Chapter 2
SSA Analysis of One-Dimensional Time Series

In the present chapter and Chap. 3, we thoroughly examine the use of SSA for one-dimensional data. This chapter is fully devoted to the SSA analysis of such data. Consideration of SSA forecasting, gap filling, and estimation of parameters of the signal is delayed until Chap. 3. The main difference between the materials of these two chapters is the use of the models. In the present chapter, the use of models is minimal; on the contrary, the methodologies of Chap. 3 are model-based.

In the terminology of Chap. 1, SSA for one-dimensional data should be referred to as 1D-SSA. However, but for the sake of brevity, in this chapter we will refer to it simply as SSA. The SSA input for all algorithms of this chapter is a collection $\mathbb{X}_N = (x_1, \ldots, x_N)$ of N real numbers, which can be thought of as a time series.

Let us start with the common parts of all versions of SSA algorithms considered in this chapter. These common parts are the embedding procedure at Step 1 of SSA and the diagonal averaging which makes the reconstruction at Step 4 (see Fig. 1.1).

Let L $(1 < L < N)$ be some integer called *window length* and set $K = N - L + 1$. Construct L-lagged vectors of size L as

$$X_i = (x_i, \ldots, x_{i+L-1})^{\mathrm{T}}, \quad i = 1 \ldots, K.$$

Define the embedding operator $\mathcal{T} = \mathcal{T}_{\mathrm{SSA}}$ by

$$\mathcal{T}_{\mathrm{SSA}}(\mathbb{X}) = \mathbf{X} = [X_1 : \ldots : X_K] = \begin{pmatrix} x_1 & x_2 & x_3 & \ldots & x_K \\ x_2 & x_3 & x_4 & \ldots & x_{K+1} \\ x_3 & x_4 & x_5 & \ldots & x_{K+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_L & x_{L+1} & x_{L+2} & \ldots & x_N \end{pmatrix}. \tag{2.1}$$

Matrix $\mathbf{X}$ of (2.1) is called trajectory (or L-trajectory) matrix. There are two important properties of this matrix:

(a) both the rows and columns of $\mathbf{X}$ are subseries of the original series, and
(b) $\mathbf{X}$ has equal elements on its anti-diagonals, which is equivalent to saying that $\mathbf{X}$ is a Hankel matrix.

The operator $\mathcal{T} = \mathcal{T}_{\mathrm{SSA}} : \mathsf{R}^N \to \mathcal{M}^{(\mathrm{H})}_{L,K}$ makes a correspondence between time series (collections of N numbers) and $\mathcal{M}^{(\mathrm{H})}_{L,K}$, the set of Hankel matrices of size $L \times K$. Since the correspondence defined by $\mathcal{T}$ is one-to-one, there exists the inverse $\mathcal{T}^{-1}$, which transfers any Hankel matrix of size $L \times K$ to a series of length N.

Let us also introduce the projector $\Pi_{\mathcal{H}} : \mathsf{R}^{L\times K} \to \mathcal{M}^{(\mathrm{H})}_{L,K}$ into the space of Hankel matrices as the operator of hankelization

$$(\Pi_{\mathcal{H}}\mathbf{Y})_{ij} = \sum_{(l,k)\in A_s} y_{lk} \big/ w_s, \tag{2.2}$$

where $s = i + j - 1$, $A_s = \{(l,k) : l + k = s + 1, 1 \le l \le L, 1 \le k \le K\}$ and $w_s = |A_s|$ denotes the number of elements in the set A_s. This corresponds to averaging the matrix elements over the "anti-diagonals." The weights w_s are equal to the number of series elements x_s in the trajectory matrix (2.1) and has a trapezoidal shape, decreasing towards both ends of the series.

On the base of (2.2), any matrix $\mathbf{Y} \in \mathsf{R}^{L\times K}$ can be transferred to a series of length N by applying $\mathcal{T}^{-1} \circ \Pi_{\mathcal{H}}$. Note that this is done in an optimal way.

2.1 Basic SSA

Basic SSA is the SSA for the analysis of one-dimensional series such that the decomposition into rank-one matrices at Step 2 (see the general scheme in Fig. 1.1) is done by the SVD.

2.1.1 Method

2.1.1.1 Step 1: Embedding

The series $\mathbb{X}$ is mapped to a sequence of L-lagged vectors, which form the trajectory matrix $\mathbf{X} = \mathcal{T}_{\mathrm{SSA}}(\mathbb{X})$, as shown in (2.1).

2.1.1.2 Step 2: Decomposition

Set $\mathbf{S} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$ and denote by $\lambda_1, \ldots, \lambda_d$ the positive *eigenvalues* of $\mathbf{S}$ taken in the decreasing order of magnitude ($\lambda_1 \geq \ldots \geq \lambda_d > 0$) and by $U_1, \ldots, U_d$ an orthonormal system of the *eigenvectors* of the matrix $\mathbf{S}$ corresponding to these eigenvalues; $V_i = \mathbf{X}^{\mathrm{T}} U_i / \sqrt{\lambda_i}$ are called factor vectors. At this step, we perform the singular value decomposition (SVD) of the trajectory matrix:

$$\mathbf{X} = \sum_{i=1}^{d} \sqrt{\lambda_i} U_i V_i^{\mathrm{T}} = \mathbf{X}_1 + \ldots + \mathbf{X}_d. \tag{2.3}$$

The matrices $\mathbf{X}_i = \sqrt{\lambda_i} U_i V_i^{\mathrm{T}}$ in (2.3) have rank 1; such matrices are called *elementary matrices*. The collection $(\sqrt{\lambda_i}, U_i, V_i)$ consisting of the singular value $\sqrt{\lambda_i}$, the left singular vector U_i and the right singular vector V_i will be called ith *eigentriple* (abbreviated as ET). Note that $\lambda_i = \|\mathbf{X}_i\|_{\mathrm{F}}^2$ and $\|\mathbf{X}\|_{\mathrm{F}}^2 = \|\mathbf{X}_1\|_{\mathrm{F}}^2 + \ldots + \|\mathbf{X}_d\|_{\mathrm{F}}^2$. The contribution of ith component $\mathbf{X}_i$ can thus be measured by $\lambda_i / \sum_j \lambda_j$.

For real-world time series, $d = \operatorname{rank} \mathbf{X}$ is typically equal to $\min(L, K)$; that is, the trajectory matrix is of full rank.

2.1.1.3 Step 3: Eigentriple Grouping

The input in this step is the expansion (2.3) and the specification of how to group the components of (2.3).

Let $I = \{i_1, \ldots, i_p\} \subset \{1, \ldots, d\}$ be a set of indices. Then the resultant matrix $\mathbf{X}_I$ corresponding to the group I is defined as $\mathbf{X}_I = \mathbf{X}_{i_1} + \ldots + \mathbf{X}_{i_p}$.

Assume that a partition of the set of indices $\{1, \ldots, d\}$ into m disjoint subsets $I_1, \ldots, I_m$ is specified. Then the expansion (2.3) leads to the decomposition

$$\mathbf{X} = \mathbf{X}_{I_1} + \ldots + \mathbf{X}_{I_m}. \tag{2.4}$$

The procedure of choosing the sets $I_1, \ldots, I_m$ is called *eigentriple grouping*. If $m = d$ and $I_j = \{j\}$, $j = 1, \ldots, d$, then the corresponding grouping is called *elementary*.

The grouping is performed by analyzing the eigentriples so that each group corresponds to an identifiable series component. The choice of several leading eigentriples corresponds to an optimal approximation of the time series, in accordance with the well-known optimality property of the SVD.

2.1.1.4 Step 4: Reconstruction (Diagonal Averaging)

The diagonal averaging (2.2) applied to a resultant matrix $\mathbf{X}_{I_k}$ produces a *reconstructed series* $\widetilde{\mathbb{X}}^{(k)} = (\widetilde{x}_1^{(k)}, \ldots, \widetilde{x}_N^{(k)}) = \mathcal{T}^{-1} \circ \Pi_{\mathcal{H}}(\mathbb{X}^{(k)})$. This way, the initial series $(x_1, \ldots, x_N)$ is decomposed into a sum of m reconstructed series:

$$x_n = \sum_{k=1}^{m} \widetilde{x}_n^{(k)}, \quad n = 1, \ldots, N. \tag{2.5}$$

The reconstructed series produced by the elementary grouping will be called *elementary reconstructed series*.

If grouping is sensible, then we obtain a reasonable decomposition into identifiable series components. Typical resultant decompositions are signal plus noise or trend plus seasonality plus noise.

As well as in the generic scheme, Steps 1 and 2 of Basic SSA are sometimes combined into the so-called "Decomposition stage" and Steps 3 and 4 are combined into "Reconstruction stage."

2.1.2 Appropriate Time Series

2.1.2.1 Time Series of Finite Rank

Although the SSA method is model-free and therefore SSA can be considered as an exploratory method, there is a model that perfectly suits SSA.

We say that a series has L-rank r if its L-trajectory matrix has rank r. Series $\mathbb{S}$ is called a series of finite rank r ($\operatorname{rank} \mathbb{S} = r$) if $\operatorname{rank}_L \mathbb{S} = r$ for any sufficiently large series length N and window length L. The term "finite rank" also has a meaning for the case of infinite series. For a generic infinite times series (cut at some N), $L = L(N)$ can tend to infinity and in this case the rank of the trajectory matrix would typically tend to infinity too. For a time series of finite rank, the rank r of the trajectory matrix is finite and fixed for large enough N and L such that $r \le \min(L, N - L + 1)$.

It is useful to know rank of a time series and the form of the singular vectors of its trajectory matrix, since knowing rank means knowing the number of the SVD components, which we should group for extraction of the corresponding series component, while the form of the singular vectors along with properties of eigenvalues helps in finding these SVD components.

We refer to Golyandina et al. (2001; Chapter 5) for details and full description. Here we mention that an exponential series $s_n = Ae^{\alpha n}$, $n = 1, 2, \ldots$, has rank 1, a linear function $s_n = an + b, a \neq 0$, has rank 2, a sinusoid with $s_n = A \sin(2\pi\omega n + \phi)$ has rank 2 for $0 < \omega < 0.5$ and rank 1 for $\omega = 0.5$. Singular vectors of trajectory matrices of time series have the same form as the series itself, which follows from the fact that rows and columns of the trajectory matrices are subseries of the original series. This information helps in choosing the groups at Grouping step.

2.1.2.2 Linear Recurrence Relations, Characteristic Polynomials and Roots

Time series of finite rank are closely related to the series governed by linear recurrence relations (LRRs). In particular, for infinite time series, the class of time series governed by LRRs coincides with the class of time series of finite rank.

Definition 2.1 A time series $\mathbb{S}_N = (s_i)_{i=1}^N$ is governed by an LRR, if there exist $a_1, \ldots, a_t$ such that

$$s_{i+t} = \sum_{k=1}^{t} a_k s_{i+t-k}, \ 1 \le i \le N-t, \ a_t \ne 0, \ t < N-1. \tag{2.6}$$

The number t is called the order of the LRR and $a_1, \ldots, a_t$ are the coefficients of the LRR. If $t = r$ is the minimal order of an LRR that governs the time series S_N, then the corresponding LRR is called minimal.

The minimal LRR is unique and its order is equal to the series rank.

As was mentioned in Sect. 1.4, it is well known that the time series $\mathbb{S}_\infty = (s_1, \ldots, s_n, \ldots)$ satisfies the LRR (2.6) for all $i \ge 0$ if and only if

$$s_n = \sum_{m=1}^{p} \left(\sum_{j=0}^{k_m-1} c_{mj} n^j \right) \mu_m^n, \tag{2.7}$$

where the complex coefficients c_{mj} depend on the first t points $s_1, \ldots, s_t$.

For real-valued time series, (2.7) implies that the class of time series governed by the LRRs consists of sums of products of polynomials, exponentials, and sinusoids

$$s_n = \sum_{m=1}^{\widetilde{p}} \left(\sum_{j=0}^{k_m-1} \widetilde{c}_{mj} n^j \right) e^{\alpha_m n} \cos(2\pi \omega_m n + \phi_m), \quad 0 \le \omega_m \le 0.5. \tag{2.8}$$

The minimal LRR determines all, except c_{mj}, parameters in (2.7) and all, except $\widetilde{c}_{mj}$ and ϕ_m, parameters in (2.8).

Definition 2.2 The polynomial $P_t(\mu) = \mu^t - \sum_{k=1}^t a_k \mu^{t-k}$ is called *characteristic polynomial* of the LRR (2.6).

Roots of the characteristic polynomial are called *characteristic roots* of the corresponding LRR. The roots of the characteristic polynomial of the minimal LRR governing the series, which can be called *signal roots of the LRR* or *characteristic roots of the series*, determine the values of parameters μ_m and k_m in (2.7) as follows. Let the time series $\mathbb{S}_\infty = (s_1, \ldots, s_n, \ldots)$ satisfy the LRR (2.6) with $a_t \ne 0$ and $i \ge 1$. Consider the characteristic polynomial of the LRR (2.6) and denote its different (complex) roots by $\mu_1, \ldots, \mu_p$, where $p \le t$. All these roots are non-zero as $a_t \neq 0$ with k_m being the multiplicity of the root μ_m ($1 \leq m \leq p$, $k_1 + \ldots + k_p = t$).

Let $c_{k_p-1,j} \neq 0$ for all j; this corresponds to the case of the minimal LRR. Then the rank of time series $\mathbb{S}_\infty$ given in (2.7) is equal to $r = \sum_{m=1}^{p} k_m$. In the real-valued case, if $\widetilde{c}_{k_{\widetilde{p}}-1,j} \neq 0$ for all j, then the rank of time series $\mathbb{S}_\infty$ given in (2.8) is equal to $r = \sum_{m=1}^{\widetilde{p}} \delta_m k_m$, where $\delta_m = 1$ for ω_m equal 0 or 0.5 and $\delta_m = 2$ for $0 < \omega_m < 0.5$.

