Chapter 4
SSA for Multivariate Time Series

In this chapter we consider the problem of simultaneous decomposition, reconstruction, and forecasting for a collection of time series from the viewpoint of SSA. The main method of this chapter is usually called either Multichannel SSA or Multivariate SSA, shortly MSSA. The principal idea of the algorithm is the same as for Basic SSA, the difference is in the way of how the trajectory matrix is constructed. The aim of MSSA is to take into consideration the combined structure of a multivariate series to obtain more accurate results.

MSSA is usually considered as an extension of 1D-SSA. However, the algorithm of MSSA was published even earlier than the algorithm of 1D-SSA; see Weare and Nasstrom (1982), where MSSA was named Extended Empirical Orthogonal Function (EEOF) analysis. The MSSA algorithm in the framework of SSA was formally formulated in Broomhead and King (1986b). Here we consider the algorithm of MSSA for the analysis and forecasting of multivariate time series following the approach described in Golyandina et al. (2001; Chapter 2) for one-dimensional series and in Golyandina and Stepanov (2005) for multidimensional ones.

Section 4.1 starts this chapter by describing the complex-valued version of 1D-SSA (called Complex SSA), which is a natural generalization of 1D-SSA for the analysis and forecasting of a system of two time series (Keppenne and Lall 1996).

In Sect. 4.2 we expand the methodology of Chap. 2 for the SSA analysis of a system of several time series. It is important to note that there are two main ways of stacking individual trajectory matrices into a joint trajectory matrix: horizontal stacking and vertical stacking. For the MSSA analysis, these two stacking procedures are equivalent.

Section 4.3 considers forecasting in MSSA. There are four main variants of MSSA forecasting: recurrent column forecasting, recurrent row forecasting, vector column forecasting, and vector row forecasting. We carefully describe the commonalities and differences between these four variants and make their comparison on a simulated data. Finally, in Sect. 4.4 we discuss features of the MSSA analysis,

N. Golyandina et al., *Singular Spectrum Analysis with R*, Use R!,
https://doi.org/10.1007/978-3-662-57380-8_4

forecasting and data filling on several real-world data sets. The examples considered in Sects. 4.3 and 4.4 and the discussion of Sect. 4.3.4 demonstrate that essentially all the techniques that have been developed in Chaps. 2 and 3 for 1D-SSA can be naturally extended to the multivariate case. This concerns, in addition to the SSA analysis and forecasting, the practical problems of smoothing, filtering, imputation of missing values, estimation of parameters of the signal, and also subspace tracking for monitoring stability and change-point detection.

4.1 Complex SSA

Any real-valued 1D-SSA variation can be transferred to the complex-valued case. At present, only Basic 1D-SSA is implemented in the Rssa package in the complex-valued form. Therefore, in this section we briefly discuss the complex-valued version of Basic 1D-SSA and call it Complex SSA. A comparison of Complex SSA with other methods of multivariate SSA will be made in subsequent sections of this chapter.

4.1.1 Method

Assume that a system of two time series of the same length N is given. Then we can consider the one-dimensional complex-valued series $\mathbb{X} = \mathbb{X}^{(1)} + \mathrm{i}\mathbb{X}^{(2)}$ and apply the complex version of 1D-SSA to this one-dimensional series. Since the Basic SSA algorithm in Sect. 2.1 is written in the real-valued form, there is some difference in the form of the SVD performed in the complex-valued space, where the transposition should be Hermitian.

Also, there is a specificity related to the uniqueness of the SVD expansion. If the singular values are different, then the SVD is unique up to multiplication of left and right singular vectors by c, where $|c| = 1$. In the real-valued case, $c = \pm 1$, while in the complex-valued case there are infinitely many complex numbers c with $|c| = 1$.

Complex-valued SSA forecasting and parameter estimation are straightforward extensions of the corresponding techniques for the real-valued time series. In the current version of Rssa, forecasting (recurrent and vector ones), Cadzow iterations and gap-filling are implemented but the parameter estimation and the shaped version of SSA are not.

4.1.2 Separability

Separability in Complex SSA is defined exactly as in the real-valued 1D-SSA variations. However, the conditions for separability are different. Conditions for

Complex SSA separability of time series are more restrictive than the separability conditions for the one-dimensional series (Golyandina et al. 2015; Appendix A.1). In particular, the following sufficient condition for weak separability is valid.

Proposition 4.1 *If time series* $\mathbb{F}^{(1)}$ *and* $\mathbb{F}^{(2)}$, $\mathbb{G}^{(1)}$ *and* $\mathbb{G}^{(2)}$, $\mathbb{F}^{(1)}$ *and* $\mathbb{G}^{(2)}$, *and also* $\mathbb{G}^{(1)}$ *and* $\mathbb{F}^{(2)}$ *are weakly L-separable by 1D-SSA, then the complex-valued time series* $\mathbb{F}^{(1)} + \mathrm{i}\mathbb{F}^{(2)}$ *and* $\mathbb{G}^{(1)} + \mathrm{i}\mathbb{G}^{(2)}$ *are weakly L-separable for Complex SSA.*

The conditions of Proposition 4.1 can be extended to conditions for asymptotic separability ($N \to \infty$) and therefore for approximate separability for fixed N.

The most important difference between 1D-SSA and Complex SSA is related to the separability of imaginary exponential functions with the term $s_n = Ae^{\mathrm{i}(2\pi\omega n+\phi)} = A\cos(2\pi\omega n + \phi) + \mathrm{i}A\sin(2\pi\omega n + \phi)$, $0 < \omega < 0.5$. Such series have the Complex SSA-rank 1, which is smaller than the 1D-SSA-ranks of real and imaginary parts of these functions; these ranks are equal to 2. Recall that the rank of a series is equal to the rank of its trajectory matrix constructed by the chosen method. In particular, this feature implies that Complex SSA provides better separability of imaginary exponential functions, in comparison with other 1D-SSA variations, see Golyandina and Stepanov (2005). Note that Complex SSA for extraction of imaginary exponential functions is used as a step of the "f-xy eigenfiltering" method for noise suppression in stacked 3-D volumes of seismic traces (Trickett 2003).

Another difference is related to the eigenvalues produced by the complex exponentials like $s_n = A\cos(2\pi\omega n + \phi_1) + \mathrm{i}B\sin(2\pi\omega n + \phi_2)$. In contrast to Basic SSA, such time series frequently produce quite different eigenvalues depending on relation between amplitudes and phases (Golyandina et al. 2015; Appendix A.1). This can influence strong separability, either for the better or for the worse.

4.1.3 Algorithm

Complex SSA formally differs from Basic SSA, presented in Sect. 2.1.4, by the use of the Hermitian transpose, which we will denote by "*."

Algorithm 4.1 Complex SSA: decomposition

Input: Complex-valued time series $\mathbb{X}$ of length N and rank d, window length L.
Output: Decomposition of the trajectory matrix on elementary matrices $\mathbf{X} = \mathbf{X}_1 + \ldots + \mathbf{X}_d$, where where $d = \operatorname{rank}\mathbf{X}$ and $\mathbf{X}_i = \sqrt{\lambda_i}U_iV_i^*$.
1: Construct the trajectory matrix $\mathbf{X} = \mathcal{T}_{\mathrm{SSA}}(\mathbb{X})$.
2: Compute the SVD $\mathbf{X} = \mathbf{X}_1 + \ldots + \mathbf{X}_d$, $\mathbf{X}_i = \sqrt{\lambda_i}U_iV_i^*$.

Note that despite the difference between Algorithms 2.1 and 4.1 is just in the change of "T" by "*," numerical complex-valued algorithms can be much more complicated and less stable.

The reconstruction algorithm in Complex SSA is standard, see Algorithm 2.2.

Since the algorithms of forecasting for Complex SSA are very similar to that in the real-valued case, we do not formulate them here.

4.1.4 Complex SSA in RSSA

4.1.4.1 Description of Functions

A typical call of the `ssa` function has the form

```
s <- ssa(x, L = (N + 1) %/% 2, kind = "cssa", svd.method = "svd")
```

where `N` is the series length.

Arguments:

`x` is an object to be decomposed. For Complex SSA it is assumed to be a simple vector or vector-like object of complex numbers. Everything else is coerced to vector.

`L` is a window length. By default it is fixed to half of the series length.

`kind` specifies the kind of SSA to apply.

`svd.method` selects the SVD method to use. Unlike for the case of real-valued SSA, only straightforward implementations of the complex SVD at Decomposition step are included into the RSSA package (called `svd` and `eigen`).

Since Complex SSA is an extension of Basic SSA to the complex-valued case, calls of `reconstruct`, `forecast`, and `cadzow` functions are exactly the same as in the real-valued case; see details in Sect. 2.1.5.

4.1.4.2 Typical Code

The following code demonstrates how to extract trends from two series simultaneously. The data "Stocks" includes daily closing prices of major European stock indices, 1991–1998, included into the RSSA package. The first series is related to Germany DAX (Ibis), the second series contains data for Switzerland SMI.

Fragment 4.1.1 ("Stocks": Reconstruction)

```
> library("Rssa")
> s <- ssa(EuStockMarkets[, 1] + 1i*EuStockMarkets[, 2],
+          kind = "cssa", svd.method = "svd")
> r <- reconstruct(s, groups = list(Trend = 1:2))
> plot(r, plot.method = "xyplot", layout = c(2, 3))
> plot(s, type = "vectors", idx = 1:8)
> len = 2
> print(rforecast(s, groups = list(Trend = 1:2), len = len)[1:len])
[1] 6156.061+8492.425i 6169.006+8507.808i
```

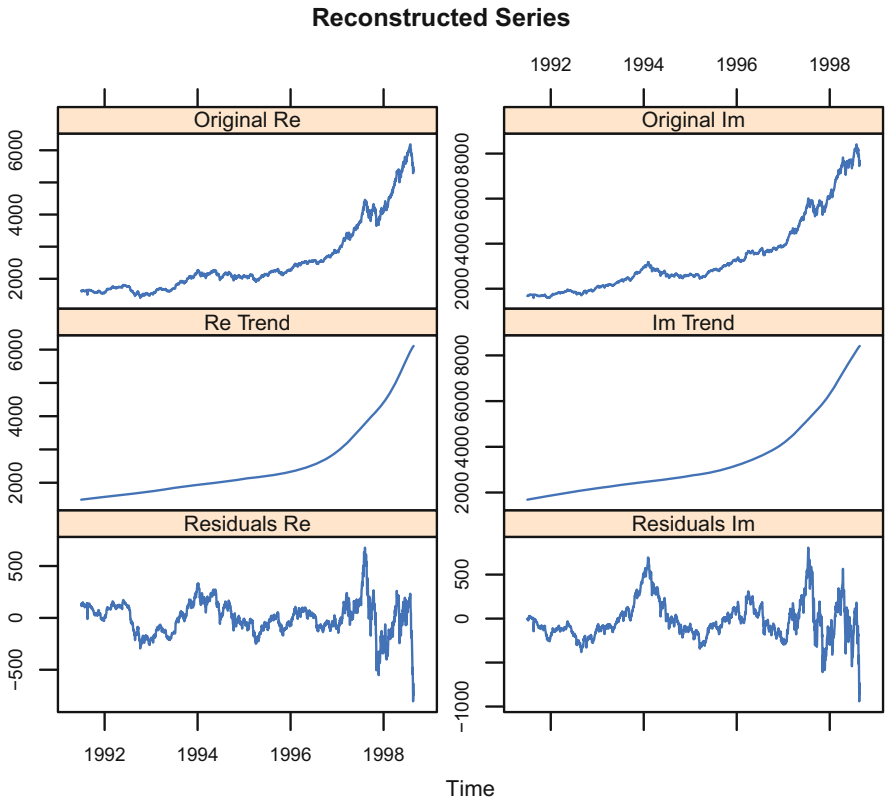

Fig. 4.1 "Stocks": Reconstructed trends

The estimated common trends are depicted in Fig. 4.1.

We choose ET1–2 by analyzing eigenvectors shown in Fig. 4.2. Similarly to the real-valued case, for trend extraction we should specify eigentriples with slowly-varying real and imaginary parts of eigenvectors; they are depicted using different colors. Note that the real part refers to the first series and the imaginary part corresponds to the second series.

However, paired scatterplots can no longer be used for detecting the sine-wave components, since eigenvectors are defined up to multiplication by a unit complex number and therefore the real and imaginary parts of two eigenvectors can differ by an arbitrary phase not equal to $\pi/2$.

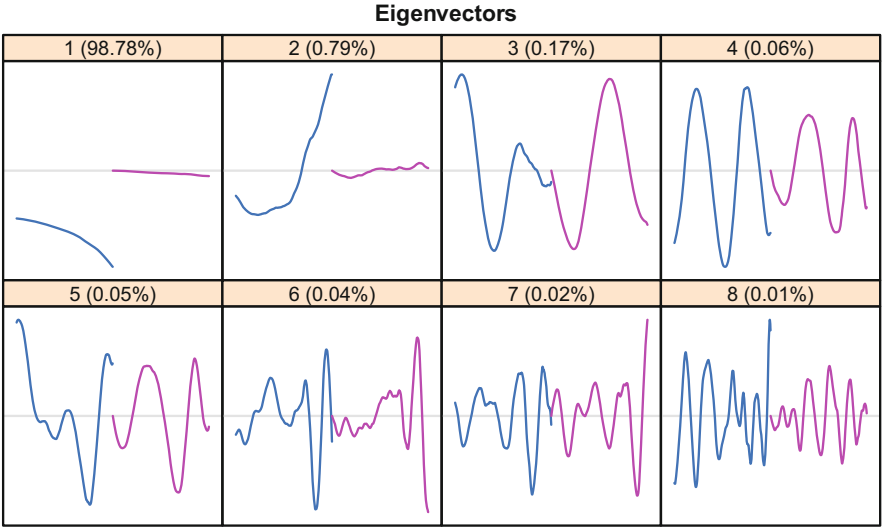

Fig. 4.2 "Stocks": Eigenvectors, real and imaginary parts

4.2 MSSA Analysis

4.2.1 Method

Consider a multivariate time series; that is, a collection $\{\mathbb{X}^{(p)} = \left(x_j^{(p)}\right)_{j=1}^{N_p}, \quad p = 1, \ldots, s\}$ of s time series of length N_p, $p = 1, \ldots, s$.

Denote $\mathbb{X} = (\mathbb{X}^{(1)}, \ldots, \mathbb{X}^{(s)})$ the initial data for the MSSA algorithm. The generic scheme of the algorithm described in Sect. 1.1 holds for MSSA; we only need to define the embedding operator $\mathfrak{T}_{\mathrm{MSSA}}(\mathbb{X}) = \mathbf{X}$.

