. Then the smooth coefficient vectors for the new window of data $(\widehat{\mathbf{c}}^J, \widehat{\mathbf{c}}^{J-1}, \ldots, \widehat{\mathbf{c}}^0)$ can be calculated from

$$\widehat{c}_{J-n,2^J-k} = c^S_{J-n,2^J-k} - \Big(\frac{1}{\sqrt{2}}\Big)^n x_0 + \Big(\frac{1}{\sqrt{2}}\Big)^n x_w, \tag{3.6.1}$$

for $n = 0, 1 \ldots, J$ and $k = 1, \ldots, 2^n$. In a similar way, the detail coefficients for the new window of data $(\widehat{\mathbf{d}}^{J-1}, \ldots, \widehat{\mathbf{d}}^1, \widehat{\mathbf{d}}^0)$, can be calculated from the following formulae

$$\widetilde{d}_{J-1-n,2^J-k} = \; d^S_{J-1-n,2^J-k} - \Big(\frac{1}{\sqrt{2}}\Big)^{n+1} x_0 + \Big(\frac{1}{\sqrt{2}}\Big)^{n+1} x_w, \tag{3.6.2}$$

for $n = 0, 1 \ldots, J-1$ and $k = 1, \ldots, 2^n$,

$$\widetilde{d}_{J-1-n,2^J-k} = \; d^S_{J-1-n,2^J-k} + \Big(\frac{1}{\sqrt{2}}\Big)^{n+1} x_0 - \Big(\frac{1}{\sqrt{2}}\Big)^{n+1} x_w \tag{3.6.3}$$

for $n = 0, 1 \ldots, J-1$ and $k = 2^{n+1}, 2^{n+1} - 1 \ldots, 2^n - 1$. The process of shifting the window and calculating the new smooth and detail coefficient vectors using equations (3.6.1)–(3.6.3) is repeated until we reach the end of the data set. In the case of the Haar wavelets, the total number of updated coefficients for each window is $\sum_{j=0}^{J} 2^j = 2^{J+1} - 1$ compared to the $J2^J$ calculations required to fully recompute the NDWT.

## 3.7 Comparison of computational cost of online classification methods

In this section, we provide an analysis of the computational cost of the various competitor classification methods outlined in Section 3.3. To this end, we run each online classification method on a set of test signals of increasing length, namely $T = 1024, 2048, 4096, 8192$. In particular, for each method and series length, we record the runtime of each method, averaged over $K = 25$ replications of series from the first example in Section 3.3. This allows us to compare how the runtime of each method scales with series length, $T$, removing external factors such as efficiency of coding and implementation programming language.

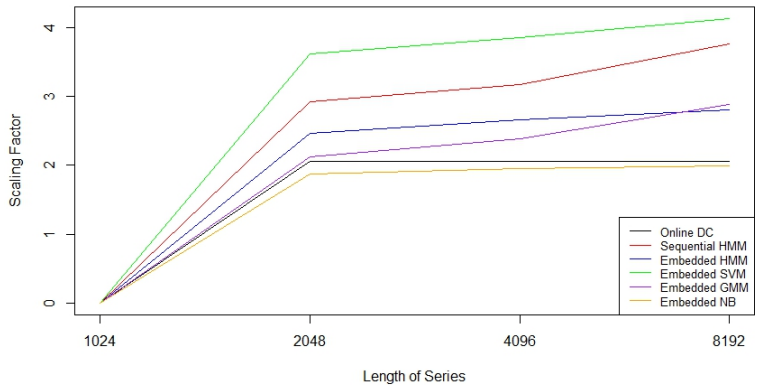



Figure 3.7.1: Comparison of computational cost of the classification methods described in Section 3.3 in terms of their scaling behaviour with length of test series.

The results of the runtime analysis are shown in Figure 3.7.1. As seen from the plot, as expected, each method increases in runtime with the length of the series. However, after an initial increase, our dynamic online classification method has a

desirable near constant scaling with the length of the series. Its scaling profile is the best after the NB classifier. Given the improvement in classification over the competitor methods across the range of examples studied in Section 3.3, we feel that this profile justifies the use of the proposed method.

# Chapter 4

# A wavelet-based approach for imputation in highdimensional series

## 4.1 Introduction

Time series data commonly arise in a variety of different areas including finance (Taylor, 2007), biology (Bar-Joseph et al., 2003) and energy (Alvarez et al., 2011; Doucoure et al., 2016). The collection and recording of time series can be interrupted in a number of ways including human error or technical faults with the recording equipment which induces missingness within the time series. Little and Rubin (2002) describe such occurrences of missing values in data through a number of "missingness mechanisms":

1. Missing completely at random (MCAR) - The probability of missingness is the same for all units, i.e. the missing value is not dependent on other variables.

2. Missing at random (MAR) - The probability of missingness depends only on available information, i.e. the missing value depends on other variables.

3. Not missing at random (NMAR) - The missingness probability depends on the variable itself, i.e. the missing observation depends on other missing values.

Regardless of the type of missingness present, further analysis of the time series such as autocovariance or spectral estimation can be difficult without first replacing the missing data with appropriate estimates. This estimation process is called imputation.

There exists a rich literature in the statistics community dedicated to the imputation of missing values within stationary time series. See Pratama et al. (2016) for a recent review of this literature. Popular multivariate approaches such as Multiple Imputation (Rubin, 1987), Hot-Deck (Ford, 1983) and Expectation-Maximisation (EM) (Dempster et al., 1977) make use of inter-variable correlations to estimate missing data. EM approaches within the literature often assume that the observed and missing data follow a multivariate normal distribution, see Junger and de Leon (2015), Honaker and King (2010) and Honaker et al. (2011) for examples. Alternative methods have been developed combining EM with other distributions including PCA fixed-effects models (Caussinus, 1986) and Gaussian Mixture Models (Ghahramani and Jordan, 1994). Other approaches to imputation within time series make use of statistical models to infer missing values. Examples of model-based methods include those which use genetic algorithms (Lobato et al., 2015; Tang et al., 2015), support

vector machines (Wu et al., 2015), random forests (Stekhoven and Bühlmann, 2011) and autoregressive processes (Sridevi et al., 2011). On the other hand, univariate time series measurements are collected from one variable only and so inter-time correlation is important. Moritz and Bartz-Beielstein (2017) discuss various univariate time series imputation approaches, ranging from simple methods that replace missing values with the mean to more advanced approaches that involve spline interpolation or model fitting combined with the use of a Kalman filter.

Another approach for coping with such missingness is to estimate the spectral information of the series in some way; a range of methods have been developed for spectral estimation in stationary time series with missing values or irregularly sampled observations within the time series and signal processing literature. The Lomb-Scargle estimator (Lomb, 1976; Scargle, 1982) estimates the Fourier spectrum from the irregularly sampled data but can be subject to strong bias which hinders it's ability to describe slopes within the spectrum. Variants of this approach have been applied in a range of different fields including astronomy (Wen et al., 1999), biology (Van Dongen et al., 1999) and biomedical engineering (Laguna et al., 1998). Other widely used techniques involve fitting time series models directly to the unequally spaced data and using this to estimate spectral information for stationary processes (Jones, 1980; Bos et al., 2002; Broersen, 2006). In practice however, the assumptions imposed by modelling an observed time series as stationary can be restrictive and unrealistic.

However, nonstationary time series, i.e. series with time-varying second order structure, are observed in various fields including finance (Stărică and Granger, 2005; Fryzlewicz et al., 2006), medicine (Cranstoun et al., 2002) and oceanography (Killick

et al., 2013). For this reason, the problem of imputing missing values in multivariate, nonstationary time series is a potentially important one. Many techniques have been developed for modelling and analysing complete multivariate, nonstationary data including the locally stationary Fourier model (Dahlhaus, 1997), the smooth localized complex exponential (SLEX) model (Ombao et al., 2005) and the multivariate locally stationary wavelet (mvLSW) framework (Park et al., 2014) but the literature on how to deal with missingness within such data is sparse. In the univariate setting, Knight et al. (2012) propose a method for estimating spectral information of a LSW process containing missing values where information is estimated at the observed time locations. However, the problem of spectral estimation within multivariate, nonstationary time series has not been widely studied.

In this article, we address the challenging problem of imputation in the multivariate locally stationary time series setting. Our approach involves first estimating the local wavelet spectral matrix of a mvLSW process with missing observations before forecasting and backcasting the missing values of the time series using a multivariate extension of the wavelet forecasting approach of Fryzlewicz et al. (2003). As a final step, we average the estimates obtained from the forward and backward pass to obtain an overall estimate of the time series. Through the use of simulated examples and a case study, we demonstrate that our method performs well for a range of realistic missingness scenarios in both the stationary and nonstationary setting.

This work is organised as follows. Within Section 4.2, we review existing methods for forecasting within locally stationary time series. In Section 4.3, we introduce the proposed imputation method. Section 4.4 contains a simulation study evaluating

the performance of the proposed imputation method using synthetic examples. In Section 4.5 we describe a case study using a dataset arising from a Carbon Capture and Storage facility. Finally, Section 4.6 includes some concluding remarks.

## 4.2 Background

In this section we provide an overview of recent work on forecasting within locally stationary time series. For a comprehensive review of nonstationary time series more generally, please refer to the excellent review by Dahlhaus (2012). This section is organised as follows; we focus on one-step ahead forecasting within univariate LSW processes in Section 4.2.1 before looking at the mvLSW setting in 4.2.2.

### 4.2.1 Forecasting univariate locally stationary time series

Fryzlewicz et al. (2003) introduce a wavelet-based method for forecasting univariate, nonstationary time series. The methodology uses the LSW framework of Nason et al. (2000), outlined in Section 2.3.1, to form generalised Yule-Walker equations which can be used to carry out $h$-step ahead prediction in this setting. We will briefly review one-step ahead prediction as this is the case that we will use in the multivariate setting.

Following Fryzlewicz et al. (2003), the one-step ahead predictor of $X_{t,T}$ given previous observations $X_{0,T}, \ldots, X_{t-1,T}$ is defined by

$$\hat{X}_{t,T} = \sum_{s=0}^{t-1} b^{(1)}_{t-1-s,T} \, X_{s,T}, \tag{4.2.1}$$

where the coefficients $b_{t-1-s,T}$ minimise the mean square prediction error (MSPE) defined by

$$\mathrm{MSPE}(\hat{X}_{t,T}, X_{t,T}) = \mathrm{E}(\hat{X}_{t,T} - X_{t,T})^2.$$

The prediction method minimises the mean square prediction error

$$\mathrm{MSPE}(\hat{X}_{t,T}, X_{t,T}) = \mathbf{b}_t^\top \mathbf{\Sigma}_{t,T} \mathbf{b}_t,$$

where $\mathbf{b}_t = (b^{(1)}_{t-1,T}, \ldots, b^{(1)}_{0,T}, -1)$ and $\mathbf{\Sigma}_{t,T}$ is the covariance matrix of $X_{0,T}, \ldots, X_{t,T}$. The MSPE can be approximated by $\mathbf{b}_t^\top \mathbf{B}_{t,T} \mathbf{b}_t$ where $\mathbf{B}_{t,T}$ is a $(t+1) \times (t+1)$ matrix whose $(m, n)$ entry is given by

$$\sum_{j=-J}^{-1} S_j\left(\frac{m+n}{2T}\right) \Psi_j(n-m),$$

where $S_j(z)$ is the evolutionary wavelet spectrum introduced in equation (2.3.2) and $\Psi$ are the autocorrelation wavelets as defined in equation (2.3.4).

The coefficients $\mathbf{b}_t$ that minimise the MSPE can be shown to solve the following prediction equations

$$\sum_{m=0}^{t-1} b^{(1)}_{t-1-m,T}\; c\left(\frac{m+n}{2T}, m-n\right) = c\left(\frac{n+t}{2T}, t-n\right), \tag{4.2.2}$$

where $c(z, \tau)$ is the local autocovariance function defined in equation (2.3.3).

### 4.2.2 Forecasting multivariate locally stationary wavelet processes

Our one-step ahead prediction approach is a straightforward extension of the foregoing work to the multivariate setting that makes use of the mvLSW model outlined in Section 2.3.2.

Due to the separable structure of the mvLSW model, we can form one-step ahead prediction equations for each channel combination $(p, q)$ using the local auto and cross-covariance defined in equations (2.3.12) and (2.3.13) respectively. The multivariate prediction equations are defined by

$$\sum_{m=0}^{t-1} b_{t-1-m,T}^{(p,q)} \, c^{(p,q)}\left(\frac{m+n}{2T}, m-n\right) = c^{(p,q)}\left(\frac{n+t}{2T}, t-n\right). \tag{4.2.3}$$

As in the univariate setting described in Section 4.2.1, the coefficients $\mathbf{b}_t^{(p,q)}$ that solve the prediction equations can be shown to minimise the quadratic equation

$$\mathbf{b}_t^{(p,q)\top} \mathbf{\Sigma}_{t,T}^{(p,q)} \mathbf{b}_t^{(p,q)},$$

where $\mathbf{b}_t^{(p,q)} = (b_{t-1,T}^{(p,q)}, \ldots, b_{0,T}^{(p,q)}, -1)$ and $\mathbf{\Sigma}_{t,T}^{(p,q)}$ is the covariance matrix of $X_{0,T}^{(p)}, \ldots, X_{t,T}^{(p)}, X_{0,T}^{(q)}, \ldots, X_{t,T}^{(q)}$.

The one-step ahead predictor of $\mathbf{X}_{t,T}$ given previous multivariate observations $\mathbf{X}_{0,T}, \ldots, \mathbf{X}_{t-1,T}$ is given by

$$\hat{X}_{t,T}^{(a)} = \sum_{b \in \{1,\ldots,P\}} \sum_{s=t-p}^{t-1} b_{t-1-s;T}^{(a,b)} X_{s,T}^{(b)} \qquad \text{for } p \in \{1, \ldots, P\}, \tag{4.2.4}$$

where $p$ is the number of recent observations used in prediction. Having outlined forecasting in the mvLSW setting, we now outline our imputation approach that uses one-step ahead prediction to replace missing values in a multivariate locally stationary time series.

## 4.3 Imputation for multivariate locally stationary wavelet processes

In this section we introduce our multivariate imputation method which uses the local auto and cross-covariance structure of a nonstationary time series to estimate missing observations. The key challenge in this context is that the usual mvLSW spectral estimation process cannot be used due to the presence of missingness. For this reason, the first step of the algorithm involves estimating the wavelet periodogram of a mvLSW process with missing observations, this will be discussed in Section 4.3.1. Using the estimate of the LWS matrix, we then form the local auto and cross-covariance structure and carry out a forward pass of the data where we forecast missing values. To obtain more accurate estimates of the time series at missing locations, we also implement a backward pass of the data where we backcast the missing values. We then average the series obtained from the forward and backward pass in order to get an overall estimate. The forecasting and backcasting steps will be described in Section 4.3.2 and 4.3.3 respectively. A complete overview of the steps carried out in one iteration of the method can be found in Algorithm 2.

**Algorithm 2** mvLSWimpute: Steps carried out in one iteration of the method.

1: **Spectral estimation step**

Estimate the LWS matrix of the signal containing missing data in the following way:

(a) Estimate the raw wavelet periodogram keeping any NAs intact, any wavelet coefficients affected by the initial NAs will also be missing.

(b) For each spectra and cross-spectra, determine the coarsest level that contains true wavelet coefficients $\hat{J}^{(p,q)}$.

(c) For the finer levels of the periodogram through to $\hat{J}^{(p,q)}$, linearly interpolate the missing wavelet coefficients by level.

(d) Recursively apply the wavelet filter equations to the level $\hat{J}^{(p,q)}$ of the periodogram to replace any levels that consist solely of NAs.

(e) Smooth the periodogram using a running mean smoother and correct using the inverse of the inner product matrix of discrete autocorrelation wavelets $\mathbf{A}$.

2: **Forecasting step**

For each missing index $i$, forecast the missing value in the following way:

(a) Consider the spectra obtained in Step 1 from time 1 to time $i-1$.

(b) Form the local auto and cross-covariance by substituting estimated spectra from time 1 to $i-1$ into equations (2.3.12) and (2.3.13).

(c) For each channel combination $(p, q)$, solve the prediction equations given in equation (4.2.3) to obtain $\mathbf{b}^{(p,q)}$.

(d) Use $\mathbf{b}^{(p,q)}$ vectors along with the clipped predictor in equation (4.2.4) to estimate the value of the time series at missing index $i$.

---

3: **Backcasting step**

For each missing index $i$, backcast the missing value in the following way:

(a) Consider the spectra obtained in Step 1 from time $T$ to time $i+1$.

(b) Form the local auto and cross-covariance by substituting estimated spectra from time $T$ to $i+1$ into equations (2.3.12) and (2.3.13).

(c) For each channel combination $(p, q)$, solve the prediction equations given in equation (4.2.3) to obtain $\mathbf{b}^{(p,q)}$.

(d) Use $\mathbf{b}^{(p,q)}$ vectors along with the clipped predictor in equation (4.3.1) to estimate the value of the time series at missing index $i$.

4: **Averaging step**

Average the estimates of the time series obtained from the forward pass described in Step 2 and the backward pass described in Step 3.

---

### 4.3.1 Spectral Estimation

Suppose that we have a $P$-variate time series of length $T = 2^J$ containing missing values which we wish to impute. The first step of the `mvLSWimpute` algorithm involves estimating the LWS matrix of the time series. The presence of missing values means that we can not use the usual estimation procedure and have to modify our approach.

First, we calculate the empirical wavelet coefficient vector, $\mathbf{d}_{j,k}$, for the time series

ensuring that any wavelet coefficients at scales and locations affected by the initial missing data will also be missing. The Haar wavelet is used within the calculation of the empirical wavelet coefficient vector as this will ensure that more levels of the wavelet transform will contain information. From this, we can form the raw wavelet periodogram matrix as described in Section 2.3.2. Since the raw wavelet periodogram will also have entries missing, we need to perform an intermediate step to fill in these missing values before smoothing and correcting as described in Section 2.3.2 to obtain an estimate of the LWS matrix.

In order to fill in the missing values, for each spectra and cross-spectra we determine the coarsest level of the periodogram $\hat{J}^{(p,q)}$ that contains true wavelet coefficients and does not consist solely of NAs. For the finer scales through to $\hat{J}^{(p,q)}$, we linearly interpolate the missing coefficients by level. For the coarsest levels of the periodogram where the coefficients are all missing, we replace them in a different way. To do this, we recursively apply the wavelet filter equations to the periodogram from level $\hat{J}^{(p,q)}$ which generates coefficients that allow us to replace the values in the coarsest levels.

To obtain an estimate of the LWS matrix, we correct the periodogram by multiplying by $\mathbf{A}^{-1}$ and then smooth the result using a running mean smoother with window length $\lfloor\sqrt{T}\rfloor$, implemented in the **mvLSW** *R* package (Taylor et al. (2017)). This estimate can then be substituted into equations (2.3.12) and (2.3.13) to form the local auto and cross-covariance which are used in the forecasting and backcasting steps of the algorithm.

### 4.3.2 Forecasting

In order to replace missing data in the time series, we first carry out a forward pass of the series where we use the one-step ahead multivariate wavelet forecasting approach outlined in Section 4.2.2. A missing index is defined to be a timepoint at which one or more channels of the $P$-variate time series has missing values present.

For each missing index $i$, we forecast the missing values sequentially in the following way. First, calculate the local auto and cross-covariance using the estimated spectra from time 1 to time $i-1$ and equations (2.3.12) and (2.3.13).

For each channel combination $(p, q)$ where $p, q \in \{1, \ldots, P\}$, form the prediction equations using the local auto and cross-covariance at certain locations and lags, as in equation (4.2.3). Solving the prediction equations allows us to obtain $\mathbf{b}^{(p,q)}$ vectors which are then used to predict the values of the series at time $i$ using the one-step ahead predictor defined in equation (4.2.4). The channels of the multivariate time series that contain missing data at time $i$ are then replaced by the corresponding predicted values from the forecasting step.

It is important to note that, for efficiency, we use a clipped predictor in the forecasting step in which only the most recent $p$ observations are used in the prediction. Currently, we use a global value of $p$ throughout the algorithm however in future we may wish to develop a more data driven approach to choosing $p$.

### 4.3.3 Backcasting

After carrying out the forward pass of the data, the next step is to backcast the missing values sequentially. This backcasting step is included in order to improve the accuracy of the imputation method since this allows us to incorporate information from both sides of the missing observation in our estimation. Similarly to the approach of Trindade (2003), we can form backward Yule-Walker equations in the mvLSW setting by beginning at time $T$ and again using the multivariate wavelet forecasting approach from Section 4.2.2 but we must be careful in how we order the spectral values in this case. Note that the backward pass is carried out independently to the forward pass and does not depend on the imputed time series obtained in the previous step.

For each missing index $i$ (considered in descending order), we proceed as in the forecasting case and form the local auto and cross-covariance using the estimated spectra from time $T$ to $i+1$. As in the forward pass, for each channel combination $(p, q)$, we can solve the prediction equations using the local auto and cross-covariance to obtain the $\mathbf{b}^{(p,q)}$ vectors. However, the one-step ahead predictor has a slightly different form in the backcasting step:

$$\hat{X}_{t,T}^{(a)} = \sum_{b\in\{1,\ldots,P\}} \sum_{s=t+p}^{t+1} b_{t+1-s;T}^{(a,b)} X_{s,T}^{(b)} \qquad \text{for } p \in \{1,\ldots,P\}. \tag{4.3.1}$$

The one-step ahead predictor in equation (4.3.1) is used to backcast the value of the time series at index $i$ and then any missing entries within channels are replaced using their corresponding predicted values.

After carrying out the forward and backward pass independently, we obtain two

imputed time series which are then averaged to get an overall estimate of the time series. The process can then be iterated but from the second iteration onwards the spectral estimation step no longer requires linear interpolation and the LWS matrix can now be estimated using equation (2.3.22). The forecasting and backcasting steps remain the same and we again average to obtain an updated estimate of the time series.

## 4.4 Simulated performance of mvLSWimpute

We now assess the performance of our proposed multivariate imputation method through a range of simulated data examples, a number of different scenarios have been chosen for the missingness in order to mimic situations arising in practice. The generating series used within the simulation study exhibit varying degrees of nonstationarity and dependence; these have been chosen to test the ability of our method to impute missing values in multivariate, nonstationary time series.