If we find the signal roots $\mu_m = \rho_m e^{\pm \mathrm{i} 2\pi\omega_m}$ of the characteristic polynomial of the LRR governing the signal, then we can estimate the signal parameters. For example, the frequency ω_m of an exponentially-modulated sinusoid can be found using the argument of the corresponding conjugate roots, whereas the root modulus ρ_m gives the exponential rate $\alpha_m = \ln \rho_m$.

2.1.3 Separability and Choice of Parameters

Understanding separability is very important for understanding how SSA works. Recall that if two time series $\mathbb{X}_N^{(1)}$ and $\mathbb{X}_N^{(2)}$ are separable, then $\mathbb{X}_N^{(1)}$ can be extracted from the observed series $\mathbb{X}_N = \mathbb{X}_N^{(1)} + \mathbb{X}_N^{(2)}$. This means that there exists a partition into groups at Grouping step such that $\widetilde{\mathbb{X}}_N^{(m)} = \mathbb{X}_N^{(m)}$.

Let us define the separability formally. Let $\mathbf{X}^{(m)}$ be the trajectory matrices of the considered series components, $\mathbf{X}^{(m)} = \sum_{i=1}^{d_m} \sqrt{\lambda_{m,i}} U_{m,i} V_{m,i}^{\mathrm{T}}$, $m = 1, 2$, be their SVDs. The column and row spaces of the trajectory matrices are called *column* and *row trajectory spaces* correspondingly.

Definition 2.3 Let L be fixed. Two series $\mathbb{X}_N^{(1)}$ and $\mathbb{X}_N^{(2)}$ are called *weakly separable* (or simply *separable*) if their column trajectory spaces are orthogonal and the same is valid for their row trajectory spaces; that is, $(\mathbf{X}^{(1)})^{\mathrm{T}}\mathbf{X}^{(2)} = \mathbf{0}_{K,K}$ and $\mathbf{X}^{(1)}(\mathbf{X}^{(2)})^{\mathrm{T}} = \mathbf{0}_{L,L}$.

Definition 2.4 Two series $\mathbb{X}_N^{(1)}$ and $\mathbb{X}_N^{(2)}$ are called *strongly separable*, if they are weakly separable and the sets of singular values of their L-trajectory matrices are disjoint; that is, $\lambda_{1,i} \neq \lambda_{2,j}$ for any i and j.

Weak separability means that at Decomposition step there exists such an SVD that allows the proper grouping. A possibility of a non-separating SVD expansion which does not allow a proper grouping is related to the non-uniqueness of the SVD in the case of equal singular values. Strong separability means that any SVD of the trajectory matrix admits the proper grouping. Therefore, in order to be sure that SSA makes an accurate separation we have to require strong (approximate) separability.

By the definition, weak separability means orthogonality of the column and row spaces of the trajectory matrices of the series components $\mathbb{X}_N^{(1)}$ and $\mathbb{X}_N^{(2)}$. For approximate (asymptotic) separability with $\widetilde{\mathbb{X}}_N^{(m)} \approx \mathbb{X}_N^{(m)}$, we need the condition of approximate (asymptotic) orthogonality of subseries of the considered components. Asymptotic separability is considered if L and/or K tend to infinity.

For sufficiently long time series, SSA can approximately separate, for example, signal and noise, sine waves with different frequencies, trend and seasonality (Golyandina et al. 2001; Chapter 6, Section 1.5; Golyandina and Zhigljavsky 2013; Section 2.3.3).

Let us demonstrate the separability of two sinusoids with different frequencies ω_1 and ω_2: $x_n^{(i)} = A_i \cos(2\pi\omega_i n + \phi_i)$. These sinusoids are asymptotically weakly separable; that is, their subseries are asymptotically orthogonal as their lengths tend to infinity. However, the rate of convergence depends on the difference between the frequencies. If the frequencies are close and the time series length is not long enough, the two series can be far from orthogonal and therefore not separable. Note that two sinusoids with equal amplitudes are asymptotically weakly separable, but not strongly asymptotically separable and therefore are mixed in the SSA decompositions.

2.1.3.1 Separability Measure

The so-called **w**-correlation matrix contains very helpful information that can be used for detection of separability and identification of groups. This matrix consists of weighted cosines of angles between the reconstructed time series components.

Let w_n ($n = 1, 2, \ldots, N$) be the weights defined in (2.2): w_n is equal to the number of times the series element x_n appears in the trajectory matrix. Define the **w**-scalar product of time series of length N as $(\mathbb{Y}_N, \mathbb{Z}_N)_{\mathbf{w}} = \sum_{n=1}^{N} w_n y_n z_n = \langle \mathbf{Y}, \mathbf{Z} \rangle_{\mathrm{F}}$, where $\mathbf{Y}$ and $\mathbf{Z}$ are the L-trajectory matrices of the series $\mathbb{Y}_N$ and $\mathbb{Z}_N$, respectively. Define the so-called **w**-correlation between $\mathbb{Y}_N$ and $\mathbb{Z}_N$ as

$$\rho_{\mathbf{w}}(\mathbb{Y}_N, \mathbb{Z}_N) = (\mathbb{Y}_N, \mathbb{Z}_N)_{\mathbf{w}} / (\|\mathbb{Y}_N\|_{\mathbf{w}} \|\mathbb{Z}_N\|_{\mathbf{w}}).$$

Well-separated components in (2.5) have weak (or zero) correlation whereas poorly separated components typically have high correlation. Therefore, looking at the matrix of **w**-correlations between elementary reconstructed series $\widetilde{\mathbb{X}}_N^{(i)}$ and $\widetilde{\mathbb{X}}_N^{(j)}$ one can find groups of correlated series components and use this information for the subsequent grouping. One of the main rules is: "do not include highly correlated components into different groups." The **w**-correlations can also be used for checking the grouped decomposition.

It is very instructive to depict the matrix of absolute values of **w**-correlations between the series components graphically in the white-black scale, where small correlations are shown in white and correlations with their absolute values close to 1 are shown in black; see, for example, Figs. 2.4 and 2.15.

2.1.3.2 Choice of Parameters

The conditions of (approximate) separability yield recommendations for the choice of the window length L: it should be large enough ($L \sim N/2$) and if we want to extract a periodic component with known period, then the window lengths, which are divisible by the period, provide better separability. Choice of parameters is discussed in Golyandina et al. (2001; Section 1.6) and Golyandina (2010). If we choose a few leading eigentriples, then SSA with small L performs smoothing of the series as a filter of order $2L-1$, see Golyandina and Zhigljavsky (2013; Section 3.9). Generally, the choice of the window length is important but the result is usually stable with respect to small changes in the values of L.

If the time series has a complex structure, then the so-called Sequential SSA (Golyandina et al. 2012a; Section 2.5.5) is recommended. Sequential SSA consists of two stages; at the first stage, trend is extracted with a small window length and then periodic components are detected and extracted from the residual with $L \sim N/2$.

2.1.3.3 Justification

If we use SSA as a model-free and exploratory technique, then the justification of the decomposition cannot be formal; it must be based on the separability theory and the interpretability of the results. Real-time or batch processing by SSA is possible if the class of series is not too broad and well-determined so that one can fix the rule for choosing proper parameters. For performing statistical testing, a model of the time series should be specified.

2.1.4 Algorithm

In Sect. 2.1.1 we described the Basic SSA method. Here we formally present the algorithm of Basic SSA. Note that the RSSA package implements the algorithms efficiently (see Sect. 1.5.4 for a brief discussion). Since effective implementation is complicated and hides the sense of algorithm steps, we write down the algorithms in the original form.

Input data for the whole algorithm of Basic SSA are the window length and the way of grouping of the elementary components $\mathbf{X}_i$ in (2.3). However, the rule for grouping is made after the decomposition (2.3) is made. Therefore, the grouping becomes the input data for Reconstruction stage. For this reason, we split the algorithm into two parts. Note that modifications of Basic SSA mostly differ by Decomposition step only; Reconstruction stage is the same for virtually all SSA versions.

Algorithm 2.1 Basic SSA: decomposition

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

Reconstruction algorithms are almost the same for different versions of SSA; their inputs have a decomposition of the trajectory matrix into a sum of rank-one matrices and the split of the rank-one components into groups. We therefore formulate a general algorithm of reconstruction and will make comments concerning specific features of modifications in the corresponding sections. The specific feature of Basic SSA is: the input decomposition is the SVD and hence the biorthogonal decomposition into the rank-one components is ordered according to component contribution $\sigma_i^2 = \lambda_i$ so that $\sigma_1 \geq \ldots \geq \sigma_d$.

Algorithm 2.2 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 $\|U_i\| = \|V_i\| = 1$; partition
    of indices: $\{1, \ldots, d\} = \bigsqcup_{j=1}^{m} I_j$.
Output: Decomposition of the time series $\mathbb{X}$ into 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: Compute $\mathbb{X} = \mathbb{X}_1 + \ldots + \mathbb{X}_m$, where $\mathbb{X}_i = \mathcal{T}_{\mathrm{SSA}}^{-1} \circ \Pi_{\mathcal{H}}(\mathbf{X}_{I_i})$.
```

2.1.5 *Basic SSA in* RSSA

2.1.5.1 Description of Functions

The main function of RSSA is `ssa` which constructs the so-called `ssa` object holding the decomposition and various auxiliary information necessary for performing a particular implementation. The function has many arguments; some of them are common for different types of SSA, some are specific. Below we will outline the main arguments of `ssa` in a typical function call:

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

where `N` is the series length.

Arguments:

`x` is an object to be decomposed. For Basic SSA it is assumed to be a simple vector or vector-like object (e.g., univariate `ts` or `zooreg` object). Everything else is coerced to a vector.

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

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

`kind` specifies the version of SSA to be used; it can be omitted in non-ambiguous cases (e.g., when `x` is a vector or a `ts` object).

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

In addition to constructing an `ssa` object `s`, by default the `ssa` function also performs Decomposition step and thus corresponds to Algorithm 2.1. If necessary, Decomposition stage can be skipped setting the argument `force.decompose` to `FALSE`.

The function returns an `ssa` object. The precise layout of the object is hidden and can be different in different versions of the package. However, there are several fields that are available to users and can be extracted with the help of `$` operator, namely:

`s$sigma` is a vector of singular values;

`s$U` is a matrix of eigenvectors;

`s$V` is a matrix of factor vectors. Note that it may not be calculated for particular selections of the SVD method.

The number of the calculated singular values, eigenvectors, and factor vectors can be obtained by means of the functions `nsigma(s)`, `nu(s)`, and `nv(s)` correspondingly. Call of `summary(s)` provides a consolidated information about the `ssa` object.

The next function is `reconstruct` which implements Reconstruction stage (Algorithm 2.2). The basic signature of the function call is

```
r <- reconstruct(s, groups = list(trend = 1:2, c(3:6,9)))
```

Arguments:

`s` is an `ssa` object holding the decomposition.

`groups` is a list of numeric vectors consisting of indices of the elementary components used for reconstruction; the entries of the list can be named.

`drop` acts only if one group is chosen; `TRUE` value means that the result is transformed from the list of the reconstructed series to the reconstructed series itself (`FALSE` is default).

The function returns a list of reconstructed objects. Elements of the list have the same names as elements of `groups` (e.g., `r$trend`). If a group is unnamed, then the corresponding component will obtain the name `Fn`, where `n` is its index in the `groups` list (e.g. `r$F2`).

By default, the routine tries to preserve all the attributes of the input object. In this way, for example, the reconstruction result of the `ts` object is the `ts` object with the same time scale. This feature can be disabled by setting the argument `drop.attributes` to `TRUE`.

2.1.5.2 SVD Methods

In many cases only few leading eigentriples are of interest for the SSA analysis. Thus the full SVD of the trajectory matrix can yield large computational and memory space burdens. Instead, the so-called Truncated SVD can be used and only a number of desired leading eigentriples can be computed. Four different SVD implementations are available in RSSA and can be specified via the argument `svd.method` of the function `ssa`:

- `"auto"`—Automatic method of selection depending on the series length, the window length, and the number of desired eigentriples.
- `"nutrlan"`—Truncated SVD via thick-restart Lanczos bidiagonalization algorithm (Yamazaki et al. 2008). The method internally calculates the eigenvalues and eigenvectors of the matrix $\mathbf{X}\mathbf{X}^{\mathrm{T}}$. Factor vectors are calculated on-fly during Reconstruction stage when necessary.
- `"propack"`—Implicitly restarted Lanczos bidiagonalization with partial reorthogonalization (Larsen 1998). The method calculates the truncated SVD of the trajectory matrix $\mathbf{X}$ (and hence calculates the factor vectors as well).
- `"eigen"` and `"svd"`—Full decomposition of the trajectory matrix using either eigendecomposition or SVD routines from `LAPACK` (Anderson et al. 1999). Using `ssa` with these `svd.methods` yields the straightforward implementations of Basic SSA algorithm without computational and space complexity reductions via additional sophisticated algorithms. Note that both methods perform full decompositions and thus the argument `neig` (which allows one to request a desired number of eigentriples) is silently ignored for these methods.

Selecting the best method for performing the SVD is not easy. However, there are several simple rules of thumb which work well in most situations.

First of all, it is unwise to use the Lanczos-based truncated SVD methods if the trajectory matrix is small or "wide." This corresponds to small series lengths (say, $N < 100$) or small window lengths ($L < 50$). Also, it is unwise to ask for too many eigentriples: when more than $L/2$ eigentriples are needed then it is better to use the full SVD instead of a truncated one. The SVD method `eigen` works best for small L.

Usually the method `propack` tends to be slightly faster and more numerically stable than `nutrlan`; however, it may yield considerable memory consumption when factor vectors are large. For example, for a time series of length 87000 and window length 43500, the decomposition with the method `nutrlan` took 16 s while with `propack` it took only 13 s (we are not aware of any other implementation of SVD, besides RSSA implementations, which can perform the decomposition with such a large window length at all). The memory consumption for the latter method is twice higher than the consumption of the former. This difference is more important for multivariate versions of SSA and should not be a problem in the 1D case.

A specific feature of the Lanczos-based truncated SVD methods is their possible non-convergence in the case of coinciding eigenvalues. In real-life time series, the exact coincidence of eigenvalues happens very rarely and hence we can often enjoy the outstanding effectiveness of these SVD methods. For a time series of finite rank r, zero eigenvalue has the multiplicity $L - r$; therefore, the number of the truncated components should be chosen appropriately, e.g. $r + 1$ components can be requested for calculation. Since for a time series of finite rank r even a small window length $L \simeq r + 1$ can be sufficient for the analysis and forecasting, the use of the `eigen` method is recommended.

By default, the method `nutrlan` is selected. However, the function `ssa` tries to correct the selection, when the chosen method is clearly not the most suitable. In particular, for short series, small window lengths or large number of desired eigentriples, the method `eigen` is automatically selected.

It should be noted that the truncated SVD implementations were extracted from the RSSA package into a separate package SVD (Korobeynikov et al. 2016) and thus can be used independently.

2.1.5.3 Typical Code

For demonstration, we consider the series of sales of fortified wines (shortly “FORT”) taken from the dataset “Australian Wines” (monthly wine sales in thousands of liters). The full dataset contains sales from January, 1980, to July, 1995 (187 points). However, the data after June, 1994 have missing values. Therefore, we analyze the first 174 points.

Fragment 2.1.1 contains the standard code for loading the package RSSA and for the input of the data included into the package.

Fragment 2.1.1 (“Australian Wines”: Input)