4.2.1.1 Embedding

Let L be an integer called window length, $1 < L < \min(N_p, p = 1, \ldots, s)$. For each time series $\mathbb{X}^{(p)}$, we form $K_p = N_p - L + 1$ L-lagged vectors $X_j^{(p)} = (x_j^{(p)}, \ldots, x_{j+L-1}^{(p)})^{\mathrm{T}}$, $1 \le j \le K_p$. Denote $K = \sum_{p=1}^{s} K_p$. The *trajectory matrix* of the multidimensional series $\mathbb{X}$ is a matrix of size $L \times K$ and has the form

$$\mathfrak{T}_{\mathrm{MSSA}}(\mathbb{X}) = \mathbf{X} = [X_1^{(1)} : \ldots : X_{K_1}^{(1)} : \ldots : X_1^{(s)} : \ldots : X_{K_s}^{(s)}] = [\mathbf{X}^{(1)} : \ldots : \mathbf{X}^{(s)}], \quad (4.1)$$

where $\mathbf{X}^{(p)} = \mathfrak{T}_{\mathrm{SSA}}(\mathbb{X}^{(p)})$ is the trajectory matrix of the one-dimensional series $\mathbb{X}^{(p)}$ defined by (2.1). Thus, the trajectory matrix of a system of time series has

stacked Hankel structure. Note that

$$\mathfrak{T}^{-1}_{\mathrm{MSSA}}(\mathbf{X}) = [\mathfrak{T}^{-1}_{\mathrm{SSA}}(\mathbf{X}^{(1)}) : \ldots : \mathfrak{T}^{-1}_{\mathrm{SSA}}(\mathbf{X}^{(s)})]. \tag{4.2}$$

Let us make an important comment concerning Embedding step formulated above.

In papers devoted to MSSA, Embedding step can differ. First, in some papers, including papers on climatological applications of MSSA (see, e.g., Broomhead and King (1986); Allen and Robertson (1996); Hannachi et al. (2007)), the trajectory matrix is transposed; that is, the trajectory matrices $\mathbf{X}^{(p)}$ are stacked vertically. We stack them horizontally with the purpose of getting the same structure of the column space in MSSA as in 1D-SSA. Since the time series can have different lengths, one dimension of their trajectory matrices $\mathbf{X}^{(p)}$, $p = 1, \ldots, s$, is the same, while the other dimension can differ. We call the coinciding dimension the window length L and stack the matrices horizontally to obtain the trajectory space (the column subspace of the trajectory matrix, i.e., the space produced by L-lagged vectors of the system of series) of dimension L.

In the horizontally-stacked case, the column trajectory space of a system of identical series coincides with the trajectory space of one time series, while in the vertically-stacked case it is not so. Also, the horizontal stacking is consistent with the continuation of time series, since the increase of series lengths changes the number of lagged vectors and does not change their dimension.

The discussion on similarities and dissimilarities of the horizontally-stacked and vertically-stacked versions of MSSA will be continued at the end of this section (see Remarks 5 and 6) and in Sect. 4.2.4.1.

4.2.1.2 Decomposition

The conventional rank-one matrix decomposition at Decomposition step of MSSA is constructed by applying the SVD to the trajectory matrix; that is, the standard MSSA is an extension of Basic SSA and therefore it can be called Basic MSSA.

Oblique modifications of MSSA are the same as in the 1D case; that is, one can perform nested decompositions by Iterative O-SSA and Filter-adjusted O-SSA. The use of these nested variations is exactly the same as in the 1D case and we refer the reader to Sects. 2.4 and 2.5 for details.

4.2.1.3 Reconstruction

Since $\mathfrak{M}^{(\mathrm{H})}_{L,K}$ in MSSA is the set of stacked Hankel matrices, the orthogonal projector $\Pi_{\text{stacked } \mathcal{H}}$ to $\mathfrak{M}^{(\mathrm{H})}_{L,K}$ has the form

$$\Pi_{\text{stacked } \mathcal{H}}(\mathbf{Y}) = [\Pi_{\mathcal{H}}(\mathbf{Y}^{(1)}) : \ldots : \Pi_{\mathcal{H}}(\mathbf{Y}^{(s)})], \tag{4.3}$$

where $\Pi_{\mathcal{H}}$ is defined in (2.2). The equality (4.3) follows from the general form of the projection described in Sect. 1.1.2.6. Similar to the 1D case, the reconstructed series are obtained by means of the composition of $\mathfrak{T}^{-1}_{\mathrm{MSSA}}$ and $\Pi_{\text{stacked }\mathcal{H}}$.

4.2.1.4 Comments

Let us make some comments concerning characteristic features of MSSA.

The eigenvectors $\{U_i\}$ in the SVD of the trajectory matrix $\mathbf{X} = \sum_i \sqrt{\lambda_i} U_i V_i^{\mathrm{T}}$ form the common basis of the column trajectory spaces of all time series from the system. Factor vectors $\{V_i\}$ (often called extended empirical orthogonal functions (EEOF) in climatology applications, starting from Weare and Nasstrom (1982)) consist of parts related to each time series separately; that is,

$$V_i = \begin{pmatrix} V_i^{(1)} \\ \vdots \\ V_i^{(s)} \end{pmatrix}, \tag{4.4}$$

where the pth factor subvector $V_i^{(p)} \in \mathsf{R}^{K_p}$ belongs to the row trajectory space of the pth series.

The eigenvectors U_i reflect the common features of time series, while the factor subvectors $V_i^{(p)}$ show how these common features appear in each series. It is natural to transform a factor vector to a *factor system* of factor subvectors $V_i^{(p)}$. Then the form of transformed factor vectors will be similar to the initial system of series.

Similarly to the one-dimensional case, the main result of application of MSSA is a decomposition of the multivariate time series into a sum of m multivariate series; the parameters are the window length L and the way of grouping. For the frequently used case of two groups, we denote by $\widetilde{\mathbb{X}}^{(k)} = \left(\tilde{x}_j^{(k)}\right)_{j=1}^{N}$, $k = 1, \ldots, s$, the reconstructed series (usually, the signal) corresponding to the first group of eigentriples I_1.

4.2.1.5 Remarks

1. The indexing of time points $1, \ldots, N_p$ $(p = 1, \ldots, s)$ starting from 1 does not mean that all s series start at the same time; they can also finish at different times. The resultant decomposition obtained by the MSSA algorithm does not depend on the shift between the one-dimensional series and therefore this indexing is only a formality. In particular, MSSA decompositions of two one-dimensional series measured at the same time interval and at disjoint time intervals do not differ.

2. The original time ranges for series $\mathbb{X}^{(p)}$ can be useful for depicting and interpreting them. The reconstructed series have the same time ranges as the original series. Factor subvectors from the factor system can also be synchronized for plotting based on the ranges of the initial series; although factor vectors are shorter than the initial series, their time shifts are the same.
3. For simultaneous analysis of several time series, it is recommended to transfer them into the same scale. Otherwise, the structure of one particular time series will have too much influence on the MSSA results. To balance the time series, they can be either standardized (centered and normalized, in additive models) or only normalized (in multiplicative models). From the other viewpoint, scaling of individual series can be used to influence the importance of a particular series of the system when, for example, this particular series is more important or has a smaller noise component.
4. The MSSA algorithm can be modified in the same ways as the 1D-SSA algorithm. For example, Toeplitz MSSA and MSSA with projection (including centering) can be considered. (However, these options are not implemented in the current version of Rssa.) Nested oblique variations of 1D-SSA (Iterative O-SSA and Filter-adjusted O-SSA) are implemented in Rssa.
5. In climatology, the SVD of the transposed (vertically-stacked) trajectory matrix defined in (4.1) is traditionally considered (Hannachi et al. 2007) as the trajectory matrix. Therefore, the eigenvectors $\{U_i\}$ correspond to normalized extended principal components in Hannachi et al. (2007), while the factor vectors $\{V_i\}$ are called Extended Empirical Orthogonal Functions (EEOFs).
6. The computational cost of the SVD at Decomposition step depends on the size of the matrix $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ and hence this computational cost may be significantly different for the horizontally-stacked and vertically-stacked versions of MSSA.

4.2.2 Multi-Dimensional Time Series and LRRs

The model of a system of time series which well suits MSSA is related to times series governed by LRRs. Instead of one LRR of the form (1.8) in the 1D case, see Sects. 1.4 and 2.1.2.2, we have a system of LRRs, which can reflect similarity of time series in the system.

Consider a system of infinite time series $\mathbb{X}^{(1)}, \dots, \mathbb{X}^{(s)}$, choose the window length L, and denote $\mathcal{X}^{(1)}, \dots, \mathcal{X}^{(s)}$ the column trajectory spaces of the series. Let $\mathcal{X} = \mathrm{span}(\mathcal{X}^{(1)}, \dots, \mathcal{X}^{(s)})$ be the column trajectory space of the collection of time series $(\mathbb{X}^{(1)}, \dots, \mathbb{X}^{(s)})$. Similarly to the 1D case, we call the dimension of the trajectory space $\mathcal{X}$ (equal to the rank of the trajectory matrix $\mathbf{X}$ of the series collection) the MSSA-rank of the series collection, see Sect. 1.1.2 for short description of general notions.

Denote the 1D-SSA-ranks of $\mathbb{X}^{(l)}$ by $r_l = \dim \mathcal{X}^{(l)} \le L$, $l = 1, \ldots, s$. For each time series $\mathbb{X}^{(l)}$, we can write the minimal LRR governing the series:

$$x^{(l)}_{j+r_l} = \sum_{k=1}^{r_l} a^{(l)}_k x^{(l)}_{j+r_l-k}, \quad \text{where} \quad a^{(l)}_{r_l} \neq 0, \quad l = 1, \ldots, s. \tag{4.5}$$

The characteristic polynomials of the LRRs (4.5) are

$$P^{(l)}_{r_l}(\mu) = \mu^{r_l} - \sum_{k=1}^{r_l} a^{(l)}_k \mu^{r_l-k}, \quad l = 1, \ldots, s. \tag{4.6}$$

Recall that the roots of the characteristic polynomials of the minimal LRRs governing the series are called signal roots.

Let

$p^{(l)}$ be the number of different roots of the polynomial $P^{(l)}_{r_l}(\mu)$,
$\mu^{(l)}_m$ be the m-th root of the polynomial $P^{(l)}_{r_l}(\mu)$,
$k^{(l)}_m$ be the multiplicity of the root $\mu^{(l)}_m$.

Then

$$k^{(l)}_1 + \ldots + k^{(l)}_{p^{(l)}} = r_l, \quad l = 1, \ldots, s.$$

The characteristic roots determine the series behavior. For example, if $k^{(l)}_m = 1$, then

$$x^{(l)}_n = \sum_{m=1}^{r_l} C^{(l)}_m \left(\mu^{(l)}_m\right)^n.$$

Also let

$\mu_1, \ldots, \mu_p$ be the pooled set of roots of polynomials $P^{(1)}_{r_1}, \ldots, P^{(s)}_{r_s}$,
$k_1, \ldots, k_p$ be the multiplicities of the roots $\mu_1, \ldots, \mu_p$,

where multiplicity of a root in the pooled set is equal to the maximum of multiplicities of the corresponding roots in the initial sets.

Since the roots are determined by the structure of the trajectory space, the following proposition holds, see Golyandina et al. (2015; Appendix A.2).

Proposition 4.2 *Let $r = \sum_{i=1}^{p} k_i < L$. Then the rank of the infinite multi-dimensional time series $(\mathbb{X}^{(1)}, \mathbb{X}^{(2)}, \ldots, \mathbb{X}^{(s)})$ is equal to r.*

Consider a simple example.

Example 4.1 Let $\mathbb{F}^{(1)} = (f_1^{(1)}, \ldots, f_N^{(1)})$ and $\mathbb{F}^{(2)} = (f_1^{(2)}, \ldots, f_N^{(2)})$ with

$$f_k^{(1)} = A\cos(2\pi\omega_1 k + \varphi_1), \quad f_k^{(2)} = B\cos(2\pi\omega_2 k + \varphi_2), \tag{4.7}$$

where $0 < \omega < 1/2$, $0 \le \varphi_1, \varphi_2 < 2\pi$ and $A, B \ne 0$. Let us fix the window length $L > 4$ and find the 1D-SSA-rank of the time series $\mathbb{F}^{(1)}$, the MSSA-rank of the system $(\mathbb{F}^{(1)}, \mathbb{F}^{(2)})$ and the Complex SSA-rank of $\mathbb{F}^{(1)} + \mathrm{i}\mathbb{F}^{(2)}$:

1. For $\omega_1 = \omega_2$, the 1D-SSA ranks of $\mathbb{F}^{(1)}$ and $\mathbb{F}^{(2)}$, as well as the MSSA-rank of $(\mathbb{F}^{(1)}, \mathbb{F}^{(2)})$, are equal to 2. The Complex SSA-rank of $\mathbb{F}^{(1)} + \mathrm{i}\mathbb{F}^{(2)}$ is equal to 1 if $A = B$ and $|\varphi_1 - \varphi_2| = \pi/2 \bmod \pi$ and is equal to 2 otherwise.
2. For $\omega_1 \ne \omega_2$ the 1D-SSA-rank of $\mathbb{F}^{(1)}$ and $\mathbb{F}^{(2)}$ is equal to 2. The MSSA rank of $(\mathbb{F}^{(1)}, \mathbb{F}^{(2)})$ and the Complex SSA-rank of $\mathbb{F}^{(1)} + \mathrm{i}\mathbb{F}^{(2)}$ are both equal to 4.

4.2.2.1 Matching of Series

Simultaneous analysis of several time series is usually performed to identify their inter-relation and to extract their common structure. Recall that for 1D-SSA, a time series has a structure if and only if the trajectory matrix of this series is rank-deficient. Certainly, for a typical real-world series, the trajectory matrix has full rank. Therefore, in what follows we talk about the rank of signal (a part of times series with structure) or its components.

Consider a system of signals $\mathbb{H} = (\mathbb{H}^{(1)}, \mathbb{H}^{(2)})$ with a rank-deficient trajectory matrix. The structure of a series is reflected in its trajectory space. We can say that two time series have the same structure if their trajectory spaces coincide. For example, for two sine waves with equal periods their trajectory spaces coincide, whatever the values of their amplitudes and phases. This follows from the fact that the trajectory space is the span of subseries of length L of the initial series. On the other hand, sine waves with different frequencies have entirely different structure and the combined trajectory space of their system is a direct sum of the series trajectory spaces.

If two time series are fully matched, then the trajectory space of one time series can be used for reconstructing or forecasting of the second series. If two series are unrelated and have totally different structure, then neither series contains any useful information about the other series for the MSSA analysis.

For MSSA, any shift between two time series and any difference between phases of two matched sine waves have no influence on the result of analysis. Therefore, one cannot say anything about the direction of causality. Moreover, asymmetry of influence of one time series to the another series can be caused by different levels of noise. However, the time series $\mathbb{X}^{(2)}$ can be called *supportive* for the time series $\mathbb{X}^{(1)} = \mathbb{H}^{(1)} + \mathbb{R}^{(1)}$ if the accuracy of either reconstruction or forecasting of $\mathbb{H}^{(1)}$ improves if we analyze the system of two series $\mathbb{X} = (\mathbb{X}^{(1)}, \mathbb{X}^{(2)})$ rather than the series $\mathbb{X}^{(1)}$ alone.