For datasets containing missing entries, traditional analysis based on complete cases has proven to be reasonably accurate provided that the amount of missing values is small (Graham, 2009). However, such methods yield poor results when the proportion of missing entries increases. To evaluate the performance proposed method as the amount of missingness increases, we remove 10%, 20%, 30% and 40% of values from the generating series at random, either from all channels simultaneously or from one channel independently.

In practice, time series obtained from industrial applications can contain gaps that

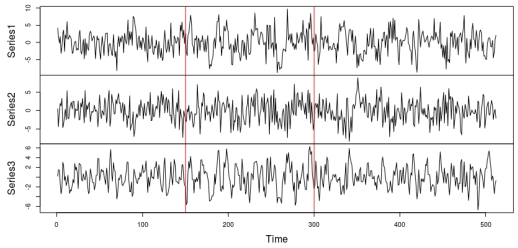
may extend over hours or even days due to faults in the recording equipment or human error. In order to reflect this, we consider the case where information is missing from one or more variables of the time series for a period of 20 consecutive time points. As a third scenario, we also include the situation where the missingness occurs in bursts up to length 20 before the signal returns to normal for a set period of time.

For all missingness scenarios, the coefficients of the generating series randomly switch at set times in order to test the ability of the imputation methods to deal with slowly and rapidly evolving dependence within a signal. The time series used in each case have the following forms

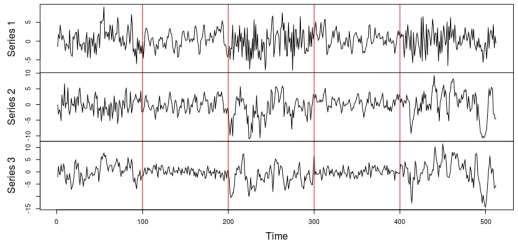
Slow changes: Trivariate signal of length 512, two changes in the generating coefficients of the series, occurring at time 150 and 300.

Rapid changes: Trivariate signal of length 512, four changes in the generating coefficients of the series, occurring at time 100, 200, 300 and 400.

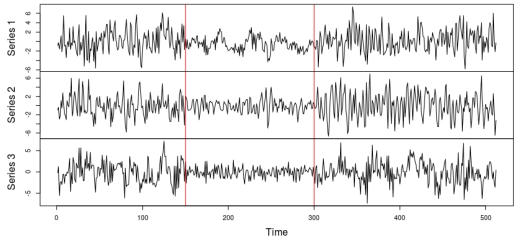
The first example we consider is a mvLSW process with changing spectral structure, chosen in such a way that there is strong coherence between channels of the signal. In this case, the example consists of two underlying classes with differing LWS



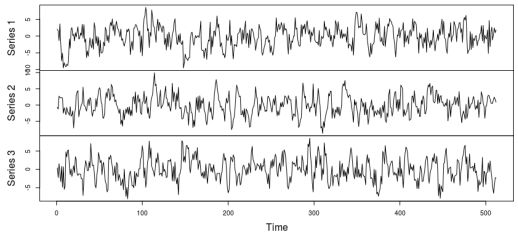

(a) Slow changes, mvLSW process




(b) Rapid changes, vector autoregressive moving average series




(c) Slow changes, vector autoregressive series




(d) Stationary vector moving average series

Figure 4.4.1: Example realisations of generating processes for the different scenarios used in the simulation study. (a), (c) Slowly evolving dependence, class changes at time 150 and 300; (b) Rapidly changing dependence structure, class changes at time 100, 200, 300 and 400; (d) Stationary signal, no changes in the generating coefficient matrices of the process.

matrices as defined below

$$\text{Class 1}: \quad S_j^{(1,2)}(z) = \begin{cases} 5 & \text{for } j=1, \\ -6 & \text{for } j=2, \\ -4 & \text{for } j=5. \end{cases} \quad S_j^{(1,3)}(z) = \begin{cases} 2 & \text{for } j=3, \\ -4 & \text{for } j=4. \end{cases} \quad S_j^{(2,3)}(z) = -6 \text{ for } j=2.$$

$$\text{Class 2}: \quad S_j^{(1,2)}(z) = \begin{cases} -5 & \text{for } j=1, \\ 6 & \text{for } j=2, \\ 4 & \text{for } j=5. \end{cases} \quad S_j^{(1,3)}(z) = \begin{cases} 8 & \text{for } j=3, \\ 4 & \text{for } j=4. \end{cases} \quad S_j^{(2,3)}(z) = 6 \text{ for } j=2.$$

Example simulated data for this process using the different dependence structures described above are shown in Figure 4.4.1(a).

The second example we examine is a time-varying vector autoregressive moving average process with three different classes defined by the following coefficient matrices

$$\text{Class 1}: \ \mathbf{X}_t = \begin{pmatrix} 0.4 & 0.1 & -0.2 \\ 0.1 & 0.3 & -0.3 \\ -0.2 & -0.3 & -0.2 \end{pmatrix} \mathbf{X}_{t-1} + \mathbf{Z}_t + \begin{pmatrix} 1 & 0.8 & 0.4 \\ 0.8 & 1 & 0.1 \\ 0.4 & 0.1 & 1 \end{pmatrix} \mathbf{Z}_{t-1}$$

$$\text{Class 2}: \ \mathbf{X}_t = \begin{pmatrix} -0.3 & -0.2 & 0.3 \\ -0.2 & -0.3 & 0.1 \\ 0.3 & 0.1 & 0.2 \end{pmatrix} \mathbf{X}_{t-1} + \mathbf{Z}_t + \begin{pmatrix} 1 & -0.6 & 0.3 \\ -0.6 & 1 & -0.3 \\ 0.3 & -0.3 & 1 \end{pmatrix} \mathbf{Z}_{t-1}$$

$$\text{Class 3}: \ \mathbf{X}_t = \begin{pmatrix} -0.6 & 0.4 & 0.1 \\ 0.4 & 0.2 & 0.3 \\ 0.1 & 0.3 & 0.5 \end{pmatrix} \mathbf{X}_{t-1} + \mathbf{Z}_t + \begin{pmatrix} 1 & 0.2 & -0.7 \\ 0.2 & 1 & 0.6 \\ -0.7 & 0.6 & 1 \end{pmatrix} \mathbf{Z}_{t-1}$$

where $\mathbf{Z}_t$ and $\mathbf{Z}_{t-1}$ are zero-mean multivariate normal realisations, distributed with class-dependent covariances

$$\Sigma_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad \Sigma_2 = \Sigma_3 = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{pmatrix}.$$

The signal switches randomly between each of the three classes at different times depending on whether we are considering slowly or rapidly evolving dependence. This ensures that the intra and cross-channel dependence of the process changes over time (see Figure 4.4.1(b)).

The third example we consider is a time-varying vector autoregressive process with two classes defined by the following coefficient matrices

$$\text{Class 1}: \ \mathbf{X}_t = \begin{pmatrix} 0.3 & 0.2 & -0.2 \\ 0.2 & 0.4 & -0.2 \\ -0.2 & -0.2 & -0.1 \end{pmatrix} \mathbf{X}_{t-1} + \begin{pmatrix} 0.4 & -0.2 & 0.3 \\ -0.2 & -0.4 & 0.1 \\ 0.3 & 0.1 & -0.2 \end{pmatrix} \mathbf{X}_{t-2} + \epsilon_1$$

$$\text{Class 2}: \ \mathbf{X}_t = \begin{pmatrix} 0.2 & -0.2 & 0 \\ -0.2 & 0.4 & -0.2 \\ 0 & -0.2 & 0.2 \end{pmatrix} \mathbf{X}_{t-1} + \begin{pmatrix} -0.1 & 0 & 0 \\ 0 & -0.4 & 0.3 \\ 0 & 0.3 & 0.3 \end{pmatrix} \mathbf{X}_{t-2} + \epsilon_2$$

where the noise vectors $\epsilon_i$ are zero-mean multivariate normal realisations, distributed with class-dependent covariances

$$\Sigma_{\epsilon_1} = \begin{pmatrix} 1 & 0.2 & 0 \\ 0.2 & 1 & 0.1 \\ 0 & 0.1 & 1 \end{pmatrix}, \qquad \Sigma_{\epsilon_2} = \begin{pmatrix} 5 & 1.2 & 2 \\ 1.2 & 5 & 1.5 \\ 2 & 1.5 & 5 \end{pmatrix}.$$

A realisation of such a process can be seen in Figure 4.4.1(c).

Within the simulation study, we compare our method to a range of multivariate imputation approaches, some of which assume that the data follows a multivariate normal distribution. For this reason, we also include a stationary example where the coefficients of the moving average process do not change over time. The coefficient matrices

for this process are defined as follows

$$\mathbf{X}_t = \mathbf{Z}_t + \begin{pmatrix} 1 & 0.5 & -0.2 \\ 0.5 & 1 & 0.3 \\ -0.2 & 0.3 & 1 \end{pmatrix} \mathbf{Z}_{t-1} + \begin{pmatrix} 1 & -0.4 & 0.2 \\ -0.4 & 1 & -0.6 \\ 0.2 & -0.6 & 1 \end{pmatrix} \mathbf{Z}_{t-2} + \begin{pmatrix} 1 & 0.1 & -0.5 \\ 0.1 & 1 & -0.3 \\ -0.5 & -0.3 & 1 \end{pmatrix} \mathbf{Z}_{t-3}$$

where $\mathbf{Z}_{t-1}, \mathbf{Z}_{t-2}$ and $\mathbf{Z}_{t-3}$ are zero-mean multivariate normal realisations, with covariances given by

$$\Sigma = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}.$$

An example of this stationary process can be found in Figure 4.4.1(d).

### 4.4.1 Competitor methods

In the simulation study, we compare our method with a number of alternative multivariate imputation approaches. Firstly we consider the modified Expectation-Maximization (EM) approach implemented in the *R* package `mtsdi` (Junger and de Leon, 2018). Within this method, cross-channel correlations are taken into account within the multivariate normal modelling structure and inter-time behaviour is accounted for using a level estimation step in which temporal behaviour of each of the univariate time series is estimated. Within the `mtsdi` package, a number of different methods are implemented for estimating the level of the univariate time series. For all simulated examples, we fit a cubic spline to each univariate component where the number of degrees of freedom of each spline is chosen by cross-validation.

Similarly, we also apply the multiple imputation method that combines expectation-maximization with bootstrapping, available in the `Amelia II` *R* package (Honaker

et al., 2015). As this is a multiple imputation approach, the method produces $m$ completed datasets which are then averaged to obtain a final imputed dataset. Within the simulations, we choose $m = 5$ as this is suggested by the authors to be suitable unless the rate of missingness is very high. As both of these methods assume that the data can be modelled using a multivariate normal distribution, we would expect them to perform poorly in cases where the underlying time series is highly nonstationary.

As we wish to impute missing values in a multivariate, nonstationary time series, it is important to compare our method to a range of model-based approaches. We apply the iterative PCA method from the $R$ package `missMDA` (Husson and Josse, 2018). Within the simulations, we apply the regularised iterative PCA algorithm with the number of random initializations set to 10 and with the default parameter values. In addition, we compare to the non-parametric random forest imputation method implemented in the $R$ package `missForest` (Stekhoven, 2013) where again we use the default parameters.

Since our method involves using a one-step ahead forecasting and backcasting step within the mvLSW framework, as a direct comparison to this we also apply the vector autoregressive prediction approach from the $R$ package `MTS` (Tsay, 2015). For each missing index, the approach fits a vector autoregressive process to the available observations and then produces one-step ahead forecasts to predict the missing values. In the same way as the mvLSWimpute method, we carry out a backward pass of the data and combine the estimates from the forecasting and backcasting steps by averaging (denoted VAR-fb). For completeness, we also include the results from applying one-step ahead forecasting only within this setting (denoted VAR-f).

### 4.4.2 Evaluation measures

For each of the missingness scenarios and dependence structures described in Section 4.4, 100 replications of the test signals are simulated and four different evaluation measures are considered. In order to assess the performance of the imputation measures, we consider a modified version of the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). The majority of the simulated examples we consider contain changes in variability over time; this volatility affects the standard RMSE and MAE and makes it difficult to directly compare results for slowly and rapidly evolving dependence. For this reason, we scale the results over time using the true standard deviation.

Let $\sigma_{t,P}$ denote the true standard deviation of the signal at time $t$ for channel $P$. The calculations of the modified RMSE and MAE only include the predicted values at the missing time points and not the full time series. Let $N$ denote the total number of missing values across all channels and timepoints, $t_{\text{mis}} = \{t_1, t_2, \cdots\}$ contains the timepoints where observations are missing and $P_{\text{mis}} = \{P_{t_1}, P_{t_2}, \cdots\}$ denotes the corresponding channels which are affected. Then the modified RMSE can be defined by

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{b \in P_{\text{mis}}} \sum_{s \in t_{\text{mis}}} \frac{(X^b_{s,T} - \hat{X}^b_{s,T})^2}{\sigma^2_{s,b}}}, \tag{4.4.1}$$

and the modified MAE is given by

$$\text{MAE} = \frac{1}{N} \sum_{b \in P_{\text{mis}}} \sum_{s \in t_{\text{mis}}} \frac{|X^b_{s,T} - \hat{X}^b_{s,T}|}{\sigma_{s,b}}. \tag{4.4.2}$$

In addition to this, we also rank the imputation methods based on the modified RMSE and MAE results. For each of the 100 replications carried out, we track which

of the imputation methods gives the lowest scaled RMSE and MAE and sum the results.

The imputation results for each of the examples described above (as in Figure 4.4.1) can be seen in Tables 4.4.1-4.4.4. For each example, we consider 10%, 20%, 30% and 40% missingness at random as well as chunks of 20 consecutive time points missing and bursts of missingness up to length 20. A description of how the bursts of missingness are generated can be found in Section 4.7. Note that we include the results for the situation where the missingness occurs in all channels simultaneously. In each case, we record the modified RMSE, modified MAE and rankings over the 100 replications based on these errors (as described above); the numbers within the brackets represent the standard deviation of these quantities.

When we consider rapidly evolving dependence within the time varying vector autoregressive moving average setting (Table 4.4.2), it can be seen that the mvL-SWimpute method performs well both in terms of the modified error measures and the rankings when percentages of the data are missing at random. On the other hand, the competitor methods which rely on the assumption of an underlying stationary model cannot cope with the changing dependence structure. Note that the addition of a backcasting step into the vector autoregressive prediction approach (VAR-fb) provides an improvement in performance. However, it can be seen that the results weaken when we look at more extreme missingness scenarios such as bursts or chunks missing. When imputing missing values in areas of a signal where consecutive time points are missing, all methods can struggle to accurately reconstruct the dependence behaviour within these areas; our proposed method still gives mild improvements in

Table 4.4.1: Performance of the imputation methods over 100 replications of mvLSW process with slowly evolving dependence for different missingness scenarios occurring simultaneously across all channels, using the evaluation measures described in the text. Numbers in brackets represent the standard deviation of estimation errors. Bold numbers indicate best result.

| | 10% | 20% | 30% | 40% | Chunks | Bursts |
|---|---|---|---|---|---|---|
| Method | Scaled by true standard deviation - RMSE | | | | | |
| mvLSWimpute-f | 0.88 (0.06) | 0.90 (0.05) | 0.92 (0.04) | 0.94 (0.04) | 1.00 (0.14) | 0.92 (0.08) |
| mvLSWimpute-fb | **0.81 (0.06)** | **0.85 (0.04)** | **0.87 (0.04)** | **0.89 (0.04)** | **1.00 (0.14)** | **0.88 (0.08)** |
| mtsdi | 0.93 (0.06) | 0.95 (0.05) | 0.97 (0.04) | 1.01 (0.07) | 1.15 (0.22) | 0.97 (0.09) |
| Amelia | 1.10 (0.07) | 1.10 (0.06) | 1.11 (0.05) | 1.09 (0.04) | 1.11 (0.15) | 1.09 (0.09) |
| VAR-f | 0.92 (0.06) | 0.94 (0.05) | 0.95 (0.04) | 0.97 (0.04) | 1.03 (0.15) | 0.95 (0.08) |
| VAR-fb | 0.86 (0.06) | 0.88 (0.05) | 0.90 (0.04) | 0.92 (0.04) | 1.02 (0.15) | 0.89 (0.08) |
| PCA | 1.00 (0.07) | 1.00 (0.05) | 1.00 (0.04) | 0.99 (0.04) | 1.02 (0.14) | 0.99 (0.08) |
| Random forest | 1.06 (0.09) | 1.06 (0.09) | 1.06 (0.07) | 1.04 (0.05) | 1.08 (0.15) | 1.04 (0.09) |
| Method | Scaled by true standard deviation - MAE | | | | | |
| mvLSWimpute-f | 0.70 (0.05) | 0.72 (0.04) | 0.73 (0.03) | 0.75 (0.03) | 0.81 (0.12) | 0.73 (0.06) |
| mvLSWimpute-fb | **0.65 (0.05)** | **0.67 (0.04)** | **0.69 (0.03)** | **0.71 (0.03)** | **0.80 (0.12)** | **0.70 (0.06)** |
| mtsdi | 0.75 (0.06) | 0.76 (0.04) | 0.77 (0.03) | 0.80 (0.04) | 0.93 (0.18) | 0.77 (0.07) |
| Amelia | 0.88 (0.06) | 0.88 (0.05) | 0.88 (0.04) | 0.87 (0.03) | 0.89 (0.12) | 0.87 (0.07) |
| VAR-f | 0.73 (0.05) | 0.75 (0.04) | 0.76 (0.03) | 0.77 (0.03) | 0.83 (0.13) | 0.75 (0.06) |
| VAR-fb | 0.68 (0.05) | 0.70 (0.04) | 0.71 (0.03) | 0.73 (0.03) | 0.82 (0.13) | 0.71 (0.06) |
| PCA | 0.80 (0.06) | 0.80 (0.04) | 0.81 (0.04) | 0.79 (0.03) | 0.82 (0.12) | 0.79 (0.06) |
| Random forest | 0.85 (0.07) | 0.85 (0.07) | 0.85 (0.05) | 0.83 (0.04) | 0.87 (0.13) | 0.84 (0.07) |
| Method | Ranking - RMSE | | | | | |
| mvLSWimpute-f | 3 | 1 | 0 | 0 | 29 | 10 |
| mvLSWimpute-fb | **90** | **91** | **97** | **98** | **32** | **67** |
| mtsdi | 1 | 0 | 0 | 0 | 7 | 4 |
| Amelia | 0 | 0 | 0 | 0 | 2 | 0 |
| VAR-f | 0 | 0 | 0 | 0 | 11 | 0 |
| VAR-fb | 6 | 8 | 3 | 2 | 12 | 19 |
| PCA | 0 | 0 | 0 | 0 | 3 | 0 |
| Random forest | 0 | 0 | 0 | 0 | 4 | 0 |
| Method | Ranking - MAE | | | | | |
| mvLSWimpute-f | 5 | 2 | 1 | 1 | 24 | 10 |
| mvLSWimpute-fb | **88** | **90** | **92** | **89** | **31** | **66** |
| mtsdi | 0 | 0 | 0 | 0 | 10 | 4 |
| Amelia | 0 | 0 | 0 | 0 | 3 | 0 |
| VAR-f | 1 | 0 | 0 | 0 | 11 | 0 |
| VAR-fb | 6 | 8 | 7 | 10 | 11 | 20 |
| PCA | 0 | 0 | 0 | 0 | 2 | 0 |
| Random forest | 0 | 0 | 0 | 0 | 8 | 0 |

Table 4.4.2: Performance of the imputation methods over 100 replications of vector moving average, autoregressive series with rapidly changing dependence structure for different missingness scenarios occurring simultaneously across all channels, using the evaluation measures described in the text. Numbers in brackets represent the standard deviation of estimation errors. Bold numbers indicate best result.

| | 10% | 20% | 30% | 40% | Chunks | Bursts |
|---|---|---|---|---|---|---|
| Method | Scaled by true standard deviation - RMSE | | | | | |
| mvLSWimpute-f | 1.67 (0.13) | 1.76 (0.10) | 1.80 (0.10) | 1.88 (0.10) | 2.02 (0.52) | 1.81 (0.16) |
| mvLSWimpute-fb | **1.45 (0.11)** | **1.54 (0.09)** | **1.60 (0.09)** | **1.70 (0.09)** | **2.00 (0.52)** | **1.62 (0.14)** |
| mtsdi | 1.69 (0.12) | 1.74 (0.11) | 1.78 (0.10) | 1.89 (0.11) | 2.33 (0.64) | 1.82 (0.18) |
| Amelia | 2.40 (0.19) | 2.42 (0.14) | 2.41 (0.14) | 2.44 (0.14) | 2.36 (0.51) | 2.37 (0.22) |
| VAR-f | 1.75 (0.15) | 1.84 (0.11) | 1.90 (0.12) | 1.98 (0.11) | 2.05 (0.53) | 1.84 (0.16) |
| VAR-fb | 1.54 (0.12) | 1.65 (0.09) | 1.71 (0.10) | 1.82 (0.09) | 2.02 (0.52) | 1.68 (0.16) |
| PCA | 2.10 (0.18) | 2.12 (0.13) | 2.11 (0.13) | 2.15 (0.13) | 2.07 (0.53) | 2.07 (0.20) |
| Random forest | 2.19 (0.21) | 2.19 (0.14) | 2.21 (0.15) | 2.26 (0.18) | 2.15 (0.54) | 2.16 (0.22) |
| Method | Scaled by true standard deviation - MAE | | | | | |
| mvLSWimpute-f | 1.32 (0.10) | 1.39 (0.08) | 1.42 (0.08) | 1.48 (0.07) | 1.61 (0.43) | 1.42 (0.13) |
| mvLSWimpute-fb | **1.14 (0.09)** | **1.22 (0.07)** | **1.26 (0.07)** | **1.34 (0.07)** | **1.59 (0.43)** | **1.27 (0.11)** |
| mtsdi | 1.34 (0.10) | 1.37 (0.09) | 1.41 (0.07) | 1.48 (0.08) | 1.86 (0.52) | 1.44 (0.14) |
| Amelia | 1.89 (0.15) | 1.91 (0.11) | 1.91 (0.11) | 1.92 (0.11) | 1.90 (0.42) | 1.87 (0.17) |
| VAR-f | 1.38 (0.11) | 1.45 (0.09) | 1.50 (0.09) | 1.56 (0.08) | 1.64 (0.44) | 1.46 (0.13) |
| VAR-fb | 1.21 (0.09) | 1.30 (0.07) | 1.35 (0.08) | 1.43 (0.07) | 1.62 (0.43) | 1.33 (0.13) |
| PCA | 1.65 (0.14) | 1.67 (0.10) | 1.66 (0.10) | 1.68 (0.10) | 1.66 (0.44) | 1.64 (0.16) |
| Random forest | 1.72 (0.16) | 1.72 (0.11) | 1.74 (0.12) | 1.78 (0.15) | 1.73 (0.44) | 1.71 (0.17) |
| Method | Ranking - RMSE | | | | | |
| mvLSWimpute-f | 0 | 0 | 0 | 0 | 19 | 0 |
| mvLSWimpute-fb | **92** | **95** | **99** | **97** | **27** | **80** |
| mtsdi | 0 | 0 | 0 | 1 | 3 | 2 |
| Amelia | 0 | 0 | 0 | 0 | 2 | 0 |
| VAR-f | 0 | 0 | 0 | 0 | 11 | 0 |
| VAR-fb | 8 | 5 | 1 | 2 | 17 | 18 |
| PCA | 0 | 0 | 0 | 0 | 7 | 0 |
| Random forest | 0 | 0 | 0 | 0 | 14 | 0 |
| Method | Ranking - MAE | | | | | |
| mvLSWimpute-f | 1 | 0 | 0 | 0 | 20 | 1 |
| mvLSWimpute-fb | **90** | **98** | **98** | **99** | **28** | **77** |
| mtsdi | 0 | 0 | 0 | 1 | 5 | 3 |
| Amelia | 0 | 0 | 0 | 0 | 3 | 0 |
| VAR-f | 0 | 0 | 0 | 0 | 9 | 0 |
| VAR-fb | 9 | 2 | 2 | 0 | 15 | 19 |
| PCA | 0 | 0 | 0 | 0 | 7 | 0 |
| Random forest | 0 | 0 | 0 | 0 | 13 | 0 |