```
> library("Rssa")
> data("AustralianWine", package = "Rssa")
> wine <- window(AustralianWine, end = time(AustralianWine)[174])
```

Fragment 2.1.2 contains a typical code for extraction of the trend and seasonality. The resultant decomposition is depicted in Fig. 2.1.

Fragment 2.1.2 (“FORT”: Reconstruction)

```
> fort <- wine[, "Fortified"]
> s.fort <- ssa(fort, L = 84, kind = "1d-ssa")
> r.fort <- reconstruct(s.fort,
+                       groups = list(Trend = 1,
+                                     Seasonality = 2:11))
> plot(r.fort, add.residuals = TRUE, add.original = TRUE,
+      plot.method = "xyplot",
+      superpose = TRUE, auto.key = list(columns = 2))
```

Roughly speaking (see details in Golyandina and Korobeynikov (2013)), `ssa` performs Steps 1 and 2 of the algorithmic scheme described in Sect. 1.1.1, while `reconstruct` performs steps 3 and 4 of the algorithm. The argument values `kind = "1d-ssa"` and `svd.method = "auto"` are default and can be omitted. The

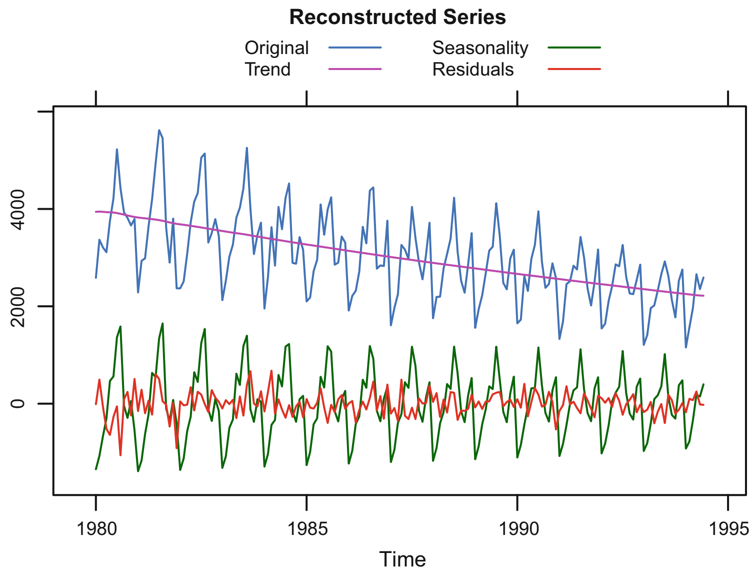


Fig. 2.1 "FORT": Decomposition

function `plot` applied to the reconstruction object performs different special kinds of plotting. In addition to specific parameters, this function can include parameters of the function `xyplot` from the standard package LATTICE (see the last two parameters of `plot` in Fragment 2.1.2).

The choice of groups for reconstruction was made on the base of the following information obtained from the `ssa` object:

1. one-dimensional (1D) figures of the eigenvectors U_i (Fig. 2.2),
2. two-dimensional (2D) figures of the eigenvectors (U_i, U_{i+1}) (Fig. 2.3), and
3. matrix of **w**-correlations $\rho_\mathbf{w}$ between the elementary reconstructed series (functions `wcor` and `plot`, Fig. 2.4).

The following fragment shows the code that reproduces Figs. 2.2–2.5.

Fragment 2.1.3 ("FORT": Identification)

```
> plot(s.fort, type = "vectors", idx = 1:8)
> plot(s.fort, type = "paired", idx = 2:11, plot.contrib = FALSE)
> print(parestimate(s.fort, groups = list(2:3, 4:5),
+                   method = "pairs"))
$F1
   period      rate   |     Mod      Arg  |      Re        Im
   11.971   0.000000  |  1.00000    0.52  |  0.86540   0.50109
$F2
   period      rate   |     Mod      Arg  |      Re        Im
    4.005   0.000000  |  1.00000    1.57  |  0.00177   1.00000
> plot(wcor(s.fort, groups = 1:30),
+             scales = list(at = c(10, 20, 30)))
> plot(reconstruct(s.fort, groups = list(G12 = 2:3, G4 = 4:5,
+                                        G6 = 6:7, G2.4 = 8:9)),
+       plot.method = "xyplot", layout = c(2, 2),
+       add.residuals = FALSE, add.original = FALSE)
```

Let us explain how the figures obtained by means of Fragment 2.1.3 can help to perform the grouping. Figure 2.2 shows that the first eigenvector is slowly-varying and therefore the eigentriple ET1 should be included into the trend group. Figure 2.3 shows that the pairs 2–3, 4–5, 6–7, 8–9, 10–11 are produced by modulated sine-waves, since the corresponding 2D-scatterplots of eigenvectors resemble regular

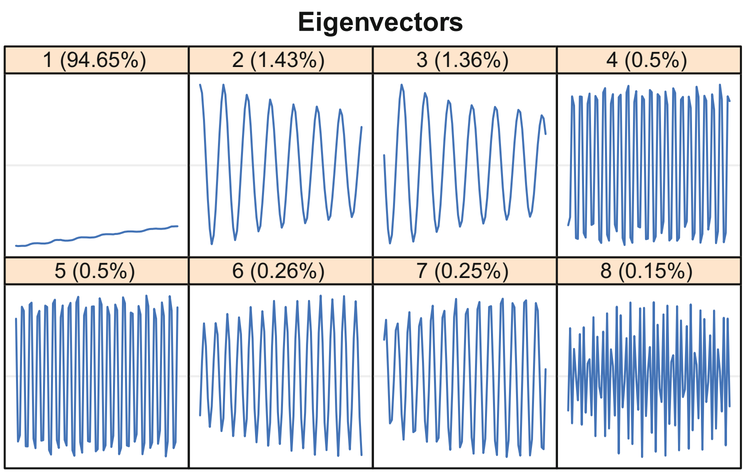


Fig. 2.2 “FORT”: 1D graphs of eigenvectors

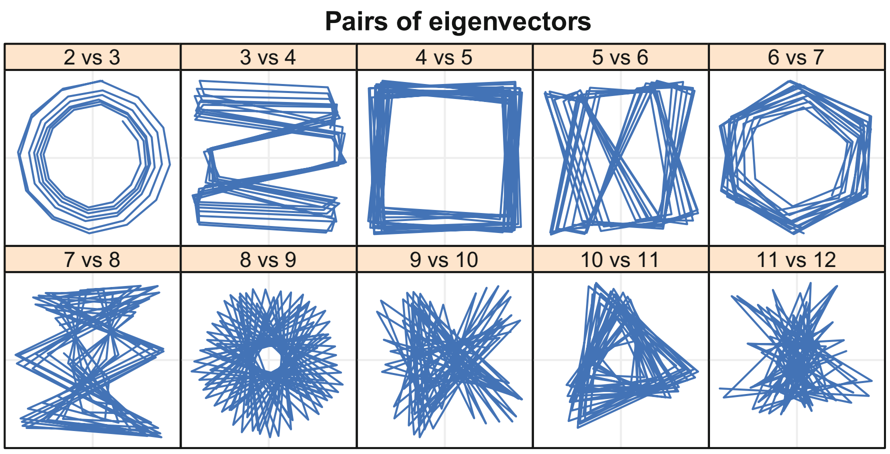


Fig. 2.3 “FORT”: 2D scatterplots of eigenvectors

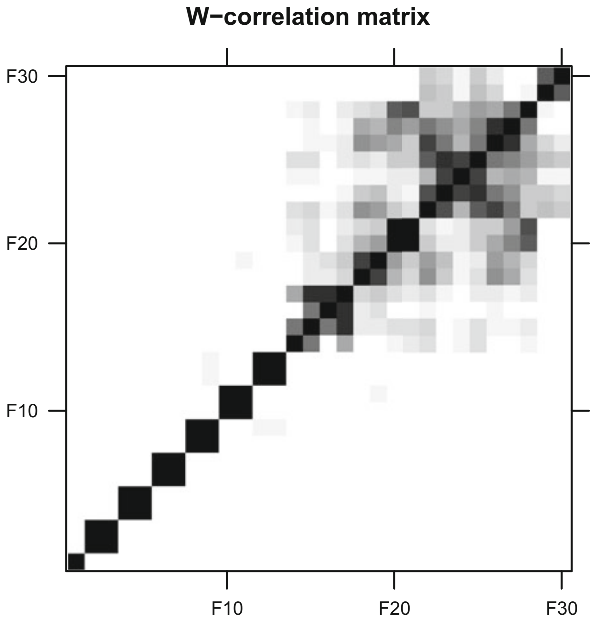


Fig. 2.4 "FORT": Weighted correlations

polygons. This way of identification is based on the following properties: a sine wave has rank 2 and produces two eigentriples, which are sine waves with the same frequency and have a phase shift exactly or approximately equal to $\pi/2$, due to the orthogonality of eigenvectors.

By counting the numbers of polygon vertices in Fig. 2.3, the periods of the sine-waves can be determined as 12, 4, 6, 2.4, 3. Alternatively, automatic methods of frequency calculation can be employed, such as LS-ESPRIT and TLS-ESPRIT methods (Roy and Kailath 1989). These methods are implemented in RSSA in the function `parestimate`, see Sect. 3.1, and are described in Golyandina et al. (2001; Sections 2.4.2.4. and 3.8.2) and Golyandina and Korobeynikov (2013) for one-dimensional time series. The periods, calculated by the automatic `parestimate` method in Fragment 2.1.3, agree with the numbers of vertices in Fig. 2.3 for the five pairs listed.

The matrix of absolute values of **w**-correlations in Fig. 2.4 is depicted in grayscale (white color corresponds to zero and black color corresponds to the absolute values equal to 1). Figure 2.4 confirms that the indicated pairs are separated between themselves and also from the trend component, since the **w**-correlations between the pairs are small, while **w**-correlations between the components from the same pair are very large. The block of 12–84 components is "gray," therefore we can expect that these components are mixed and are largely produced by noise.

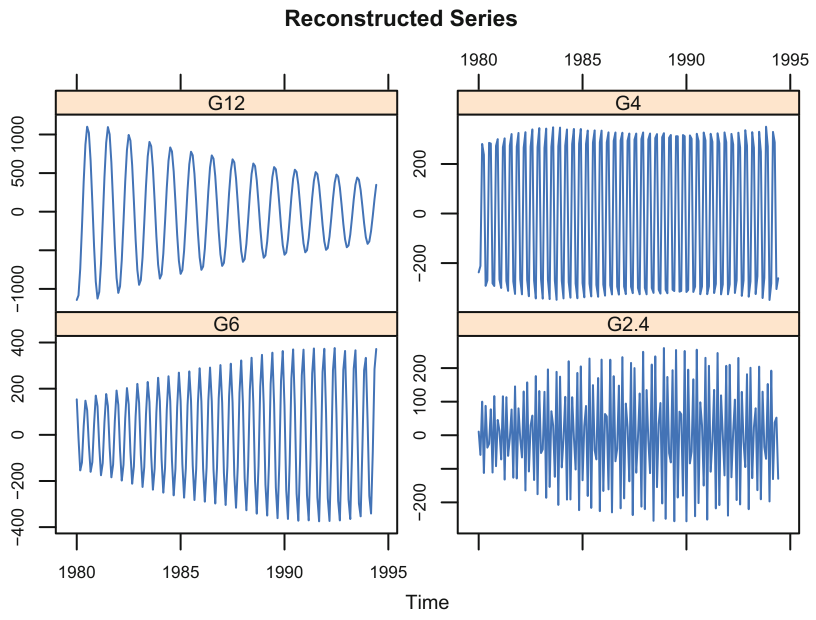


Fig. 2.5 “FORT”: Reconstructed sine waves

Figure 2.5 contains four reconstructed modulated sine waves and shows that several sine waves have increasing amplitudes, while others are decreasing; the same can be seen in Fig. 2.2. In Fig. 2.1, we grouped the modulated sine waves and obtained the seasonal component with varying annual behavior.

2.2 Toeplitz SSA

2.2.1 Method

Toeplitz SSA differs from Basic SSA only in Step 2 of the generic scheme presented in Fig. 1.1; that is, in the decomposition of $\mathbf{X}$ into rank-one matrices. Basic SSA uses the SVD at this step with U_i calculated as eigenvectors of $\mathbf{S} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$. If the length N of the series $\mathbb{X}$ is not large and the series is assumed to be stationary, then the usual recommendation is to replace the matrix $\mathbf{S}$ by some other matrix, which is constructed under the assumption of stationarity.

Note first that in Basic SSA we can consider the *lag-covariance matrix* $\mathbf{C} = \mathbf{S}/K$ instead of $\mathbf{S}$ for obtaining the SVD of the trajectory matrix $\mathbf{X}$. Indeed, the eigenvectors of the matrices $\mathbf{S}$ and $\mathbf{C}$ are the same and the eigenvalues differ only by the factor $1/K$.

Denote by $c_{ij} = c_{ij}(N)$ the elements of the lag-covariance matrix $\mathbf{C}$. If the time series is stationary with zero mean, L is fixed and $K \to \infty$, then $\lim c_{ij}(N) = R_{\mathbb{X}}(|i-j|)$ as $N \to \infty$, where $R_{\mathbb{X}}(k)$ stands for the lag-k term of the time series autocovariance function. We can therefore define a Toeplitz version of the lag-covariance matrix by putting equal values $\widetilde{c}_{ij}$ at each matrix auxiliary diagonal $|i-j| = k$. The most natural way for defining the values $\widetilde{c}_{ij}$ and the corresponding matrix $\widetilde{\mathbf{C}}$ is to compute

$$\widetilde{c}_{ij} = \frac{1}{N-|i-j|} \sum_{m=1}^{N-|i-j|} x_m x_{m+|i-j|}, \quad 1 \le i, j \le L. \tag{2.9}$$

While using this formula it is usually assumed that the time series $\mathbb{X}$ is centered so that the mean $\bar{x} = \sum_{i=1}^{N} x_i/N$ is subtracted from all $x_i \in \mathbb{X}$.

Let $L \le K$ and denote by P_i $(i = 1, \ldots, L)$ the eigenvectors of $\widetilde{\mathbf{C}}$; these vectors form an orthonormal basis of R^L. Then the decomposition on elementary matrices can be written as $\mathbf{X} = \sum_{i=1}^{L} P_i(\mathbf{X}^{\mathrm{T}} P_i)^{\mathrm{T}}$. Ordering of addends is performed by the magnitudes of $\sigma_i = \|\mathbf{X}^{\mathrm{T}} P_i\|$. Note that this ordering generally differs from the ordering of eigenvalues of the matrix $\widetilde{\mathbf{C}}$ corresponding to the eigenvectors P_i. Some of these eigenvalues could even be negative as the matrix $\widetilde{\mathbf{C}}$ is not necessarily positive definite.

If the original series is stationary with zero mean, then the use of *Toeplitz lag-covariance matrix* $\widetilde{\mathbf{C}}$ can be more appropriate than the use of the lag-covariance matrix $\mathbf{C}$. On the other hand, Toeplitz SSA is not suitable for nonstationary series; if the original series has an influential nonstationary component, then Basic SSA works better than Toeplitz SSA. For example, if we are dealing with a pure exponential series, then it is described by a single eigentriple for any window length, while Toeplitz SSA produces L eigentriples for the window length L; moreover, the eigenvectors in Toeplitz SSA have some special properties (Andrew 1973), which do not depend on the series. The same effect takes place for the linear series, exponential-cosine series, etc.

A number of papers devoted to SSA analysis of climatic time series (e.g., Ghil et al. (2002), where Toeplitz SSA is often referred to as a VG method) consider Toeplitz SSA as the main version of SSA and state that the difference between the Basic and Toeplitz versions of SSA is marginal. However, using the Toeplitz version of SSA is unsafe if the series contains a trend or oscillations with increasing or decreasing amplitude. Examples of effects observed when Toeplitz SSA is applied to non-stationary time series are presented in Golyandina (2010). For the study of theoretical properties of Toeplitz SSA, see, for example, Harris and Yan (2010).

2.2.2 *Algorithm*

Algorithm 2.3 Toeplitz SSA: decomposition