Numerical experiments confirm that for two matching signals the series with any level of noise, which is not larger than the other one, is always supportive (see, e.g., Sect. 4.3.3.3).

4.2.3 Separability

The notion of separability for multidimensional time series is similar to that for one-dimensional series; the latter was briefly considered in Sect. 1.1.2 and thoroughly described in Golyandina et al. (2001; Sections 1.5 and 6.1).

Separability is the key notion in the SSA theory. Indeed, separability of $\mathbb{H}$ from $\mathbb{R}$ means the ability of SSA to extract $\mathbb{H}$ from the sum $\mathbb{H}+\mathbb{R}$. Recall that there is a weak separability, which means orthogonality of the trajectory spaces, and a strong separability which is equivalent to empty intersection of the sets of singular values produced by the series which we are trying to separate.

Conditions of separability of multidimensional time series are more restrictive than that for one-dimensional series. The following sufficient condition of weak separability is valid (exactly the same as for Complex SSA, see Sect. 4.1.2), see (Golyandina et al. 2015; Appendix A.1).

Proposition 4.3 *If time series $\mathbb{F}^{(1)}$ and $\mathbb{F}^{(2)}$, $\mathbb{G}^{(1)}$ and $\mathbb{G}^{(2)}$, $\mathbb{F}^{(1)}$ and $\mathbb{G}^{(2)}$, and also $\mathbb{G}^{(1)}$ and $\mathbb{F}^{(2)}$ are weakly L-separable by 1D-SSA, then the two-dimensional time series $(\mathbb{F}^{(1)}, \mathbb{F}^{(2)})$ and $(\mathbb{G}^{(1)}, \mathbb{G}^{(2)})$ are weakly L-separable by MSSA.*

Proposition 4.3 can be extended to cover the case of asymptotic separability (as series lengths $N_i \to \infty$) and therefore approximate separability for fixed large N_i.

Example 4.2 Consider an example of four harmonic real-valued time series $\mathbb{F}^{(1)}$, $\mathbb{F}^{(2)}$, $\mathbb{G}^{(1)}$, and $\mathbb{G}^{(2)}$ of length N:

$$f_k^{(1)} = A_1 \cos(2\pi\omega_1 k + \varphi_1), \qquad f_k^{(2)} = B_1 \cos(2\pi\omega_1 k + \varphi_2),$$

$$g_k^{(1)} = A_2 \cos(2\pi k\omega_2 k + \phi_1), \qquad g_k^{(2)} = B_2 \cos(2\pi k\omega_2 k + \phi_2),$$

$\omega_1 \neq \omega_2$, $k = 0, \ldots, N-1$, $A_1, A_2, B_1, B_2 \neq 0$. If $L\omega_i$ and $K\omega_i$ $(i = 1, 2)$ are integers, then $(\mathbb{F}^{(1)}, \mathbb{F}^{(2)})$ and $(\mathbb{G}^{(1)}, \mathbb{G}^{(2)})$ are L-separable by MSSA.

Note that if either $L\omega_i$ or $K\omega_i$ (or both) is not integer, then the two series are not L-separable; however, asymptotic (and approximate for finite lengths) separability takes place.

Weak separability is not enough for extraction of time series components. Therefore, let us look at strong separability related to eigenvalues produced by time series components. It appears that the same pair of time series $(\mathbb{F}^{(1)}, \mathbb{F}^{(2)})$ can produce different eigenvalues in 1D-SSA, MSSA, and Complex SSA. Therefore, by applying a more suitable multivariate extension of 1D-SSA we can improve strong separability.

Example 4.3 Let

$$f_k^{(1)} = A\cos(2\pi\omega k + \varphi_1), \qquad f_k^{(2)} = B\cos(2\pi k\omega k + \varphi_2).$$

If $L\omega$ and $K\omega$ are integers, then $(\mathbb{F}^{(1)}, \mathbb{F}^{(2)})$ produces two equal eigenvalues in MSSA: $\lambda_1 = \lambda_2 = (A^2+B^2)LK/4$. This implies that there is no strong separability in the respective version of MSSA. Note, however, that there is strong separability in this example for Complex SSA, see Golyandina et al. (2015).

4.2.4 Comments on 1D-SSA, MSSA and Complex SSA

4.2.4.1 Covariance Structure

Consider in more detail the case of two time series $\mathbb{X} = (\mathbb{F}, \mathbb{G})$ and let $\mathbf{F}$ and $\mathbf{G}$ be the trajectory matrices of $\mathbb{F}$ and $\mathbb{G}$ correspondingly. Then, since in MSSA we stack the individual trajectory matrices horizontally, the trajectory matrix of $\mathbb{X}$ is $\mathbf{X} = [\mathbf{F} : \mathbf{G}]$. In accordance with (2.3), the SVD of $\mathbf{X} = \mathbf{X}^{(\mathrm{H})}$ is $\mathbf{X} = \sum_i \sqrt{\lambda_i} U_i V_i^{\mathrm{T}}$, where λ_i and U_i are eigenvalues and eigenvectors of the matrix $\mathbf{S} = \mathbf{S}^{(\mathrm{H})}_{\mathrm{MSSA}} = \mathbf{X}\mathbf{X}^{\mathrm{T}} = \mathbf{F}\mathbf{F}^{\mathrm{T}} + \mathbf{G}\mathbf{G}^{\mathrm{T}}$ and $V_i = \mathbf{X}^{\mathrm{T}} U_i/\sqrt{\lambda_i}$.

Consider now the vertical stacking of the trajectory matrices of $\mathbb{F}$ and $\mathbb{G}$ in the trajectory matrix: $\mathbf{X}^{(\mathrm{V})} = \begin{pmatrix}\mathbf{F}^{\mathrm{T}}\\ \mathbf{G}^{\mathrm{T}}\end{pmatrix} = \left(\mathbf{X}^{(\mathrm{H})}\right)^{\mathrm{T}}$.

The SVD of $\mathbf{X}^{(\mathrm{V})}$ is then $\mathbf{X}^{(\mathrm{V})} = \sum_i \sqrt{\lambda_i} V_i U_i^{\mathrm{T}}$, the transposed SVD of $\mathbf{X}$. Here λ_i, V_i, and U_i are exactly the same as above but now they have different interpretation: in particular, V_i (they are often called EEOFs, see Remark 5 in Sect. 4.2.1) are the eigenvectors of

$$\mathbf{S}^{(\mathrm{V})}_{\mathrm{MSSA}} = \mathbf{X}^{(\mathrm{V})}\left(\mathbf{X}^{(\mathrm{V})}\right)^{\mathrm{T}} = \begin{pmatrix}\mathbf{F}^{\mathrm{T}}\mathbf{F} & \mathbf{F}^{\mathrm{T}}\mathbf{G}\\ \mathbf{G}^{\mathrm{T}}\mathbf{F} & \mathbf{G}^{\mathrm{T}}\mathbf{G}\end{pmatrix}.$$

The last formula clearly demonstrates the relation between the two versions (horizontal stacking and vertical stacking) of MSSA and shows that MSSA takes into consideration cross-covariances of time series (more precisely, we obtain the cross-covariances if centering of the one-dimensional series is done at the preprocessing stage).

Consider now Complex SSA. Since the eigendecomposition of a complex-valued matrix $\mathbf{A} + \mathrm{i}\mathbf{B}$ can be reduced to the eigendecomposition of the real-valued matrix

$$\mathbf{D} = \begin{pmatrix}\mathbf{A} & -\mathbf{B}\\ \mathbf{B} & \mathbf{A}\end{pmatrix},$$

in the case of Complex SSA we in fact analyze eigenvectors of the matrix

$$\mathbf{S}_{\mathrm{CSSA}} = \begin{pmatrix} \mathbf{F}^{\mathrm{T}}\mathbf{F} & \mathbf{F}^{\mathrm{T}}\mathbf{G} \\ \mathbf{G}^{\mathrm{T}}\mathbf{F} & \mathbf{G}^{\mathrm{T}}\mathbf{G} \end{pmatrix} + \begin{pmatrix} \mathbf{G}^{\mathrm{T}}\mathbf{G} & -\mathbf{G}^{\mathrm{T}}\mathbf{F} \\ -\mathbf{F}^{\mathrm{T}}\mathbf{G} & \mathbf{F}^{\mathrm{T}}\mathbf{F} \end{pmatrix}.$$

We can observe that the structures of the two one-dimensional series in Complex SSA are mixed more than in MSSA.

4.2.4.2 Separability

Conditions of separability for multidimensional time series are more restrictive than that for one-dimensional series. In particular, a sufficient condition for separability of a two-series system is the separability of each series from the first collection with each series from the second one. However, for matched signals, their (weak) 1D-SSA-separability from noise can be considerably improved by their simultaneous MSSA analysis.

Since weak separability is not enough for extraction of time series components, we should pay attention to strong separability related to eigenvalues produced by the time series components. It appears (see Example 4.3) that the two-dimensional time series $(\mathbb{F}^{(1)}, \mathbb{F}^{(2)})$ typically produce different eigenvalues in 1D-SSA, MSSA, and Complex SSA. Therefore, an application of more suitable multidimensional version of SSA can improve strong separability. However, non-matching of one-dimensional time series in a system of series increases the number of eigenvalues related to the signal and hence increases the chance of mixing the signal with the residual. To overcome this effect, the modification DerivSSA (see Sect. 2.5 for the 1D case), which is able to considerably improve strong separability, is implemented in Rssa for the MSSA analysis of a collection of time series.

4.2.4.3 Ranks

Rank of a signal is a very important notion in SSA, since it reflects the complexity of signals and hence the difficulty of the problem of their extraction. For MSSA and Complex SSA, the notions of the time series of finite rank and of time series satisfying LRRs are similar to the related notions in 1D-SSA, although the rank of the same time series may be different and depend on the method used. Let us compare 1D-SSA-, MSSA-, and Complex SSA-ranks, i.e., ranks of the corresponding trajectory matrices. By 1D-SSA-rank for a collection of time series we mean the rank of each one-dimensional series separately.

If two time series have the same structure (and therefore the same 1D-SSA-ranks), then the MSSA-rank is equal to the 1D-SSA-rank of each of the two series. The Complex SSA-rank can be even smaller than the individual 1D-SSA-ranks in the specific case of imaginary exponentials.

Consider a collection $\mathbb{H}^{(k)} = \left(h_j^{(k)}\right)_{j=1}^{N}$, $k = 1, \ldots, s$, of s signals of length N. Let r_k denote the 1D-SSA rank of $\mathbb{H}^{(k)}$ (i.e., the dimension of the trajectory space generated by one-dimensional SSA applied to this time series) and r denote the MSSA rank of $(\mathbb{H}^{(1)}, \ldots, \mathbb{H}^{(s)})$. The relation between r and r_k, $k = 1, \ldots, s$, is considered in Sect. 4.2.2. In particular, it is shown that $r_{\min} \le r \le r_{\max}$, where $r_{\min} = \max\{r_k, k = 1, \ldots, s\}$ and $r_{\max} = \sum_{k=1}^{s} r_k$. The case $r = r_{\max}$ is the least favorable for MSSA and means that different time series do not have matched components. The case $r < r_{\max}$ indicates the presence of matched components and hence simultaneous processing of the time series system can be more effective than their individual analysis.

In terms of matching, if all s series have the same characteristic roots (see Sect. 4.2.2 for the definition), then the time series $\mathbb{H}^{(m)}$, $m = 1, \ldots, s$, consist of additive components of the same 1D-SSA-structure. Such time series are fully matched. For fully matched time series, the MSSA-rank is much smaller than the sum of the 1D-SSA-ranks of the separate time series from the system. On the other hand, if the sets of characteristic roots do not intersect, then the time series have no common structure. In this case, the MSSA-rank is equal to the sum of the 1D-SSA-ranks of the separate time series from the system. For a typical system of real-world time series, we are in-between these two extreme cases.

4.2.4.4 Choice of the Window Length

The choice of the window length for one-dimensional SSA was reviewed in Sect. 2.1.3.2; see Golyandina et al. (2001; Section 1.6) and Golyandina (2010) for more thorough discussions. The problem of the choice of the window length in MSSA is more complicated than that in 1D-SSA. Until now there is no in-depth study of the problem of the choice of the optimal window length for analysis and, to an even greater extent, for forecasting of multidimensional time series. Moreover, the choice of the best window length for MSSA forecasting differs for different types of forecasting methods, see numerical comparison in Sect. 4.4. Some numerical investigation of this problem has been performed in Golyandina and Stepanov (2005), Golyandina et al. (2015); it is extended in Sect. 4.4.

By analogy with the one-dimensional case, we can formulate some key principles for the choice of L. The main principle is the same as for 1D-SSA and states that the choice of L should provide (approximate) separability of series. However, the MSSA case has additional features. Different approaches to the choice of the window length can be partly explained as follows. In 1D-SSA, it makes sense to constrain the window length to the interval $2 \le L \le [(N+1)/2]$, since the SVD expansions for window lengths L and $N - L + 1$ coincide. For the MSSA-analysis of more than one time series, the expansions for all possible window lengths $2 \le L \le \min_i N_i - 1$ are generally different. In particular, while in the 1D-SSA analysis it makes no sense to take $L > (N+1)/2$, in the MSSA analysis it makes

perfect sense choosing large L (and hence small $K_i = N_i - L + 1$) for trend extraction and smoothing.

Since L in 1D-SSA does not exceed half of the time series length, the divisibility of $L = \min(L, K)$ on possible periods of oscillations is recommended in 1D-SSA. In MSSA, $\min(L, K_i)$ is not necessarily equal to L and therefore one also has to pay attention to the values of K_i.

In 1D-SSA, the most detailed decomposition can be obtained if the trajectory matrix $\mathbf{X}$ has maximal rank. In the general case of SSA-family methods, this corresponds to the case of a square trajectory matrix. Thus, for a system of s time series of length N the window length in MSSA providing the square trajectory matrix $\mathbf{X}$ is approximately $sN/(s+1)$. For the case of two time series this corresponds to $2L/3$ for MSSA, while for Complex SSA applied to one complex-valued series this gives $N/2$.

Numerical investigations show that the formula $L \simeq sN/(s+1)$ is appropriate for the decomposition of a small number of time series (see simulation results in Sect. 4.4), but does not look suitable for the system of many short series (the values of K_i become too small for achieving separability). Generally, the choice $L \simeq N/2$ is still appropriate for MSSA.

Various special techniques can be transferred from 1D-SSA to MSSA, such as Sequential SSA, see Sect. 2.8 for examples of Sequential 1D-SSA analysis. Sequential 1D-SSA is based on successive application of 1D-SSA with different window lengths, see Golyandina and Zhigljavsky (2013; Section 2.5.5) for more details. Sequential MSSA can be applied in a similar manner. In addition to the reasons which are similar to the 1D case, we may find extra arguments in favor of Sequential MSSA. In particular, if trends of different one-dimensional series are of different structure, a smaller window length can be applied to achieve similarity of eigenvectors and to improve separability. After that, the residuals with a common structure (e.g., containing the seasonality) can be simultaneously decomposed with a larger window length.