Table 4.4.3: Performance of the imputation methods over 100 replications of vector autoregressive series with slowly evolving dependence structure for different missingness scenarios occurring simultaneously across all channels, using the evaluation measures described in the text. Numbers in brackets represent the standard deviation of estimation errors. Bold numbers indicate best result.

| | 10% | 20% | 30% | 40% | Chunks | Bursts |
|---|---|---|---|---|---|---|
| Method | Scaled by true standard deviation - RMSE | | | | | |
| mvLSWimpute-f | 1.10 (0.08) | 1.13 (0.05) | 1.14 (0.05) | 1.16 (0.04) | 1.14 (0.13) | 1.14 (0.08) |
| mvLSWimpute-fb | **1.04 (0.07)** | **1.07 (0.05)** | **1.08 (0.05)** | **1.11 (0.04)** | **1.13 (0.13)** | **1.09 (0.07)** |
| mtsdi | 1.14 (0.08) | 1.17 (0.05) | 1.18 (0.05) | 1.19 (0.05) | 1.21 (0.18) | 1.17 (0.09) |
| Amelia | 1.34 (0.09) | 1.36 (0.07) | 1.36 (0.07) | 1.37 (0.05) | 1.35 (0.25) | 1.34 (0.09) |
| VAR-f | 1.11 (0.09) | 1.15 (0.05) | 1.15 (0.06) | 1.17 (0.04) | 1.15 (0.14) | 1.14 (0.08) |
| VAR-fb | 1.06 (0.08) | 1.09 (0.05) | 1.10 (0.05) | 1.13 (0.04) | 1.15 (0.20) | 1.10 (0.08) |
| PCA | 1.16 (0.08) | 1.17 (0.05) | 1.16 (0.06) | 1.17 (0.04) | 1.17 (0.18) | 1.17 (0.08) |
| Random forest | 1.24 (0.12) | 1.26 (0.12) | 1.27 (0.11) | 1.27 (0.10) | 1.21 (0.17) | 1.25 (0.12) |
| Method | Scaled by true standard deviation - MAE | | | | | |
| mvLSWimpute-f | 0.88 (0.07) | 0.90 (0.04) | 0.91 (0.04) | 0.92 (0.03) | 0.94 (0.14) | 0.90 (0.06) |
| mvLSWimpute-fb | **0.82 (0.06)** | **0.85 (0.04)** | **0.86 (0.04)** | **0.88 (0.03)** | **0.93 (0.14)** | **0.86 (0.07)** |
| mtsdi | 0.91 (0.06) | 0.93 (0.04) | 0.94 (0.04) | 0.95 (0.04) | 0.97 (0.14) | 0.93 (0.07) |
| Amelia | 1.06 (0.07) | 1.08 (0.05) | 1.08 (0.04) | 1.08 (0.04) | 1.08 (0.20) | 1.06 (0.08) |
| VAR-f | 0.88 (0.07) | 0.91 (0.05) | 0.92 (0.05) | 0.93 (0.03) | 0.95 (0.15) | 0.91 (0.06) |
| VAR-fb | 0.84 (0.06) | 0.87 (0.04) | 0.87 (0.04) | 0.89 (0.03) | 0.94 (0.15) | 0.87 (0.07) |
| PCA | 0.92 (0.06) | 0.94 (0.04) | 0.93 (0.04) | 0.94 (0.03) | 0.95 (0.15) | 0.93 (0.07) |
| Random forest | 0.99 (0.10) | 1.00 (0.09) | 1.01 (0.08) | 1.01 (0.08) | 0.99 (0.20) | 0.99 (0.10) |
| Method | Ranking - RMSE | | | | | |
| mvLSWimpute-f | 0 | 0 | 0 | 0 | **22** | 2 |
| mvLSWimpute-fb | **68** | **66** | **72** | **73** | 17 | **57** |
| mtsdi | 0 | 0 | 0 | 0 | 12 | 1 |
| Amelia | 0 | 0 | 0 | 0 | 2 | 0 |
| VAR-f | 0 | 0 | 0 | 0 | 14 | 1 |
| VAR-fb | 32 | 34 | 28 | 27 | 21 | 38 |
| PCA | 0 | 0 | 0 | 0 | 3 | 1 |
| Random forest | 0 | 0 | 0 | 0 | 9 | 0 |
| Method | Ranking - MAE | | | | | |
| mvLSWimpute-f | 1 | 0 | 0 | 1 | **22** | 3 |
| mvLSWimpute-fb | **63** | **64** | **67** | **67** | 17 | **55** |
| mtsdi | 0 | 0 | 0 | 0 | 11 | 0 |
| Amelia | 0 | 0 | 0 | 0 | 2 | 0 |
| VAR-f | 1 | 0 | 0 | 0 | 14 | 2 |
| VAR-fb | 35 | 36 | 33 | 32 | 20 | 39 |
| PCA | 0 | 0 | 0 | 0 | 3 | 1 |
| Random forest | 0 | 0 | 0 | 0 | 11 | 0 |

Table 4.4.4: Performance of the imputation methods over 100 replications of stationary vector moving average series for different missingness scenarios occurring simultaneously across all channels, using the evaluation measures described in the text. Numbers in brackets represent the standard deviation of estimation errors. Bold numbers indicate best result.

| | 10% | 20% | 30% | 40% | Chunks | Bursts |
|---|---|---|---|---|---|---|
| Method | Scaled by true standard deviation - RMSE | | | | | |
| mvLSWimpute-f | 1.81 (0.12) | 1.89 (0.09) | 1.94 (0.07) | 2.00 (0.07) | 2.14 (0.26) | 1.97 (0.15) |
| mvLSWimpute-fb | **1.63 (0.11)** | **1.71 (0.09)** | **1.77 (0.06)** | **1.83 (0.07)** | **2.13 (0.25)** | **1.81 (0.14)** |
| mtsdi | 1.71 (0.11) | 1.79 (0.10) | 1.89 (0.10) | 2.02 (0.12) | 2.33 (0.31) | 1.94 (0.19) |
| Amelia | 2.36 (0.15) | 2.41 (0.11) | 2.42 (0.10) | 2.42 (0.10) | 2.41 (0.26) | 2.43 (0.18) |
| VAR-f | 1.83 (0.12) | 1.91 (0.10) | 1.96 (0.08) | 2.03 (0.08) | 2.19 (0.26) | 2.01 (0.16) |
| VAR-fb | 1.66 (0.11) | 1.75 (0.09) | 1.81 (0.07) | 1.89 (0.08) | 2.16 (0.26) | 1.85 (0.14) |
| PCA | 2.15 (0.14) | 2.21 (0.10) | 2.21 (0.08) | 2.22 (0.10) | 2.20 (0.25) | 2.20 (0.16) |
| Random forest | 2.30 (0.19) | 2.37 (0.18) | 2.35 (0.17) | 2.37 (0.16) | 2.39 (0.35) | 2.35 (0.23) |
| Method | Scaled by true standard deviation - MAE | | | | | |
| mvLSWimpute-f | 1.44 (0.10) | 1.51 (0.08) | 1.54 (0.06) | 1.59 (0.06) | 1.73 (0.23) | 1.56 (0.11) |
| mvLSWimpute-fb | **1.29 (0.09)** | **1.36 (0.07)** | **1.40 (0.05)** | **1.45 (0.06)** | **1.72 (0.22)** | **1.43 (0.11)** |
| mtsdi | 1.36 (0.09) | 1.43 (0.08) | 1.49 (0.07) | 1.58 (0.08) | 1.86 (0.25) | 1.52 (0.14) |
| Amelia | 1.89 (0.13) | 1.93 (0.09) | 1.93 (0.07) | 1.93 (0.09) | 1.94 (0.24) | 1.94 (0.16) |
| VAR-f | 1.45 (0.10) | 1.52 (0.08) | 1.56 (0.06) | 1.61 (0.07) | 1.78 (0.23) | 1.59 (0.12) |
| VAR-fb | 1.32 (0.09) | 1.39 (0.07) | 1.44 (0.05) | 1.50 (0.07) | 1.76 (0.23) | 1.47 (0.11) |
| PCA | 1.72 (0.12) | 1.76 (0.09) | 1.76 (0.06) | 1.77 (0.09) | 1.79 (0.23) | 1.76 (0.14) |
| Random forest | 1.84 (0.16) | 1.90 (0.15) | 1.88 (0.14) | 1.90 (0.13) | 1.95 (0.31) | 1.88 (0.20) |
| Method | Ranking - RMSE | | | | | |
| mvLSWimpute-f | 1 | 0 | 0 | 0 | 26 | 1 |
| mvLSWimpute-fb | **67** | **79** | **84** | **95** | **31** | **71** |
| mtsdi | 12 | 8 | 6 | 0 | 11 | 14 |
| Amelia | 0 | 0 | 0 | 0 | 0 | 0 |
| VAR-f | 0 | 0 | 0 | 0 | 5 | 1 |
| VAR-fb | 20 | 13 | 10 | 5 | 13 | 13 |
| PCA | 0 | 0 | 0 | 0 | 3 | 0 |
| Random forest | 0 | 0 | 0 | 0 | 11 | 0 |
| Method | Ranking - MAE | | | | | |
| mvLSWimpute-f | 1 | 0 | 0 | 0 | 25 | 1 |
| mvLSWimpute | **63** | **78** | **90** | **92** | **30** | **68** |
| mtsdi | 12 | 9 | 5 | 4 | 10 | 17 |
| Amelia | 0 | 0 | 0 | 0 | 0 | 0 |
| VAR-f | 0 | 0 | 0 | 0 | 10 | 0 |
| VAR-fb | 24 | 13 | 5 | 4 | 10 | 14 |
| PCA | 0 | 0 | 0 | 0 | 4 | 0 |
| Random forest | 0 | 0 | 0 | 0 | 11 | 0 |

this challenging scenario.

When we consider the stationary moving average process (Table 4.4.4), it can be seen that the mvLSWimpute approach again produces more accurate results followed by both VAR-fb and mtsdi. Despite the competitor methods being designed for imputation within stationary time series, the mvLSWimpute method outperforms them both in terms of the modified error measures and the rankings.

As expected, for the mvLSW process exhibiting slowly varying spectral structure (Table 4.4.1), the mvLSWimpute method performs strongly across all evaluation measures. For the slowly evolving vector autoregressive series (Table 4.4.3), the mvLSWimpute method consistently performs better than the competitors. However, the VAR-fb approach also performs well in this setting due to the underlying model used for the one step ahead predictions being designed for this scenario. In this case, the results produced are comparable to mvLSWimpute in terms of the modified RMSE and MAE.

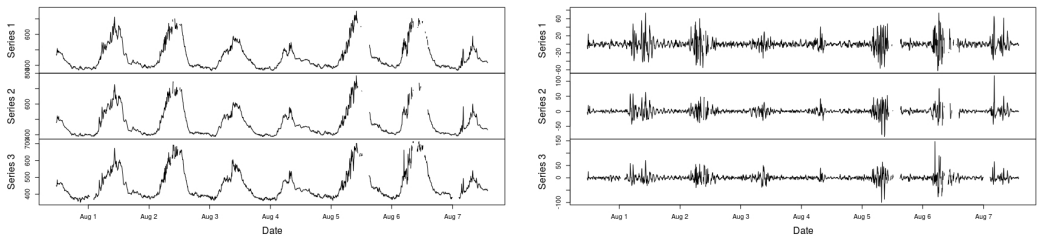
Note that in nearly all cases across the examples, mvLSWimpute performs consistently well in terms of the error measures considered, despite some of the competitor methods being designed specifically for imputation within those scenarios. We also note that the use of a backcasting step within both the mvLSW and VAR imputation methods improves their performance, justifying its inclusion. For the scenarios discussed above, missingness in one channel only was also considered and similar performance was observed. The full results for missingness in one channel only are included in Section 4.8.

## 4.5 Case study

In the previous section, we evaluated the performance of our multivariate imputation method against a range of alternatives for a variety of simulated scenarios. We now consider an application related to the use of atmospheric monitoring techniques within the oil and gas industry.

There has been an increased interest in Carbon Capture and Storage (CCS) projects over recent years due to the environmental benefits that such operations can bring (Bachu, 2008; Leung et al., 2014). However, the associated climate benefits are highly dependent on the efficient containment of the injected gases. For this reason, there has been a particular focus on developing reliable atmospheric monitoring techniques to detect and locate $CO_2$ leakages within CCS sites. Current approaches include methods based on atmospheric tomography (Jenkins et al., 2011; Levine et al., 2016), Lagrangian particle dispersion models (Luhar et al., 2014) and Gaussian plume dispersion models (Hirst et al., 2017). Any attempts to detect such leaks can be hindered by the presence of missingness within the data as missing entries must either be removed or replaced with suitable estimates before further analysis can take place.

Within this section, we focus on the problem of imputing missing values in a multivariate time series arising from a CCS project using data provided by our industrial collaborator. We consider a trivariate signal of length 2048 corresponding to approximately one week of measurements; Figure 4.5.1(a) shows the $CO_2$ concentrations over time for three sensors. The signal exhibits a range of missingness, including data missing both at random or for consecutive time points across one or more sensors. The



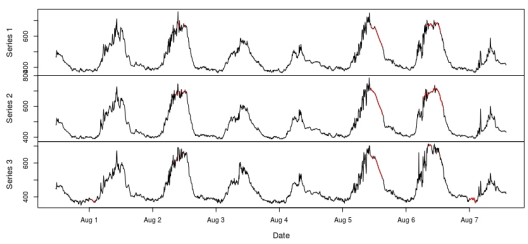

(a) (b)

Figure 4.5.1: Time series plots of the $CO_2$ concentration for three sensors over the same time period: (a) original series; (b) detrended series.

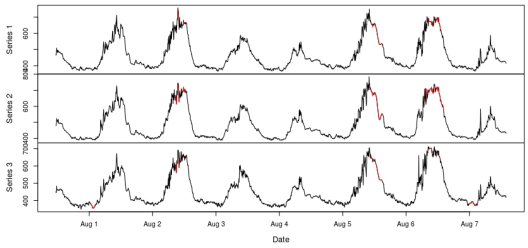
total number of missing values across the signal corresponds to approximately 6.5%. Due to the zero-mean assumption of the mvLSW model, before analysis we detrend the series by fitting a smoothing spline to each of the components and considering the residuals, see Figure 4.5.1(b).

It is important to note that after removing the trend effect and imputing missing values in the now zero-mean process, the trend must then be introduced back into the series in some way to ensure direct comparisons can be made between the original and imputed time series. However, this means that we must be aware that any interesting behaviour within an imputed time series may be a feature of the imputation method or could be due to the trend being added back into the series.

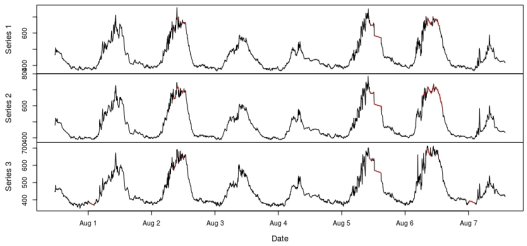
We apply the mvLSW-based imputation approach with $p = 20$ points considered in the clipped predictor for both the forecasting and backcasting steps. For comparison, we apply the mtsdi method and the VAR-fb approach as, of the existing methods, these performed better in the simulation study. Since the true values of the test signal are unknown at the missing time points, we compare the results visually. The




(a) MvLSWimpute-fb



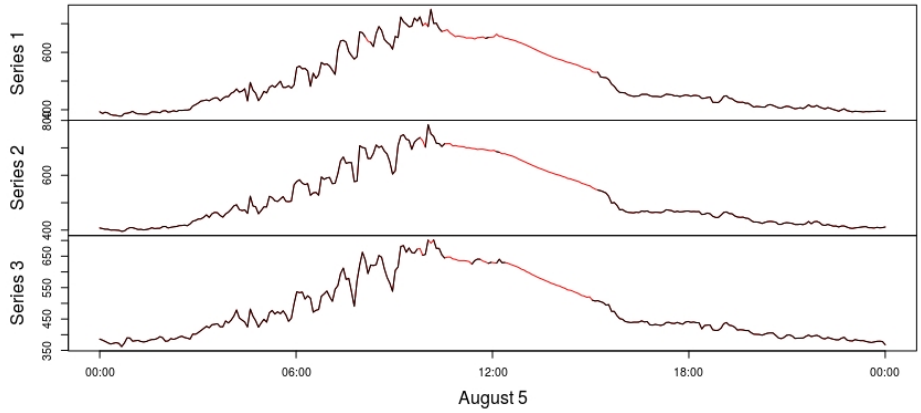

(b) mtsdi



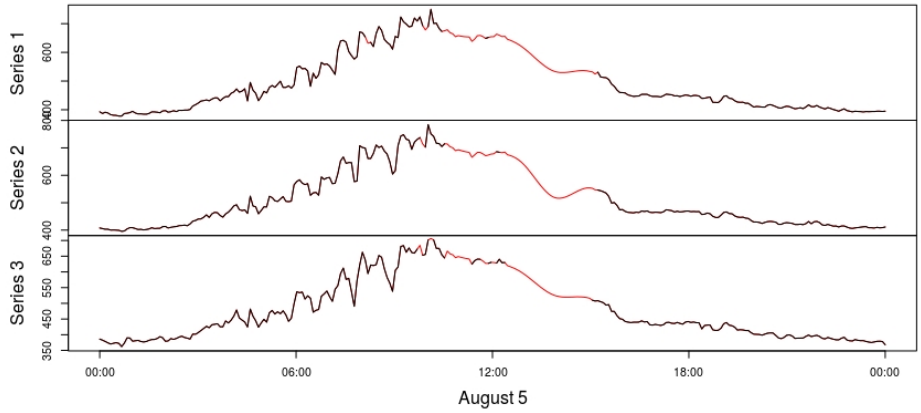

(c) VAR-fb

Figure 4.5.2: Imputation results obtained from applying mvLSWimpute-fb, mtsdi and VAR-fb approaches to $CO_2$ data, imputed values are shown in red.

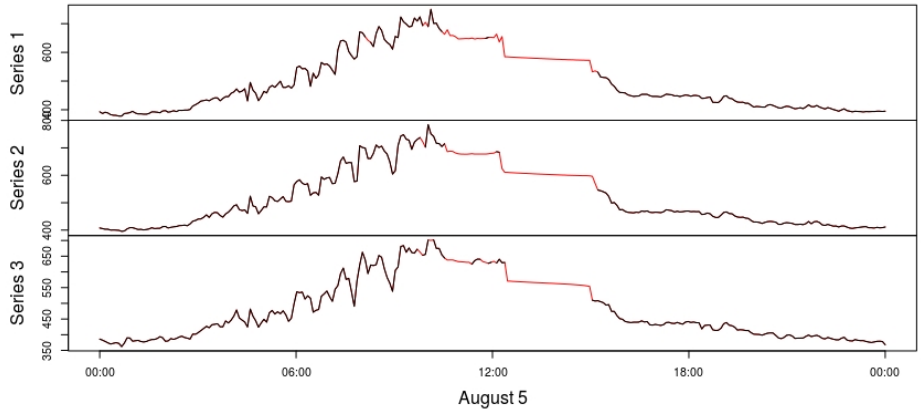
imputation results for each of the methods are found in Figure 4.5.2; imputed values are shown in red. It can be seen that, whilst the imputation results for all three methods are quite similar, the mvLSWimpute method produces the most reliable estimate for the missing data between August 5 and 6 when compared to the daily behaviour over the rest of the week. Figure 4.5.3 shows the imputation results for each of the methods, focusing on August 5 only. As the background concentration of $CO_2$ within the atmosphere naturally varies over the course of a day, it would be unlikely that we would see any features as imputed by mtsdi and VAR-fb (Figures 4.5.3(b) and 4.5.3(c)) where the concentration suddenly changes over a short period of time.




(a) MvLSWimpute-fb




(b) mtsdi




(c) VAR-fb

Figure 4.5.3: Imputation results for August 5 only, imputed values are shown in red.

Recall that the overall aim of atmospheric monitoring within CCS regions is to be able to detect anomalous regions that could indicate releases of $CO_2$. It is therefore important to be able to replace missing values with reasonable estimates which will then allow further analysis to be carried out. Our mvLSW imputation approach can be used as a first step to infill any missing values before attempting to detect anomalous regions or other secondary analysis tasks of interest.

## 4.6 Concluding remarks

In this work, we have introduced a wavelet-based imputation method that can be used to replace missing values within a multivariate, nonstationary time series. We compared the performance of our method against existing imputation approaches using simulated data examples and a dataset from a Carbon Capture and Storage facility. The simulated data examples demonstrate that the use of a backcasting step within imputation can improve the performance of the prediction methods. The case study shows that our method can be used to successfully impute missing values within time series containing both nonstationarity and seasonality, resulting in a more reliable imputation estimate compared to existing approaches.