```
Input: Time series $\mathbb{X}$ of length $N$, window length $L$.
Output: Decomposition of the trajectory matrix on elementary matrices $\mathbf{X} = \mathbf{X}_1 + \ldots + \mathbf{X}_L$, where
    $\mathbf{X}_i = \sigma_i P_i Q_i^{\mathrm{T}}$, $\|P_i\| = \|Q_i\| = 1$.
1: Construct the trajectory matrix $\mathbf{X} = \mathcal{T}_{\mathrm{SSA}}(\mathbb{X})$.
2: Obtain the decomposition
    $$\mathbf{X} = \sum_{i=1}^{L} \sigma_i P_i Q_i^{\mathrm{T}} = \mathbf{X}_1 + \ldots + \mathbf{X}_L, \qquad (2.10)$$
    where $\{P_i\}_{i=1}^{L}$ are eigenvectors of the matrix $\widetilde{\mathbf{C}}$ with entries computed by (2.9), $S_i = \mathbf{X}^{\mathrm{T}} P_i$,
    $Q_i = S_i / \|S_i\|$ and $\sigma_i = \|\mathbf{X}_i\|_{\mathrm{F}} = \|S_i\|$. Components are ordered by the magnitudes of $\sigma_i$:
    $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_L$.
```

The reconstruction algorithm is exactly the same as Algorithm 2.2 with vectors (P_i, Q_i) substituted for (U_i, V_i).

2.2.3 *Toeplitz SSA in* Rssa

2.2.3.1 Description of Functions

In Rssa, Toeplitz SSA is implemented via the same `ssa` function as Basic SSA. One should use `kind="toeplitz-ssa"` to enable the Toeplitz version. All other arguments have the meaning as described in Sect. 2.1.5:

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

where `N` is the series length.

Note that the triples (σ_i, P_i, Q_i), which are generated by the decomposition (2.10), are also called eigentriples in Rssa and the access to $\{P_i\}$ and $\{Q_i\}$ is provided by the codes `s$U` and `s$V`.

2.2.3.2 Typical Code

To our mind, Toeplitz SSA has a limited range of applications, since it requires stationarity for both signal and noise, which is first unnatural and second impossible to verify. Hence, in the example below we use simulated data (Fragment 2.2.1).

As mentioned above, before using Toeplitz SSA it is recommended to center the series. Then Toeplitz SSA can be used in exactly the same way as Basic SSA.

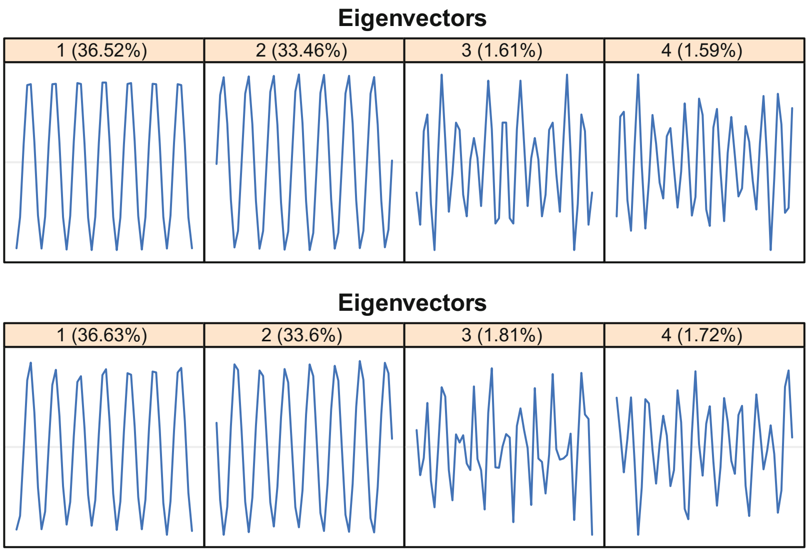


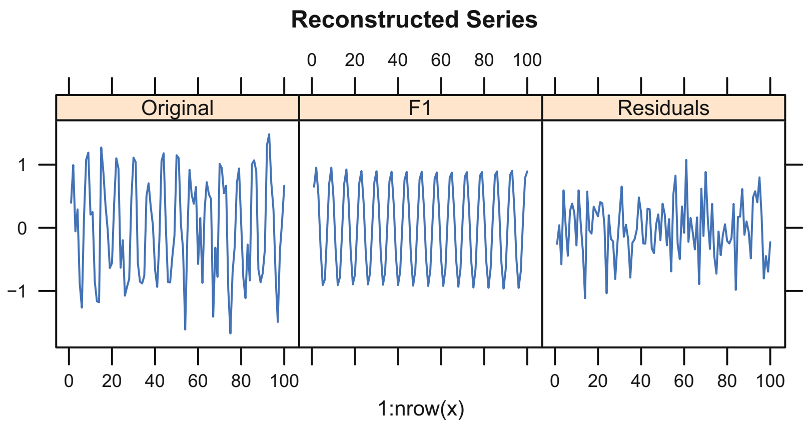
Fig. 2.6 Noisy sinusoid: 1D graphs of eigenvectors (top: Toeplitz SSA, bottom: Basic SSA)

Fragment 2.2.1 (Noisy Sinusoid: Toeplitz SSA)

```
> N <- 100
> sigma <- 0.5
> set.seed(1)
> F <- sin (2 * pi * (1:N) / 7) + sigma * rnorm(N)
> Fcenter <- F - mean(F)
> st <- ssa(Fcenter, L = 50, kind = "toeplitz-ssa")
> s <- ssa(F, L = 50, kind = "1d-ssa")
> p <- plot(s, type = "vectors", idx = 1:4, layout = c(4, 1))
> pt <- plot(st, type = "vectors", idx = 1:4, layout = c(4, 1))
> plot(pt, split = c(1, 1, 1, 2), more = TRUE)
> plot(p, split = c(1, 2, 1, 2), more = FALSE)
> pt <- plot(reconstruct(st, groups = list(1:2)),
+            plot.method = "xyplot", layout = c(3, 1))
> p <- plot(reconstruct(s, groups = list(1:2)),
+           plot.method = "xyplot", layout = c(3, 1))
> plot(pt, split = c(1, 1, 1, 2), more = TRUE)
> plot(p, split = c(1, 2, 1, 2), more = FALSE)
```

Here we see that for Toeplitz SSA the amplitude of the sinusoid reconstruction is closer to a constant than that for Basic SSA (Figs. 2.6 and 2.7). Generally, eigenvectors for Toeplitz SSA are more regular, even for the noise decomposition. This is due to the properties of eigenvectors of Toeplitz matrices (Fig. 2.7).

Note that typically the window length for Toeplitz SSA should be rather small, since the used estimate of auto-covariance matrix of the series tends to the true auto-covariance matrix only if L is fixed and K tends to infinity.



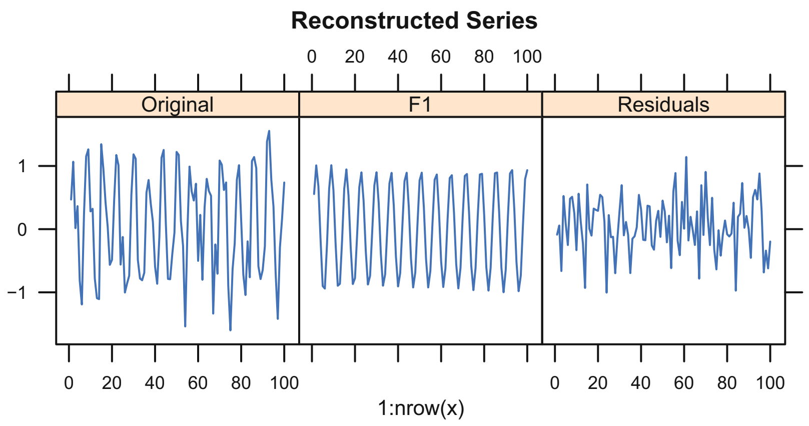


Fig. 2.7 Noisy sinusoid: Reconstruction (top: Toeplitz SSA, bottom: Basic SSA)

2.2.3.3 Simulated Example

As was mentioned above, for stationary time series the use of Toeplitz SSA is appropriate, while it makes no sense to apply Toeplitz SSA for trend extraction. Also, if a periodic component (e.g., a seasonal behavior) is changing in time, the accuracy of signal reconstruction is worse than that for Basic SSA.

Let us demonstrate this by means of simulation. We consider the signal in the form $s_n = \exp(\alpha n)\sin(2\pi n/7), n = 1, \ldots, 100$, and the noisy series $x_n = s_n + \sigma\varepsilon_n$, where $\sigma = 0.5$, ε_n is white Gaussian noise.

For $\alpha = 0$ this series can be considered as stationary with stationary deterministic signal (see definition in Golyandina et al. (2001; Sections 1.7.2 and 6.4)). For non-zero α, this series is not stationary. Thus, let us consider α from [0, 0.01].

Fragment 2.2.2 (Simulation: Comparison of Toeplitz and Basic SSA)