4.2.5 Algorithm

The algorithm of MSSA decomposition differs from Algorithm 2.1 of Basic SSA decomposition only by the form of the embedding operator.

Algorithm 4.2 MSSA: decomposition

Input: Collection $\{\mathbb{X}^{(p)} = \left(x_j^{(p)}\right)_{j=1}^{N_p},\ p = 1, \ldots, s\}$ of s time series of length N_p, $p = 1, \ldots, s$, window length L.

Output: Decomposition of the trajectory matrix on elementary matrices $\mathbf{X} = \mathbf{X}_1 + \ldots + \mathbf{X}_d$, where $\mathbf{X}_i = \sqrt{\lambda_i} U_i V_i^{\mathrm{T}}$.

1: Construct the trajectory matrix $\mathbf{X} = \mathfrak{T}_{\mathrm{MSSA}}(\mathbb{X})$, where $\mathfrak{T}_{\mathrm{MSSA}}$ is defined by (4.1).
2: Compute the SVD $\mathbf{X} = \mathbf{X}_1 + \ldots + \mathbf{X}_d$, $\mathbf{X}_i = \sqrt{\lambda_i} U_i V_i^{\mathrm{T}}$.

Reconstruction stage is also very similar to Algorithm 2.2 for Basic SSA reconstruction.

Algorithm 4.3 MSSA reconstruction

Input: Decomposition $\mathbf{X} = \mathbf{X}_1 + \ldots + \mathbf{X}_d$, where $\mathbf{X}_i = \sigma_i U_i V_i^{\mathrm{T}}$ and $\|U_i\| = \|V_i\| = 1$, grouping $\{1, \ldots, d\} = \bigsqcup_{j=1}^{m} I_j$.

Output: Decomposition of the time series system on identifiable components $\mathbb{X} = \mathbb{X}_1 + \ldots + \mathbb{X}_m$.

1: Construct the grouped matrix decomposition $\mathbf{X} = \mathbf{X}_{I_1} + \ldots + \mathbf{X}_{I_m}$, where $\mathbf{X}_I = \sum_{i \in I} \mathbf{X}_i$.

2: $\mathbb{X} = \mathbb{X}_1 + \ldots + \mathbb{X}_m$, where $\mathbb{X}_i = \mathfrak{T}^{-1}_{\mathrm{MSSA}} \circ \Pi_{\text{stacked } \mathcal{H}}(\mathbf{X}_{I_i})$.

Recall that $\mathfrak{T}_{\mathrm{MSSA}}$ and $\Pi_{\text{stacked } \mathcal{H}}$ can be expressed through $\mathfrak{T}_{\mathrm{SSA}}$ and $\Pi_{\mathcal{H}}$ introduced in Chap. 2 for 1D-SSA (see Sect. 4.2.1).

4.2.6 MSSA Analysis in Rssa

4.2.6.1 Description of Functions

Typical call of `ssa` for the MSSA analysis is

```
s <- ssa(x, L = (min(N) + 1)%/%2, kind = "mssa")
```

where `N` is the vector of the series lengths.

Arguments:

`x` is an object containing a collection of time series to be decomposed.

`L` is a window length. By default it is fixed to half of the minimal series length.

`neig` is the number of desired eigentriples. If `neig = NULL`, a sane default value which depends on L and N will be used.

`kind` specifies the kind of SSA to apply.

`svd.method` selects the SVD method to use. Full description is given in Sect. 2.1.5.2.

Additional details and the description of the `reconstruct` function can be found in Sect. 2.1.5, since there is no difference between one-dimensional and multivariate cases here.

4.2.6.2 Typical Code

Here we demonstrate how the MSSA decomposition of a system of time series can be performed by means of the Rssa package. Since the analysis and forecasting of one-dimensional time series by Rssa are thoroughly described in previous chapters and in Golyandina and Korobeynikov (2013), we pay more attention to the differences between 1D-SSA and MSSA.

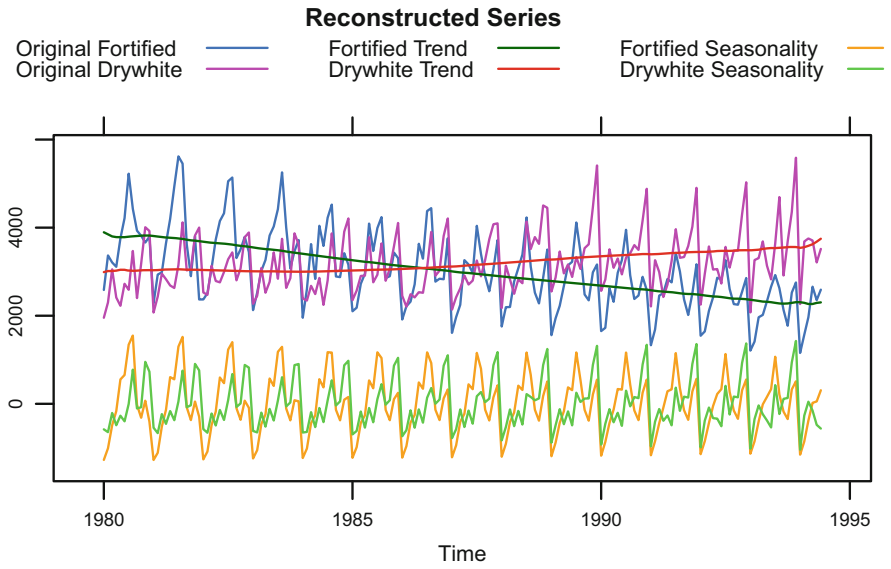

Fig. 4.3 "FORT" and "DRY": Reconstructed trend and seasonality

In Sect. 2.1.5.3 we decomposed the one-dimensional series "FORT" (sales of fortified wines) from the dataset "AustralianWine." Here we add one more series, the sales of dry wines (shortly "DRY"), for simultaneous analysis.

For loading the data we use the code from Fragment 2.1.1.

Fragment 4.2.1 ("FORT" and "DRY": Reconstruction)

```
> wineFortDry <- wine[, c("Fortified", "Drywhite")]
> L <- 84
> s.wineFortDry <- ssa(wineFortDry, L = L, kind = "mssa")
> r.wineFortDry <- reconstruct(s.wineFortDry,
+                              groups = list(Trend = c(1, 6),
+                                 Seasonality = c(2:5, 7:12)))
> plot(r.wineFortDry, add.residuals = FALSE,
+      plot.method = "xyplot",
+      superpose = TRUE, auto.key = list(columns = 3))
```

Fragment 4.2.1 contains a typical code for simultaneous extraction of the trend and seasonality (compare with Fragment 2.1.2) and produces Fig. 4.3. A clear difference between the two fragments is in the indicated value of the parameter `kind` in the `ssa` function. A more significant difference is related to plotting the results. For a multivariate series, there is, in a sense, a matrix of series, where one index is the series number in the system and the second index indicates the component number in the decomposition. The `plot` function for the reconstruction object allows to indicate which subset (slice) of this matrix one wants to depict by

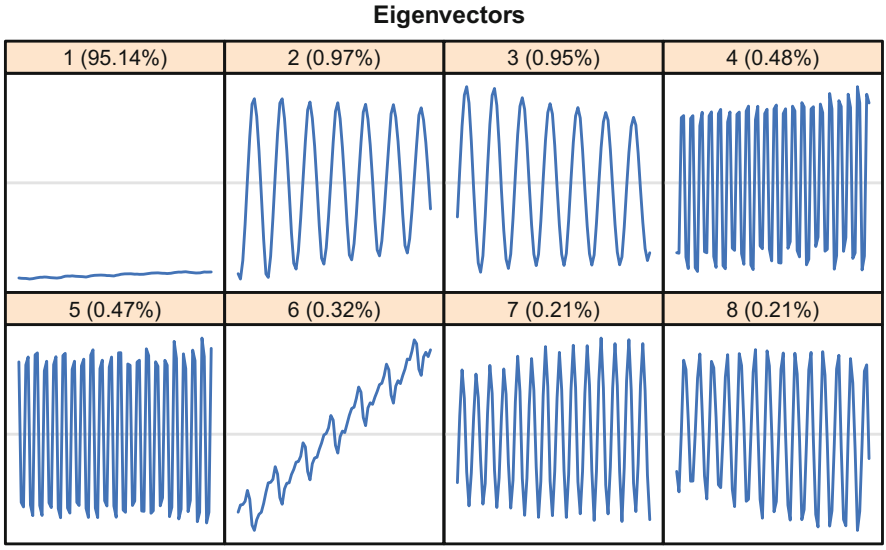

Fig. 4.4 "FORT" and "DRY": 1D graphs of eigenvectors

means of the parameter `slice`. The parameter `slice` consists of the list of indices of series and indices of decomposition components. The use of `slice` is demonstrated on several examples in Sects. 4.4.1 and 4.4.3.

The code for the component identification in MSSA is very similar to that in SSA, compare Fragments 4.2.2 and 2.1.3. The difference is in the structure of the factor vectors; however, they are not necessary for the identification. Figure 4.4 (compare it with Fig. 2.2) shows that the trend is described by ET1 and ET6, which is slightly mixed with seasonality. Figure 4.5 (compare with Fig. 2.3) demonstrates those pairs of ETs that are related to seasonality.

The results of MSSA analysis are similar to the results of 1D-SSA analysis. However, the separability is sometimes slightly worse for MSSA. Iterative O-SSA (Sect. 2.4) and DerivSSA (Sect. 2.5) can be applied to MSSA objects to improve separability. One can see in Fig. 4.6 (compare ET2 here with ET6 in Fig. 4.4) that after application of the Iterative O-SSA the trend is no longer mixed with seasonality. The eigentriples are reordered and the trend is described by the first two eigentriples. Note that if the signal components are mixed up between themselves, then the signal forecasting is not affected by this. However, if one wants to forecast the trend only, then the mixture would typically worsen the forecast accuracy.

Since the implemented methods of parameter estimation are based on eigenvectors only, they can be applied to eigenvectors in multidimensional case in exactly the same way as in the one-dimensional case.

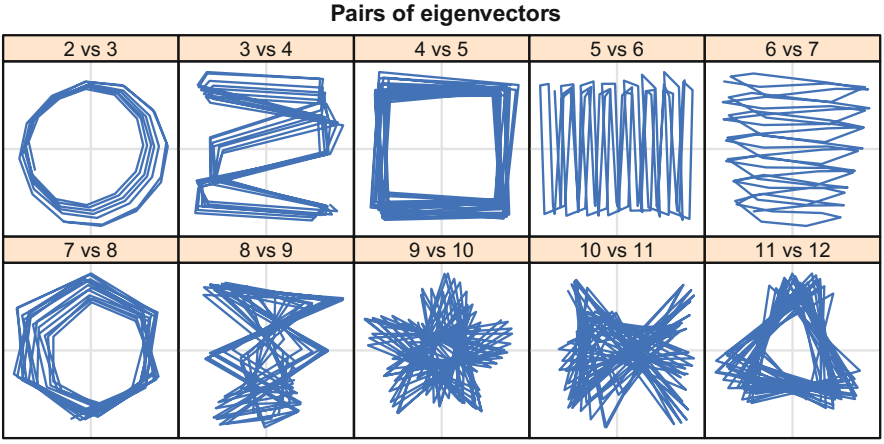

Fig. 4.5 "FORT" and "DRY": 2D scatterplots of eigenvectors

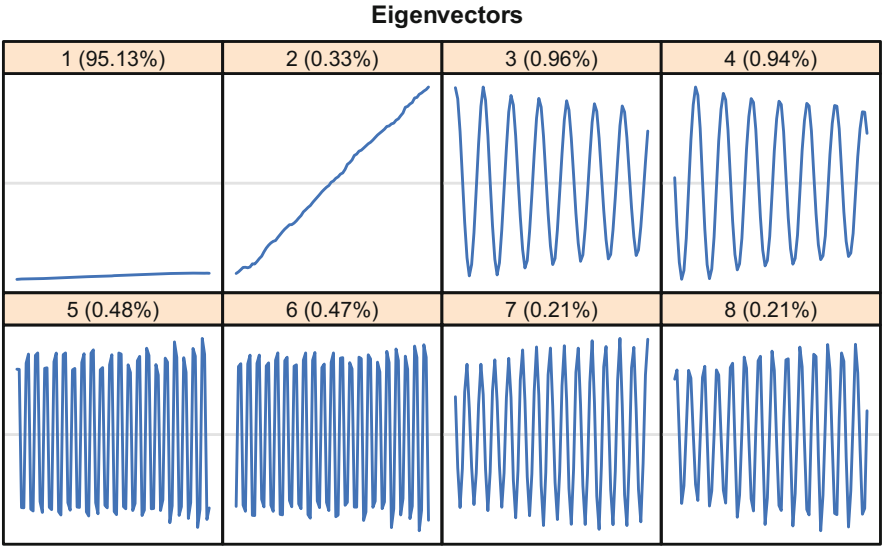

Fig. 4.6 "FORT" and "DRY": 1D graphs of eigenvectors after Iterative O-SSA

Fragment 4.2.2 ("FORT" and "DRY": Identification)

```
> plot(s.wineFortDry, type = "vectors", idx = 1:8)
> plot(s.wineFortDry, type = "paired", idx = 2:11,
+      plot.contrib = FALSE)
> print(parestimate(s.wineFortDry, groups = list(2:3, 4:5),
+                   method = "esprit"))
$F1
   period      rate    |     Mod      Arg   |      Re         Im
```

```
   12.128  -0.004789 |  0.99522   0.52 |  0.86463   0.49283
  -12.128  -0.004789 |  0.99522  -0.52 |  0.86463  -0.49283
$F2
   period     rate    |    Mod     Arg  |     Re        Im
    4.007  -0.001226 |  0.99877   1.57 |  0.00279   0.99877
   -4.007  -0.001226 |  0.99877  -1.57 |  0.00279  -0.99877
> plot(wcor(s.wineFortDry, groups = 1:30),
+      scales = list(at = c(10, 20, 30)))
> si.wineFortDry <- iossa(s.wineFortDry,
+                         nested.groups = list(c(1,6), c(2:5, 7:12)))
> plot(si.wineFortDry, type = "vectors", idx = 1:8)
```

4.2.6.3 Comments

Formats of Input and Output Data

While the representation of a one-dimensional time series in R is pretty obvious, there are many possible ways of defining a multivariate time series. Let us outline some common choices.

- A matrix with separate series in the columns. Optionally, an additional time structure like in `mts` objects can be embedded.
- A matrix-like (e.g., a `data.frame`) object with series in the columns. In particular, `data.frame` would be a result of reading the series from a file via the `read.table` function.
- A list of separate time series objects (e.g., a `list` of `ts` or `zoo` objects).

Also, the time scales of the individual time series can be normalized via head or tail padding with `NA` (for example, as a result of the `ts.union` call) or specified via time series attributes.