In practice, we have found that, as with other competitor methods, the performance of our approach suffers when we have extreme scenarios such as chunks of consecutive time points missing or bursts of missingness. An avenue for future research could be to look at ways in which we could improve the imputation results for these cases.

## 4.7 Bursts of missingness

For all the simulations described in Section 4.4, we generate 5 bursts of missingness that have maximum length $l = 20$ and are a minimum of $d = 70$ timepoints apart. First, we randomly sample 50% of the indices without replacement which form the set of candidate locations for missing data. Next, we select the start location for the first burst by sampling an index $k$ between 30 and 70. The start locations $s$ of the bursts are then defined by $k + ln + dn$ where $n \in \{1, \ldots, 5\}$ and the end locations are determined by $s + l$. For each of the bursts, the missing indices are chosen by selecting the candidate locations (determined in Step 1) that are between the start and end point of the burst. An example time series containing bursts of missingness can be seen in Figure 4.7.1, the filled black circles represent the locations of the missing values.

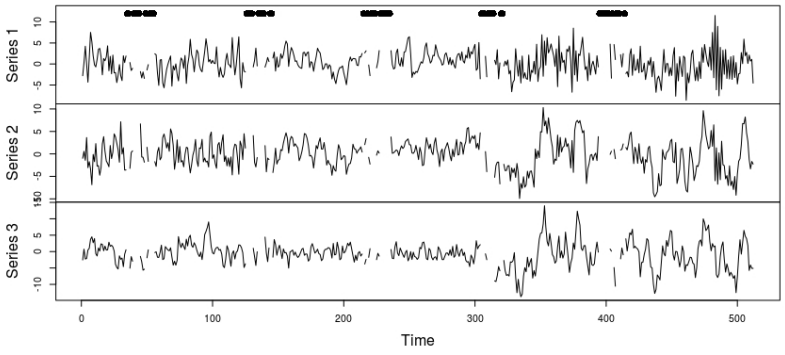



Figure 4.7.1: Example time series containing bursts of missingness.

## 4.8 Simulated performance of mvLSWimpute - additional tables

In this section, we present the results for the simulated examples considered in Section 4.4 where the missingness occurs across one channel only. Specifically, we include the imputation results for slowly evolving mvLSW and vector autoregressive series in Tables 4.8.1 and 4.8.3 respectively. Table 4.8.2 contains the results for vector moving average, autoregressive series with rapidly changing dependence structure and Table 4.8.4 contains the results for stationary vector moving average series with missingness in one channel only.

Similarly to the results included in Section 4.4, we can see from Tables 4.8.1-4.8.4 that the mvLSWimpute method performs consistently well in terms of the error measures across all cases. Note that the missingness mechanism appears to affect the results more than the underlying time series model. As in Section 4.4, it is clear that all methods struggle to accurately impute missing values when there are bursts of missingness or chunks of consecutive time points missing.

Table 4.8.1: Performance of the imputation methods over 100 replications of mvLSW process with slowly evolving dependence for different missingness scenarios occurring across one channel, using the evaluation measures described in the text. Numbers in brackets represent the standard deviation of estimation errors. Bold numbers indicate best result.

| | 10% | 20% | 30% | 40% | Chunks | Bursts |
|---|---|---|---|---|---|---|
| Method | Scaled by true standard deviation - RMSE | | | | | |
| mvLSWforecast | 0.84 (0.08) | 0.86 (0.06) | 0.87 (0.05) | 0.87 (0.05) | 0.90 (0.17) | 0.90 (0.09) |
| mvLSWimpute | **0.78 (0.08)** | **0.80 (0.05)** | **0.81 (0.05)** | **0.82 (0.04)** | **0.88 (0.18)** | **0.83 (0.08)** |
| mtsdi | 0.81 (0.09) | 0.83 (0.06) | 0.82 (0.05) | 0.84 (0.06) | 0.93 (0.27) | **0.83 (0.09)** |
| Amelia | 1.01 (0.10) | 1.04 (0.07) | 1.02 (0.07) | 1.03 (0.06) | 1.03 (0.23) | 1.06 (0.12) |
| VARforecast | 0.90 (0.09) | 0.91 (0.06) | 0.91 (0.06) | 0.93 (0.05) | 0.99 (0.23) | 0.94 (0.09) |
| VARpred | 0.83 (0.08) | 0.85 (0.06) | 0.85 (0.05) | 0.87 (0.05) | 0.94 (0.20) | 0.87 (0.09) |
| PCA | 0.93 (0.10) | 0.95 (0.06) | 0.94 (0.07) | 0.94 (0.06) | 0.92 (0.23) | 0.97 (0.12) |
| Random forest | 0.99 (0.11) | 1.01 (0.07) | 1.00 (0.07) | 1.01 (0.07) | 1.01 (0.24) | 1.03 (0.12) |
| Method | Scaled by true standard deviation - MAE | | | | | |
| mvLSWforecast | 0.67 (0.07) | 0.69 (0.05) | 0.69 (0.05) | 0.70 (0.04) | 0.73 (0.14) | 0.72 (0.08) |
| mvLSWimpute | **0.63 (0.07)** | **0.64 (0.05)** | **0.64 (0.04)** | **0.65 (0.04)** | **0.71 (0.15)** | **0.66 (0.07)** |
| mtsdi | 0.65 (0.07) | 0.65 (0.05) | 0.65 (0.04) | 0.67 (0.05) | 0.75 (0.23) | **0.66 (0.07)** |
| Amelia | 0.81 (0.08) | 0.82 (0.06) | 0.81 (0.06) | 0.82 (0.05) | 0.84 (0.20) | 0.85 (0.11) |
| VARforecast | 0.72 (0.07) | 0.72 (0.05) | 0.73 (0.05) | 0.74 (0.05) | 0.80 (0.20) | 0.75 (0.08) |
| VARpred | 0.66 (0.07) | 0.67 (0.05) | 0.67 (0.04) | 0.69 (0.04) | 0.76 (0.17) | 0.69 (0.07) |
| PCA | 0.75 (0.08) | 0.75 (0.05) | 0.74 (0.06) | 0.75 (0.05) | 0.75 (0.20) | 0.77 (0.10) |
| Random forest | 0.79 (0.10) | 0.80 (0.05) | 0.79 (0.06) | 0.80 (0.06) | 0.81 (0.20) | 0.82 (0.10) |
| Method | Ranking - RMSE | | | | | |
| mvLSWforecast | 7 | 3 | 1 | 1 | 7 | 4 |
| mvLSWimpute | **58** | **60** | **66** | **65** | 21 | 37 |
| mtsdi | 25 | 31 | 32 | 32 | **25** | **42** |
| Amelia | 0 | 0 | 0 | 0 | 4 | 0 |
| VARforecast | 1 | 0 | 0 | 0 | 8 | 1 |
| VARpred | 9 | 6 | 1 | 2 | 16 | 12 |
| PCA | 0 | 0 | 0 | 0 | 15 | 3 |
| Random forest | 1 | 0 | 0 | 0 | 4 | 1 |
| Method | Ranking - MAE | | | | | |
| mvLSWforecast | 11 | 3 | 0 | 3 | 11 | 6 |
| mvLSWimpute | **47** | **50** | **60** | **60** | 22 | 35 |
| mtsdi | 30 | 35 | 35 | 35 | **26** | **43** |
| Amelia | 0 | 0 | 0 | 0 | 3 | 0 |
| VARforecast | 0 | 0 | 0 | 0 | 7 | 2 |
| VARpred | 10 | 12 | 5 | 2 | 12 | 12 |
| PCA | 1 | 0 | 0 | 0 | 13 | 2 |
| Random forest | 1 | 0 | 0 | 0 | 6 | 0 |

Table 4.8.2: Performance of the imputation methods over 100 replications of vector moving average, autoregressive series with rapidly changing dependence structure for different missingness scenarios occurring across one channel, using the evaluation measures described in the text. Numbers in brackets represent the standard deviation of estimation errors. Bold numbers indicate best result.

| | 10% | 20% | 30% | 40% | Chunks | Bursts |
|---|---|---|---|---|---|---|
| Method | Scaled by true standard deviation - RMSE | | | | | |
| mvLSWforecast | 1.62 (0.21) | 1.63 (0.13) | 1.68 (0.11) | 1.69 (0.11) | 1.75 (0.40) | 1.61 (0.18) |
| mvLSWimpute | **1.39 (0.17)** | **1.41 (0.12)** | **1.47 (0.09)** | **1.48 (0.09)** | **1.71 (0.39)** | **1.41 (0.17)** |
| mtsdi | 1.72 (0.18) | 1.68 (0.15) | 1.74 (0.11) | 1.75 (0.12) | 2.00 (0.54) | 1.76 (0.19) |
| Amelia | 2.26 (0.26) | 2.23 (0.16) | 2.30 (0.13) | 2.27 (0.14) | 2.12 (0.52) | 2.20 (0.26) |
| VARforecast | 1.72 (0.22) | 1.73 (0.13) | 1.77 (0.11) | 1.77 (0.12) | 1.87 (0.47) | 1.70 (0.20) |
| VARpred | 1.48 (0.19) | 1.51 (0.11) | 1.56 (0.09) | 1.57 (0.10) | **1.71 (0.42)** | 1.51 (0.18) |
| PCA | 2.01 (0.21) | 2.01 (0.16) | 2.06 (0.13) | 2.04 (0.14) | 1.89 (0.44) | 1.97 (0.26) |
| Random forest | 2.08 (0.24) | 2.06 (0.17) | 2.13 (0.15) | 2.12 (0.15) | 1.98 (0.42) | 2.04 (0.28) |
| Method | Scaled by true standard deviation - MAE | | | | | |
| mvLSWforecast | 1.29 (0.17) | 1.29 (0.10) | 1.33 (0.09) | 1.34 (0.09) | 1.44 (0.34) | 1.29 (0.14) |
| mvLSWimpute | **1.10 (0.13)** | **1.12 (0.09)** | **1.16 (0.07)** | **1.17 (0.07)** | **1.40 (0.34)** | **1.13 (0.14)** |
| mtsdi | 1.37 (0.14) | 1.34 (0.12) | 1.38 (0.09) | 1.39 (0.10) | 1.65 (0.48) | 1.41 (0.16) |
| Amelia | 1.80 (0.21) | 1.79 (0.13) | 1.89 (0.10) | 1.80 (0.12) | 1.73 (0.43) | 1.76 (0.21) |
| VARforecast | 1.36 (0.17) | 1.37 (0.10) | 1.40 (0.08) | 1.40 (0.09) | 1.54 (0.41) | 1.36 (0.16) |
| VARpred | 1.17 (0.15) | 1.19 (0.08) | 1.23 (0.08) | 1.24 (0.08) | **1.40 (0.37)** | 1.19 (0.14) |
| PCA | 1.60 (0.18) | 1.61 (0.13) | 1.63 (0.10) | 1.62 (0.12) | 1.55 (0.39) | 1.58 (0.22) |
| Random forest | 1.64 (0.19) | 1.63 (0.13) | 1.67 (0.11) | 1.66 (0.12) | 1.61 (0.33) | 1.61 (0.23) |
| Method | Ranking - RMSE | | | | | |
| mvLSWforecast | 2 | 0 | 0 | 0 | 18 | 4 |
| mvLSWimpute | **73** | **91** | **90** | **89** | 19 | **78** |
| mtsdi | 1 | 0 | 0 | 0 | 7 | 0 |
| Amelia | 0 | 0 | 0 | 0 | 2 | 0 |
| VARforecast | 2 | 0 | 0 | 0 | 7 | 0 |
| VARpred | 22 | 9 | 10 | 11 | **33** | 18 |
| PCA | 0 | 0 | 0 | 0 | 5 | 0 |
| Random forest | 0 | 0 | 0 | 0 | 9 | 0 |
| Method | Ranking - MAE | | | | | |
| mvLSWforecast | 2 | 1 | 1 | 0 | 10 | 5 |
| mvLSWimpute | **74** | **90** | **89** | **86** | 14 | **76** |
| mtsdi | 1 | 0 | 0 | 0 | 6 | 0 |
| Amelia | 0 | 0 | 0 | 0 | 2 | 0 |
| VARforecast | 1 | 0 | 0 | 0 | 10 | 0 |
| VARpred | 22 | 9 | 10 | 14 | **40** | 18 |
| PCA | 0 | 0 | 0 | 0 | 6 | 0 |
| Random forest | 0 | 0 | 0 | 0 | 12 | 1 |

Table 4.8.3: Performance of the imputation methods over 100 replications of vector autoregressive series with slowly evolving dependence structure for different missingness scenarios occurring across one channel, using the evaluation measures described in the text. Numbers in brackets represent the standard deviation of estimation errors. Bold numbers indicate best result.

| | 10% | 20% | 30% | 40% | Bursts | Chunks |
|---|---|---|---|---|---|---|
| Method | Scaled by true standard deviation - RMSE | | | | | |
| mvLSWforecast | 1.09 (0.11) | 1.08 (0.08) | 1.10 (0.06) | 1.10 (0.06) | 1.12 (0.19) | 1.12 (0.11) |
| mvLSWimpute | **1.03 (0.10)** | **1.02 (0.08)** | **1.04 (0.06)** | **1.05 (0.06)** | **1.11 (0.18)** | **1.06 (0.10)** |
| mtsdi | 1.15 (0.12) | 1.13 (0.10) | 1.14 (0.07) | 1.14 (0.07) | 1.18 (0.30) | 1.16 (0.12) |
| Amelia | 1.34 (0.15) | 1.32 (0.10) | 1.32 (0.09) | 1.33 (0.08) | 1.27 (0.27) | 1.33 (0.14) |
| VARforecast | 1.14 (0.12) | 1.11 (0.09) | 1.12 (0.07) | 1.12 (0.06) | 1.15 (0.17) | 1.15 (0.11) |
| VARpred | 1.08 (0.12) | 1.05 (0.09) | 1.06 (0.06) | 1.07 (0.05) | 1.14 (0.18) | 1.08 (0.11) |
| PCA | 1.17 (0.13) | 1.14 (0.10) | 1.15 (0.07) | 1.16 (0.07) | 1.13 (0.18) | 1.16 (0.12) |
| Random forest | 1.29 (0.14) | 1.27 (0.11) | 1.29 (0.09) | 1.30 (0.08) | 1.25 (0.20) | 1.30 (0.14) |
| Method | Scaled by true standard deviation - MAE | | | | | |
| mvLSWforecast | 0.87 (0.10) | 0.86 (0.07) | 0.87 (0.05) | 0.88 (0.05) | 0.91 (0.27) | 0.89 (0.09) |
| mvLSWimpute | **0.83 (0.09)** | **0.82 (0.07)** | **0.83 (0.05)** | **0.84 (0.05)** | **0.89 (0.16)** | **0.85 (0.08)** |
| mtsdi | 0.92 (0.10) | 0.90 (0.08) | 0.91 (0.06) | 0.91 (0.05) | 0.96 (0.27) | 0.92 (0.10) |
| Amelia | 1.06 (0.13) | 1.05 (0.08) | 1.05 (0.07) | 1.04 (0.06) | 1.03 (0.21) | 1.06 (0.11) |
| VARforecast | 0.91 (0.11) | 0.89 (0.07) | 0.89 (0.05) | 0.91 (0.05) | 0.93 (0.15) | 0.92 (0.09) |
| VARpred | 0.86 (0.11) | 0.83 (0.07) | 0.84 (0.05) | 0.85 (0.04) | 0.93 (0.15) | 0.86 (0.09) |
| PCA | 0.93 (0.11) | 0.91 (0.08) | 0.92 (0.06) | 0.92 (0.06) | 0.91 (0.15) | 0.93 (0.10) |
| Random forest | 1.02 (0.12) | 1.00 (0.08) | 1.02 (0.07) | 1.03 (0.07) | 1.01 (0.18) | 1.02 (0.11) |
| Method | Ranking - RMSE | | | | | |
| mvLSWforecast | 12 | 5 | 1 | 1 | 19 | 9 |
| mvLSWimpute | **58** | **60** | **65** | **76** | **25** | **47** |
| mtsdi | 5 | 5 | 2 | 3 | 14 | 7 |
| Amelia | 1 | 0 | 0 | 0 | 7 | 1 |
| VARforecast | 3 | 0 | 0 | 0 | 5 | 3 |
| VARpred | 18 | 29 | 29 | 19 | 5 | 30 |
| PCA | 3 | 1 | 3 | 1 | 18 | 3 |
| Random forest | 0 | 0 | 0 | 0 | 7 | 0 |
| Method | Ranking - MAE | | | | | |
| mvLSWforecast | 16 | 8 | 5 | 3 | 18 | 11 |
| mvLSWimpute | **48** | **55** | **60** | **66** | **24** | **45** |
| mtsdi | 7 | 4 | 5 | 2 | 19 | 10 |
| Amelia | 0 | 0 | 0 | 0 | 10 | 0 |
| VARforecast | 4 | 4 | 0 | 2 | 5 | 3 |
| VARpred | 22 | 26 | 28 | 26 | 6 | 24 |
| PCA | 3 | 3 | 2 | 1 | 9 | 7 |
| Random forest | 0 | 0 | 0 | 0 | 9 | 0 |

Table 4.8.4: Performance of the imputation methods over 100 replications of stationary vector moving average series for different missingness scenarios occurring across one channel, using the evaluation measures described in the text. Numbers in brackets represent the standard deviation of estimation errors. Bold numbers indicate best result.

| | 10% | 20% | 30% | 40% | Chunks | Bursts |
|---|---|---|---|---|---|---|
| Method | Scaled by true standard deviation - RMSE | | | | | |
| mvLSWforecast | 1.78 (0.18) | 1.87 (0.15) | 1.94 (0.13) | 2.00 (0.12) | 2.12 (0.52) | 1.81 (0.18) |
| mvLSWimpute | **1.62 (0.17)** | **1.69 (0.13)** | **1.76 (0.11)** | **1.83 (0.11)** | **2.11 (0.52)** | **1.65 (0.17)** |
| mtsdi | 1.67 (0.18) | 1.77 (0.15) | 1.85 (0.13) | 2.00 (0.17) | 2.20 (0.69) | 1.71 (0.19) |
| Amelia | 2.28 (0.25) | 2.34 (0.17) | 2.32 (0.14) | 2.34 (0.14) | 2.32 (0.55) | 2.29 (0.22) |
| VARforecast | 1.80 (0.19) | 1.90 (0.15) | 1.97 (0.14) | 2.05 (0.13) | 2.26 (0.53) | 1.85 (0.18) |
| VARpred | 1.64 (0.18) | 1.74 (0.13) | 1.81 (0.11) | 1.89 (0.11) | 2.18 (0.52) | 1.67 (0.17) |
| PCA | 2.09 (0.24) | 2.13 (0.15) | 2.13 (0.12) | 2.15 (0.13) | **2.11 (0.55)** | 2.12 (0.21) |
| Random forest | 2.25 (0.23) | 2.30 (0.17) | 2.31 (0.13) | 2.31 (0.14) | 2.28 (0.54) | 2.28 (0.23) |
| Method | Scaled by true standard deviation - MAE | | | | | |
| mvLSWforecast | 1.42 (0.14) | 1.49 (0.13) | 1.54 (0.10) | 1.58 (0.10) | 1.71 (0.43) | 1.45 (0.15) |
| mvLSWimpute | **1.29 (0.13)** | **1.36 (0.11)** | **1.40 (0.10)** | **1.45 (0.09)** | **1.70 (0.43)** | **1.32 (0.15)** |
| mtsdi | 1.34 (0.15) | 1.41 (0.12) | 1.47 (0.10) | 1.57 (0.12) | 1.77 (0.58) | 1.38 (0.16) |
| Amelia | 1.82 (0.21) | 1.88 (0.15) | 1.85 (0.12) | 1.87 (0.12) | 1.87 (0.45) | 1.84 (0.18) |
| VARforecast | 1.45 (0.16) | 1.52 (0.13) | 1.57 (0.11) | 1.63 (1.07) | 1.83 (0.44) | 1.49 (0.15) |
| VARpred | 1.31 (0.15) | 1.39 (0.11) | 1.44 (0.10) | 1.50 (0.10) | 1.77 (0.43) | 1.35 (0.14) |
| PCA | 1.68 (0.21) | 1.71 (0.13) | 1.70 (0.11) | 1.72 (0.11) | **1.70 (0.45)** | 1.69 (0.17) |
| Random forest | 1.79 (0.21) | 1.83 (0.15) | 1.85 (0.12) | 1.85 (0.12) | 1.84 (0.44) | 1.83 (0.20) |
| Method | Ranking - RMSE | | | | | |
| mvLSWforecast | 3 | 2 | 0 | 0 | **19** | 3 |
| mvLSWimpute | **33** | **54** | **65** | **75** | 14 | **40** |
| mtsdi | 31 | 24 | 23 | 11 | **19** | 32 |
| Amelia | 0 | 0 | 0 | 0 | 5 | 0 |
| VARforecast | 2 | 0 | 1 | 0 | 7 | 3 |
| VARpred | 31 | 20 | 11 | 14 | 11 | 22 |
| PCA | 0 | 0 | 0 | 0 | 16 | 0 |
| Random forest | 0 | 0 | 0 | 0 | 9 | 0 |
| Method | Ranking - MAE | | | | | |
| mvLSWforecast | 4 | 2 | 1 | 0 | 15 | 2 |
| mvLSWimpute | **38** | **52** | **68** | **72** | 18 | **45** |
| mtsdi | 31 | 30 | 25 | 12 | **20** | 25 |
| Amelia | 0 | 0 | 0 | 0 | 8 | 0 |
| VARforecast | 3 | 0 | 0 | 0 | 11 | 4 |
| VARpred | 24 | 16 | 6 | 15 | 6 | 24 |
| PCA | 0 | 0 | 0 | 1 | 12 | 0 |
| Random forest | 0 | 0 | 0 | 0 | 10 | 0 |

# Chapter 5

# `mvLSWimpute`: An R package for imputation in multivariate locally stationary time series

## 5.1 Introduction

This chapter focuses on the implementation of the mvLSWimpute method proposed in Chapter 4 within R. We describe the package functionality and demonstrate how it can be used to replace missing values within multivariate locally stationary time series. Missing values commonly occur in real world datasets across a range of applications including environmental research (Junninen et al., 2004; Plaia and Bondi, 2006), epidemiology (Greenland and Finkle, 1995; Clark and Altman, 2003) and signal processing (Gemmeke et al., 2010; Smaragdis et al., 2011). Secondary analysis of such series, for example clustering (Tuikkala et al., 2008) and forecasting (Haworth

and Cheng, 2012), can often be hindered by the presence of missingness. With that in mind, the task of replacing these missing values accurately is an important one and depends both on the application and the structure of the dataset. This task is commonly called imputation.