```
> SIMUL <- FALSE
> N <- 100
> sigma <- 0.5
> set.seed(1)
> alpha <- seq(0.0, 0.01, 0.001)
> L <- 50
> Q <- 1000
> if (SIMUL) {
+   RMSE <-
+     sapply(alpha,
+            function(a) {
+              sqrt(rowMeans(replicate(Q, {
+                S <- exp(a * (1:N)) * sin(2 * pi * (1:N) / 7)
+                F <- S + sigma * rnorm(N)
+                Fcenter <- F - mean(F)
+                st <- ssa(Fcenter, L = L, kind = "toeplitz-ssa")
+                s <- ssa(F, L = L, kind = "1d-ssa")
+                rec <- reconstruct(s, groups = list(1:2))$F1
+                rec.t <- reconstruct(st, groups = list(1:2))$F1
+                c("1d-ssa" = mean((rec - S)^2),
+                  "toeplitz" = mean((rec.t - S)^2))
+              })))
+            })
+
+   toeplitz.sim <- as.data.frame(t(RMSE))
+ } else {
+   data("toeplitz.sim", package = "ssabook")
+ }
> matplot(alpha, toeplitz.sim, type = "l", ylim = c(0, 0.25))
```

Figure 2.8 shows the dependence of the reconstruction accuracy on the exponential rate α constructed with the help of the code from Fragment 2.2.2. The window length $L = 50$ was chosen and RMSE was taken as a measure of accuracy. One can see that the accuracy of Basic SSA reconstruction does not depend on α, while the error of Toeplitz SSA increases as α increases. If a series is very close to a stationary series, Toeplitz SSA has a slightly smaller error than Basic SSA. However, already for $\alpha = 0.01$ Toeplitz SSA makes considerably larger errors than Basic SSA.

2.3 SSA with Projection

2.3.1 Method

As was mentioned in Sect. 1.2.1.1, the goal of SSA with projection is an efficient use of a known information about series components. The well-known methods of SSA with centering and SSA with double centering for extraction of constant and linear trends, respectively, are special cases of SSA with projection.

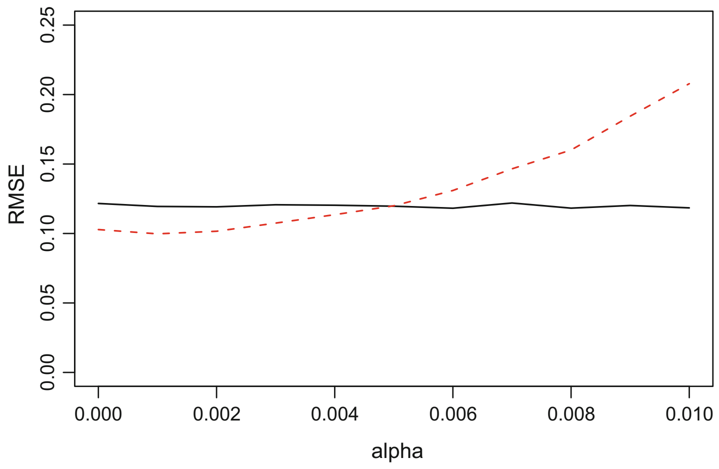


Fig. 2.8 Simulation: Reconstruction error of Toeplitz (dash red line) and Basic SSA (solid black line)

Let $\mathbb{X}$ be a time series of length N, L be the window length, $K=N-L+1$, and $\mathbf{X}$ be the trajectory matrix. The general form of centering can be expressed as follows:

1. Calculation of a special matrix $\mathbf{C}^{(\text{center})} = \mathbf{C}(\mathbf{X})$ based on a priori information.
2. Computation of $\mathbf{X}^\star = \mathbf{X} - \mathbf{C}^{(\text{center})}$.
3. Construction of the SVD: $\mathbf{X}^\star = \sum_{i=1}^{d^\star} \sqrt{\lambda_i^\star} U_i^\star (V_i^\star)^{\mathrm{T}}$.

As a result, we obtain the decomposition $\mathbf{X} = \mathbf{C}^{(\text{center})} + \sum\limits_{i=1}^{d^\star} \sqrt{\lambda_i^\star} U_i^\star (V_i^\star)^{\mathrm{T}}$.

Denote $E_M = (1, \ldots, 1)^{\mathrm{T}} \in \mathsf{R}^M$ the M-vector of ones. The following three types of centering can be considered (Golyandina et al. 2001; Sections 1.7.1 and 6.3):

- *Single row centering* when $\mathbf{C}_{\text{row}}^{(\text{center})}(\mathbf{X}) = (\mathbf{X}E_K/K)E_K^{\mathrm{T}}$ corresponds to averaging by rows; that is, each element of a row of $\mathbf{C}_{\text{row}}^{(\text{center})}$ consists of the average of the corresponding row of the trajectory matrix.
- *Single column centering* when $\mathbf{C}_{\text{col}}^{(\text{center})}(\mathbf{X}) = E_L(\mathbf{X}^{\mathrm{T}}E_L/L)^{\mathrm{T}}$ corresponds to averaging by columns.
- *Double centering* when $\mathbf{C}_{\text{both}}^{(\text{center})} = \mathbf{C}_{\text{row}}^{(\text{center})} + \mathbf{C}_{\text{col}}^{(\text{center})}\big(\mathbf{X} - \mathbf{C}_{\text{row}}^{(\text{center})}(\mathbf{X})\big)$ corresponds to averaging by both rows and columns.

Note that the single centering can be considered as a projection of rows or columns of $\mathbf{X}$ on $\operatorname{span}(E_K)$ or $\operatorname{span}(E_L)$, respectively, since $E_K E_K^{\mathrm{T}}$ and $E_L E_L^{\mathrm{T}}$ are exactly the matrices of the projection operators. Therefore, centering in SSA can be considered as a preliminary projection of the trajectory matrix on a given subspace; the residual matrix will be subsequently expanded by the SVD or any other decomposition.

Let us generalize this approach to projections to arbitrary spaces following Golyandina and Shlemov (2017). Let $\Pi_{\mathrm{col}} : \mathsf{R}^L \to \mathcal{L}_{\mathrm{col}}$ and $\Pi_{\mathrm{row}} : \mathsf{R}^K \to \mathcal{L}_{\mathrm{row}}$ be orthogonal projectors, where $\mathcal{L}_{\mathrm{col}} \in \mathsf{R}^L$ is called the column projection space and $\mathcal{L}_{\mathrm{row}} \in \mathsf{R}^K$ is called the row projection space. For any $\mathbf{Y} \in \mathsf{R}^{L\times t}$, denote $\Pi_{\mathrm{col}}(\mathbf{Y})$ the matrix consisting of the columns, which result from projections of the columns of $\mathbf{Y}$. Similarly, for any $\mathbf{Y} \in \mathsf{R}^{t\times K}$, denote $\Pi_{\mathrm{row}}(\mathbf{Y})$ the matrix consisting of the rows, which result from projections of the rows of $\mathbf{Y}$.

Denote a basis of the column projection space $\{P_i, i = 1, \ldots, p\}$ and a basis of the row projection space $\{Q_i, i = 1, \ldots, q\}$. Let $\mathbf{P} = [P_1 : \ldots : P_p]$ and $\mathbf{Q} = [Q_1 : \ldots : Q_q]$. Without loss of generality we assume that $\{P_i, i = 1, \ldots, p\}$ and $\{Q_i, i = 1, \ldots, q\}$ are orthonormal bases of $\mathcal{L}_{\mathrm{col}}$ and $\mathcal{L}_{\mathrm{row}}$ (otherwise, we can perform ortho-normalization).

In SSA with projection, the scheme of SSA with centering is extended to arbitrary projections; that is, $\mathbf{C} = \Pi_{\mathrm{col}}(\mathbf{X})$ for the column projection, $\mathbf{C} = \Pi_{\mathrm{row}}(\mathbf{X})$ for the row projection and $\mathbf{C} = \Pi_{\mathrm{both}}(\mathbf{X})$ for the double projection, where

$$\begin{aligned}\Pi_{\mathrm{both}}(\mathbf{X}) = \Pi_{\mathrm{row}}(\mathbf{X}) + \Pi_{\mathrm{col}}(\mathbf{X} - \Pi_{\mathrm{row}}(\mathbf{X}))\\ = \Pi_{\mathrm{col}}(\mathbf{X}) + \Pi_{\mathrm{row}}(\mathbf{X} - \Pi_{\mathrm{col}}(\mathbf{X}))\\ = \Pi_{\mathrm{row}}(\mathbf{X}) + \Pi_{\mathrm{row}}(\mathbf{X}) - (\Pi_{\mathrm{col}} \circ \Pi_{\mathrm{row}})(\mathbf{X}).\end{aligned} \tag{2.11}$$

If either the column or row basis is absent (that is, the space for column or row projection consists of zero), then we formally set the corresponding projector to be zero operator implying $\mathbf{C} = \Pi_{\mathrm{both}}(\mathbf{X})$ in any mode.

A general form of the decomposition provided by SSA with projection is

$$\mathbf{X} = \mathbf{C} + \sum_{i=1}^{d^\star} \sqrt{\lambda_i^\star} U_i^\star (V_i^\star)^{\mathrm{T}}, \tag{2.12}$$

where $\mathbf{C} = \Pi_{\mathrm{both}}(\mathbf{X})$ and $\sum_{i=1}^{d^\star} \sqrt{\lambda_i^\star} U_i^\star (V_i^\star)^{\mathrm{T}}$ is the SVD of $\mathbf{X}^\star = \mathbf{X} - \mathbf{C}$.

It is shown in Golyandina and Shlemov (2017) that (2.12) can be represented as a sum of elementary matrices of rank 1. The matrix $\mathbf{C}$ can be considered as a sum of $q + p$ elementary matrices of the forms $\sigma_i^{(r)} \widetilde{P}_i Q_i^{\mathrm{T}}$, $i = 1, \ldots, q$, and $\sigma_i^{(c)} P_i \widetilde{Q}_i^{\mathrm{T}}$, $i = 1, \ldots, p$. The triples $(\sigma_i^{(r)}, \widetilde{P}_i, Q_i)$ and $(\sigma_i^{(c)}, P_i, \widetilde{Q}_i)$ have the same meaning as eigentriples. For double projection, this representation depends on the order of projections; we will apply the row projector first. Therefore, the decomposition (2.12) can be transformed into a decomposition of $\mathbf{X}$ into a sum of $q + p + d^\star$ elementary rank-one matrices, which are orthogonal with respect to the Frobenius norm $\|\cdot\|$, by construction. As a consequence, the contribution of the projection term $\mathbf{C}$ into the decomposition is measured by $\|\mathbf{C}\|^2/\|\mathbf{X}\|^2$.

Thus, (2.12) is a decomposition of $\mathbf{X}$ on elementary matrix components unambiguously defined. Reconstruction stage is exactly the same as in the Basic SSA method. Note that it has little sense to include the eigentriples produced by projections to different groups, since the projections are performed on the subspaces as a whole. It follows from Golyandina and Shlemov (2017; Lemma 1) that the rank $d^\star$ of the matrix $\mathbf{X}^\star$ obtained after projection cannot be larger than the rank of the original matrix $\mathbf{X}$ and can not be smaller than rank $\mathbf{X} - (q + p)$.

When we use projections, we should expect some specific form for one of the series component. For example, to extract a sine wave using projections, we should know its period, and to extract exponential trend, we should know its rate. These conditions are often too restrictive. A clear exception is extraction of the polynomial trends, when we should assume only the degree of the polynomial to define its trajectory space.

Note finally that SSA with projection can be applied in the shaped version, when the series has gaps.

2.3.2 Appropriate Time Series

For SSA with projection, a known series component with a trajectory matrix $\mathbf{Y}$ should be in agreement with projections so that $\Pi_{\mathrm{col}}(\mathbf{Y}) = \mathbf{Y}$ for the column projection, $\Pi_{\mathrm{row}}(\mathbf{Y}) = \mathbf{Y}$ for the row projection, and $\Pi_{\mathrm{both}}(\mathbf{Y}) = \mathbf{Y}$ for the double projection.

Clearly, for column and row projections, this is true if the corresponding projection is performed on the column or row trajectory space of the known series component. For example, the trajectory space of an exponential component $s_n = \mu^n$ spans $(1, \mu, \ldots, \mu^L)^{\mathrm{T}}$, while the trajectory space of the linear function $s_n = an + b$ spans $(1, 1, \ldots, 1)^{\mathrm{T}}$ and $(1, 2, \ldots, L)^{\mathrm{T}}$ for any a and b.

Let us introduce a condition sufficient for $\Pi_{\mathrm{both}}(\mathbf{X}) = \mathbf{X}$ to hold for the general case of the double projection.

Recall that a series governed by an LRR, whose characteristic polynomial has the given set of roots called characteristic roots, is of the form (1.9).

Theorem 2.1 (Golyandina and Shlemov (2017)) *Let series $\mathbb{Y}^{(m)}$, $m = 1, 2$, be governed by minimal LRRs of orders r_m, $\mathbf{Y}^{(m)}$ be their trajectory matrices. Denote $\{\mu_j;\ j = 1, \ldots, s\}$ the set containing the characteristic roots of both series. Assume that $\mathbb{Y}^{(m)}$, $m = 1, 2$, have the signal roots μ_j, $j = 1, \ldots, s$, with multiplicities $d_j^{(m)} \geq 0$, $\sum_{j=1}^{s} d_j^{(m)} = r_m$. Let Π_{col} be the projector on the column space of $\mathbf{Y}^{(1)}$, Π_{row} be the projector on the row space of $\mathbf{Y}^{(2)}$, Π_{both} be given in (2.11). Then $\Pi_{\mathrm{both}}(\mathbf{X}) = \mathbf{X}$ if and only if the set of characteristic roots of the series $\mathbb{X}$ consists of the roots μ_j, $j = 1, \ldots, s$, of multiplicities $d_j \leq d_j^{(1)} + d_j^{(2)}$.*

Corollary 2.1 *Let $\mathbb{Y}$ be a series of dimension r, $\mathbf{Y}$ be its trajectory matrix, Π_{row} be the projector on its row trajectory space, Π_{col} be the projector on its column trajectory space. Consider the series $\mathbb{X}$ with $x_n = (an + b)y_n$. Then $\Pi_{\mathrm{both}}(\mathbf{X}) = \mathbf{X}$.*

Remark 2.1 Note that multiplication by $an + b$ means that the multiplicities of the characteristic roots increase by 1.

Since for polynomial trends of a degree k there is the unique characteristic root equal to 1 of multiplicity $k+1$ (Golyandina et al. 2001; Example 5.3) and we should assume only the degree of the polynomial trend to obtain its trajectory space, this case is of particular interest.

Corollary 2.2 *Let Π_{row} be the projection on the row trajectory space of the polynomial of order m, Π_{col} be the projection on the column trajectory space of the polynomial of order k. Then for the polynomial $\mathbb{X} = P_{m+k+1}$ of order $m+k+1$ we have $\Pi_{\text{both}}(\mathbf{X}) = \mathbf{X}$.*

It immediately follows from the projection definition that in the conditions of Corollary 2.2, for any polynomial $\mathbb{X} = P_m$ of degree m we have $\Pi_{\text{row}}(\mathbf{X}) = \mathbf{X}$ and for any polynomial $\mathbb{X} = P_k$ of degree k we have $\Pi_{\text{col}}(\mathbf{X}) = \mathbf{X}$.

2.3.3 Separability

We can expect that if the bases of the spaces to be projected to are chosen properly (for example, if an LRR governing a time series component is known), then SSA with projection improves the resultant decomposition, in comparison with Basic SSA.

Using the notion of separability, we can formulate this improvement as follows. Let $\mathbb{X} = \mathbb{X}^{(1)} + \mathbb{X}^{(2)}$. We will say that a time series component $\mathbb{X}^{(1)}$ is separated by SSA with projection if its trajectory matrix $\mathbf{X}^{(1)}$ coincides with $\mathbf{C}$, where $\mathbf{C}$ is as in (2.12). Therefore, separability by SSA with projection means that the series component $\mathbb{X}^{(1)}$ can be reconstructed by projecting components of the matrix decomposition (2.12) only.

Let $\mathbb{X}^{(1)}$ be a series of finite rank, $\mathbb{X} = \mathbb{X}^{(1)} + \mathbb{X}^{(2)}$. Similar to Golyandina et al. (2001; Sections 1.7.1 and 6.3), where conditions for separability by SSA with centering are considered, the following conditions of separability can be obtained:

1. Basic SSA: $\mathbb{X}^{(1)}$ and $\mathbb{X}^{(2)}$ are separable if (if and only if, by the definition) their row and column spaces are orthogonal.
2. SSA with row projection on the row space of $\mathbb{X}^{(1)}$: $\mathbb{X}^{(1)}$ and $\mathbb{X}^{(2)}$ are separable if their row spaces are orthogonal.
3. SSA with column projection on the column space of $\mathbb{X}^{(1)}$: $\mathbb{X}^{(1)}$ and $\mathbb{X}^{(2)}$ are separable if their column spaces are orthogonal.
4. SSA with double projection on the row and column space of $\mathbb{Y}$, where $\mathbb{X}^{(1)}$ and $\mathbb{Y}$ are such that $x_n^{(1)} = (an+b)y_n$, $a \neq 0$: $\mathbb{X}^{(1)}$ and $\mathbb{X}^{(2)}$ are separable by SSA with double projection if $\mathbb{Y}$ and $\mathbb{X}^{(2)}$ are separable by Basic SSA.

For an approximate separability $\mathbf{X}^{(1)} \approx \mathbf{C}$, we need an approximate orthogonality. Also, an asymptotic separability and the rate of convergence can be considered by analogy with the conventional separability for Basic SSA and SSA with centering.

Recall that the usual double centering in SSA corresponds to a constant series $\mathbb{Y}$ and therefore a linear series $\mathbb{X}^{(1)}$. Orthogonality to a constant series is a much weaker condition than that to a linear series. Therefore, for extraction of a linear trend the use of double centering is recommended.

We can expect that in the case of a polynomial trend, SSA with double centering can work better than SSA with row or column centering and also than Basic SSA.

Also, the separability conditions imply that for extraction of polynomial trends double projection can be used for better separability and therefore the result can be more accurate than the ordinary least-squares polynomial regression provides; see Golyandina et al. (2001; Section 1.7.1) and Golyandina and Shlemov (2017) containing comparison with least-squares regression estimates.

It is important that separability of SSA with projection, if it takes place, is always strong, since the elementary components, which are produced by the projection, precede the SVD components, by construction of the decomposition.

2.3.4 Algorithm

Algorithm 2.4 SSA with projection: decomposition