The package is designed to allow any of the input cases outlined above and produces the reconstructed series in the same format. All the attributes, names of the series, `NA` padding, etc. are carefully preserved. For forecasted series, the time scale attributes for several known time series objects (e.g., `ts`) are inferred automatically where possible.

The examples in Fragments 4.2.1 and 4.3.2 provide an overview of the possible input series formats.

Plotting Specifics

Plotting of the reconstructed series is performed by the function `plot` applied to the reconstructed collection of time series. The parameter `plot.method` can be `"native"` or `"xyplot"`. Keep in mind that the default (`"native"`) plotting method for reconstruction objects may or may not be suitable for multivariate time series plotting. For example, it provides many useful plotting possibilities for `ts` and `mts` objects, but may be totally unusable in the case of `data.frame` objects, because it will only call the `pairs` function on the resulting data frame at the end.

Efficient Implementation

All ideas from the one-dimensional case can be extended to the multivariate case. In the one-dimensional case, the complexity is determined by the series length N and the window length L and the worst case corresponds to $L \sim K \sim N/2$ with overall complexity of $O(L^3 + L^2K) = O(N^3)$.

In the multidimensional case (for simplicity assume that all the series have equal lengths N), the worst case corresponds to $L \sim K \sim sN/(s+1)$; the order of complexity is the same $O(N^3)$ but the constant can be considerably larger. Therefore, the speed-up (due to efficient implementation) giving the order $O(kN\log(N) + k^2N)$, where k is the number of calculated eigentriples, in the multivariate case can be much higher than in the one-dimensional case.

Note that MSSA can be viewed as a special case of Shaped 2D-SSA (see Sect. 5.2.1.3) and the current implementation in the package implicitly uses this.

4.3 MSSA Forecasting

Recall from Sect. 3.2 that forecasting in 1D-SSA is performed for a signal component which can be separated by 1D-SSA and is governed, perhaps approximately, by an LRR. For brevity, we will talk about forecasting of the whole signal. 1D-SSA provides an estimate of the signal subspace and thereby an estimate of one of LRRs governing the signal. The recurrent 1D-SSA forecasting continues the estimated signal by the estimated LRR. The vector 1D-SSA forecasting continues the reconstructed vectors in the given subspace.

Methods of one-dimensional SSA forecasting in a given subspace are described in Sect. 3.2. For CSSA, the forecasting algorithms are straightforward extensions of 1D-SSA forecasting algorithms to the complex-valued case, therefore we do not discuss them here. On the other hand, the methods of MSSA forecasting require special attention.

As in 1D-SSA, methods of MSSA forecasting can be subdivided into recurrent and vector forecasting. In contrast with 1D-SSA, rows and columns of the trajectory matrix in MSSA have different structure. Therefore, there exist two kinds of MSSA forecasting: row forecasting and column forecasting; this depends on which of the two spaces the forecasting is made (row or column space respectively). In total, there are four main variants of MSSA forecasting: recurrent column forecasting, recurrent row forecasting, vector column forecasting, and vector row forecasting.

There are different names for the same forecasting methods. In Golyandina and Stepanov (2005), column and row forecasting are called L- and K-forecasting. In Hassani and Mahmoudvand (2013), these methods are called horizontal and vertical forecasts and the trajectory matrix is transposed. In Sects. 4.2.1 and 4.2.4.1 we have explained the choice of orientation of the MSSA trajectory matrix and the connection between the horizontally-stacked and vertically-stacked trajectory matrices of separate time series. We use the name “column” and “row” with respect to the horizontally-stacked trajectory matrices as defined in Sect. 4.2.1.

In the column forecasting methods, each time series in the system is forecasted separately but in a given common subspace (i.e., using the common LRR). In the row forecasting methods, each series is forecasted with the help of its own LRR applied to the whole set of series from the system. Let us describe all four variants of MSSA forecasting.

4.3.1 Method

4.3.1.1 Common Notation

First, we introduce some common notation used for description of all the variants of MSSA forecasting.

Denote by $\overline{A} \in \mathsf{R}^{Q-1}$ the vectors consisting of the last $Q-1$ coordinates of $A \in \mathsf{R}^Q$; that is, the vectors with the first coordinate removed are indicated by the line on the top of the vector. Denote by $\underline{A} \in \mathsf{R}^{Q-1}$ the vectors consisting of the first $Q-1$ coordinates of A; by $\pi(A)$ we denote the last coordinate of the vector. For a matrix $\mathbf{A} = [A_1 : \ldots : A_r]$, we denote $\overline{\mathbf{A}} = [\overline{A}_1 : \ldots : \overline{A}_r]$ and $\underline{\mathbf{A}} = [\underline{A}_1 : \ldots : \underline{A}_r]$ and let $\boldsymbol{\pi}(\mathbf{A}) = (\pi(A_1), \ldots, \pi(A_r))^{\mathrm{T}}$ be the last row of the matrix $\mathbf{A}$.

Consider the following form of $B \in \mathsf{R}^K$, where $K = \sum_{i=1}^s K_i$, induced by the structure of the row trajectory space:

$$B = \begin{pmatrix} B^{(1)} \\ B^{(2)} \\ \vdots \\ B^{(s)} \end{pmatrix}, \qquad \underline{B} = \begin{pmatrix} \underline{B}^{(1)} \\ \underline{B}^{(2)} \\ \vdots \\ \underline{B}^{(s)} \end{pmatrix}, \qquad \overline{\underline{B}} = \begin{pmatrix} \overline{\underline{B}}^{(1)} \\ \overline{\underline{B}}^{(2)} \\ \vdots \\ \overline{\underline{B}}^{(s)} \end{pmatrix}, \tag{4.8}$$

where $B^{(j)} \in \mathsf{R}^{K_j}$, and let $\boldsymbol{\mu}(B) = (\pi(B^{(1)}), \ldots, \pi(B^{(s)}))^{\mathrm{T}}$. Also, for $\mathbf{B} = [B_1 : \ldots : B_r]$ let $\underline{\mathbf{B}} = [\underline{B}_1 : \ldots : \underline{B}_r]$ and $\mathbf{B}^{(j)} = [B_1^{(j)} : \ldots : B_r^{(j)}]$.

Assume that the group I corresponding to the forecasted component is given by the set of the leading components at Decomposition step of Algorithm 4.2; this assumption is made just for simplifying the formulas. Thus, let r leading eigentriples $(\sqrt{\lambda_j}, U_j, V_j)$ be identified and chosen as related to the signal of rank r so that $I = I_1 = \{1, \ldots, r\}$, $\mathbf{U} = [U_1 : \ldots : U_r]$, $\mathbf{V} = [V_1 : \ldots : V_r]$. The reconstructed series $\widetilde{\mathbb{X}}$, its trajectory matrix $\widetilde{\mathbf{X}}$, and the reconstructed matrix $\widehat{\mathbf{X}}$ are defined in Sect. 1.1.1. Define $\mathcal{L}^{\mathrm{col}} = \mathrm{span}(U_i, i \in I)$, $\mathcal{L}^{\mathrm{row}} = \mathrm{span}(V_i, i \in I)$. The reconstructed matrix $\widehat{\mathbf{X}} = [\widehat{\mathbf{X}}_1 : \ldots : \widehat{\mathbf{X}}_K]$ consists of the column vectors which are the projections of the column vectors of the trajectory matrix on the chosen subspace $\mathcal{L}^{\mathrm{col}}$.

To avoid repeating the transpose sign, denote $\widetilde{\mathbf{Y}} = [\widetilde{\mathbf{Y}}_1 : \ldots : \widetilde{\mathbf{Y}}_L] = \widetilde{\mathbf{X}}^{\mathrm{T}}$, $\widehat{\mathbf{Y}} = [\widehat{\mathbf{Y}}_1 : \ldots : \widehat{\mathbf{Y}}_L] = \widehat{\mathbf{X}}^{\mathrm{T}}$, $\widehat{\mathbf{Y}}_k = \widehat{\mathbf{X}}_k^{\mathrm{T}}$.

4.3.1.2 Recurrent MSSA Forecast

We denote the vector of forecasted signal values for each time series by $R_N = \left(\tilde{x}^{(1)}_{N_1+1}, \tilde{x}^{(2)}_{N_2+1}, \ldots, \tilde{x}^{(s)}_{N_s+1}\right)^{\mathrm{T}}$. Recurrent forecasting is closely related to missing data imputation for components of vectors from the given subspace and in fact uses the formula (1) from Golyandina and Osipov (2007). Following Golyandina and Stepanov (2005), we will write the forecasting formulas for two versions of the recurrent MSSA forecast: row (generated by $\{U_j\}_{j=1}^r$) and column (generated by $\{V_j\}_{j=1}^r$). These one-term ahead forecasting formulas can be applied for M-term ahead forecasting by using the recurrence.

The column recurrent forecasting performs forecast by an LRR of order $L-1$ applied to the last $L-1$ points of the reconstructed signal; that is, the same LRR and different initial data. The row recurrent forecasting constructs s different linear relations, each is applied to the set of $K_i - 1$ last points of series; that is, the LRRs are different but the initial data for them is the same.

Column Forecast

Denote by $\mathbf{Z}$ the matrix consisting of the last $L-1$ values of the reconstructed signals:

$$\mathbf{Z} = \begin{pmatrix} \tilde{x}^{(1)}_{N_1-L+2} & \cdots & \tilde{x}^{(1)}_{N_1} \\ \tilde{x}^{(2)}_{N_2-L+2} & \cdots & \tilde{x}^{(2)}_{N_2} \\ \vdots & \vdots & \vdots \\ \tilde{x}^{(s)}_{N_s-L+2} & \cdots & \tilde{x}^{(s)}_{N_s} \end{pmatrix},$$

$\nu^2 = \sum_{j=1}^{r} \pi(U_j)^2$. If $\nu^2 < 1$, then the column MSSA forecast is uniquely defined and can be calculated by the formula

$$R_N = \mathbf{Z}\mathcal{R}_L, \qquad \text{where} \qquad \mathcal{R}_L = \frac{1}{1-\nu^2}\sum_{j=1}^{r} \pi(U_j)\underline{U_j} \ \in \mathsf{R}^{L-1}. \tag{4.9}$$

Note that (4.9) implies that the forecasting of all individual signals is made using the same linear recurrent formula which is generated by the whole system.

Row Forecast

Introduce the vectors of the last $K_m - 1$ values of the reconstructed signals

$$Z^{(m)} = \left(\tilde{x}^{(m)}_{N-K_m+2}, \ldots, \tilde{x}^{(m)}_{N_m}\right)^{\mathrm{T}}, \quad m = 1, \ldots, s,$$

and denote

$$Z = \begin{pmatrix} Z^{(1)} \\ Z^{(2)} \\ \vdots \\ Z^{(s)} \end{pmatrix}, \qquad \mathbf{S} = [\boldsymbol{\mu}(V_1) : \ldots : \boldsymbol{\mu}(V_r)].$$

In this notation, $Z = \overline{\overline{\widetilde{Y}}}_L$.

If the inverse matrix $(\mathbf{I}_s - \mathbf{S}\mathbf{S}^{\mathrm{T}})^{-1}$ exists and $r \le K - s$, then the row MSSA recurrent forecast exists and can be calculated by the formula

$$R_N = \mathcal{R}_K Z, \quad \text{where } \mathcal{R}_K = (\mathbf{I}_s - \mathbf{S}\mathbf{S}^{\mathrm{T}})^{-1} \mathbf{S}\underline{\underline{V}}^{\mathrm{T}}. \tag{4.10}$$

Note that (4.10) implies that the forecasting of the individual signals is made using the LRRs which are different for different series. The forecasting value generally depends on the last values of all time series from the system of time series.

4.3.1.3 Vector MSSA Forecasting

Denote $\underline{\mathcal{L}}^{\mathrm{col}} = \mathrm{span}(\underline{U}_1, \ldots, \underline{U}_r)$ and $\underline{\underline{\mathcal{L}}}^{\mathrm{row}} = \mathrm{span}(\underline{\underline{V}}_1, \ldots, \underline{\underline{V}}_r)$. Let Π^{col} be the orthogonal projector of R^{L-1} on $\underline{\mathcal{L}}^{\mathrm{col}}$ and Π^{row} be the orthogonal projector of R^{K-s} on $\underline{\underline{\mathcal{L}}}^{\mathrm{row}}$.

An explicit form of the matrices of the column and row projectors can be found in Golyandina and Osipov (2007; formula (4)). However, the calculation by that formula is time-consuming. A fast algorithm of calculation is presented in Golyandina et al. (2015; Section 6.3).

Column Forecast

We have mentioned above that for a given subspace ($\mathcal{L}^{\mathrm{col}}$ in our case) the column forecast is performed independently for each time series. Define the linear operator $\mathcal{P}^{\mathrm{col}}_{\mathrm{Vec}} : \mathsf{R}^L \mapsto \mathcal{L}^{\mathrm{col}}$ by the formula

$$\mathcal{P}^{\mathrm{col}}_{\mathrm{Vec}} Z = \begin{pmatrix} \Pi^{\mathrm{col}} \overline{Z} \\ \mathcal{R}_L^{\mathrm{T}} \overline{Z} \end{pmatrix}, \tag{4.11}$$

where $\mathcal{R}_L$ is defined in (4.9).

The *vector forecasting algorithm* for jth series is as follows:

1. In the notation above, define vectors Z_i as follows:

$$Z_i = \begin{cases} \widehat{X}_i^{(j)} & \text{for } i = 1, \ldots, K_j, \\ \mathcal{P}^{\mathrm{col}}_{\mathrm{Vec}} Z_{i-1} & \text{for } i = K_j + 1, \ldots, K_j + M + L - 1. \end{cases} \tag{4.12}$$

2. By constructing the matrix $\mathbf{Z} = [Z_1 : \ldots : Z_{K_j+M+L-1}]$ and making its diagonal averaging we obtain the series $z_1, \ldots, z_{N_j+M+L-1}$.
3. The numbers $z_{N_j+1}, \ldots, z_{N_j+M}$ form the M terms of the vector forecast.

Row Forecast

Define the linear operator $\mathcal{P}^{\mathrm{row}}_{\mathrm{Vec}} : \mathsf{R}^K \mapsto \mathcal{L}^{\mathrm{row}}$ by the formula

$$\mathcal{P}^{\mathrm{row}}_{\mathrm{Vec}} Z = A, \tag{4.13}$$

such that $\underline{\underline{A}} = \Pi^{\mathrm{row}} \overline{\overline{Z}}$ and $\boldsymbol{\mu}(A) = \mathfrak{R}_K \overline{\overline{Z}}$, where $\mathfrak{R}_K$ is defined in (4.10).