For univariate time series data, any method to impute missing data must take into account the inherent temporal dependence across successive time points, see Moritz et al. (2015) for an extensive summary. Simple approaches to univariate time series imputation include replacing missing values with the mean, carrying forward the last observed value or linear interpolation. All of these are implemented in the R package **imputeTS** (Moritz, 2018) which provides the most comprehensive suite of functions for imputation within this setting. In many cases, the underlying time series containing missing data may have trend and seasonality and for this reason it can be beneficial to model the series in some way to accurately capture this behaviour. Packages available on CRAN that contain advanced time series imputation functionality and allow for imputation within such series include **zoo** (Zeileis et al., 2018) which uses seasonal Kalman filters and **forecast** (Hyndman, 2017) which employs Seasonal Trend Decomposition using Loess to replace missing values.

Turning to the multivariate setting, a variety of different imputation methods have been proposed for imputing missing values, see García-Laencina et al. (2010) for an indepth review. A range of existing R packages are dedicated to this task, some popular examples include **missMDA** (Husson and Josse, 2018), **mice** (van Buuren and Groothuis-Oudshoorn, 2018), **missForest** (Stekhoven, 2013) and **VIM** (Templ et al., 2017). These implement a selection of algorithms for multivariate imputation

within continuous, categorical and count variables including methods based on random forests (Ishwaran et al., 2008), k-nearest neighbours (Hudak et al., 2008; Li and Parker, 2014), PCA (Josse et al., 2011) and fully conditional specification models (Bouhlila and Sellaouti, 2013). In terms of multivariate time series data, there are a number of packages available on CRAN that specifically deal with imputing missing values in this context. Each of these contain model-based algorithms that use the expectation-maximization algorithm to infer missing values, this is implemented in **mtsdi** (Junger and de Leon, 2018) and **Amelia** (Honaker et al., 2015) for Gaussian distributions and in **MARSS** (Holmes et al., 2018) for Vector Autoregressive models. However, current model-based approaches only cover imputation within stationary time series where characteristics such as the mean and variance remain constant over time. Time series that arise in practice across a range of applications often exhibit changes in characteristics such as trend, variance and autocorrelation. Hence, existing imputation methods have not been designed with the requisite nonstationary structure in mind.

In this chapter we introduce the **mvLSWimpute** package. This package can be used to replace missing values in a multivariate locally stationary time series. The method extends the one-step ahead forecasting approach of Fryzlewicz et al. (2003) to the multivariate Locally Stationary Wavelet setting (Park et al., 2014); using the auto and cross-covariance structure of the time series to accurately predict missing values. Specifically, **mvLSWimpute** provides functionality for: the estimation of the Local Wavelet Spectral (LWS) matrix for a given multivariate locally stationary time series containing missing entries; and, imputation of the missing values within this setting.

The package has been developed in R and makes use of a number of functions from the **mvLSW** package (Taylor et al., 2017). Note that the mvLSWimpute approach is designed for imputation within zero-mean time series containing missing values, for this reason we must detrend the series before carrying out any analysis to ensure this assumption holds. For the data used within the case study in Section 5.3, we fit a smoothing spline to each component of the time series and consider the residuals.

This chapter is organised as follows. Within Section 5.2, we briefly review the methodology for the mvLSWimpute approach before demonstrating the R package functions through the use of a worked example. Specifically, we focus on the spectral estimation step in Section 5.2.1. One-step ahead forecasting within the mvLSW framework is discussed in Section 5.2.2. Section 5.2.3 describes the backcasting step, included to improve imputation results. In Section 5.3 we describe a case study using the `EuStockMarkets` data from the **datasets** package. Finally, Section 5.4 includes some concluding remarks.

## 5.2 mvLSWimpute approach

Each iteration of the mvLSWimpute method consists of three steps: (i) an initial spectral estimation step, (ii) a forward pass of the data in which missing values are forecasted using one-step ahead prediction and (iii) a backward pass of the data where missing entries are backcasted. An overall estimate of the imputed values of the time series is then obtained by averaging the series obtained from the forward and backward pass of the data. We briefly describe the LWS matrix estimation process

in Section 5.2.1 before presenting a worked example for a trivariate locally stationary wavelet (LSW) time series containing missing values. For a more comprehensive discussion please read Chapter 4.

The theory behind forecasting multivariate locally stationary time series is introduced in Section 5.2.2 along with an example of how this is implemented within R in the **mvLSWimpute** package. The backcasting step, which was introduced to improve imputation results, is summarized in Section 5.2.3. A complete demonstration of one iteration of the imputation process is then included.

The **mvLSWimpute** package consists of functions that allow us to impute missing values in a multivariate locally stationary time series, following the method outlined in Algorithm 2. The main functions contained in this package can be found in Table 5.2.1.

### 5.2.1 Spectral Estimation

The first step of the imputation approach involves estimating the Local Wavelet Spectral (LWS) matrix of the time series. The theory on how to accurately estimate this quantity was introduced by Park et al. (2014) for the case where the underlying data is complete. However, when the time series contains missing values, the estimation procedure can no longer be applied in its existing form and must be modified. Following the work in Chapter 4, we first calculate the Haar empirical wavelet coefficient vector $\mathbf{d}_{j,k}$ for the time series, ensuring that any wavelet coefficients at scales and locations affected by the initial missing values are also missing. From this, we can obtain the raw wavelet periodogram matrix and linearly interpolate missing coefficients in

| Function | Description |
|---|---|
| `spec_estimation` | Estimates the LWS matrix for a multivariate time series containing missing values. Returns a `mvLSW` object containing the spectral array and information on the estimation procedure. |
| `mv_impute` | Can be used to carry out one iteration of the mvLSWimpute method, using either one-step ahead forecasting only or combined forecasting and backcasting. Returns a list containing the imputed time series and information on the times and locations of the original missing data points. |

Table 5.2.1: **mvLSWimpute** functions

the spectra and cross-spectra across the levels containing information. For the levels of the periodogram where the coefficients are all missing, we replace them by recursively applying the wavelet filter equations to the coarsest level of the periodogram that contains information. Finally, we smooth and correct the periodogram in the usual manner, as described in Park et al. (2014) and implemented in the **mvLSW** R package. A comprehensive description of this estimation procedure can be found in Chapter 4.

We demonstrate the performance of the spectral estimation function on a simulated example, using the mvLSW process with slowly evolving dependence structure from the simulation study in Section 4.4. The trivariate EWS is defined in R using the **mvLSW** package for a time series of length $T = 512$ (i.e. $J = 9$).

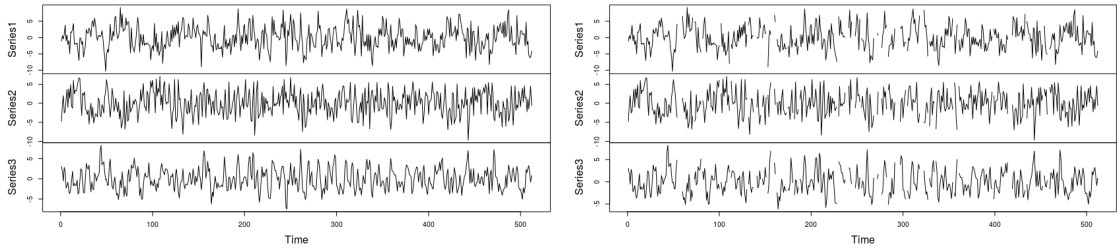
```
R> library("mvLSW")
R> P <- 3
R> T <- 512
R> J <- log2(T)
R> true_spec <- array(0,dim=c(3,3,9,512))
R> true_spec[1,2,1,] <- true_spec[2,1,1,] <-
+   c(rep(5,150),rep(-5,150),rep(5,212))
R> true_spec[2,3,2,] <- true_spec[3,2,2,] <-
+   c(rep(-7,150),rep(7,150),rep(-7,212))
R> true_spec[1,3,3,] <- true_spec[3,1,3,] <-
+   c(rep(2,150),rep(8,150),rep(2,212))
R> true_spec[1,2,5,] <- true_spec[2,1,5,] <- true_spec[1,3,4,] <-
+   true_spec[3,1,4,] <- c(rep(-4,150),rep(4,150),rep(-4,212))
R> true_spec[1,2,2,] <- true_spec[2,1,2,] <- true_spec[2,3,2,] <-
+   true_spec[3,2,2,] <- c(rep(-6,150),rep(6,150),rep(-6,212))
R> true_spec <- as.mvLSW(true_spec)
R> true_spec <- mvLSW:::AdjPositiveDef(true_spec, 1e-10)
```

To obtain a LSW time series containing missing values, we first generate a realisation from the EWS structure defined above with Gaussian innovations, again using the **mvLSW** package. To ensure reproducibility in this article, we set the seed.

```
R> set.seed(1)
R> X <- rmvLSW(Spectrum = true_spec, noiseFN = rnorm)
```

We then randomly sample 10% of indices to contain missing values across all variables, Figure 5.2.1(b) shows an example realisation with this type of missingness induced.

```
R> missing.indices <- sample(1:T,size=floor(0.1*T))
R> test.missing <- X
R> test.missing[missing.indices,] <- NA
```




(a) (b)

Figure 5.2.1: Example realisation of the process. (a) True signal; (b) Signal after missingness has been induced.

Estimation of the LWS matrix for a multivariate locally stationary time series containing missing values is implemented using the `spec_estimation` function. The wavelet transform is performed using the `HaarWT` function and we use routines from the **mvLSW** package to smooth and correct the estimates and ensure the spectra and cross-spectra are positive definite. This ensures that our spectral estimates should be comparable to those generated using the `mvEWS` function from **mvLSW**.

```
R> est.LWS <- spec_estimation(data=test.missing)
```

The input arguments of `spec_estimation` are :

- `data`: A $P$-variate time series with dyadic length containing missing values.

Note that the input time series must be of class `matrix`, `ts`, `xts` (Ryan et al., 2018) or `zoo` (Zeileis et al., 2018).

Within the spectral estimation step, we use the `Smooth_EWS` and `AdjPositiveDef` functions from the **mvLSW** package to smooth our estimates and ensure the spectra and cross-spectra are positive definite. The function returns a `mvLSW` object containing the following elements:

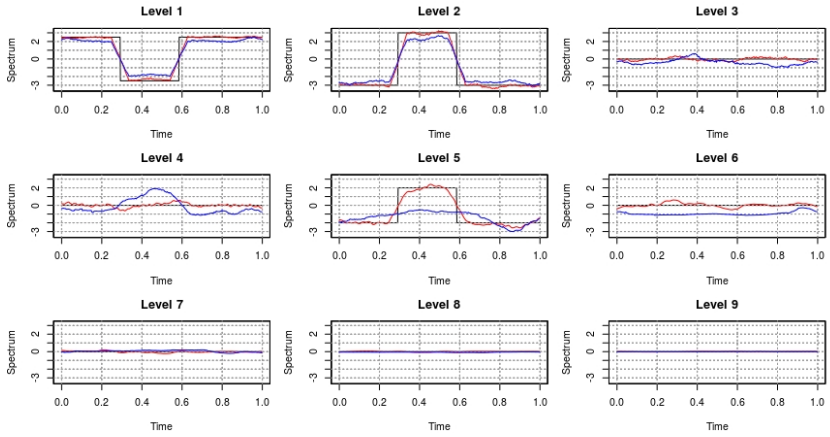
- `spectrum`: A $P \times P \times J \times T$ array containing the estimated LWS matrix.
- `Information`: A list containing information on the estimation procedure, see Taylor et al. (2017) for further details.

To illustrate the performance of the `spec_estimation` function, we simulate 100 multivariate locally stationary wavelet processes from this spectral structure and estimate the LWS matrix of each complete series using the **mvLSW** R package. Missingness is then induced within the data, for each replication we randomly select 10% of indices to contain missing values across all variables. The spectral structure for the signal containing missing data is then estimated using the `spec_estimation` function described above. Over the 100 replications, we compare the average spectra and cross-spectra at different levels for each of the methods to see how our spectral estimation approach for missing data compares with the true values generated using the complete data and the `mvEWS` function. Figure 5.2.2 shows the average cross-spectra for the

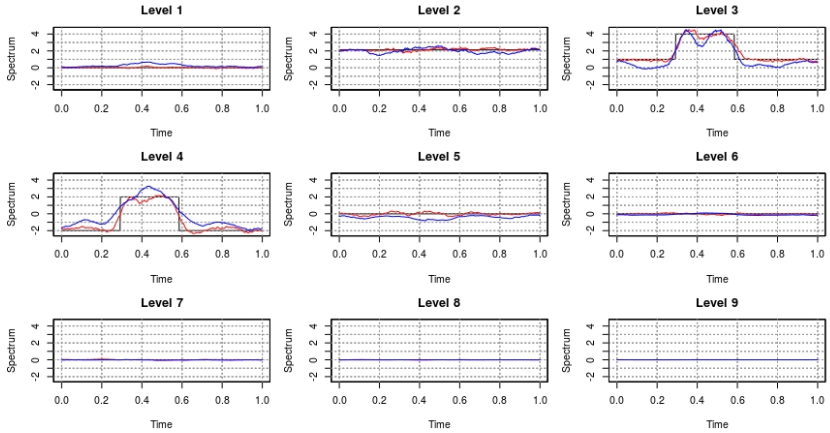
channel combinations $(1,2),(1,3)$ and $(2,3)$ across all levels. The true spectral values are plotted in black, the red lines represent the average cross-spectra over the replications generated by applying the `mvEWS` function to the complete time series and the blue lines are the average cross-spectra generated by applying the `spec_estimation` function to the time series containing missingness.

It can be seen that our spectral estimation method produces cross-spectra that broadly match those produced using the `mvEWS` function on the complete time series, however there appear to be some differences across Levels $4, 5$ and 6. In this case, across all replications the coarsest level of the cross-spectra $\hat{J}$ that contains true wavelet coefficients is 5. This level of the raw wavelet periodogram contains very little true information from the original time series, and so it can be difficult to accurately replace the missing wavelet coefficients. This can result in the imputed spectra differing from the truth, an example of which can be seen in Figure 5.2.2(a). It can also be seen from Figure 5.2.2 that our spectral estimation method is able to reproduce the features in the cross-spectra at finer levels 1–4. However, it is important to note that if we wanted to estimate the LWS matrix for a multivariate locally stationary wavelet process which had power only at coarser scales then the utility of our approach may be limited due to the difficulties mentioned above.

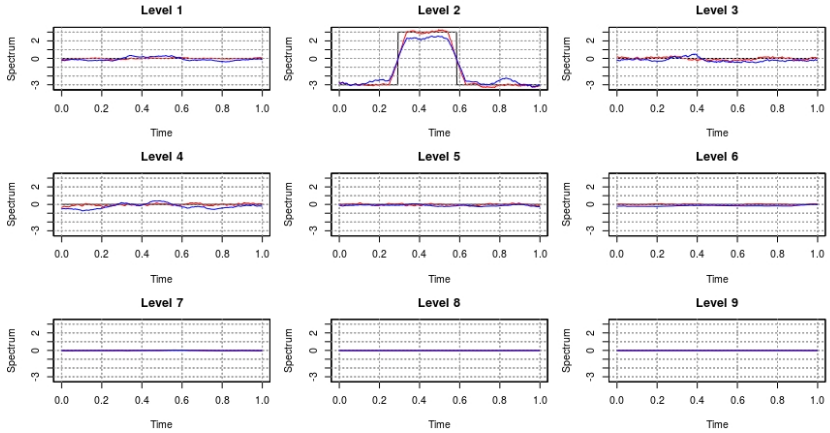
We also consider a further example, this time using the time-varying vector autoregressive process with slowly evolving dependence structure from the simulation study in Section 4.4. Again we simulate 100 replications from this process and repeat the imputation process described above. The average cross-spectra across all levels for the channel combinations $(1,2),(1,3)$ and $(2,3)$ can be seen in Figure 5.2.3.




(a) Cross-spectra for channels 1 and 2 across all levels

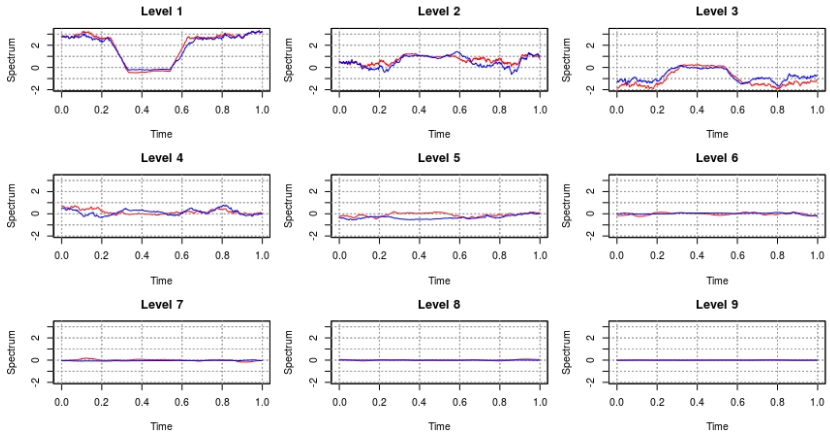



(b) Cross-spectra for channels 1 and 3 across all levels

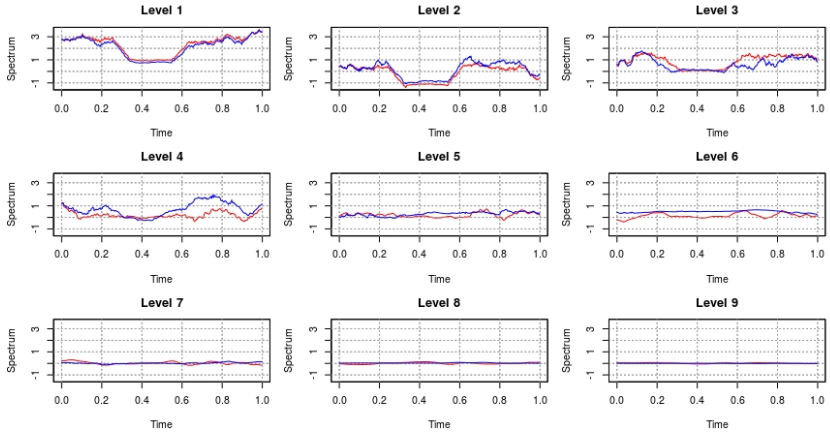



(c) Cross-spectra for channels 2 and 3 across all levels

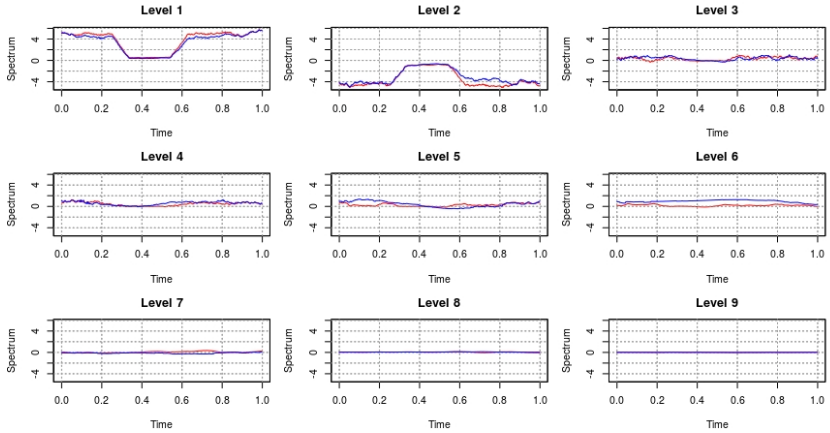
Figure 5.2.2: Average spectra across 100 replications of the slowly evolving mvLSW process; True spectral values shown in black, spectral values generated using `mvEWS` shown in red and the spectral estimates for the missing data are shown in blue.




(a) Cross-spectra for channels 1 and 2 across all levels




(b) Cross-spectra for channels 1 and 3 across all levels




(c) Cross-spectra for channels 2 and 3 across all levels

Figure 5.2.3: Average spectra across 100 replications of the time varying vector autoregressive process; spectral values generated using `mvEWS` shown in red and the spectral estimates for the missing data are shown in blue.

Again the red lines represent the average cross-spectra over the replications generated by applying the `mvEWS` function to the complete time series and the blue lines are the average cross-spectra obtained by applying the `spec_estimation` function to the time series containing missingness. It can be seen that the estimates produced by the `spec_estimation` function match those produced by `mvEWS` across most levels. The largest difference can be seen in the coefficients at level 4 in Figure 5.2.3(b), as in the previous example this can be explained by the fact that few true wavelet coefficients will be generated at levels 4–5 for time series with this structure and so it is difficult to produce accurate estimates of the missing coefficients.

The examples within this section show that the mvLSWimpute method produces accurate estimates of the spectra and cross-spectra of a multivariate locally stationary wavelet process containing missing values at finer scales. However they also show some of the downsides of the imputation approach, in that the wavelet coefficients produced by the method at coarser levels can contain little true information from the original time series. For this reason, it is important to note that the mvLSWimpute approach may produce poor imputation results for processes that contain power only at coarser scales.

### 5.2.2 Forecasting

Within the mvLSWimpute method, the LWS matrix estimated as described in Section 5.2.1 is used to produce one-step ahead forecasts of missing values. The forecasting step consists of an extensions of the wavelet-based forecasting approach of Fryzlewicz et al. (2003) to the multivariate setting which uses the Multivariate Lo-

cally Stationary Wavelet (MvLSW) framework of Park et al. (2014). The local auto and cross-covariance structure of the multivariate time series are used to form Yule-Walker equations which can be solved to carry out one-step ahead predictions and replace missing values with suitable estimates. A full description of the forecasting procedure can be found in Chapter 4.