```
Input: Time series $\mathbb{X}$ of length $N$, window length $L$, orthonormal basis of the column projection
    space $\{P_i, i = 1, \ldots, p\}$ and orthonormal basis of the row projection space $\{Q_i, i =
    1, \ldots, q\}$. Either $p$ or $q$ can be zero.
Output: Decomposition of the trajectory matrix on elementary matrices $\mathbf{X} = \mathbf{X}_1 + \ldots + \mathbf{X}_d$, where
    $\mathbf{X}_i = \sigma_i U_i V_i^{\mathrm{T}}$ are either zero or rank-one matrices.
 1: Construct the trajectory matrix $\mathbf{X} = \mathcal{T}_{\mathrm{SSA}}(\mathbb{X})$.
 2: Subtract the row projection: $\mathbf{X}' = \mathbf{X} - \mathbf{C}_{\mathrm{row}}$, where
        $\mathbf{C}_{\mathrm{row}} = \Pi_{\mathrm{row}}(\mathbf{X}) = \sum_{i=1}^{q} \sigma_i^{(r)} \widetilde{P}_i Q_i^{\mathrm{T}},$
    $\sigma_i^{(r)} = \|\mathbf{X} Q_i\|$, $\widetilde{P}_i = \mathbf{X} Q_i / \sigma_i^{(r)}$ if $\sigma_i^{(r)} > 0$; otherwise, $\widetilde{P}_i$ is the zero vector.
 3: Subtract the column projection: $\mathbf{X}^\star = \mathbf{X}' - \mathbf{C}_{\mathrm{col}}$, where
        $\mathbf{C}_{\mathrm{col}} = \Pi_{\mathrm{col}}(\mathbf{X}') = \sum_{i=1}^{p} \sigma_i^{(c)} P_i \widetilde{Q}_i^{\mathrm{T}},$
    $\sigma_i^{(c)} = \|\mathbf{X}'^{\mathrm{T}} P_i\|$, $\widetilde{Q}_i = \mathbf{X}'^{\mathrm{T}} P_i / \sigma_i^{(c)}$ if $\sigma_i^{(c)} > 0$; otherwise, $\widetilde{Q}_i$ is the zero vector.
 4: Construct an SVD of the matrix $\mathbf{X}^\star$: $\mathbf{X}^\star = \sum_{i=1}^{d^\star} \mathbf{X}_i^\star$, where $\mathbf{X}_i^\star = \sqrt{\lambda_i^\star} U_i^\star (V_i^\star)^{\mathrm{T}}$.
 5: As a result, $\mathbf{X} = \sum_{i=1}^{d} \mathbf{X}_i$, where $d = q + p + d^\star$, $\mathbf{X}_i = \sigma_i^{(r)} \widetilde{P}_i Q_i^{\mathrm{T}}$ for $i = 1, \ldots, q$,
    $\mathbf{X}_{i+q} = \sigma_i^{(c)} P_i \widetilde{Q}_i^{\mathrm{T}}$ for $i = 1, \ldots, p$, and $\mathbf{X}_{i+q+p} = \sqrt{\lambda_i^\star} U_i^\star (V_i^\star)^{\mathrm{T}}$ for $i = 1, \ldots, d^\star$.
```

Algorithm 2.5 SSA with projection: reconstruction

```
Input: Decomposition $\mathbf{X} = \mathbf{X}_1 + \ldots + \mathbf{X}_d$, $\mathbf{X}_i = \sigma_i U_i V_i^{\mathrm{T}}$, number $q$ of row-projection components,
    number $p$ of column-projection components, grouping $\{1, \ldots, d\} = \bigsqcup_{j=1}^{m} I_j$, which does not
    split the first $q + p$ projection components.
Output: Decomposition of the time series 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: Compute $\mathbb{X} = \mathbb{X}_1 + \ldots + \mathbb{X}_m$, where $\mathbb{X}_i = \mathcal{T}_{\mathrm{SSA}}^{-1} \circ \Pi_{\mathcal{H}}(\mathbf{X}_{I_i})$.
```

The only essential difference with the reconstruction by Basic SSA is that the set of the matrices $\mathbf{X}_i$, $i = 1, \ldots, q + p$, produced by projections, should be included to the same group.

2.3.5 *SSA with Projection in* RSSA

2.3.5.1 Description of Functions

In RSSA, Basic SSA with projection is a special case of Basic SSA, hence `ssa` function should be used with additional arguments that would specify column and row projection bases. The meaning of all other arguments is the same as described in Sect. 2.1.5. A typical call is as follows:

```
s <- ssa(x, L = (N + 1) %/% 2, neig = NULL,
         kind = "1d-ssa", svd.method = "auto",
         column.projector = "centering",
         row.projector = "centering")
```

where `N` is the series length.

Arguments:

`column.projector`, `row.projector` Each may be a matrix of orthonormal basis of the projection subspace, or a single integer, which will be interpreted as the dimension of the orthogonal polynomial basis (note that the dimension equals to the degree of the basis plus 1, e.g. the quadratic basis has dimension 3), or one of following character strings: `"none"`, `"constant"` (or `"centering"`), `"linear"`, `"quadratic,"` or `"cubic"` for orthonormal bases of the corresponding polynomial series.

The `ssa` call when both projectors are set to `"none"` corresponds to ordinary Basic SSA, `column.projector = "centering"` (or, the same, `column.projector=1`) corresponds to Basic SSA with centering, `column.projector = "centering"` and `row.projector = "centering"` corresponds to Basic SSA with double centering. The mode `kind = "toeplitz-ssa"` is unavailable for any choice of projections.

Note that the special triples generated by projections are included into the whole set of triples produced by the adaptive decomposition used. In RSSA, all the triples are named eigentriples for uniformity. The first `nspecial(s)` triples correspond to projections. Hence, reconstruction by `groups = list(1:nspecial(n))` corresponds to reconstruction of the projection term. For example, for double centering mode, one obtains a linear trend estimation.

2.3.5.2 Typical Code

Let us consider the example "CO2" (Mauna Loa Atmospheric CO_2 Concentration). It seems that for such kind of time series, many methods can yield very similar results. Basic SSA provides very natural way for trend extraction, it is demonstrated in Golyandina and Korobeynikov (2013), where the choice $L = 120$, $ET1, 4$ was considered.

Fragment 2.3.1 demonstrates the code, which allows to perform reconstruction by means of SSA with projection.

Fragment 2.3.1 ("CO2": SSA with Projection)