The *vector forecasting algorithm* is as follows:

1. In the notation above, define vectors Z_i as follows:

$$Z_i = \begin{cases} \widehat{Y}_i & \text{for } i = 1, \ldots, L, \\ \mathcal{P}^{\mathrm{row}}_{\mathrm{Vec}} Z_{i-1} & \text{for } i = L+1, \ldots, L+M+K^*-1, \end{cases} \tag{4.14}$$

   where $K^* = \max(K_i, i = 1, \ldots, s)$.
2. By constructing the matrix $\mathbf{Z} = [Z_1 : \ldots : Z_{L+M+K^*-1}]$ and making Reconstruction step we obtain the series $z_1^{(j)}, \ldots, z_{N_j+M+K^*-1}^{(j)}$, $j = 1, \ldots, s$.
3. The numbers $z_{N_j+1}^{(j)}, \ldots, z_{N_j+M}^{(j)}$, $j = 1, \ldots, s$, form the M terms of the vector forecast.

Remark 4.1 For the M-step ahead vector forecast, $M + K^* - 1$ new lagged vectors for the row forecasting and $M + L - 1$ ones for the column forecasting are constructed. The reason for this is to make the M-step forecast inheriting the $(M-1)$-step forecast as its part. This specific feature of the vector forecasting provides its stability and accuracy if the accurately extracted component of finite rank is forecasted; that is, if a long-term forecast is appropriate. Otherwise (if the MSSA approximation is inadequate), the long-term vector forecasting can be misleading and even a short-term vector forecasting can be inaccurate for large K^* or L correspondingly.

4.3.2 Algorithms

Algorithms of MSSA column forecasting are very similar to the algorithms of 1D-SSA forecasting (see Algorithms 3.5 and 3.6). Let a version of MSSA be applied to the system of s time series $\mathbb{X}$ and let the eigentriples $\{(\sigma_i, P_i, Q_i),\ i \in I\}$ be chosen for reconstruction. The suggested forecasting algorithms are formulated for the forecasting in the subspace $\mathcal{L}_r = \mathrm{span}\{P_i, i \in I\} \subset \mathsf{R}^L$. For simplicity, we assume that $I = \{1, \ldots, r\}$ and the vectors $P_i, i \in I$, are orthonormal. Note that the forecasting values do not depend on the choice of a basis of $\mathcal{L}_r$.

Algorithm 4.4 Recurrent MSSA column forecasting

Input: Collection of time series $\mathbb{X}^{(p)}$ of length N_p, where $p = 1, \dots, s$, window length L, orthonormal system of vectors $\{P_i\}_{i=1}^r$, forecast horizon M.

Output: Forecast values $(\widetilde{x}_{N_p+1}^{(p)}, \dots, \widetilde{x}_{N_p+M}^{(p)})$, $p = 1, \dots, s$.

1: Construct the vector $\mathcal{R} = (a_{L-1}, \dots, a_1)^{\mathrm{T}}$ of coefficients of the min-norm LRR by Algorithm 3.1 applied to $\{P_i, i \in I\}$.
2: Construct the reconstructed matrices $\widehat{\mathbf{X}}^{(p)} = \mathbf{P}\mathbf{P}^{\mathrm{T}}\mathbf{X}^{(p)}$, where $\mathbf{P} = [P_1 : \dots : P_r]$, and the reconstructed series $\widetilde{\mathbb{X}}^{(p)} = (\widetilde{x}_1^{(p)}, \dots, \widetilde{x}_{N_p}^{(p)})$ as $\widetilde{\mathbb{X}}^{(p)} = \mathfrak{T}_{\mathrm{SSA}}^{-1} \circ \Pi_{\mathcal{H}}(\widehat{\mathbf{X}}^{(p)})$; $p = 1, \dots, s$.
3: Calculate the forecast values recurrently by

$$\widetilde{x}_n^{(p)} = \sum_{i=1}^{L-1} a_i \widetilde{x}_{n-i}^{(p)}, \ n = N_p + 1, \dots, N_p + M; \ p = 1, \dots, s.$$

Algorithm 4.4 is written for the version, where the reconstructed series is taken as the base for forecasting. In the other version, where the original series itself is the base of forecasting, $x_n^{(p)}$ are taken instead of $\widetilde{x}_n^{(p)}$ at Step 3 for $n = N_p - L + 1, \dots, N_p$.

The next algorithm implements vector forecasting.

Algorithm 4.5 Vector MSSA column forecasting

Input: Collection of time series $\mathbb{X}^{(p)}$ of length N_p, where $p = 1, \dots, s$, window length L, orthonormal system of vectors $\{P_i\}_{i=1}^r$, forecast horizon M.

Output: Forecast values $(\widetilde{x}_{N_p+1}^{(p)}, \dots, \widetilde{x}_{N_p+M}^{(p)})$, $p = 1, \dots, s$.

1: Obtain the vector $\mathcal{R} = (a_{L-1}, \dots, a_1)^{\mathrm{T}}$ of coefficients of the min-norm LRR by Algorithm 3.1 applied to $\{P_i, i \in I\}$.
2: Calculate the matrix Π defining the projection in (3.5).
3: Compute the reconstructed matrices $\widehat{\mathbf{X}}^{(p)} = \mathbf{P}\mathbf{P}^{\mathrm{T}}\mathbf{X}^{(p)}$, where $\mathbf{P} = [P_1 \dots : P_r]$; $p = 1, \dots, s$.
4: Extend the reconstructed matrices $\widehat{\mathbf{X}}^{(p)} = [\widehat{X}_1^{(p)} : \dots : \widehat{X}_{K_p}^{(p)}]$ by column vectors:

$$\widehat{X}_n^{(p)} = \mathcal{P}_{\mathrm{Vec}} \widehat{X}_{n-1}^{(p)} \text{ for } n = K_p + 1, \dots, K_p + M + L - 1; \ p = 1, \dots, s,$$

where $\mathcal{P}_{\mathrm{Vec}}$ is given in (3.6) and uses the vector $\mathcal{R}$ of coefficients of the min-norm LRR. Denote the extended matrices by $\widehat{\mathbf{X}}_{\mathrm{ext}}^{(p)}$; $\widehat{\mathbf{X}}_{\mathrm{ext}}^{(p)} \in \mathsf{R}^{L \times (K_p + M + L - 1)}$; $p = 1, \dots, s$.
5: Obtain the extended reconstructed series $\widetilde{\mathbb{X}}_{\mathrm{ext}}^{(p)} = (\widetilde{x}_1^{(p)}, \dots, \widetilde{x}_{N_p+M+L-1}^{(p)})$ by $\widetilde{\mathbb{X}}_{\mathrm{ext}}^{(p)} = \mathfrak{T}_{\mathrm{SSA}}^{-1} \circ \Pi_{\mathcal{H}}(\widehat{\mathbf{X}}_{\mathrm{ext}}^{(p)})$; $p = 1, \dots, s$.
6: Return the forecast values $(\widetilde{x}_{N_p+1}^{(p)}, \dots, \widetilde{x}_{N_p+M}^{(p)})$; $p = 1, \dots, s$.

Algorithms of row forecasting have more complicated form and follow the steps outlined in Sect. 4.3.1.

4.3.3 *MSSA Forecasting in* Rssa

4.3.3.1 Description of Functions

MSSA forecasting functions differ from the functions described in Sect. 3.2.3.1 for one-dimensional case by the additional parameter `direction`, which can be equal to either `"row"` or `"column"`. The other parameters are exactly the same as described in Sect. 3.2.3.1. For example, a call for the recurrent row forecast based on the `ssa` object `s` can be made as follows:

```
f <- rforecast(s, groups = list(1:2, 3:4),
               direction = "row", len = 12, only.new = FALSE)
```

The wrappers `predict` is able to construct forecasts in a unified manner. For example, the presented call of `rforecast` is similar to the call

```
f <- predict(s, groups = list(1:2, 3:4),
             method = "recurrent", direction = "row", len = 12)
```

The function `forecast`, the bootstrap intervals, and backward forecasting are not implemented for MSSA forecasting.

4.3.3.2 Typical Code

The code for forecasting is very similar to that in 1D-SSA, compare Fragments 4.3.1 and 3.2.1. For demonstration, we use the monthly sales of fortified ("FORT") and dry ("DRY") wines taken from the dataset "AustralianWine" (Fig. 4.7).

Fragment 4.3.1 ("FORT" and "DRY": Forecast)

```
> f.wineFortDry <- rforecast(s.wineFortDry,
+                            groups = list(1, 1:12),
+                            len = 60, only.new = TRUE)
> plot(cbind(wineFortDry[, "Fortified"],
+            f.wineFortDry$F2[, "Fortified"]),
+      plot.type = "single",
+      col = c("black", "red"), ylab = "Fort")
> plot(cbind(wineFortDry[, "Drywhite"],
+            f.wineFortDry$F2[, "Drywhite"]),
+      plot.type = "single",
+      col = c("black", "red"), ylab = "Dry")
> par(mfrow = c(1, 1))
```

4.3.3.3 Simulated Example: Numerical Comparison

In this section, we demonstrate how the accuracy of MSSA is related to the structure of the multivariate time series. The aim is to compare accuracy for separate analysis and forecasting of time series with simultaneous processing of the series system. We

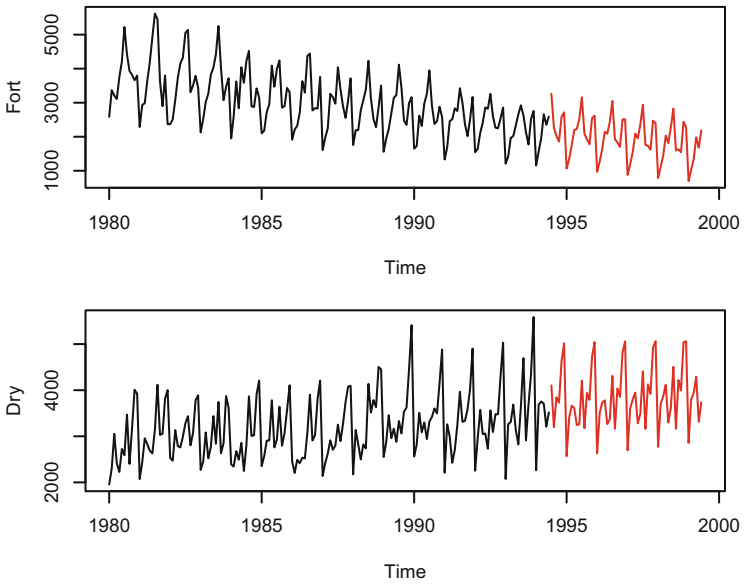

Fig. 4.7 "FORT" and "DRY": Forecast of the signal

summarize the results from Golyandina and Stepanov (2005) and Golyandina et al. (2015) and supplement them with new comparisons. In particular, the comparison results explain the choice of the default forecasting method.

In the study below, we consider the case $s = 2$ and examine the following SSA methods: (a) 1D-SSA applied twice, (b) MSSA, and (c) CSSA. The investigated model examples include the least favorable and the most favorable cases for MSSA as well as some cases well suited for the application of CSSA.

Let us assume that we observe $(\mathbb{X}^{(1)}, \mathbb{X}^{(2)}) = (\mathbb{H}^{(1)}, \mathbb{H}^{(2)}) + (\mathbb{N}^{(1)}, \mathbb{N}^{(2)})$, where $(\mathbb{H}^{(1)}, \mathbb{H}^{(2)})$ is a two-dimensional signal consisting of two harmonic time series, $\mathbb{N}^{(1)}$ and $\mathbb{N}^{(2)}$ are realizations of independent white Gaussian noises. Then we can use the standard simulation techniques to obtain estimates of the mean square errors (MSE) for the reconstruction and forecasting of $(\mathbb{H}^{(1)}, \mathbb{H}^{(2)})$ by the indicated SSA methods. The resultant MSE is calculated as the mean of $\mathrm{MSE}^{(1)}$ and $\mathrm{MSE}^{(2)}$ for $\mathbb{H}^{(1)}$ and $\mathbb{H}^{(2)}$ correspondingly.

We take the following parameters for the simulation of the time series: $N = 71$, the variance of all noise components is $\sigma^2 = 25$, the number of replications is 10000. We consider the following three versions of the signal $(\mathbb{H}^{(1)}, \mathbb{H}^{(2)})$.

Example A (the same periods; the difference between the phases is different from $\pi/2$):

$$h_k^{(1)} = 30\cos(2\pi k/12), \quad h_k^{(2)} = 20\cos(2\pi k/12 + \pi/4), \quad k = 1, \ldots, N.$$

Example B (the same periods and amplitudes; the difference between the phases is equal to $\pi/2$):

$$h_k^{(1)} = 30\cos(2\pi k/12), \quad h_k^{(2)} = 30\cos(2\pi k/12 + \pi/2), \quad k = 1, \ldots, N.$$

Example C (different periods):

$$h_k^{(1)} = 30\cos(2\pi k/12), \quad h_k^{(2)} = 20\cos(2\pi k/8 + \pi/4), \quad k = 1, \ldots, N.$$

The choice of these examples is determined by the observation that the dimensions of the signal trajectory spaces (i.e., ranks) are different for different extensions of the 1D-SSA method, see Table 4.1. For each example the rank in blue corresponds to the method with the best accuracy for this example. Cells in the row corresponding to 1D-SSA contain one number, since the ranks of the times series from the considered collections coincide.

The results of investigation for different window lengths L are summarized in Tables 4.2 and 4.3. The 24 term-ahead forecast was performed. For each example, the cells corresponding to the method with the reconstruction/forecast accuracy, which is closed to the best one, are shown in bold and the overall minimum is in blue color.

Comparison of Tables 4.2 and 4.3 with Table 4.1 clearly demonstrates the relation between the accuracy of the signal reconstruction (forecast) and the dimension of the signal trajectory space. Since the structure of the series from Example B and Example A is the same from the viewpoint of MSSA and 1D-SSA, we omit the corresponding results for Example B in Table 4.3.