The forward pass of the data involves predicting the missing values at each time point using the one-step ahead multivariate wavelet forecasting approach described in Section 4.3.2. For each timepoint $i$ at which one or more channels of the $P$-variate time series has missing values, we first consider the spectra from time 1 to $i-1$ and use this to form the local auto and cross-covariance using Equations (2.3.12) and (2.3.13). Following this, the prediction equations defined in equation (4.2.3) can be formed for each channel combination $(p, q)$. These are then solved to obtain $\mathbf{b}^{(p,q)}$ vectors that are substituted into the one-step ahead predictor in equation (4.2.4) to predict the values of the series at time $i$. The channels of the multivariate time series that contain missing data at time $i$ are then replaced by the corresponding predicted values.

Within the **mvLSWimpute** package, the `mv_impute` function implements the multivariate wavelet-based imputation approach. Setting the argument `type="forward"` allows the user to impute missing values using one-step ahead forecasting only and not include the backcasting step of the algorithm. The input arguments for `mv_impute` are:

- `data`: A $P$-variate time series with of dyadic length containing missing values.
- `p`: The number of terms to consider in the clipped predictor.

- `type`: Choose either `"forward"` for one-step ahead forecasting only or `"forward-backward"` to apply both the forecasting and backcasting step.
- `ind`: Vector containing the indices where datapoints are missing. By default, this is set to `NULL` and the indices are determined from the input data.

For efficiency, the `mv_impute` function calculates and returns a slice of the lacv array corresponding to the lags and locations that are needed to solve the prediction equations for the particular missing index and clipped predictor length rather than the full lacv array. This function returns a list containing the following elements:

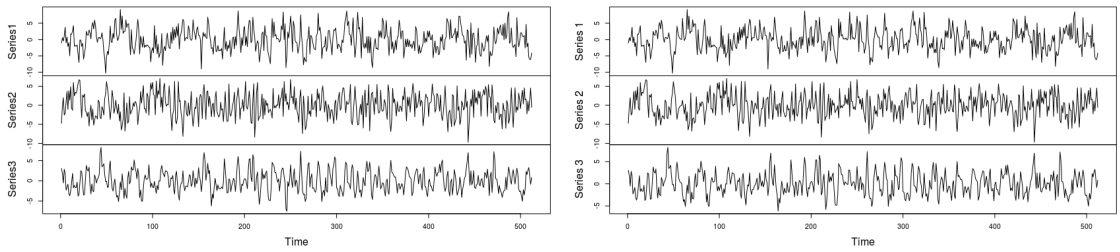
- `ImputedData`: A $T \times P$ matrix containing the imputed time series.
- `missing.index`: Vector containing the indices where datapoints are missing.

For the time series shown in Figure 5.2.1(b), we use `mv_impute` to forecast missing values, compare the results visually and calculate the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). Within the mvLSW forecasting method, we choose to use $p = 2$ for the number of terms to include in the clipped predictor. Note that if the time series contained larger periods of missing values then we would choose a larger value of $p$ to reflect this.

```
R> forecasted.series = mv_impute(test.missing, p = 2, type = "forward")
```

The results of applying one iteration of the mvLSW forecasting method can be seen in Figure 5.2.4, along with the true time series.

Finally, we apply the `accuracy` function from the **forecast** R package to assess the performance of the forecasting step. Later we will compare this to the corresponding




(a) True signal (b) Forecasted signal

Figure 5.2.4: True series and signal after applying the mvLSWimpute forecasting method.

results obtained from the combined imputation approach using a forecasting and backcasting step.

```
R> accuracy(as.ts(true),as.ts(forecasted.series$ImputedData))[1,2:3]
```

```
                RMSE       MAE
Test set  0.8877950 0.2157173
```

### 5.2.3 Backcasting

Independently of the forward pass of the data, we now carry out a backward pass in which any missing values are backcasted sequentially. This step is included to improve the imputation results as we now incorporate information from $p$ points on either side of each missing value within the prediction. In a similar way to the approach of Trindade (2003), we form backward Yule-Walker equations in the mvLSW setting by ordering the spectral values in a different way within the multivariate wavelet forecasting approach. The backcasting step is described in more detail in Section 5.2.3.

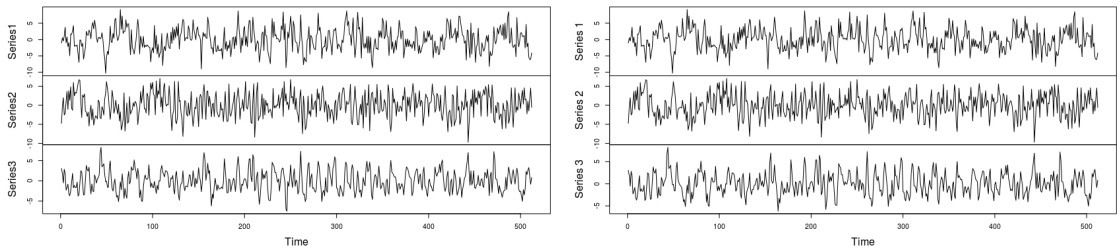
In this case, we consider the missing indices in descending order. For each time $i$ in which the series contains missing values, we consider the estimated spectra from time $T$ to $i+1$. As in the forecasting case, we can form the local auto and cross-covariance using the estimated spectra from time $T$ to $i+1$ and Equations (2.3.12) and (2.3.13). The local auto and cross-covariance can then be used to solve the prediction equations for each channel combination $(p, q)$ and obtain the $\mathbf{b}^{(p,q)}$ vectors. The one-step ahead predictor in Equation (4.3.1) is used to backcast the value of the time series at index $i$ and then any missing entries within channels are replaced using their corresponding predicted values.

Since the forward and backward pass of the time series are carried out independently, we must combine them in some way to obtain an overall estimate of the series, we choose to do this by averaging them. Note the backcasting approach is included as a way to enhance the imputation results when combined with the one-step ahead forecasts, and as such is not meant to be applied independently.

We use the `mv_impute` function to estimate missing values in the time series shown in Figure 5.2.1(b), again we choose $p = 2$ as the number of terms to include in the clipped predictor for the forecasting and backcasting step, the results can be seen in Figure 5.2.5.

```
R> imputed.series = mv.impute(test.missing, p = 2,
+   type = "forward-backward")
```

Applying the `accuracy` function to the imputed time series results in the following RMSE and MAE estimates




(a) True signal (b) Imputed signal

Figure 5.2.5: True series and signal after applying mvLSWimpute method.

```
R> accuracy(as.ts(true), as.ts(imputed.series$ImputedData))[1,2:3]

                RMSE       MAE
Test set  0.8058244 0.1945510
```

It can be seen that the RMSE and MAE are lower for the time series obtained from the complete mvLSWimpute method than for the individual forecasted series. This is to be expected as each missing point will be estimated using $2p$ total points rather than $p$ points for forecasting individually (where $p$ is the number of terms in the clipped predictor). This illustrates how the inclusion of the backcasting step helps improve the accuracy of the imputation results.

The mvLSWimpute method can be iterated by applying the `mv_impute` function to the imputed time series obtained from the initial step and specifying the time and location of the initial missing values. In practice we have typically found that the first application of the method produces an accurate imputation estimate and subsequent iterations do not provide much improvement in terms of accuracy, for this reason we do not include the option to iterate within the `mv_impute` function.

## 5.3 Case Study

In this section, we demonstrate the **mvLSWimpute** package functionality through the use of a case study. We consider the `EuStockMarkets` data from the **datasets** package that consists of the daily financial index of the German (DAX), Swiss (SMI), French (CAC) and British (FTSE) markets from 1991 to 1999. In order to remove any trend from the series, we first fit a smoothing spline to each financial index before considering the residuals. We then select a window of the time series of length $T = 1024$ on which to perform the analysis, a plot of this series can be seen in Figure 5.3.1.

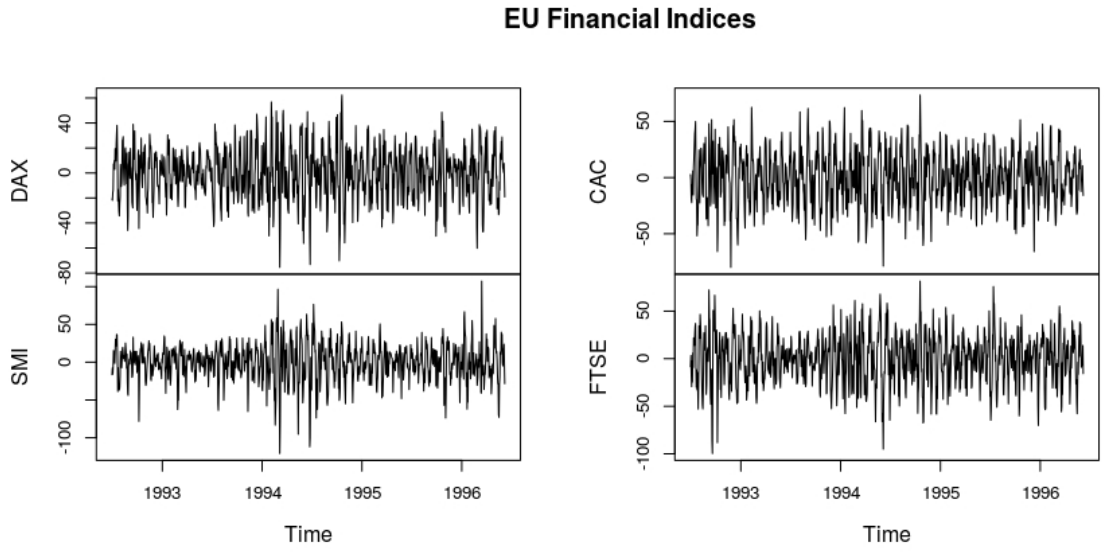



Figure 5.3.1: Time series of four European financial indices.

```
R> data("EuStockMarkets", package = "datasets")

R> T <- 1024; J <- log2(T); N <- nrow(EuStockMarkets)

R> detrend_fun = function(data)return(residuals(smooth.spline(data)))
```

```
R> apply(EuStockMarkets, FUN = detrend_fun, MARGIN = 2)
R> EU.ret = ts(EU.ret, start = start(EuStockMarkets),
+   end = end(EuStockMarkets), frequency = frequency(EuStockMarkets))
R> EU.ret <- window(EU.ret, start = c(1992,130), end = c(1996,113))
R> plot(x = EU.ret, main = "EU Financial Indices", nc = 2)
```

As in Section 5.2.1, we sample 10% of indices at random that will contain missing values across all variables, a plot of the time series containing missing values can be seen in Figure 5.3.2.

```
R> set.seed(123)
R> missing.indices <- sample(10:(T-1), size = floor(0.1*T))
R> EU.missing <- EU.ret
R> EU.missing[missing.indices,] <- NA
R> plot(x = EU.missing, main = "EU Financial Indices", nc = 2)
```

Imputation of the time series in Figure 5.3.2 is carried out using the `mv_impute` function, we consider the results from the combined forecasting and backcasting approach. The following code extract applies the wavelet-based imputation method and creates a plot of the imputed time series which can be seen in Figure 5.3.3. Note that in this case we choose to set the length of the clipped predictor $p = 2$ since there are not many consecutive missing values in the time series. In situations where longer periods of time contain missing values, selecting a larger value for the length of clipped predictor may be more suitable.

```
R> EU.imputed <- mv_impute(EU.missing, p = 2,
```

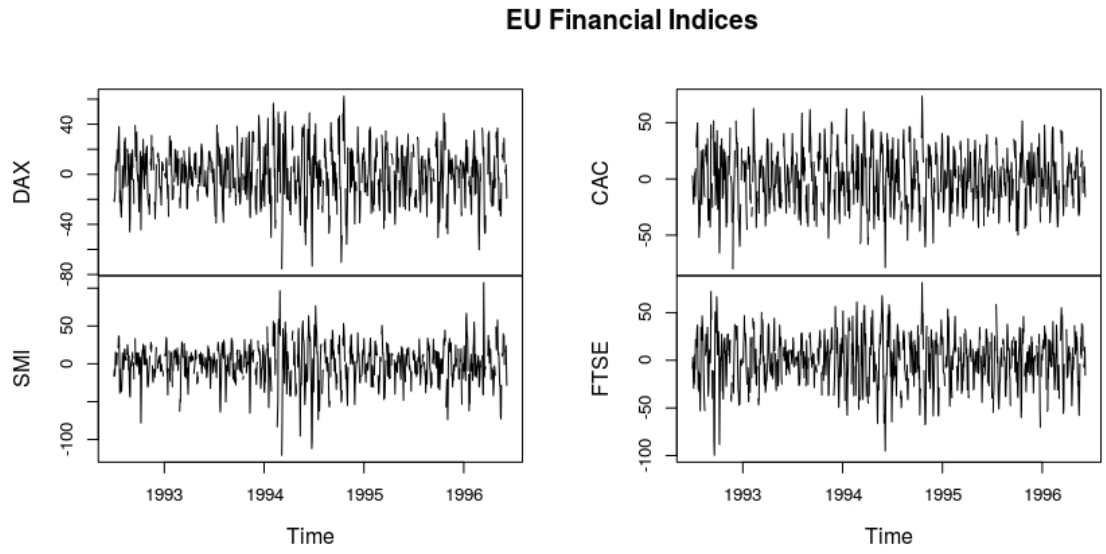



Figure 5.3.2: Time series of four European financial indices, for each series 10% of values are missing.

```
+   type = "forward-backward")
R> plot(x = EU.imputed$ImputedData, nc = 2,
+   main = "EU Financial Indices - mvLSWimpute")
```

Applying the `accuracy` function to the imputed time series results in the following RMSE and MAE estimates

```
R> accuracy(EU.ret, EU.imputed$ImputedData)[1,2:3]

                RMSE       MAE
Test set  5.194638 1.290103
```

We compare these imputation results with those obtained from the **mtsdi** R package which, of the competitor methods, has produced the strongest results. The fol-

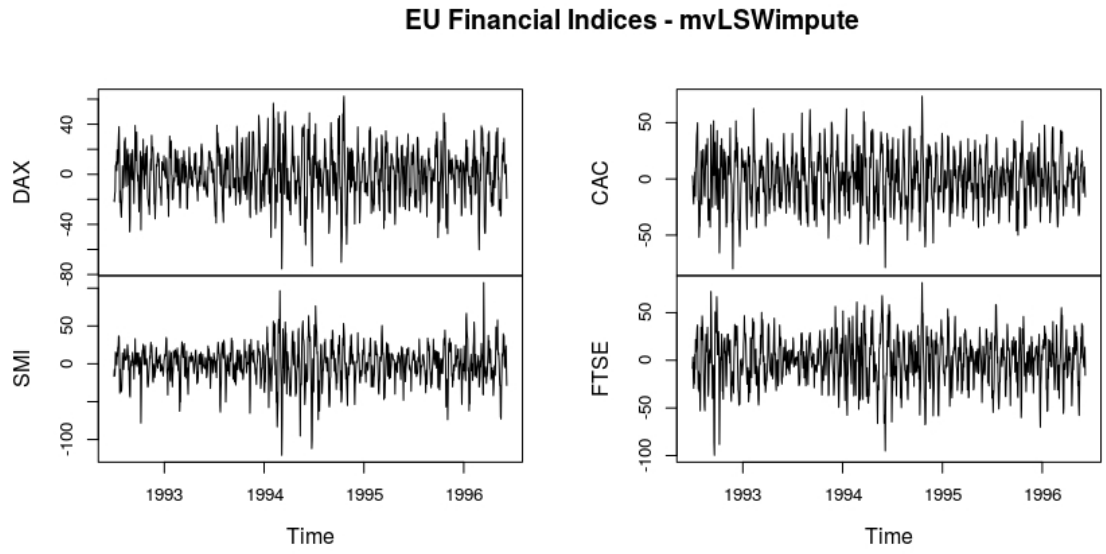



Figure 5.3.3: European Financial Indices time series after imputing missing values using mvLSWimpute.

lowing code extract applies the EM-based method and creates a plot of the imputed time series which can be seen in Figure 5.3.4.

```
R> library("mtsdi")
R> EU.mtsdi <- mnimput(formula=~DAX+SMI+CAC+FTSE, dataset=EU.missing)
R> plot(x = ts(EU.mtsdi$filled.dataset, start = start(EU.ret),
+   end = end(EU.ret), nc = 2, frequency = frequency(EU.ret)),
+   main = "EU Financial Indices - mtsdi")
```

Again, we apply the `accuracy` function to the imputed time series to obtain the following RMSE and MAE estimates

```
R> accuracy(EU.ret, EU.mtsdi$filled.dataset))[1,2:3]
                RMSE       MAE
```

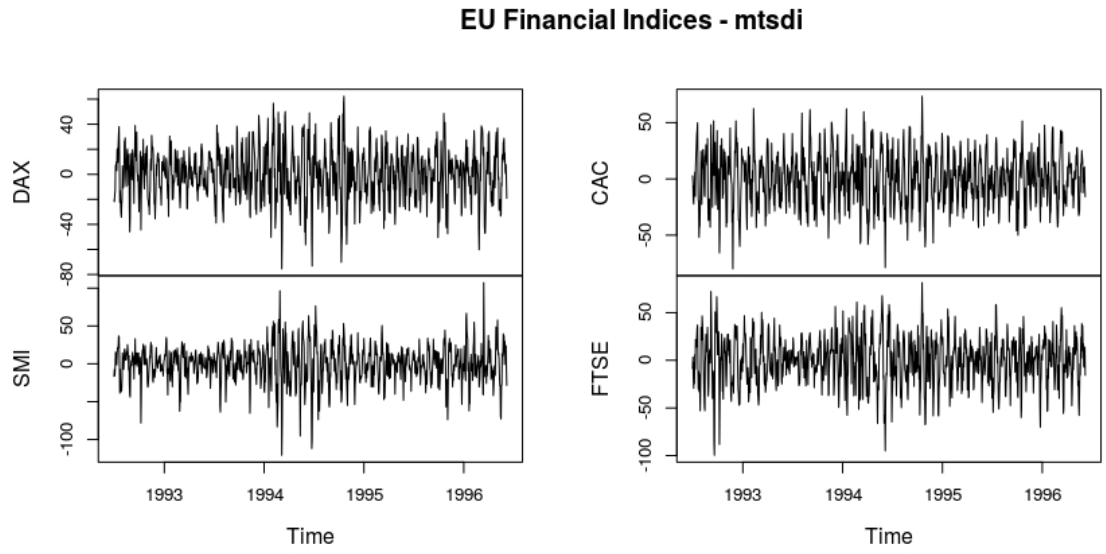
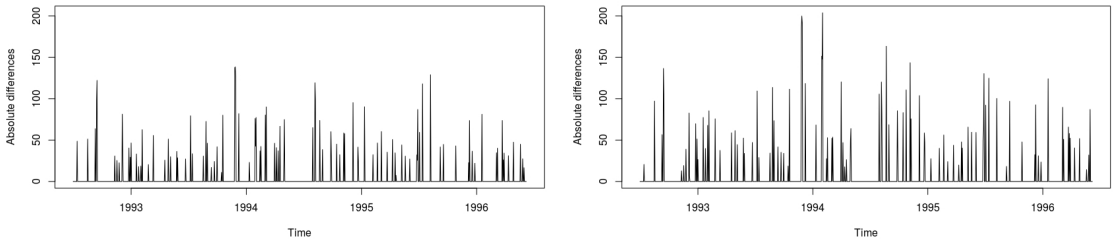



Figure 5.3.4: European Financial Indices time series after imputing missing values using mtsdi.

```
Test set  7.219201 1.740379
```

For the European Financial Indices time series, the mvLSWimpute method produces better imputation results than the competitor both in terms of the RMSE and MAE. For each of the imputed time series, we considered the absolute difference between the imputed estimate and the true values over time. Figure 5.3.5 shows the sum of this absolute difference over the four financial indices of interest. It can clearly be seen that the magnitude of these differences are greater for the imputed time series produced by mtsdi.

Within this case study, we have demonstrated that the **mvLSWimpute** R package can be used to replace missing values within a multivariate locally stationary time series with suitable predictions based on the local auto and cross-covariance of the series.




(a) (b)

Figure 5.3.5: Sum of the absolute differences between the imputed time series and the truth. (a) mvLSWimpute; (b) mtsdi.

## 5.4 Summary

The **mvLSWimpute** package contains a number of useful tools for analysing multivariate nonstationary time series containing missing values, using the multivariate locally stationary wavelet framework defined in Park et al. (2014). The command `spec_estimation`, discussed in Section 5.2.1, allows the user to estimate the Local Wavelet Spectral matrix of a time series containing missing entries by first estimating the raw wavelet periodogram, keeping any missing coefficients intact. Depending on the level of the periodogram, these missing wavelet coefficients are then replaced by either linearly interpolating or recursively applying the wavelet filter equations. The main function `mv_impute`, can be used to impute missing values in nonstationary time series using either one-step ahead forecasting, described in Section 5.2.2, or a combined approach which incorporates a backcasting step to enhance the forecasting results, discussed in Section 5.2.3.

A case study that demonstrates the package utilities using the `EuStockMarkets`,

European financial indices dataset can be found in Section 5.3. Through this, we show that **mvLSWimpute** can be used to replace missing values in a nonstationary time series where the spectral power of the process varies over time. The nonstationarity is taken into account within the local auto and cross-covariance functions that are used to replace the missing values within the series with suitable estimates.

# Chapter 6

# Conclusions and Discussion of Future Work

In this thesis, we considered two different problems related to nonstationary time series analysis. In particular, we have focused on classifying and imputing missing values within multivariate time series with slowly evolving dependence structure. In Chapter 2, we review the extensive literature on wavelets and time series modelling. A brief overview of wavelet analysis and some of the most common wavelet transforms is provided before summarising popular approaches for modelling time series, concentrating on the nonstationary setting with the Locally Stationary Fourier processes (Dahlhaus, 1997) and the Smooth Localized Complex Exponential model (Ombao et al., 2002). We conclude the chapter by introducing alternative wavelet-based methods for modelling nonstationary time series with changing second-order structure, including the (univariate) Locally Stationary Wavelet model (Nason et al., 2000) and the multivariate Locally Stationary Wavelet framework (Park et al., 2014) that forms a key part

of the classification and imputation approaches developed in this thesis.

In Chapter 3, we presented an extension of the dynamic classification approach of Park et al. (2018) to the online setting. In order to allow for classification of data streams, we make use of a dyadic length moving window. For each shift of the window, we update the wavelet coefficients and transformed coherence estimate from which we can obtain the probability that the signal belongs to a particular class for the time points contained in the window. This procedure results in the collection of multiple probability estimates corresponding to the different windows, these are combined by averaging the estimates at each time point. Finally, we present a case study demonstrating the utility of our method as a tool for classifying and detecting periods of anomalous behaviour within distributed acoustic sensing data. The details of the software that implements the online dynamic classification method can be found in Appendix A.