```
> s2 <- ssa(co2, column.projector = "centering",
+          row.projector = "centering")
> plot(reconstruct(s2, groups =
+                  list(Linear.trend = seq_len(nspecial(s2)))),
+      add.residuals = FALSE, plot.method = "matplot")
> s4 <- ssa(co2, column.projector = 2, row.projector = 2)
> plot(reconstruct(s4, groups =
+                  list(Trend = seq_len(nspecial(s4)))),
+      add.residuals = FALSE, plot.method = "matplot")
> plot(s4, type = "vectors", idx = 1:12)
> r <- reconstruct(s4,
+                 groups =
+                   list(Signal = c(seq_len(nspecial(s4)), 5:8)))
> plot(r, plot.method = "xyplot")
```

We start with extraction of linear trend and therefore choose `column.projector = "centering"`, `row.projector = "centering"` to perform SSA with double centering. Note that the same choice of projectors can be achieved by setting `column.projector` and `row.projector` equal to 1, where 1 is the dimension of the trajectory space of a polynomial series of degree 0; that is, of a constant series. Recall that the choice `column.projector = p`, `row.projector = q` corresponds to extraction of a polynomial trend of degree $p + q - 1$. To select all the projection components in the decomposition `s2` for extraction, we set the trend group consisting of the first `nspecial(s2)` components. The extracted trend is close to linear, see Fig. 2.9. Certainly, the accurate trend of "CO2" series is not linear.

To extract a more accurate trend, let us choose other subspaces for projections, `column.projector = 2` and `row.projector = 2`, to extract a trend, which is close to a cubic polynomial. Figure 2.10 shows that the extracted trend is quite accurate. This trend is very similar to that in Golyandina and Korobeynikov (2013), which was extracted by Basic SSA. Note that in this example the trend

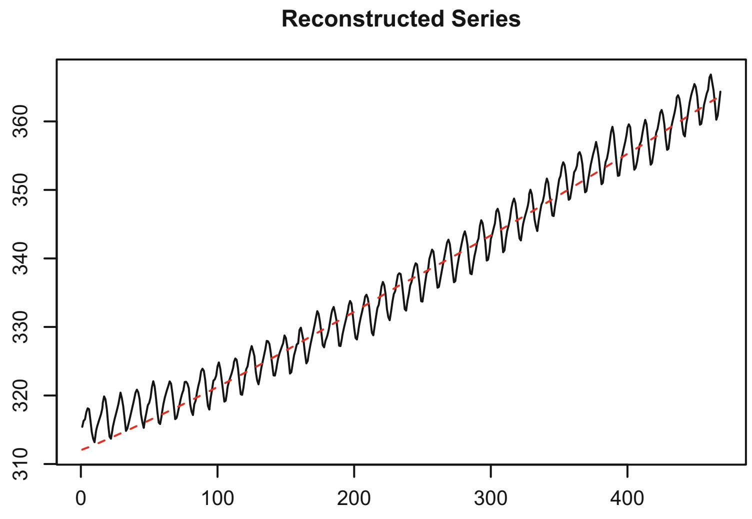


Fig. 2.9 "CO2": Reconstruction of linear trend

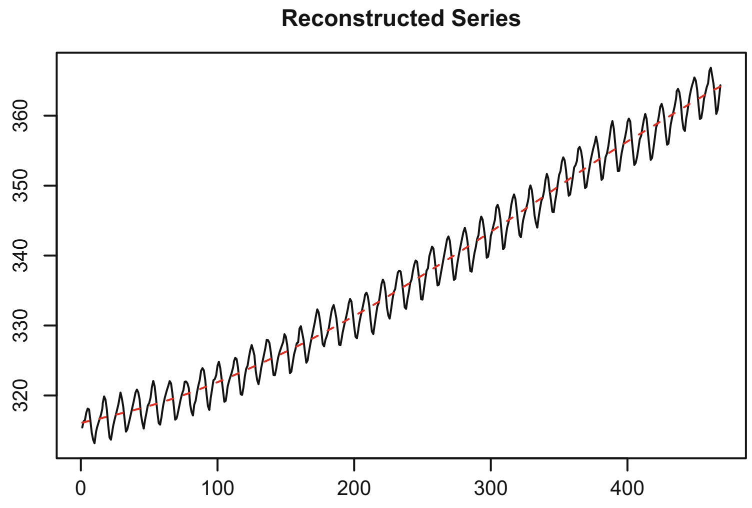


Fig. 2.10 "CO2": Reconstruction of the cubic trend

is approximated by a sum of two exponentials in Basic SSA (ET1,4 for $L = 120$) and a polynomial of order 3 in SSA with projection; both approximations have four parameters and achieve similar accuracy.

Note that SSA with projection allows us to extract not only a trend but also other kinds of series components, similar to what Basic SSA does. Figure 2.11 presents graphs of eigenvectors for the "CO2" example. The first two components contain two vectors produced by projecting rows on the row projection space and the next

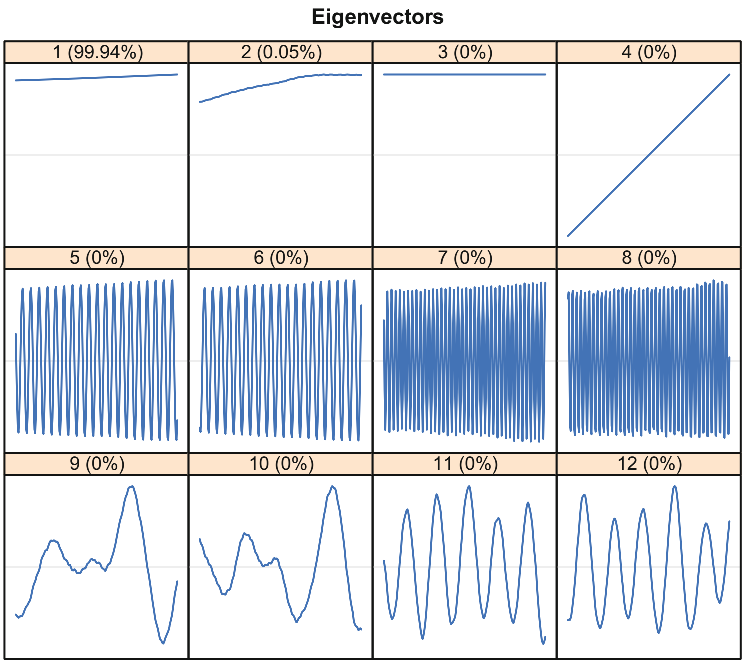


Fig. 2.11 “CO2”: 1D graphs of eigenvectors

two components contain a basis of the column space (two linear functions here). Other graphs show singular vectors of the matrix $\mathbf{X}''$ obtained by subtraction of the projection matrices. Choice of the trend components ET1–4 and the seasonality components ET5–8 leads to the signal extraction depicted in Fig. 2.12.

2.3.5.3 Simulated Examples: Polynomial Regression

Here we consider an example showing the difference between SSA with projection and the least-squares parametric regression for polynomial trend extraction. Let us take a polynomial trend $t_n = 10(n/N - 0.5)^5$ of order 5, $x_n = t_n + \sin(2\pi n/10)$, where $N = 199$ and $n = 1, \ldots, N$.

Projections, which keep a polynomial of degree 5, can be composed in different ways. It can be purely either column or row projection on the 6-dimensional polynomial trajectory space. Also, a double projection can be considered. For example, we can take both row and column projections on the row and column trajectory spaces of a polynomial of degree 2 (and of dimension 3). By Corollary 2.2, this double projection with $k = m = 2$ keeps polynomial of degree 5.

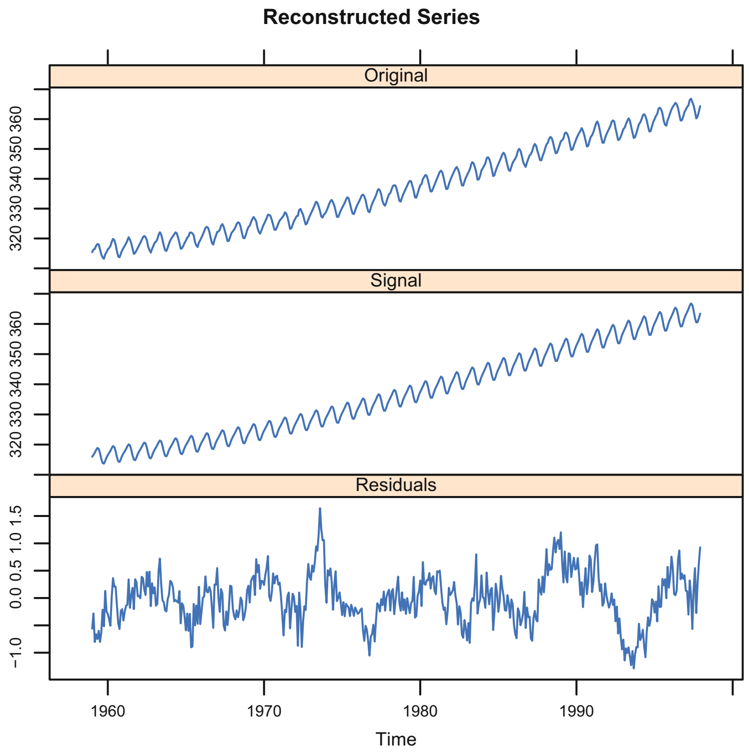


Fig. 2.12 “CO2”: Reconstruction of signal

SSA with projection for neither choice provides approximate separability of the polynomial trend from a sinusoid. However, in view of the separability conditions we can expect that the choice $k = m = 2$ probably corresponds to the best accuracy.

Fragment 2.3.2 (Polynomial Trend: SSA with Projection)