Table 4.1 Dimension of the signal trajectory space

	Example A	Example B	Example C
MSSA	**2**	2	4
1D-SSA	2	2	**2**
CSSA	2	**1**	4

Table 4.2 MSE of signal reconstruction

Example A	$L = 12$	$L = 24$	$L = 36$	$L = 48$	$L = 60$
MSSA	3.18	1.83	1.59	**1.47**	2.00
1D-SSA	3.25	**2.01**	**2.00**	**2.01**	3.25
CSSA	3.25	**2.02**	**2.01**	**2.02**	3.25
Example B	$L = 12$	$L = 24$	$L = 36$	$L = 48$	$L = 60$
MSSA	3.18	1.82	1.58	**1.47**	1.97
1D-SSA	3.25	**2.01**	**2.00**	**2.01**	3.25
CSSA	1.57	**1.00**	**0.99**	**1.00**	1.57
Example C	$L = 12$	$L = 24$	$L = 36$	$L = 48$	$L = 60$
MSSA	6.91	3.77	3.07	**2.88**	3.84
1D-SSA	3.23	**2.01**	**2.00**	**2.01**	3.23
CSSA	6.98	4.06	**3.82**	4.06	6.98

Table 4.3 MSE of signal forecast

Example A	$L = 12$	$L = 24$	$L = 36$	$L = 48$	$L = 60$
Recurrent					
MSSA-column	5.36	**3.67**	3.73	3.70	4.43
MSSA-row	6.02	4.25	3.83	**3.32**	3.98
1D-SSA	7.24	**5.59**	6.30	6.42	7.93
CSSA	7.30	**5.60**	6.32	6.41	7.86
Vector					
MSSA-column	5.93	3.77	3.62	**3.11**	3.65
MSSA-row	4.00	**3.03**	3.39	3.17	4.24
1D-SSA	7.74	5.43	5.85	**5.14**	6.76
CSSA	7.79	5.44	5.86	**5.12**	6.87
Example C	$L = 12$	$L = 24$	$L = 36$	$L = 48$	$L = 60$
Recurrent					
MSSA-column	25.76	**7.39**	7.55	7.43	9.00
MSSA-row	19.82	8.47	8.00	**6.66**	8.30
1D-SSA	7.36	**5.61**	6.28	6.44	8.00
CSSA	38.79	**11.21**	13.37	13.09	24.89
Vector					
MSSA-column	25.34	7.56	7.57	**6.20**	7.67
MSSA-row	57.59	**6.04**	7.03	6.30	8.69
1D-SSA	7.84	5.47	5.84	**5.18**	6.88
CSSA	35.77	10.89	13.44	**10.22**	69.04
Example B	$L = 12$	$L = 24$	$L = 36$	$L = 48$	$L = 60$
CSSA recurrent	3.48	**2.76**	3.10	3.19	3.99
CSSA vector	3.82	2.70	2.89	**2.56**	3.18

Note that the reconstructions by 1D-SSA and CSSA are the same for window lengths L and $N - L + 1$ (12 and 60, 24 and 48 for the considered examples). Reconstructions by MSSA are different for different L. Also note that the trajectory matrix for 1D-SSA has rank $\min(L, N - L + 1)$ and the rank is maximal for $L \simeq N/2$. The MSSA-trajectory matrix has rank equal to $\min(L, (N - L + 1)s)$, where s is the number of time series in the system. This rank is maximal for $L \simeq sN/(s+1)$. Although the maximality of the rank does not guarantee the minimality of errors, this consideration means that to achieve better separability the choice of the window length L larger than $N/2$ can often be recommended. Simulations confirm this: the minimum of the reconstruction error for MSSA is achieved at $L = 48 = 72 \times 2/3$.

The forecasting errors have much more complicated structure, see Golyandina (2010). In particular, these errors for forecasting depend on the reconstruction errors for the last time series points; therefore, the error may have a dependence on L, which is different from that for the average reconstruction errors. The considered examples show that the vector forecast is more accurate than the recurrent one and that the row MSSA forecast is slightly more accurate than the column MSSA forecast.

The considered examples confirm the following assertions:

- The accuracy of the SSA-based methods is closely related to the structure of the signal trajectory spaces generated by these methods. MSSA has an advantage if time series from the system have matched components. (Note that we considered equal levels of noise.)
- Optimal window lengths for analysis and forecasting can differ. Despite the accuracy of forecast is related to the accuracy of reconstruction, this relation is not straightforward.
- The vector forecast with the best window length is more accurate than the recurrent forecast. However, it is not always the case if we compare forecast accuracies for the same window length. This is probably valid for forecasting of well-separated signal of finite rank only, see Remark 4.1.
- In MSSA, the recommendations for the choice of the window length (e.g., "take L larger (or smaller) than the half of the time series length") for recurrent forecasting are in a sense opposite to that for the vector forecasting.
- For the row and column forecasting (1D-SSA and CSSA forecasting methods are particular cases of the column forecasting), the recommendations are also opposite. This is not surprising since L and K have swapped places in MSSA relative to 1D-SSA and CSSA.

Fragment 4.3.2 demonstrates how the Rssa package allows estimation of the reconstruction and forecast accuracy on the example of MSSA and CSSA analysis and vector forecasting applied to Example A with $R = 10$ replications. Note that the numbers in Tables 4.2 and 4.3 were obtained by another complicated code, where $R = 10000$ (see the replicated code to Golyandina et al. (2015)).

Fragment 4.3.2 (Simulation for Accuracy Estimation)

```
> N <- 71
> sigma <- 5
> Ls <- c(12, 24, 36, 48, 60)
> len <- 24
> signal1 <- 30 * cos(2*pi * (1:(N + len)) / 12)
> signal2 <- 30 * cos(2*pi * (1:(N + len)) / 12 + pi / 4)
> signal <- cbind(signal1, signal2)
> R <- 10
> mssa.errors <- function(Ls) {
+   f1 <- signal1[1:N] + rnorm(N, sd = sigma)
+   f2 <- signal2[1:N] + rnorm(N, sd = sigma)
+   f <- cbind(f1, f2)
+   err.rec <- numeric(length(Ls)); names(err.rec) <- Ls
+   err.for <- numeric(length(Ls)); names(err.for) <- Ls
+   for (l in seq_along(Ls)) {
+     L <- Ls[l]
+     s <- ssa(f, L = L, kind = "mssa")
+     rec <- reconstruct(s, groups = list(1:2))[[1]]
+     err.rec[l] <- mean((rec - signal[1:N, ])^2)
+     pred <- vforecast(s, groups = list(1:2), direction = "row",
+                       len = len, drop = TRUE)
```

```
+     err.for[l] <- mean((pred - signal[-(1:N), ])^2)
+   }
+   list(Reconstruction = err.rec, Forecast = err.for)
+ }
> mres <- replicate(R, mssa.errors(Ls))
> err.rec <- rowMeans(simplify2array(mres["Reconstruction", ]))
> err.for <- rowMeans(simplify2array(mres["Forecast", ]))
> print(err.rec)
      12       24       36       48       60
2.869683 1.587789 1.248881 1.153730 1.855115
> print(err.for)
      12       24       36       48       60
2.671251 2.578059 1.501565 2.595378 4.564218
> signal <- signal1 + 1i*signal2
> cssa.errors <- function(Ls) {
+   f1 <- signal1[1:N] + rnorm(N, sd = sigma)
+   f2 <- signal2[1:N] + rnorm(N, sd = sigma)
+   f <- f1 + 1i*f2
+   err.rec <- numeric(length(Ls)); names(err.rec) <- Ls
+   err.for <- numeric(length(Ls)); names(err.for) <- Ls
+
+   for (l in seq_along(Ls)) {
+     L <- Ls[l]
+     s <- ssa(f, L = L, kind = "cssa", svd.method = "svd")
+     rec <- reconstruct(s, groups = list(1:2))[[1]]
+     err.rec[l] <- mean(abs(rec - signal[1:N])^2)
+     pred <- vforecast(s, groups = list(1:2), len = len,
+                       drop = TRUE)
+     err.for[l] <- mean(abs(pred - signal[-(1:N)])^2)
+   }
+   list(Reconstruction = err.rec, Forecast = err.for)
+ }
> cres <- replicate(R, cssa.errors(Ls))
> err.rec <- rowMeans(simplify2array(cres["Reconstruction", ]))
> err.for <- rowMeans(simplify2array(cres["Forecast", ]))
> print(err.rec)
      12       24       36       48       60
7.349316 4.298144 4.101666 4.298144 7.349316
> print(err.for)
      12       24       36       48       60
24.67425 13.60116 14.54819 11.72135 15.86380
```

4.3.4 Other Subspace-Based MSSA Extensions

In view of the common structure of all SSA-family algorithms (see Sect. 1.1), many SSA-related techniques can be naturally extended from 1D objects (i.e., series) to other objects, and particularly to the systems of series.

In Sect. 4.3.1, we have considered methods of MSSA forecasting. Let us now describe some extensions. Since the algorithms and the codes are either exactly or almost identical to the 1D case, we are not writing them out. In particular, all methods and algorithms that are based on the use of the column subspaces are exactly the same in MSSA and 1D-SSA. For example, parameter estimation based on the column subspace can be performed using the same function `parestimate`.

The shaped version of MSSA is almost the same as Shaped 1D-SSA (Sect. 2.6) except for the following difference. If `NA` are placed at the ends of the time series, then the corresponding series is considered as a series of smaller length. If there is no other missing data, then the resultant series of smaller length does not have any missing values and is decomposed by a non-shaped version of MSSA. It is important to mention that if there are `NA` at the right end of a series, then forecasts start from the last not-NA values.

Formally, forecasting can be applied to shaped `ssa` object. However, it is generally recommended to fill gaps first and then forecast the series.

Iterative gap-filling (Sect. 3.3.3, the function `igapfill`) and low-rank approximation by Cadzow iterations (Sect. 3.4, the function `cadzow`) are implemented in a general manner and therefore can be applied to systems of time series with gaps in exactly the same manner as in the 1D case.

Subspace-based gap-filling (Sect. 3.3.3, the function `gapfill`) is implemented on the base of the recurrent column forecasting only and therefore does not have the parameters `direction` and `method`.

4.4 Case Studies

4.4.1 Analysis of Series in Different Scales (Normalization)

Assume that different time series in the system of series are measured in different scales. In statistics, this problem is typically resolved by making a standardization of the data. In SSA, centering may not be an appropriate preprocessing. Therefore, two types of preprocessing can be applied, conventional standardization and normalization, which is the division by the square root of the mean sum of squares. The normalization can be more appropriate for positive series, since it changes only the scale of data.

Let us consider fortified (“FORT”) and rosé (“ROSE”) wine sales from the dataset “AustralianWine.” Sales of fortified wines are measured in thousands but sales of rosé wines are measured in tens and hundreds. Fragment 4.4.1 shows how the scale influences the reconstruction result.

Fragment 4.4.1 ("FORT" and "ROSE": Influence of Series Scales)

```
> wineFortRose <- wine[, c("Fortified", "Rose")]
> summary(wineFortRose)
   Fortified          Rose
 Min.   :1154   Min.   : 30.00
 1st Qu.:2372   1st Qu.: 66.00
 Median :2898   Median : 87.00
 Mean   :3010   Mean   : 93.01
 3rd Qu.:3565   3rd Qu.:114.25
 Max.   :5618   Max.   :267.00
> norm.wineFortRosen <- sqrt(colMeans(wineFortRose^2))
> wineFortRosen <-
+   sweep(wineFortRose, 2, norm.wineFortRosen, "/")
> L <- 84
> s.wineFortRosen <- ssa(wineFortRosen, L = L, kind = "mssa")
> r.wineFortRosen <- reconstruct(s.wineFortRosen,
+                       groups = list(Trend = c(1, 12, 14),
+                                     Seasonality = c(2:11, 13)))
> s.wineFortRose <- ssa(wineFortRose, L = L, kind = "mssa")
> r.wineFortRose <- reconstruct(s.wineFortRose,
+                       groups = list(Trend = 1,
+                                     Seasonality = 2:11))
> wrap.plot <- function(rec, component = 1, series,
+                       xlab = "", ylab, ...)
+   plot(rec, add.residuals = FALSE, add.original = TRUE,
+        plot.method = "xyplot", superpose = TRUE,
+        scales = list(y = list(tick.number = 3)),
+        slice = list(component = component, series = series),
+        xlab = xlab, ylab = ylab, auto.key = "", ...)
> trel1 <- wrap.plot(r.wineFortRosen, series = 2,
+                    ylab = "Rose, norm", main = NULL)
> trel2 <- wrap.plot(r.wineFortRosen, series = 1,
+                    ylab = "Fort, norm", main = NULL)
> trel3 <- wrap.plot(r.wineFortRose, series = 2,
+                    ylab = "Rose", main = NULL)
> trel4 <- wrap.plot(r.wineFortRose, series = 1,
+                    ylab = "Fort", main = NULL)
> plot(trel1, split = c(1, 1, 2, 2), more = TRUE)
> plot(trel2, split = c(1, 2, 2, 2), more = TRUE)
> plot(trel3, split = c(2, 1, 2, 2), more = TRUE)
> plot(trel4, split = c(2, 2, 2, 2))
```

Figure 4.8 demonstrates the result of a trend reconstruction, where the trend was detected in the same way as before; that is, by using the forms of eigenvectors and weighted correlations. The trend of the "ROSE" series is more complicated. However, "FORT" overweighs the decomposition and the eigentriples that refine the "ROSE" trend have very small weight and mix with the common noise. Therefore, the MSSA processing with no normalization is worse for the analysis of the series ROSE, which is measured on a smaller scale.

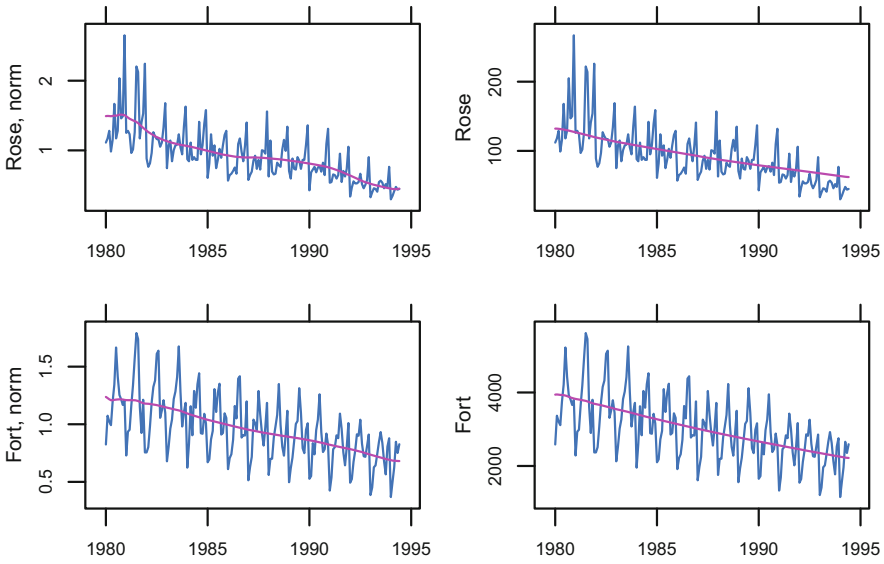

Fig. 4.8 "FORT" and "ROSE": Trends with normalization (ET1,12,14) and without (ET1)

4.4.2 Forecasting of Series with Different Lengths and Filling-In

Fragment 4.3.1 in Sect. 4.3.3.2 shows a typical code when MSSA is used for the series of equal lengths. However, the same code can be applied to series which have different lengths. The full set of "AustralianWine" data has missing values: there is no data for two months (points 175 and 176) for sales of "ROSE" and there is no data for the last 11 months of "Total" sales.

Let us perform the following actions: (A) fill-in missing data in ROSE, (B) calculate the sum of sales of the wines presented in the data, and (C) forecast this sum together with the total series in order to fill-in the missing data.

To use the analysis performed above, let us process the "ROSE" series together with the "FORT" series for (A). Fragment 4.4.2 implements (A). We fill-in the missing data by two methods. The result is presented in Fig. 4.9.