In Chapter 4, we developed the mvLSWimpute approach as a method for replacing missing values within a multivariate locally stationary time series. Our approach first estimates the local wavelet spectral information of the process containing missing values which is used to form the local auto and cross-covariance functions. These are key quantities in the prediction step where a multivariate extension of the wavelet forecasting approach of Fryzlewicz et al. (2003) is introduced, allowing us to calculate one-step ahead forecasts for the missing values. This is combined with a backcasting step to improve the performance of the imputation method. Finally, we describe a case study in which the mvLSWimpute method is applied to a dataset arising from a Carbon Capture and Storage facility. The **mvLSWimpute** R package that

implements the imputation approach is outlined in Chapter 5 along with detailed examples of the package functionality.

We conclude this thesis by discussing some avenues for future research. The motivation for the development of the online dynamic classification method detailed in Chapter 3 was to be able to detect anomalous regions within acoustic sensing data. At the moment, we have considered the two-class problem within Section 3.4 where normal behaviour and stripes are well defined classes. In practice, we may come across stripes of a different structure and there are a number of ways in which we could approach this classification problem. Firstly, we could compile a universal bank of training signals containing examples of different types of anomalous behaviour, a suitable collection of training signals would allow us to both detect stripes and determine the type present within a time series. This would require an extensive and exhaustive knowledge of the features present in this type of data. An alternative would be to treat this as a classification problem with $N$ known classes and other unseen ones. The challenge then would be to develop a dynamic classification method that can distinguish between the known classes and determine time points belonging to unknown ones.

Future work arising from Chapter 4 could include developing a data-driven approach for determining the number of terms to include in the clipped predictor $p$ and the smoothing bandwidth parameter $g$. This has been studied in the univariate forecasting setting, for example Fryzlewicz et al. (2003) uses a grid search method which involves moving backwards by a certain number of observations $s$, choosing some initial estimates for the parameters $(p_0, g_0)$ and predicting $X_{t-s}$ on the grid

$[(p_0 - 1, g_0 - \delta), (p_0, g_0 - \delta), \ldots, (p_0 + 1, g_0 + \delta)]$. The pair that corresponds to the best predictor is then chosen to be the new initial parameters, the process is iterated to predict $X_{t-s+1}, X_{t-s+2}, \ldots, X_{t-1}$ and the parameters are updated each time to obtain the parameters $(p_1,\ g_1)$ to be used in the final prediction. A similar approach could be carried out in the multivariate setting but this updating procedure would not be computationally efficient. Alternatively, Killick et al. (2018) uses the local partial autocorrelation function to select $p$. Within this method, an estimate of the local partial autocorrelation function is computed for a window centred on the last observation in the process $t - 1$. In a similar manner to the stationary setting, $\hat{p}$ is then chosen to be the lag that minimises this. The smoothing is carried out using a rectangular kernel with window size determined using the best bandwidth selection algorithm in the **locits** R package (Nason, 2016). In the multivariate setting, we currently use a global value of $p$ for each missing values of the time series and the default settings for the smoothing parameters $\lfloor\sqrt{T}\rfloor$, where $T$ is the length of the time series. For each of the different channel combinations for a $P$-variate time series, we could obtain the local windowed partial autocorrelation function but the challenge would then be how to determine an overall value of $\hat{p}$ from the estimates.

# Appendix A

# `onlineDC`: R package

## A.1 The onlineDC package functions

The **onlineDC** package implements the online dynamic classification approach described in Chapter 3. The main functions contained in this package are:

- `sim.training`: Simulates multivariate normal training signals from predefined classes. Returns a list containing the training data and the true class membership of the signals.

- `get.info`: Computes useful information such as the set of discriminative indices from training signals. Returns a list containing the indices along with the transformed coherence of the training signals grouped by class.

- `calc.prob`: Implements the offline dynamic classification method and calculates the probability that the test signal belongs to a particular class over time.

- `windowed.prob`: Implements the dynamic classification with a moving window,

the probability of belonging to a particular class for each window of data is saved in an array.

- `prob.est`: Computes the average probability that the test signal belongs to a particular class over time based on the estimates obtained from the different windows. Returns an object of class `onlineDC`.

- `plot.onlineDC`: Plot the average probability of belonging to each class over time. There are three different displays options, the average probability can be plotted as a line graph or heat/gradient plot and the average class membership over time can also be displayed.

We will now demonstrate how the **onlineDC** package works through a simulated example. We consider a three class example, where each class is defined by a trivariate normal signal with the same mean vector $\mu = (0, 0, 0)$ and differing covariance matrices. The covariance matrices are given by:

$$\Sigma^{(1)} = \begin{pmatrix} 1 & 0 & 0.3 \\ 0 & 1 & 0.7 \\ 0.3 & 0.7 & 1 \end{pmatrix}, \quad \Sigma^{(2)} = \begin{pmatrix} 1 & 0.6 & 0.1 \\ 0.6 & 1 & -0.4 \\ 0.1 & -0.4 & 1 \end{pmatrix}, \quad \Sigma^{(3)} = \begin{pmatrix} 1 & 0.2 & -0.5 \\ 0.2 & 1 & 0 \\ -0.5 & 0 & 1 \end{pmatrix}.$$

### A.1.1 Simulating multivariate normal training data

To begin, we use the function `sim.training` to simulate a set of training signals of length 128 that will be used in the classification:

```
library("onlineDC")

R> mean.vec <- list(c(0,0,0), c(0,0,0), c(0,0,0))
```

```
R> cov.mat <- list(matrix(c(1,0,0.3,0,1,0.7,0.3,0.7,1), 3, 3),
+ matrix(c(1,0.6,0.1,0.6,1,-0.4,0.1,-0.4,1), 3, 3),
+ matrix(c(1,0.2,-0.5,0.2,1,0,-0.5,0,1), 3, 3))
R> training <- sim.training(w = 128, mean = mean.vec, cov = cov.mat)
```

The input arguments for this function are:

- `w`: The length of the training signals which we wish to simulate.
- `mean`: A list containing the mean vectors for each multivariate normal class.
- `cov`: A list containing the corresponding covariance matrices for the classes.

Using the predefined input multivariate normal classes, the function returns a list containing the following elements:

- `data`: An array containing the simulated training signals.
- `class`: A matrix containing the true class membership of the training data.

As default, the signal generates two signals of length $w$ from each of the classes and further signals that contain a change in class at the halfway point. For example, the above code will generate nine training signals of length 128; two simulations from each of the three classes and a further three signals containing changes in class.

### A.1.2 Obtaining information from the training data

As described in Section 3.2, the first step in the dynamic classification method involves the calculation of the transformed coherence for each training signal and determining the set of discriminative indices as in Definition 2.4.7. We can invoke the `get.info`

function to obtain the set of discriminative indices that will then be used in the classification step. This function requires several arguments:

- `training`: An array of dimension $P \times w \times N$ containing the training data, where $P$ is the number of channels, $w$ is the window length and $N$ is the number of training signals.
- `true.class`: A matrix of dimension $w \times N$ that contains the class assignments of the training signals.
- `Jstar`: Maximum level of the wavelet transform to consider, if none is entered the default is $\log_2(w)$.
- `prop`: This is used to determine the size of the set of discriminative indices used for classification, given by prop$(\frac{P(P-1)J}{2})$, if no value is entered then the default is prop $= 0.25$.
- `filter.number`: The filter number of the wavelet function used in the calculation of the transformed coherence.
- `family`: The wavelet function family used in the calculation of the transformed coherence.
- `smooth`: Logical variable, determines whether the wavelet periodogram should be smoothed. By default, this is set to `TRUE`.
- `bias.correct`: Logical variable, determines whether to correct the wavelet periodogram using the inverse of the inner product matrix of discrete autocorrelation wavelets, as described in Section 2.3.2.

In order to compute the transformed coherence estimates, we call functions from the **mvLSW** package, specifically `mvEWS` and `coherence`. Additional arguments can be passed to `get.info` specifying the smoothing method to be applied and the smoothing parameters, as required by `mvEWS`. Note that if `smooth=TRUE`, default values are provided for all of the required smoothing arguments.

The `get.info` function returns a list containing the following elements:

- `ind`: A vector containing the set of discriminative indices.
- `Sc`: A list containing the spectral estimates of the training signals, split by class. This can then be used to easily determine the estimated mean and covariance matrices for the transformed coherence of each class.

This function is called automatically when we invoke the main classification function in the package `prob.est` but we have included the description here for completeness. For example, using the training data generated above we can use the Haar wavelet transform to obtain the transformed coherence of the signals and the set of discriminative indices using the following command:

```
R> info <- get.info(training = training$data, Jstar = 7, prop = 0.25,
+   true.class = training$class, filter.number = 1,  smooth = TRUE,
+   family = "DaubExPhase")
```

### A.1.3 Applying Dynamic Classification method

We now simulate a test signal $Y$ that we will classify using the functions in the package. This signal will be of length 450, starting in class 2 for the first 150 timepoints before

moving to class 1 at time 300 and then ending in class 3.

```
R> set.seed <- 1000
R> Y <- matrix(NA,3,450)
R> Y[,1:150] <- t(rmvnorm(150, mean = rep(0,3),
+    sigma = matrix(c(1,0.6,0.1,0.6,1,-0.4,0.1,-0.4,1), 3, 3)))
R> Y[,151:300] <- t(rmvnorm(150, mean = rep(0,3),
+    sigma = matrix(c(1,0,0.3,0,1,0.7,0.3,0.7,1), 3, 3)))
R> Y[,301:450] <- t(rmvnorm(150, mean = rep(0,3),
+    sigma = matrix(c(1,0.2,-0.5,0.2,1,0,-0.5,0,1), 3, 3)))
```

At this stage, we can use the `calc.prob` function on individual windows of the test signal to estimate the probability that the signal belongs to a particular class for the duration of the window. For example, if we wish to obtain the probabilities for the first window of data we use the following command:

```
R> window1 <- calc.prob(test = Y[,1:128], training = training$data,
+   true.class = training$class, Jstar = 7, prop = 0.25,
+   filter.number = 1, family = "DaubExPhase", smooth = TRUE)$prob
```

Again we include this function for completeness as it is invoked by both the `windowed.prob` and `prob.est` functions. We do not include a full description of the function arguments here since it shares arguments with both of these functions.

### A.1.4 Applying Dynamic Classification with a moving window

In order to carry out the online dynamic classification method, we select a suitable moving window length $w$ and classify each window of the data individually as described in Section 3.2. We use the function `windowed.prob` to generate an array that contains the probability of belonging to a particular class for each of the window of data. The arguments required by this function are:

- `test`: Multivariate test signal that we wish to classify, matrix of dimension $P \times T$ where $P$ is the number of channels and $T$ is the length of the series.
- `training`: An array containing the training data.
- `true.class`: A matrix that contains the class assignments of the training signals.
- `Jstar`: Maximum level of the wavelet transform to consider, if none is entered the default is $\log_2(w)$.
- `prop`: Determines the size of the set of discriminative indices used for classification, as in the `get.info` function.
- `filter.number`: The filter number of the wavelet function used in the calculation of the transformed coherence.
- `family`: The wavelet function family used in the calculation of the transformed coherence.

- `smooth`: Logical variable, determines whether the wavelet periodogram should be smoothed. By default, this is set to `TRUE`.

- `bias.correct`: Logical variable, determines whether to correct the wavelet periodogram using the inverse of the inner product matrix of discrete autocorrelation wavelets.

- `return`: Logical condition, determines if the probability array containing the results for each window is returned.

It is important to note that the probability array generated by this function can be very large, it is for this reason we include the logical condition `return`. If `return=FALSE` then the probability array is not returned for efficiency. If `return=TRUE` then the function returns a list containing the elements:

- `ind`: A vector containing the set of discriminative indices used in the classification.

- `window.length`: Length of the moving window used in the method.

- `prob.array`: An array containing the probability of belonging to a particular class for each window of data.

Since the probability array contains the probability of belonging to a particular class for each window of data, we can use this to plot a number of different things. The `plot.dynamic` function can be used to plot dynamically how the probability estimates change over time for each window. In addition, for a particular timepoint we can plot

the probability estimates that arise from the different windows. The `plot.dynamic` function requires the following arguments:

- `data`: An array containing the probability of belonging to a particular class for each window.
- `type`: Two different methods are available. Using the option `type="dynamic"` plots how the probability estimates change over time as a result of the different windows of data. The option `type="time"` selects a particular timepoint and plots the probability estimates for that time that arise from the different windows.
- `interval`: Sets the time interval between frames of the animation, required when `type="dynamic"`. Default is 0.5 seconds.
- `t`: The timepoint for which we want to plot the probability estimates from different windows, required when `type="time"`.

First, we use the `windowed.prob` function to generate the required array:

```
R> window.results <- windowed.prob(test = Y, training = training$data,
+   true.class = training$class, Jstar = 7, prop = 0.25,
+   filter.number = 1, family = "DaubExPhase", smooth = TRUE,
+   return = TRUE)
```

The following command can then be used to generate the dynamic plot in HTML form:

```
R> plot.dynamic(window.results$prob.array, type = "dynamic",
+   interval = 0.5)
```

Alternatively, we can plot the probability estimates for a given timepoint, for example time 200, using the following command:

```
R > plot.dynamic(window.results$prob.array, type = "time", t = 200)
```

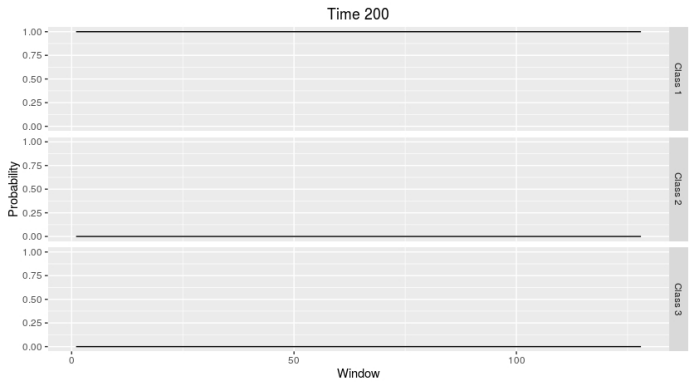



Figure A.1.1: Probability estimates from different windows at time 200.

The results can be seen in Figure A.1.1.

### A.1.5 Combining probabilities

We will now describe the main function defined in the package, `prob.est`. This function calls `get.info` to calculate the transformed coherence of the training signals and to obtain the set of discriminative indices. Then `calc.prob` and `windowed.prob` are invoked to obtain estimates of the probability that the signal belongs to a particular class over time. Finally, `prob.est` combines the probability estimates to obtain an overall average probability that the signal belongs to a particular class over time, as described in Section 3.2. The main arguments for `prob.est` are:

- `test`: Multivariate test signal that we wish to classify, matrix of dimension $P \times T$ where $P$ is the number of channels and $T$ is the length of the series.

- `training`: An array containing the training data.

- `true.class`: A matrix that contains the class assignments of the training signals.

- `Jstar`: Maximum level of the wavelet transform to consider, if none is entered the default is $\log_2(w)$.

- `prop`: Determines the size of the set of discriminative indices used for classification, as in the `get.info` function.

- `filter.number`: The filter number of the wavelet function used in the calculation of the transformed coherence.

- `family`: The wavelet function family used in the calculation of the transformed coherence.

- `smooth`: Logical variable, determines whether the wavelet periodogram should be smoothed. By default, this is set to `TRUE`.

- `bias.correct`: Logical variable, determines whether to correct the wavelet periodogram using the inverse of the inner product matrix of discrete autocorrelation wavelets.

The following code extract classifies the trivariate series $Y$ using a moving window of length 128 and the previously generated training data.

```
 R > out <- prob.est(test=Y, training = training$data, Jstar = 7,
+    true.class = training$class, filter.number = 1,  prop = 0.25,
+    family = "DaubExPhase", smooth = TRUE)
```

This function returns an object of class `onlineDC`. The `onlineDC` class objects are returned as lists. Three methods are available for this object class : `summary`, `print` and `plot`.

```
R> summary(out)

Length of data:  450
Number of channels:  3
Window length:  128
Number of training signals:  9
Levels:  7
Filter was:  Haar wavelet
Date:  Fri Dec 07 10:28:54 2018
```

Using the `summary` command in this case tells us that we classified a trivariate signal of length 450 using a moving window of length 128, we used information from 9 training signals and the wavelet used in the analysis was the Haar wavelet.

```
R> print(out)

Class 'onlineDC' : Online Dynamic Classification Object:
       ~~  : List with 7 components with names
              data training.data prob.est window.length Jstar filter
```

```
date


Created on : Fri Dec 07 10:28:54 2018
```

Invoking the `print` command tells us information about the `onlineDC` class object, in particular that it contains the following elements:

- `data`: The input test signal that has been classified.
- `training.data`: A list containing the training signals and true class assignments used in the method.
- `prob.est`: A matrix containing the average probability that the test signal belongs to each class over time.
- `window.length`: The length of the moving window used in the method.
- `Jstar`: Number of levels of the wavelet transform considered.
- `filter`: List containing information on the wavelet filter used in the method.
- `date`: The date the classification was carried out.

Finally, we can use the `plot.onlineDC` function to plot the information contained in an `onlineDC` object in a number of different ways. The function requires the following arguments:

- `object`: An `onlineDC` class object.

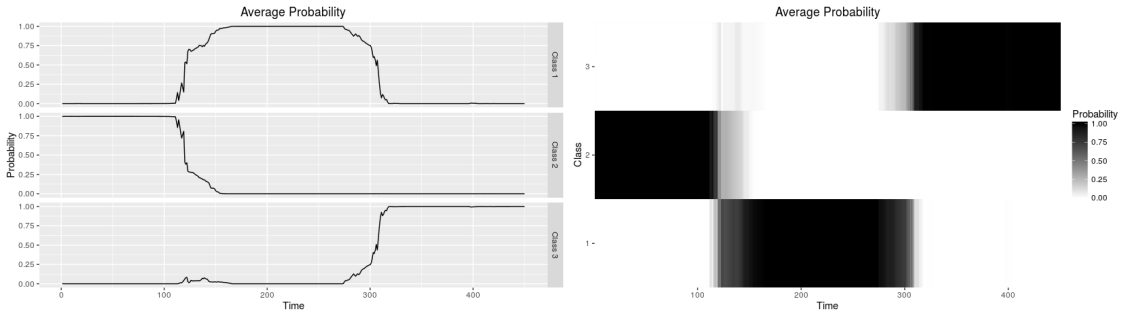
- `display`: Three display methods are available. The average probability that the signal belongs to each class over time can be plotted as a line graph using the option `display="prob.l"`. Similarly, using the option `display="prob.h"` allows us to display the average probability as a heat/gradient plot. The final option `display="class"` plots the average class membership over time of the signal.

For example, using the command

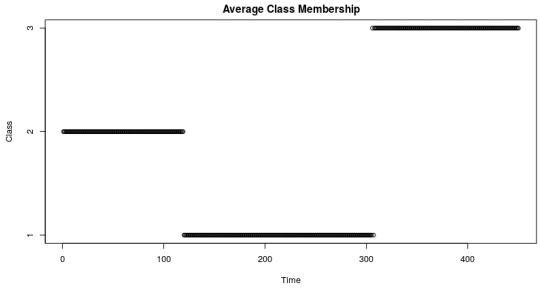
```
R> plot(out, display = "plot.l")
```

results in a plot of the average probability that the signal belongs to each of the three classes over time, as seen in Figure A.1.2(a).

The **onlineDC** package implements the online dynamic classification approach, as described in Chapter 3. **onlineDC** consists of functions that allow us to obtain the set of discriminative indices from training signals, apply dynamic classification with a moving window and then obtain an overall probability that the signal belongs to a particular class over time. As a result, **onlineDC** can be used to classify a multivariate data stream which is not of dyadic length and contains multiple class changes.




(a) Using `display="prob.l"` (b) Using `display="prob.h"`




(c) Using `display="class"`

Figure A.1.2: The result of using the different display options to plot the `onlineDC` class object.

# Bibliography

P.L. Ainsleigh, N. Kehtarnavaz, and R. L. Streit. Hidden Gauss-Markov models for signal classification. *IEEE Transactions on Signal Processing*, 50(6):1355–1367, 2002.

F. M. Alvarez, A. Troncoso, J. C. Riquelme, and J.S.A. Ruiz. Energy time series forecasting based on pattern sequence similarity. *IEEE Transactions on Knowledge and Data Engineering*, 23(8):1230–1243, 2011.

R. G. Aykroyd, S. Barber, and L. R. Miller. Classification of multiple time signals using localized frequency characteristics applied to industrial process monitoring. *Computational Statistics & Data Analysis*, 94:351–362, 2016.

S. Bachu. CO2 storage in geological media: Role, means, status and barriers to deployment. *Progress in Energy and Combustion Science*, 34(2):254–273, 2008.

A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017.

Z. Bar-Joseph, G. K. Gerber, D. K. Gifford, T. S. Jaakkola, and I. Simon. Continu-

ous representations of time-series gene expression data. *Journal of Computational Biology*, 10(3-4):341–356, 2003.

A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: Massive online analysis. *Journal of Machine Learning Research*, 11:1601–1604, 2010.

A. Bifet, J. Read, I. Žliobaitė, B. Pfahringer, and G. Holmes. Pitfalls in benchmarking data stream classification and how to avoid them. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 465–479. Springer, 2013.

R. Bos, S. De Waele, and P. M. Broersen. Autoregressive spectral estimation by application of the Burg algorithm to irregularly sampled data. *IEEE Transactions on Instrumentation and Measurement*, 51(6):1289–1294, 2002.

D. S. Bouhlila and F. Sellaouti. Multiple imputation using chained equations for missing data in TIMSS: a case study. *Large-scale Assessments in Education*, 1(1): 4, 2013.

G. E. P. Box, G. M. Jenkins, and G. C. Reinsel. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

P. M. T. Broersen. Automatic spectral analysis with missing data. *Digital Signal Processing*, 16(6):754–766, 2006.

O. Cappé, E. Moulines, and T. Rydén. Inference in Hidden Markov Models. In *Proceedings of EUSFLAT Conference*, pages 14–16, 2009.

H. Caussinus. Models and uses of principal component analysis. *Multidimensional data analysis*, 86:149–170, 1986.

K. P. Chan and W. C. Fu. Efficient time series matching by wavelets. In *Proceedings of the 15th International Conference on Data Engineering*, pages 126–133. IEEE, 1999.

C. Chatfield. *The Analysis of Time Series: An Introduction*. Chapman and Hall, 2004.

J. Chau and R. von Sachs. Functional mixed effects wavelet estimation for spectra of replicated time series. *Electronic Journal of Statistics*, 10(2):2461–2510, 2016.

H. Cho and P. Fryzlewicz. Multiple changepoint detection for high dimensional time series via sparsified binary segmentation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77(2):475–507, 2015.

T. G. Clark and D. G. Altman. Developing a prognostic model in the presence of missing data: an ovarian cancer case study. *Journal of Clinical Epidemiology*, 56 (1):28–37, 2003.

A. Cohen, I. Daubechies, and P. Vial. Wavelets on the Interval and Fast Wavelet Transforms. *Applied and Computational Harmonic Analysis*, 1(1):54–81, 1993.

R. R. Coifman and D. L. Donoho. Translation-invariant de-noising. In *Wavelets and Statistics*, pages 125–150. Springer, 1995.

R. R. Coifman and M. V. Wickerhauser. Entropy-based algorithms for best basis selection. *IEEE Transactions on information theory*, 38(2):713–718, 1992.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

S.D. Cranstoun, H.C. Ombao, R. Von Sachs, W. Guo, and B. Litt. Time-frequency spectral estimation of multichannel EEG using the auto-SLEX method. *IEEE Transactions on Biomedical Engineering*, 49(9):988–996, 2002.

R. Dahlhaus. Fitting time series models to non-stationary processes. *The Annals of Statistics*, 25(1):1–37, 1997.

R. Dahlhaus. A likelihood approximation for locally stationary processes. *The Annals of Statistics*, 28(6):1762–1794, 2000.

R. Dahlhaus. Locally stationary processes. In *Handbook of Statistics*, volume 30, pages 351–413. Elsevier, 2012.

I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 41(7):909–996, 1988.

I. Daubechies. *Ten Lectures on Wavelets*. SIAM, 1992.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, pages 1–38, 1977.

B. Doucoure, K. Agbossou, and A. Cardenas. Time series prediction using artificial wavelet neural network and multi-resolution analysis: Application to wind speed data. *Renewable Energy*, 92:202–211, 2016.

A. Ellmauthaler, M. E. Willis, X. Wu, and M. LeBlanc. Noise Sources in Fiber-optic Distributed Acoustic Sensing VSP Data. In *79th EAGE Conference and Exhibition 2017*, 2017.

Y. Ephraim and N. Merhav. Hidden Markov Processes. *IEEE Transactions on Information Theory*, 48(6):1518–1569, 2002.

B. L. Ford. An overview of hot-deck procedures. *Incomplete Data in Sample Surveys*, 2 (Part IV):185–207, 1983.

C. Fraley, A. E. Raftery, L. Scrucca, T. B. Murphy, and M. Fop. *mclust: Gaussian Mixture Modelling for Model-Based Clustering, Classification, and Density Estimation*, 2017. URL `http://cran.r-project.org/package=mclust`.

P. Fryzlewicz and G. P. Nason. Haar-Fisz Estimation of Evolutionary Wavelet Spectra. *Journal of the Royal Statistical Society, Series B (Methodological)*, 68(4):611–634, 2006.

P. Fryzlewicz and H. C. Ombao. Consistent classification of non-stationary time series using stochastic wavelet representations. *Journal of the American Statistical Association*, 104(485):299–312, 2009.

P. Fryzlewicz, S. Van Bellegem, and R. Von Sachs. Forecasting non-stationary time

series by wavelet process modelling. *Annals of the Institute of Statistical Mathematics*, 55(4):737–764, 2003.

P. Fryzlewicz, T. Sapatinas, and S. Subba Rao. A Haar–Fisz technique for locally stationary volatility estimation. *Biometrika*, 93(3):687–704, 2006.

T-C. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

P. J. García-Laencina, J. L. Sancho-Gómez, and A. R. Figueiras-Vidal. Pattern classification with missing data: a review. *Neural Computing and Applications*, 19(2): 263–282, 2010.

J. F. Gemmeke, H. Van Hamme, B. Cranen, and L. Boves. Compressive sensing for missing data imputation in noise robust speech recognition. *IEEE Journal of selected topics in Signal Processing*, 4(2):272–287, 2010.

Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an EM approach. *In Advances in Neural Information Processing Systems*, pages 120–127, 1994.

J.W. Graham. Missing data analysis: Making it work in the real world. *Annual review of psychology*, 60:549–576, 2009.

S. Greenland and W. D. Finkle. A critical look at methods for handling missing covariates in epidemiologic regression analyses. *American Journal of Epidemiology*, 142(12):1255–1264, 1995.

Y. Grenier. Time-Dependent ARMA Modeling of Nonstationary Signals. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 31(4):899–911, 1983.

J. Haworth and T. Cheng. Non-parametric regression for spacetime forecasting under missing data. *Computers, Environment and Urban Systems*, 36:538–550, 2012.

L. Himmelmann. *HMM: HMM - Hidden Markov Models*, 2010. URL `https://cran.r-project.org/package=HMM`. R package version 1.0.

B. Hirst, D. Randell, M. Jones, P. Jonathan, B. King, and M. Dean. A new technique for monitoring the atmosphere above onshore carbon storage projects that can estimate the locations and mass emission rates of detected sources. *Energy Procedia*, 114:3716–3728, 2017.

E. E. Holmes, E. J. Ward, M. Scheuerell, and K. Wills. *MARSS: Multivariate Autoregressive State-Space Modeling*, 2018. URL `https://CRAN.R-project.org/package=MARSS`.

J. Honaker and G. King. What to do about missing values in time–series cross–section data. *American Journal of Political Science*, 54(2):561–581, 2010.

J. Honaker, G. King, and M. Blackwell. Amelia II: A Program for Missing Data. *Journal of Statistical Software*, 45(7):1–47, 2011.

J. Honaker, G. King, and M. Blackwell. *Amelia: A Program for Missing Data*, 2015. URL `https://CRAN.R-project.org/package=Amelia`.

H. Y. Huang, H. C. Ombao, and D. S. Stoffer. Discrimination and classification of nonstationary time series using the SLEX model. *Journal of the American Statistical Association*, 99(467):763–774, 2004.

A. T. Hudak, N. L. Crookston, J. S. Evans, D. E. Hall, and M. J. Falkowski. Nearest neighbor imputation of species-level, plot-scale forest structure attributes from LiDAR data. *Remote Sensing of Environment*, 112(5):2232–2245, 2008.

F. Husson and J. Josse. *missMDA: Handling Missing Values with Multivariate Data Analysis*, 2018. URL `https://CRAN.R-project.org/package=missMDA`.

R. J. Hyndman. *forecast: Forecasting Functions for Time Series and Linear Models*, 2017. URL `https://CRAN.R-project.org/package=forecast`.

H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer. Random survival forests. *The Annals of Applied Statistics*, 2(3):841–860, 2008.

C. R. Jenkins, R. Leuning, and Z. M. Loh. Atmospheric tomography to locate CO2 leakage at storage sites. *Energy Procedia*, 4:3502–3509, 2011.

R. H. Jones. Maximum likelihood fitting of ARMA models to time series with missing observations. *Technometrics*, 22(3):389–395, 1980.

J. Josse, J. Pagés, and F. Husson. Multiple imputation in principal component analysis. *Advances in data analysis and classification*, 5(3):231–246, 2011.

W. Junger and A. P. de Leon. *mtsdi: Multivariate Time Series Data Imputation*, 2018. URL `https://CRAN.R-project.org/package=mtsdi`.

W. L. Junger and A. P. de Leon. Imputation of missing data in time series for air pollutants. *Atmospheric Environment*, 102:96–104, 2015.

H. Junninen, H. Niska, K. Tuppurainen, J. Ruuskanen, and M. Kolehmainen. Methods for imputation of missing values in air quality data sets. *Atmospheric Environment*, 38(18):2895–2907, 2004.

A. Kampouraki, G. Manis, and C. Nikou. Heartbeat time series classification with support vector machines. *IEEE Transactions on Information Technology in Biomedicine*, 13(4):512–518, 2009.

J. Kersten. Simultaneous feature selection and Gaussian mixture model estimation for supervised classification problems. *Pattern Recognition*, 47(8):2582–2595, 2014.

R. Killick, I. A Eckley, and P. Jonathan. A wavelet-based approach for detecting changes in second order structure within nonstationary time series. *Electronic Journal of Statistics*, 7:1167–1183, 2013.

R. Killick, M. Knight, G. P. Nason, and I. A. Eckley. The local partial autocorrelation function and some applications. *In submission*, 2018.

M. I. Knight, M. A. Nunes, and G.P. Nason. Spectral estimation for locally stationary time series with missing observations. *Statistics and Computing*, 22(4):877–895, 2012.

K. Krzemieniewska, I. A. Eckley, and P. Fearnhead. Classification of non-stationary time series. *Stat*, 3(1):144–157, 2014.

P. Laguna, G. B. Moody, and R. G. Mark. Power spectral density of unevenly sampled data by least-square analysis: performance and application to heart rate signals. *IEEE Transactions on Biomedical Engineering*, 45(6):698–715, 1998.

D. Y. C. Leung, G. Caramanna, and M. M. Maroto-Valer. An overview of current status of carbon dioxide capture and storage technologies. *Renewable and Sustainable Energy Reviews*, 39:426–443, 2014.

Z. H. Levine, A. L. Pintar, J. T. Dobler, N. Blume, M. Braun, T. S. Zaccheo, and T. G. Pernini. The detection of carbon dioxide leaks using quasi-tomographic laser absorption spectroscopy measurements in variable wind. *Atmospheric measurement techniques*, 9:1627, 2016.

Y. Li and L. E. Parker. Nearest neighbor imputation using spatialtemporal correlations in wireless sensor networks. *Information Fusion*, 15:64–79, 2014.

R. Little and D. B. Rubin. Statistical Analysis with Missing Data (Second Edition). 2002.

F. Lobato, C. Sales, I. Araujo, V. Tadaiesky, L. Dias, L. Ramos, and A. Santana. Multi-objective genetic algorithm for missing data imputation. *Pattern Recognition Letters*, 68:126–131, 2015.

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

A. K. Luhar, D. M. Etheridge, R. Leuning, Z. M. Loh, C. R. Jenkins, and E. Yee. Locating and quantifying greenhouse gas emissions at a geological CO2 storage site

using atmospheric modeling and measurements. *Journal of Geophysical Research: Atmospheres*, 119(18), 2014.

H. Lütkepohl. *New introduction to multiple time series analysis (Corr. 2nd printing. ed.)*. Berlin: Springer, 2007.

S. G. Mallat. Multiresolution approximations and wavelet orthonormal bases of $L^2$ (R). *Transactions of the American Mathematical Society*, 315(1):69–87, 1989a.

S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 11 (7):674–693, 1989b.

A. Mateeva, J. Lopez, H. Potters, J. Mestayer, B. Cox, D. Kiyashchenko, P. Wills, S. Grandi, K. Hornman, B. Kuvshinov, and W. Berlang. Distributed acoustic sensing for reservoir monitoring with vertical seismic profiling. *Geophysical Prospecting*, 62(4):679–692, 2014.

G. McLachlan and D. Peel. *Finite Mixture Models*. John Wiley & Sons, 2004.

D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, F. Leisch, C.-C. Chang, and C.-C. Lin. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2015. URL `http://cran.r-project.org/package=e1071`.

Y. Meyer. Principe d'incertitude, bases hilbertiennes et algebres d'operateurs. *Séminaire Bourbaki*, 662:209–223, 1986.

Y. Meyer. *Wavelets and Operators*. Cambridge University Press, 1992.

F. Mörchen. Time series feature extraction for data mining using DWT and DFT. Technical Report 33, University of Marburg, 2003.

S. Moritz. *imputeTS: Time Series Missing Value Imputation*, 2018. URL `https://CRAN.R-project.org/package=imputeTS`.

S. Moritz and T. Bartz-Beielstein. imputeTS: Time Series Missing Value Imputation in R. *R Journal*, 9(1):207–218, 2017.

S. Moritz, A. Sardá, T. Bartz-Beielstein, M. Zaefferer, and J. Stork. Comparison of different methods for univariate time series imputation in R. *arXiv preprint arXiv:1510.03924*, 2015.

K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions On Neural Networks*, 12(2): 181–201, 2001.

C. F. H. Nam, J. A. D. Aston, I. A. Eckley, and R. Killick. The uncertainty of storm season changes: Quantifying the uncertainty of autocovariance changepoints. *Technometrics*, 57(2):194–206, 2015.

G. P. Nason. *locits: Test of Stationarity and Localized Autocovariance*, 2016. URL `https://CRAN.R-project.org/package=locits`. R package version 1.7.3.

G. P. Nason. *wavethresh: Wavelets Statistics and Transforms*, 2016. URL `https://cran.r-project.org/package=wavethresh`. R package version 4.6.8.

G. P. Nason and B. W. Silverman. The discrete wavelet transform in S. *Journal of Computational and Graphical Statistics*, 3(2):163–191, 1994.

G. P. Nason and B. W. Silverman. The stationary wavelet transform and some statistical applications. *Lecture Notes in Statistics*, 103:281–300, 1995.

G. P. Nason, R. Von Sachs, and G Kroisandt. Wavelet processes and adaptive estimation of the evolutionary wavelet spectrum. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(2):271–292, 2000.

G. P. Nason, B. Powell, D. Elliott, and P. A. Smith. Should we sample a time series more frequently?: decision support via multirate spectrum estimation. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 180(2):353–407, 2017.

H. Ombao, J. Raz, R. Von Sachs, and W. Guo. The SLEX model of a non-stationary random process. *Annals of the Institute of Statistical Mathematics*, 54(1):171–200, 2002.

H. Ombao, R. Von Sachs, and W. Guo. SLEX analysis of multivariate nonstationary time series. *Journal of the American Statistical Association*, 100(470):519–531, 2005.

A. Owen, G. Duckworth, and J. Worsley. OptaSense: fibre optic distributed acoustic sensing for border monitoring. In *Intelligence and Security Informatics Conference (EISIC), 2012 European*, pages 362–364. IEEE, 2012.

R. Paleja, D. Mustafina, T. Park, D. Randell, J. van der Horst, R. Crickmore, et al. Velocity Tracking for Flow Monitoring and Production Profiling Using Distributed

Acoustic Sensing. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2015.

T. Park, I. A. Eckley, and H. C. Ombao. Estimating time-evolving partial coherence between signals via multivariate locally stationary wavelet processes. *IEEE Transactions on Signal Processing*, 62(20):5240–5250, 2014.

T. Park, I. A. Eckley, and H. C. Ombao. Dynamic Classification using Multivariate Locally Stationary Wavelets. *Signal Processing*, 152:118–129, 2018.

D. B. Percival and A. T. Walden. *Wavelet Methods for Time Series Analysis (Vol. 4)*. Cambridge University Press, 2006.

A. Plaia and A. L. Bondi. Single imputation method of missing values in environmental pollution data sets. *Atmospheric Environment*, 40(38):7316–7330, 2006.

R. J. Povinelli, M. T. Johnson, A. C. Lindgren, and J. Ye. Time series classification using Gaussian mixture models of reconstructed phase spaces. *IEEE Transactions on Knowledge and Data Engineering*, 16(6):779–783, 2004.

I. Pratama, A. E. Permanasari, I. Ardiyanto, and R. Indrayani. A review of missing values handling methods on time-series data. In *Information Technology Systems and Innovation (ICITSI), 2016 International Conference on*, pages 1–6. IEEE, 2016.

M. B. Priestley. Evolutionary spectra and non-stationary processes. *Journal of the Royal Statistical Society, Series B (Methodological)*, pages 204–237, 1965.

M. B. Priestley. Spectral Analysis and Time Series. 1981.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

J. Read, A. Bifet, G. Holmes, and B. Pfahringer. Scalable and efficient multi-label classification for evolving data streams. *Machine Learning*, 88(1-2):243–272, 2012.

A. Rosenberg and J. Hirschberg. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, volume 7, pages 410–420, 2007.

D. B. Rubin. Multiple Imputation for Nonresponse in Surveys (Wiley Series in Probability and Statistics). 1987.

J. A. Ryan, J. M. Ulrich, R. Bennett, and C. Joy. *xts: eXtensible Time Series*, 2018. URL `https://CRAN.R-project.org/package=xts`.

J. Sanderson, P. Fryzlewicz, and M. W. Jones. Estimating linear dependence between nonstationary time series using the locally stationary wavelet model. *Biometrika*, 97(2):435–446, 2010.

J. D. Scargle. Studies in astronomical time series analysis. II-Statistical aspects of spectral analysis of unevenly spaced data. *The Astrophysical Journal*, 263:835–853, 1982.

R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications*. Springer New York, 2000.

P. Smaragdis, B. Raj, and M. Shashanka. Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values. *Journal of Signal Processing Systems*, 65(3):361–370, 2011.

S. Sridevi, S. Rajaram, C. Parthiban, S. SibiArasan, and C. Swadhikar. Imputation for the analysis of missing values and prediction of time series data. In *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*, pages 1158–1163. IEEE, 2011.

D. J. Stekhoven. *missForest: Nonparametric Missing Value Imputation using Random Forest*, 2013. URL `https://CRAN.R-project.org/package=missForest`.

D. J. Stekhoven and P. Bühlmann. MissForest–non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2011.

C. Stărică and C. Granger. Nonstationarities in Stock Returns. *The Review of Economics and Statistics*, 87(3):503–522, 2005.

T. Subba Rao. The Fitting of Non-Stationary Time-Series Models with Time-Dependent Parameters. *Journal of the Royal Statistical Society, Series B (Methodological)*, pages 312–322, 1970.

W. Sweldens. The Lifting Scheme: A custom-design construction of biorthogonal wavelets. *Applied and computational harmonic analysis*, 3(2):186–200, 1996.

W. Sweldens. The Lifting Scheme: A Construction of Second Generation Wavelets. *SIAM Journal on Mathematical Analysis*, 29(2):511–546, 1998.

J. Tang, G. Zhang, Y. Wang, H. Wang, and F. Liu. A hybrid approach to integrate fuzzy C-means based imputation method with genetic algorithm for missing traffic volume data estimation. *Transportation Research Part C: Emerging Technologies*, 51:29–40, 2015.

S. Taylor, T. Park, I. A. Eckley, and R. Killick. *mvLSW: Multivariate Locally Stationary Wavelet Process Estimation*, 2017. URL `https://cran.r-project.org/package=mvLSW`. R package version 1.2.1.

S. J. Taylor. Modelling Financial Time Series (Second Edition). 2007.

M. Templ, A. Alfons, A. Kowarik, and B. Prantner. *VIM: Visualization and Imputation of Missing Values*, 2017. URL `https://CRAN.R-project.org/package=VIM`.

G. Tiao and R. Tsay. Model Specification in Multivariate Time Series. *Journal of the Royal Statistical Society, Series B (Methodological)*, 51(2):157–213, 1989.

A. A. Trindade. Implementing modified Burg algorithms in multivariate subset autoregressive modeling. *Journal of Statistical Software*, 8(1):1–68, 2003.

R. S. Tsay. *MTS: All-Purpose Toolkit for Analysing Multivariate Time Series (MTS) and Estimating Multivariate Volatility Models*, 2015. URL `https://CRAN.R-project.org/package=MTS`.

J. Tuikkala, L. L. Elo, O. S. Nevalainen, and T. Aittokallio. Missing value imputation

improves clustering and interpretation of gene expression microarray data. *BMC Bioinformatics*, 9(1):202, 2008.

S. Van Bellegem and R. Von Sachs. Locally Adaptive Estimation of Evolutionary Wavelet Spectra. *The Annals of Statistics*, 36(4):1879–1924, 2008.

S. van Buuren and K. Groothuis-Oudshoorn. *mice: Multivariate Imputation by Chained Equations*, 2018. URL `https://CRAN.R-project.org/package=mice`. R package version 3.3.0.

H. P. A. Van Dongen, E. Olofsen, J. H. Van Hartevelt, and E. W. Kruyt. A procedure of multiple period searching in unequally spaced time-series with the Lomb–Scargle method. *Biological Rhythm Research*, 30(2):149–177, 1999.

B. Vidakovic. *Statistical Modeling by Wavelets*. New York, Wiley, 1999.

P. L. Walker. *The Theory of Fourier Series and Integrals*. Wiley, 1986.

L. Wei and E. Keogh. Semi-supervised time series classification. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 748–753. ACM, 2006.

L. Wen, W. Cui, A. M. Levine, and H. V. Bradt. Orbital modulation of X-rays from Cygnus X-1 in its hard and soft states. *The Astrophysical Journal*, 525(2):968–977, 1999.

P. Whittle. *Hypothesis Testing in Time Series Analysis*. Almquist and Wiksell, 1951.

M. V. Wickerhauser. *Adapted Wavelet Analysis from Theory to Software*. AK Peters Series. Taylor & Francis, 1994.

J. Wijfells. *RMOA: Connect R with MOA to perform streaming classifications*, 2014. URL `https://github.com/jwijffels/RMOA`. R package version 1.0.

J. Williams. Distributed Acoustic Sensing for Pipeline Monitoring. *Pipeline & Gas Journal*, 239(7), 2012.

S. F. Wu, C. Y. Chang, and S. J. Lee. Time series forecasting with missing values. In *Industrial Networks and Intelligent Systems (INISCom), 2015 1st International Conference on*, pages 151–156. IEEE, 2015.

G. U. Yule. On a method of investigating periodicities disturbed series, with special reference to Wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 226(636-646):267–298, 1927.

A. Zeileis, G. Grothendieck, J. A. Ryan, J. M. Ulrich, and F. Andrews. *zoo: S3 Infrastructure for Regular and Irregular Time Series*, 2018. URL `https://CRAN.R-project.org/package=zoo`. R package version 1.8-3.

H. Zhang, T. B. Ho, and M. S. Lin. A Non-parametric Wavelet Feature Extractor for Time Series Classification. In H. Dai, R. Srikant, and C. Zhang, editors, *Advances in Knowledge Discovery and Data Mining*, pages 595–603. Springer Berlin Heidelberg, 2004.

W. Zucchini and I. L. MacDonald. *Hidden Markov Models for Time Series: an Introduction using R.* New York, Chapman and Hall-CRC, 2009.