Fragment 4.4.2 ("FORT" and "ROSE": Filling-in the Missing Data in "ROSE")

```
> wineFortRose <- AustralianWine[, c("Fortified", "Rose")]
> L <- 84
> wineFortRose <- AustralianWine[, c("Fortified", "Rose")]
> norm.wineFortRosen <-
+   sqrt(colMeans(wineFortRose^2, na.rm = TRUE))
> wineFortRosen <-
+   sweep(wineFortRose, 2, norm.wineFortRosen, "/")
```

```
> s.wineFortRosen <- ssa(wineFortRosen, L = L, kind = "mssa")
> g.wineFortRosen <-
+   gapfill(s.wineFortRosen, groups = list(1:14))
> ig.wineFortRosen <-
+   igapfill(s.wineFortRosen, groups = list(1:14))
> ig.wineFortRose <-
+   norm.wineFortRosen["Rose"] * ig.wineFortRosen
> g.wineFortRose <-
+   norm.wineFortRosen["Rose"] * g.wineFortRosen
> xyplot(AustralianWine[100:187, "Rose"] +
+          ig.wineFortRose[100:187, "Rose"] +
+          g.wineFortRose[100:187, "Rose"] ~
+          time(AustralianWine)[100:187],
+        type = "l", xlab = "Time", ylab = "Rose",
+        lty = c(1, 2, 1), lwd = c(2, 1, 1),
+        auto.key = list(text = c("`Rose'",
+                                 "Iterative gap filling",
+                                 "Subspace-based gap-filling")))
```

Fragment 4.4.3 implements (B) and (C). Since the series "Total" and "Mainsales" are of different lengths (recall that NA at the end of series correspond to the reduction of the series length), the forecasts correspond to different times (Fig. 4.10). If one is only interested in filling-in the missing data in "Total," then the forecasted values of "Mainsales" can be ignored.

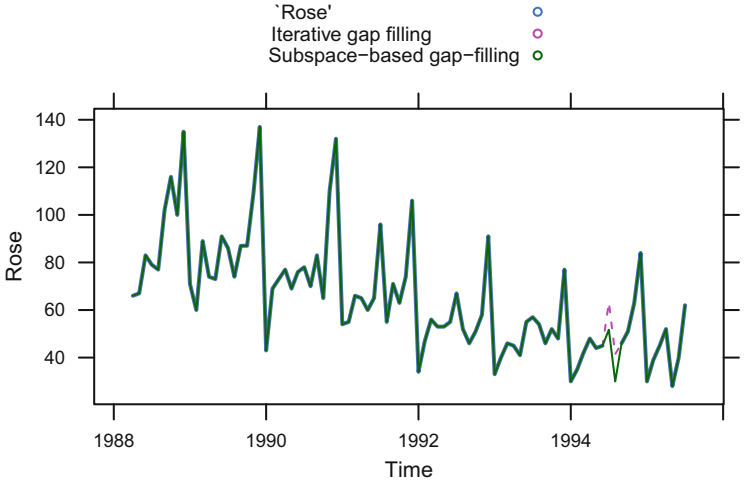

Fig. 4.9 "FORT" and "ROSE": Filling-in of "ROSE" by two methods

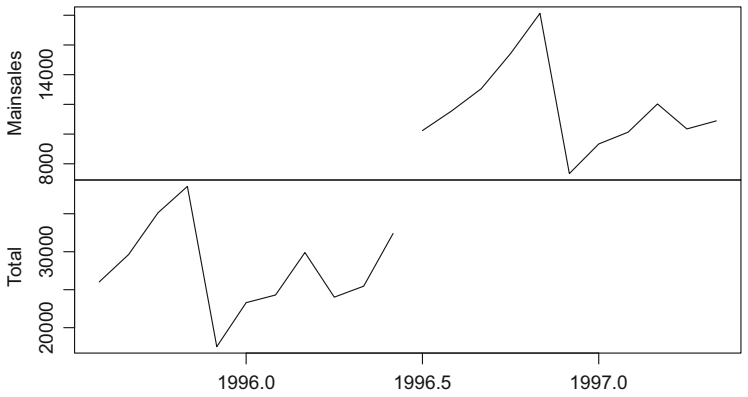

Fig. 4.10 "Total" and "Mainsales": Forecast to fill-in "Total"

Fragment 4.4.3 ("Total" and "Mainsales": Forecast to Fill-in "Total")

```
> FilledRose <- AustralianWine
> FilledRose[175:176, "Rose"] <- g.wineFortRose[175:176]
> mainsales <- ts(rowSums(FilledRose[, -1]))
> tsp(mainsales) <- tsp(AustralianWine)
> wine.add.mainsales <- cbind(FilledRose, mainsales)
> colnames(wine.add.mainsales) <-
+   c(colnames(FilledRose), "Mainsales")
> L <- 84
> s.totalmain <- ssa(wine.add.mainsales[, c("Mainsales",
+                                           "Total")],
+                    L = L, kind = "mssa")
> f.totalmain <- rforecast(s.totalmain, groups = list(1:14),
+                          len = 11, only.new = TRUE)
> plot(f.totalmain, main = "", xlab = NULL, oma = c(3, 1, 1, 1))
```

4.4.3 Simultaneous Decomposition of Many Series

In this example, we consider the system of many time series and show that the decomposition by MSSA helps to look at similar patterns in the series.

Let us consider a collection of $s = 6$ series from the "AustralianWine" dataset, which includes the series of wine sales considered in Fragment 4.2.1, see Sect. 4.2.6.2. A considerable part of this multivariate series can be described as seasonality. Therefore, MSSA can have an advantage over conventional 1D-SSA applied separately to each series from the system.

Since the time series have different scales, it may be advantageous to transform the time series to the same scale by normalizing them. We choose the window length $L = 163$, then $K_i = 12$ $(i = 1, \ldots, 6)$ and $K = 6 \cdot 12 = 72$ and therefore the

number of elementary components is equal to $72 = \min(163, 72)$. This choice of window length does not correspond to the maximal possible number of elementary components.

We can obtain a more detailed decomposition with $144 = \min(151, 144)$ elementary components if we choose $L = 151$ and $K_i = 24$ but in this case elementary components appear with almost equal weights (implying a lack of strong separability). Thus, the choice $L = 163$ and $K_i = 12$ helps to avoid mixing up of the components.

The identification of the trend (ET1,2,5) and seasonality (ET3,4, 6–12) is performed on the base of eigenvectors and uses the principles used in the typical code from Sect. 4.2.6.2. Fragment 4.4.4 contains the code which gives the reconstruction shown in Figs. 4.11 and 4.12.

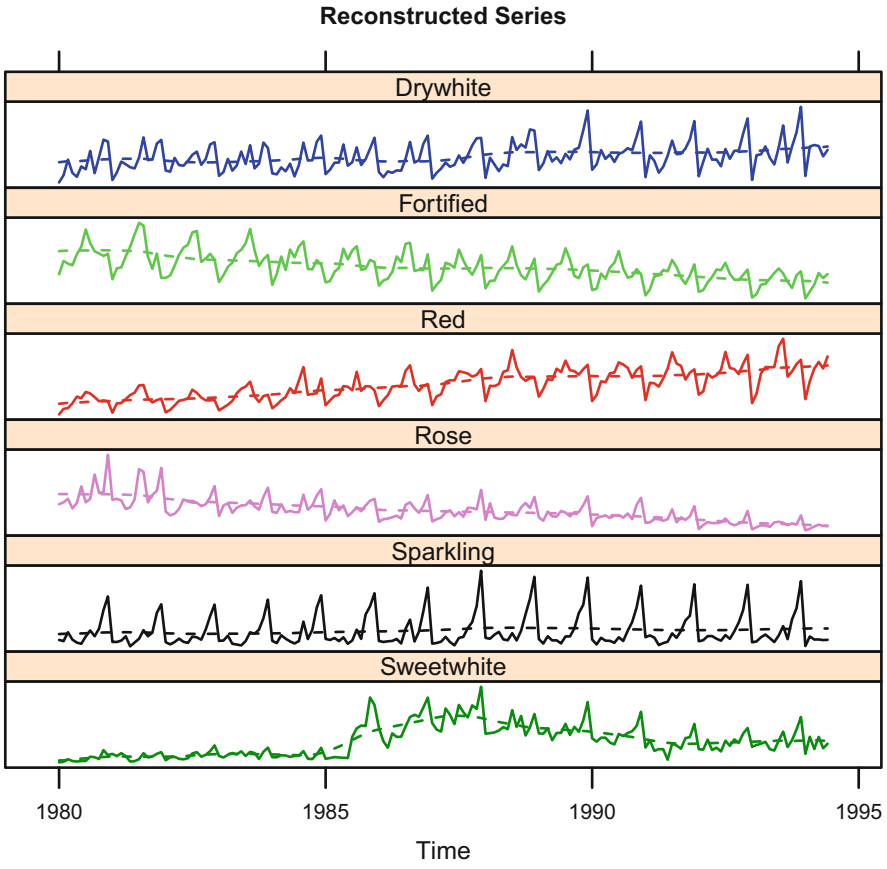

Fig. 4.11 "Australian wines": Extraction of trends

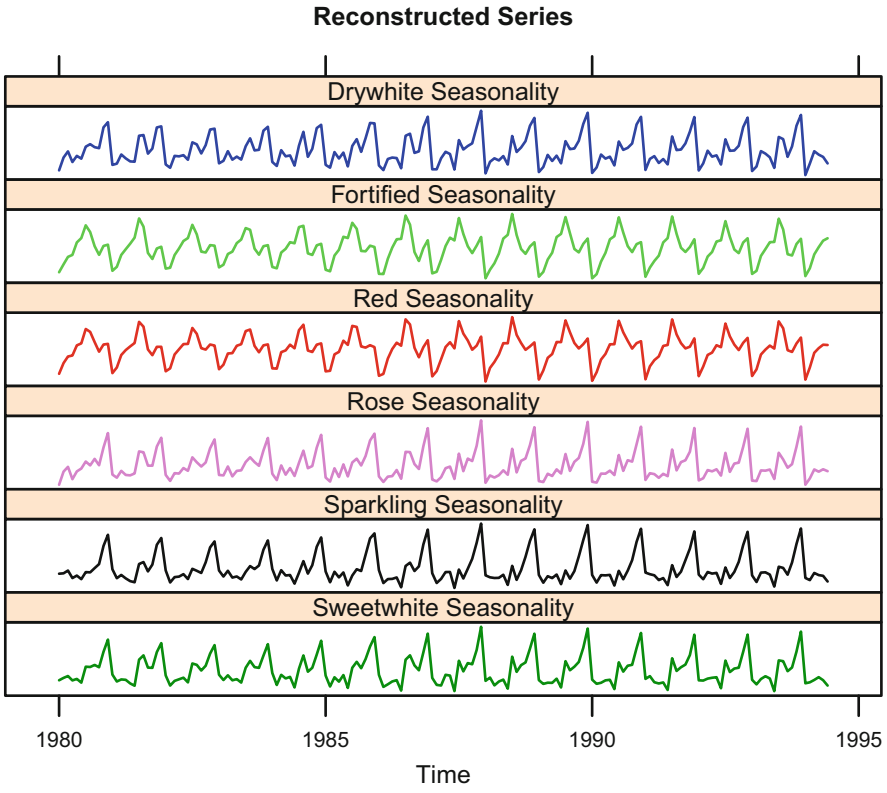

Fig. 4.12 "Australian wines": Extraction of seasonality

Fragment 4.4.4 ("Australian wines": Simultaneous Decomposition by MSSA)

```
> L <- 163
> norm.wine <- sqrt(colMeans(wine[, -1]^2))
> winen <- sweep(wine[, -1], 2, norm.wine, "/")
> s.winen <- ssa(winen, L = L, kind = "mssa")
> r.winen <- reconstruct(s.winen,
+                        groups = list(Trend = c(1, 2, 5),
+                                      Seasonality = c(3:4, 6:12)))
> plot(r.winen, add.residuals = FALSE,
+      plot.method = "xyplot",
+      slice = list(component = 1),
+      screens = list(colnames(winen)),
+      col =
+        c("blue", "green", "red", "violet", "black", "green4"),
+      lty = rep(c(1, 2), each = 6),
+      scales = list(y = list(draw = FALSE)),
+      layout = c(1, 6))
> plot(r.winen, plot.method = "xyplot", add.original = FALSE,
+      add.residuals = FALSE, slice = list(component = 2),
```

```
+       col =
+         c("blue", "green", "red", "violet", "black", "green4"),
+       scales = list(y = list(draw = FALSE)),
+       layout = c(1, 6))
```

The reconstructed trends and seasonal components look adequate. In addition, the simultaneous processing of several time series is very convenient as we obtain similar time series components all at once. In particular, it is clearly seen from Fig. 4.12 that the sales of fortified wines are maximal in June-July (that are winter months in Australia), while the sales of sparkling wines are largest in December.

References

Allen RM, Robertson WA (1996) Distinguishing modulated oscillations from coloured noise in multivariate datasets. Clim Dynam 12(11):775–784

Broomhead D, King G (1986) Extracting qualitative dynamics from experimental data. Physica D 20:217–236

Broomhead D, King G (1986b) On the qualitative analysis of experimental dynamical systems. In: Sarkar S (ed) Nonlinear phenomena and chaos. Adam Hilger, Bristol, pp 113–144

Golyandina N (2010) On the choice of parameters in singular spectrum analysis and related subspace-based methods. Stat Interface 3(3):259–279

Golyandina N, Korobeynikov A (2013) Basic singular spectrum analysis and forecasting with R. Comput Stat Data Anal 71:943–954

Golyandina N, Osipov E (2007) The "Caterpillar"-SSA method for analysis of time series with missing values. J Stat Plan Inference 137(8):2642–2653

Golyandina N, Stepanov D (2005) SSA-based approaches to analysis and forecast of multidimensional time series. In: Proceedings of the 5th St.Petersburg workshop on simulation, June 26–July 2, 2005. St. Petersburg State University, St. Petersburg, pp 293–298

Golyandina N, Zhigljavsky A (2013) Singular spectrum analysis for time series. Springer briefs in statistics. Springer

Golyandina N, Nekrutkin V, Zhigljavsky A (2001) Analysis of time series structure: SSA and related techniques. Chapman&Hall/CRC

Golyandina N, Korobeynikov A, Shlemov A, Usevich K (2015) Multivariate and 2D extensions of singular spectrum analysis with the Rssa package. J Stat Softw 67(2):1–78

Hannachi A, Jolliffe IT, Stephenson DB (2007) Empirical orthogonal functions and related techniques in atmospheric science: A review. Int J Climatol 27(9):1119–1152

Hassani H, Mahmoudvand R (2013) Multivariate singular spectrum analysis: a general view and vector forecasting approach. Int J Energy Stat 01(01):55–83

Keppenne C, Lall U (1996) Complex singular spectrum analysis and multivariate adaptive regression splines applied to forecasting the southern oscillation. In: Exp. long-lead forcst. Bull

Trickett SR (2003) F-xy eigenimage noise suppression. Geophysics 68(2):751–759

Weare BC, Nasstrom JS (1982) Examples of extended empirical orthogonal function analyses. Mon Weather Rev 110(6):481–485