```
> N <- 199
> tt <- (1:N) / N
> r <- 5
> F0 <- 10 * (tt - 0.5)^r
> F <- F0 + sin(2 * pi * (1:N) / 10)
> L <- 100
> dec <- ssa(F, L = L, column.projector = 3, row.projector = 3)
> rec1 <- reconstruct(dec, groups =
+                        list(Trend = seq_len(nspecial(dec))))
> fit1 <- rec1$Trend
> fit1_3b <- lm(fit1 ~ poly((1:N), r, raw = TRUE))
> fit3b <- lm(F ~ poly((1:N), r, raw = TRUE))
> li <- 1:199
> d <- data.frame(Initial = F[li],
+                 dproj = fit1[li],
+                     dproj_reg = predict(fit1_3b)[li],
+                     regr = predict(fit3b)[li], trend = F0[li])
> xyplot(as.formula(paste(paste(colnames(d), collapse = "+"),
+                               "~", "1:nrow(d)")),
+          data = d,
+          type = "l", ylab = "", xlab = "",
+          lty = c(1, 1, 1, 1, 1), lwd = c(1, 2, 2, 2, 2),
+          auto.key = list(columns = 3,
+                              lines = TRUE, points = FALSE))
```

We compare the following trend estimations (see Fragment 2.3.2). First, we set $L = 100$ and consider the trend obtained by double projection with $k = m = 2$ (this is called “dproj”). Then, we find the least-squares polynomial regression of order 5 for the initial series (“regr”) and for “dproj” (“dproj_regr”).

If L and K are divisible by the period, then the separability accuracy is better and the result is in a sense unbiased. Least-squares polynomial regression of order 5 does not estimate the polynomial trend in the meaning considered in this example as it minimizes the prediction mean square error, by the definition.

The results are presented in Fig. 2.13. SSA with double projection extracts the trend approximately with visible mixture with the sine-wave component. However, these oscillations are around the proper trend. Least-squares polynomial regression of order 5 applied to the result of double projection confirms it.

Least-squares parametric regression provides a poor estimator of trend in the considered example. For longer time series the difference is not so dramatic and the trend estimates are closer.

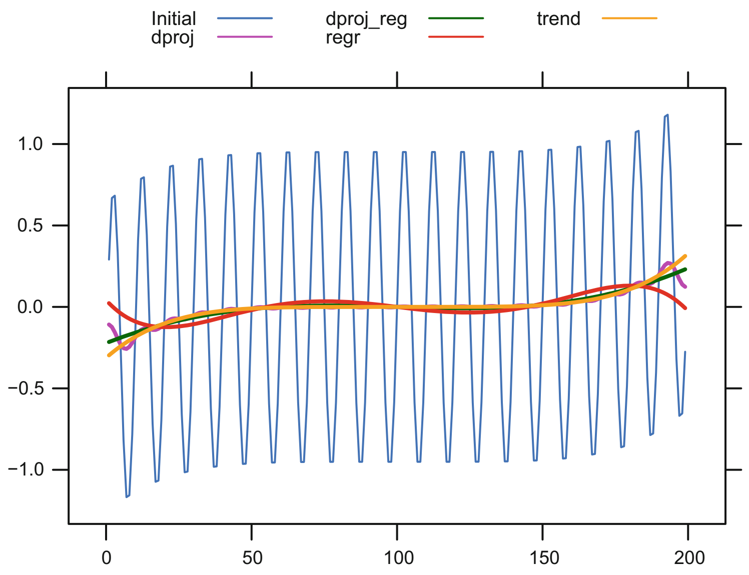


Fig. 2.13 Polynomial trend: Comparison of trend reconstructions

2.4 Iterative Oblique SSA

For reasonably long time series lengths and moderate noise levels, interpretable components such as trends, oscillations, and noise are approximately separable by Basic SSA (Golyandina et al. 2001; Sections 1.5 and 6.1). However, the conditions of approximate separability can be restrictive, especially, for short time series.

Orthogonality of subseries, which is the main condition for separability in Basic SSA, see Sect. 2.1.3, can be a strong limitation on the series which we want to separate. However, if we consider orthogonality with respect to a non-standard Euclidean inner product, conditions of separability are considerably weaker. This approach yields the method called Oblique SSA (O-SSA) with the SVD performed in a non-orthogonal coordinate system at Decomposition step. The idea of Oblique SSA is similar to that of prewhitening which is frequently used in statistics as preprocessing: if we know covariances between components, then we can perform linear transformation and obtain uncorrelated components. Since the "covariances" of the components are not known in advance, an iterative method called Iterative Oblique SSA can be used. Also, the method is able to change contributions of the components in a specific way so that their strong separability will most likely to be improved.

2.4.1 Method

2.4.1.1 Use of Oblique SVD

Although many interpretable series components like trend (a slowly varying component) and seasonality are asymptotically orthogonal, for a given time series length the orthogonality can be unreachable even approximately. Therefore, it would be helpful to weaken the orthogonality condition. Oblique SSA uses different orthogonality, which still means the equality of an inner product to 0, but this time a non-standard inner product is used; this inner product is adapted to the time series components, which we want to separate.

It is well-known that any inner product in the Euclidean space is associated with a symmetric positive-definite matrix $\mathbf{A}$ and is defined as $\langle X_1, X_2\rangle_\mathbf{A} = (\mathbf{A}X_1, X_2)$. The standard inner product corresponds to the use of the identity matrix as $\mathbf{A}$. The notion of inner product implies the notion of $\mathbf{A}$-orthogonality: two vectors X_1 and X_2 are $\mathbf{A}$-orthogonal if $\langle X_1, X_2\rangle_\mathbf{A} = 0$. If the matrix $\mathbf{A}$ is semi-definite, then it defines the inner product in its column space (also, in the row space which is the same in view of symmetry). While considering $\langle X_1, X_2\rangle_\mathbf{A}$, we will always assume that the vectors X_1 and X_2 belong to the column space of $\mathbf{A}$.

The non-standard Euclidean inner products induce such notions as oblique coordinate systems, orthogonality of vectors, which are oblique in the ordinary sense, and so on.

Let us consider an elementary example. Let $X = (1, 2)^{\mathrm{T}}$ and $Y = (1, 1)^{\mathrm{T}}$. These two vectors are not orthogonal in the usual sense as $(X, Y) = 3$. However, if we define

$$\mathbf{A} = \begin{pmatrix} 5 & -3 \\ -3 & 2 \end{pmatrix}, \tag{2.13}$$

then $\langle X, Y\rangle_{\mathbf{A}} = (\mathbf{A}X, Y) = 0$ and $(\mathbf{O}_{\mathbf{A}}X, \mathbf{O}_{\mathbf{A}}Y) = 0$ for any $\mathbf{O}_{\mathbf{A}}$ such that $\mathbf{O}_{\mathbf{A}}^{\mathrm{T}}\mathbf{O}_{\mathbf{A}} = \mathbf{A}$, e.g.,

$$\mathbf{O}_{\mathbf{A}} = \begin{pmatrix} 1 & -1 \\ -2 & 1 \end{pmatrix}.$$

This means that $\{X, Y\}$ is an orthogonal basis with respect to the $\mathbf{A}$-inner product $\langle\cdot, \cdot\rangle_{\mathbf{A}}$ and the matrix $\mathbf{O}_{\mathbf{A}}$ defines the orthogonalizing map. The matrix $\mathbf{A}$ can be chosen in such a way that X and Y have any given $\mathbf{A}$-norms. The choice (2.13) corresponds to $\mathbf{A}$-orthonormality.

To describe Oblique SSA, let us introduce the SVD of a matrix $\mathbf{X}$ produced by two oblique bases, $\mathbf{L}$-orthonormal and $\mathbf{R}$-orthonormal correspondingly, in the row and column spaces.

Definition 2.5 We say that

$$\mathbf{X} = \sum_{i=1}^{d} \sigma_i P_i Q_i^{\mathrm{T}} \tag{2.14}$$

is the $(\mathbf{L}, \mathbf{R})$-SVD, if $\{P_i\}_{i=1}^d$ is an $\mathbf{L}$-orthonormal system and $\{Q_i\}_{i=1}^d$ is an $\mathbf{R}$-orthonormal system; that is, the decomposition is $(\mathbf{L}, \mathbf{R})$-biorthogonal.

This kind of SVD is called Restricted SVD (RSVD) given by the triple $(\mathbf{X}, \mathbf{L}, \mathbf{R})$, see De Moor and Golub (1991) for details. The mathematics related to inner products $\langle\cdot, \cdot\rangle_{\mathbf{A}}$ with positive-semidefinite matrix $\mathbf{A}$ and the corresponding RSVD is shortly described in Golyandina and Shlemov (2015; Appendix A) from the viewpoint of decompositions into a sum of elementary matrices.

Oblique SSA (O-SSA) is a modification of the Basic SSA method described in Sect. 2.1, where the standard SVD at Decomposition step is replaced by the $(\mathbf{L}, \mathbf{R})$-SVD for some matrices $\mathbf{L}$ and $\mathbf{R}$. We will use all the notions related to Basic SSA for this oblique modification.

If $\mathbf{L}$ and $\mathbf{R}$ are the identity matrices, then Oblique SSA coincides with Basic SSA, $\sigma_i = \sqrt{\lambda_i}$, $P_i = U_i$, and $Q_i = V_i$.

Computationally, oblique SVD is straightforwardly reduced to the ordinary SVD (see Golyandina and Shlemov (2015; Proposition 4)) and therefore its calculation does not require special numerical techniques, see Algorithm 2.6.

2.4.1.2 Nested Oblique SSA

Unlike the ordinary SVD, the SVD with respect to a non-orthogonal coordinate system provides a matrix approximation which does not have obvious approximation properties. This implies that Oblique SSA is not a good tool for extraction of the leading components, in particular, for extraction of the signal and for denoising.

Therefore, we suggest to use Oblique SSA in the nested way. The approach is somewhat similar to factor analysis, where a factor space can be estimated by principal component analysis and then interpretable factors are extracted from the factor space.

Suppose that in a particular application Basic SSA is able to extract the signal but is not able to separate the signal components. For example, let the time series consist of a noisy sum of two sinusoids with close frequencies. Then Basic SSA can perform the denoising but it is unlikely that it will be able to separate these sinusoids. Hence Basic SSA should only be used for estimation of the subspace of the sum of sinusoids and then some other method is advised to be employed for separating the sinusoids. Note that the nested approach is similar to the refined grouping used in Golyandina and Zhigljavsky (2013; Section 2.5.4) for the SSA-ICA and Golyandina and Lomtev (2016) for the SSA-AMUSE algorithms, which use ideas taken from independent component analysis.

Thus, let us apply Basic SSA with proper parameters and let a matrix decomposition $\mathbf{X} = \mathbf{X}_{I_1} + \ldots + \mathbf{X}_{I_p}$ be obtained at Grouping step of Basic SSA; each group corresponds to a separated time series component. Let the sth group $I = I_s$ be chosen for a refined decomposition. Denote $\mathbf{Y} = \mathbf{X}_I$, $r = \operatorname{rank} \mathbf{Y}$, $\mathbb{Y} = \mathcal{T}^{-1} \circ \Pi_{\mathcal{H}}(\mathbf{Y})$ the series obtained from $\mathbf{Y}$ by diagonal averaging.

Let us describe the scheme of Nested Oblique SSA. The aim of the nested scheme is obtaining a refined decomposition of $\mathbf{Y} = \mathbf{X}_I$ in the matrix form $\mathbf{Y} = \mathbf{Y}^{(1)} + \ldots + \mathbf{Y}^{(l)}$, using the ($\mathbf{L}$, $\mathbf{R}$)-SVD and therefore getting a decomposition of the corresponding time series $\mathbb{Y} = \widetilde{\mathbb{Y}}^{(1)} + \ldots + \widetilde{\mathbb{Y}}^{(l)}$.

For correctness of the scheme, we should assume that the matrices $\mathbf{L}$ and $\mathbf{R}$ are consistent with $\mathbf{Y}$; that is, the column space of $\mathbf{Y}$ is a subset of the column space of $\mathbf{L}$ and the row space of $\mathbf{Y}$ is a subset of the column space of $\mathbf{R}$.

Nested O-SSA is very similar to Basic SSA; the difference is that there is no Embedding step and that the matrix $\mathbf{X}_I$, which is not necessarily a Hankel matrix, is used instead of the conventional trajectory matrix.

At Decomposition step, we construct the ($\mathbf{L}$, $\mathbf{R}$)-SVD of $\mathbf{Y}$ in the form

$$\mathbf{Y} = \sum_{i=1}^{r} \sigma_i P_i Q_i^{\mathrm{T}}, \tag{2.15}$$

see Algorithm 2.6.

The rest of the method coincides with Reconstruction stage of Basic SSA. After Grouping step, we obtain the decomposition $\mathbf{Y} = \mathbf{Y}_{J_1} + \ldots + \mathbf{Y}_{J_l}$ and then, as a result, the refined series decomposition $\mathbb{Y} = \widetilde{\mathbb{Y}}^{(1)} + \ldots + \widetilde{\mathbb{Y}}^{(l)}$, where $\widetilde{\mathbb{Y}}^{(m)} = \mathfrak{T}^{-1} \circ \Pi_{\mathcal{H}}(\mathbf{Y}_{J_m})$.

Therefore, after application of Nested O-SSA to the group I_s, we obtain the following decomposition of the original series $\mathbb{X}$:

$$\mathbb{X} = \widetilde{\mathbb{X}}^{(1)} + \ldots + \widetilde{\mathbb{X}}^{(p)}, \text{ where } \widetilde{\mathbb{X}}^{(s)} = \widetilde{\mathbb{Y}}^{(1)} + \ldots + \widetilde{\mathbb{Y}}^{(l)}.$$

For simplicity, below we will consider the case $l = 2$.

2.4.1.3 Iterative Approach to O-SSA

Let us describe an iterative version of Nested O-SSA; that is, an iterative algorithm for obtaining appropriate matrices $\mathbf{L}$ and $\mathbf{R}$ for the $(\mathbf{L}, \mathbf{R})$-SVD of $\mathbf{X}_I$. For proper use of nested decompositions, we should expect that the matrix $\mathbf{X}_I$ is close to a rank-deficient trajectory matrix of rank r.

To explain the main principle of the method, assume that $\mathbf{X}_I = \mathbf{Y}$ is the trajectory matrix of $\mathbb{Y}$. Let $\mathbb{Y} = \mathbb{Y}^{(1)} + \mathbb{Y}^{(2)}$ and the trajectory matrices $\mathbf{Y}_1$ and $\mathbf{Y}_2$ be of ranks r_1 and r_2, $r_1 + r_2 = r$. Then by Golyandina and Shlemov (2015; Theorem 1) there exist r-rank separating matrices $\mathbf{L}^*$, $\mathbf{R}^*$ of sizes $L \times L$ and $K \times K$ correspondingly and a partition $\{1, \ldots, r\} = J_1 \sqcup J_2$ such that we can perform the proper grouping in the $(\mathbf{L}^*, \mathbf{R}^*)$-SVD and thereby obtain $\mathbf{Y}_{J_1} = \mathbf{Y}_1$ and $\mathbf{Y}_{J_2} = \mathbf{Y}_2$.

The separating matrices $\mathbf{L}^*$ and $\mathbf{R}^*$ are unknown as they are determined by unknown trajectory spaces of $\mathbb{Y}^{(1)}$ and $\mathbb{Y}^{(2)}$. Therefore, we want to construct a sequence of $(\mathbf{L}, \mathbf{R})$-SVD decompositions (2.14), which in the limit gives the required separating decomposition.

Let us have an initial $(\mathbf{L}^{(0)}, \mathbf{R}^{(0)})$-SVD decomposition of $\mathbf{Y}$ and group its components to obtain some initial estimates $\widetilde{\mathbb{Y}}^{(1,0)}$ and $\widetilde{\mathbb{Y}}^{(2,0)}$ of $\mathbb{Y}^{(1)}$ and $\mathbb{Y}^{(2)}$. Then we can use the trajectory spaces of $\widetilde{\mathbb{Y}}^{(1,0)}$ and $\widetilde{\mathbb{Y}}^{(2,0)}$ to construct the new inner product which is expected to be closer to the separating one. Therefore, we can expect that $\widetilde{\mathbb{Y}}^{(1,1)}$ and $\widetilde{\mathbb{Y}}^{(2,1)}$ will be closer to $\mathbb{Y}^{(1)}$ and $\mathbb{Y}^{(2)}$ and therefore we take their trajectory spaces to construct a new inner product; and so on. Of course, if the initial decomposition is strongly separating, then we obtain $\widetilde{\mathbb{Y}}^{(m,1)} = \widetilde{\mathbb{Y}}^{(m,0)} = \mathbb{Y}^{(m)}$, $m = 1, 2$.

2.4.1.4 Basic Iterative Algorithm

We call the iterative version of Nested Oblique SSA *Iterative Oblique SSA* or *Iterative O-SSA*.

As before, we consider a nested O-SSA whose input is the matrix $\mathbf{Y} = \mathbf{X}_I$ of rank r. For Basic SSA and for nested O-SSA, a partition of eigentriple numbers for grouping is made after Decompositions stage. For Iterative O-SSA, a partition $I = \widetilde{J}_1 \sqcup \widetilde{J}_2$, $r_m = |\widetilde{J}_m|$, should be specified in advance, since iterations are performed in an automatic mode. Certainly, the choice of partition is not made in dark since before the use of a nested version, we have a full decomposition which we use for choosing the group I and its partition.

Iterative O-SSA is made of repeated application of nested O-SSA with recalculation of $(\mathbf{L}^{(k)}, \mathbf{R}^{(k)})$. As any iterative algorithm, Iterative O-SSA should have initial data $(\mathbf{L}^{(0)}, \mathbf{R}^{(0)})$ and a stopping rule. Standard stopping rule includes the maximum number of iterations M and the precision threshold ε. In Iterative O-SSA, the algorithm stops if the reconstructed series components $\widetilde{\mathbb{Y}}^{(m,k)}$, $m = 1, 2$, change very little. For a function $\rho(\cdot)$ defining a vector norm, the iterations stop under the condition $\max\big(\rho(\widetilde{\mathbb{Y}}^{(m,k)} - \widetilde{\mathbb{Y}}^{(m,k-1)}), m = 1, 2\big) < \varepsilon$.

Note that since the consistence of $(\mathbf{L}, \mathbf{R})$ with $\mathbf{Y}$ is needed for a correct $(\mathbf{L}, \mathbf{R})$-SVD, the initial data $(\mathbf{L}^{(0)}, \mathbf{R}^{(0)})$ should also be consistent.

Remark 2.2 The initial matrices $(\mathbf{L}^{(0)}, \mathbf{R}^{(0)})$ together with grouping can be specified so that the initial decomposition is a part of the SVD (2.3) given by the set of indices I and $I = \widetilde{J}_1 \sqcup \widetilde{J}_2$, where $\widetilde{J}_1$ and $\widetilde{J}_2$ are chosen on the base of analysis of the components in (2.3). Since (2.3) is biorthogonal, $\mathbf{L}^{(0)}$ and $\mathbf{R}^{(0)}$ are the identity matrices. It is convenient to denote by J_1 and J_2 the sets consisting of ordinal indices of the elements of $\widetilde{J}_1$ and $\widetilde{J}_2$ in I. Thereby, $\{1, \ldots, r\} = J_1 \sqcup J_2$. For example, if $\widetilde{J}_1 = \{11, 14\}$ and $\widetilde{J}_2 = \{12, 18\}$, then $I = \{11, 12, 14, 18\}$, $J_1 = \{1, 3\}$ and $J_1 = \{2, 4\}$.

To finalize the Iterative O-SSA method, we present a formal description of iterations. The separating decomposition $\mathbf{Y} = \mathbf{Y}_1 + \mathbf{Y}_2$ should satisfy the following properties:

(a) $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are Hankel;
(b) $\operatorname{rank} \mathbf{Y}_1 = r_1$, $\operatorname{rank} \mathbf{Y}_2 = r_2$;
(c) the column and row spaces of $\mathbf{Y}_1$ and $\mathbf{Y}_2$ lie in the column and row spaces of $\mathbf{Y}$;
(d) $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are $(\mathbf{L}, \mathbf{R})$-biorthogonal for $\mathbf{L} = \mathbf{L}^*$ and $\mathbf{R} = \mathbf{R}^*$.

Define Π_{col} the orthogonal projection operator (for the Euclidean norm) on the column space of $\mathbf{Y}$, Π_{row} the projection operator on the row space of $\mathbf{Y}$. The nested group is the ordered union $I = \widetilde{J}_1 \sqcup \widetilde{J}_2$, $r_m = |\widetilde{J}_m|$, J_1 and J_2 are defined in Remark 2.2; the pair of matrices $(\mathbf{L}^{(k-1)}, \mathbf{R}^{(k-1)})$ is the input for the kth iteration.

Let us formulate the kth iteration steps.

(A) To obtain Hankel matrices, we perform hankelization of the input decomposition $\widetilde{\mathbf{Y}}_m = \Pi_{\mathcal{H}} \mathbf{Y}_{J_m}^{(k-1)}$, $m = 1, 2$.
(B) Then, to obtain a low-rank approximation of ranks r_1 and r_2 correspondingly, we construct the ordinary SVDs $\widetilde{\mathbf{Y}}_m = \sum_{i=1}^{d_m} \sqrt{\lambda_i^{(m)}} U_i^{(m)} (V_i^{(m)})^{\mathrm{T}}$, $m = 1, 2$, and take the leading r_m terms.
(C) Since we should not fall outside the column space of the input matrix $\mathbf{Y}$ (we consider a nested decomposition), we find the projections $\widehat{U}_i^{(m)} = \Pi_{\mathrm{col}} U_i^{(m)}$ and $\widehat{V}_i^{(m)} = \Pi_{\mathrm{row}} V_i^{(m)}$ for $i = 1, \ldots, r_m$, $m = 1, 2$. Denote

$$\widehat{\mathbf{U}}^{(m)} = [\widehat{U}_1^{(m)} : \ldots : \widehat{U}_{r_m}^{(m)}],\ \widehat{\mathbf{V}}^{(m)} = [\widehat{V}_1^{(m)} : \ldots : \widehat{V}_{r_m}^{(m)}].$$

For the algorithm correctness, we assume that the matrices $\widehat{\mathbf{U}}^{(m)}$ and $\widehat{\mathbf{V}}^{(m)}$ are of full rank; otherwise, the algorithm may not work.

(D) Finally, calculate $\mathbf{L}^{(k)} = (\widehat{\mathbf{U}}^{\dagger})^{\mathrm{T}}\widehat{\mathbf{U}}^{\dagger}$ and $\mathbf{R}^{(k)} = (\widehat{\mathbf{V}}^{\dagger})^{\mathrm{T}}\widehat{\mathbf{V}}^{\dagger}$, where $\widehat{\mathbf{U}} = [\widehat{\mathbf{U}}^{(1)} : \widehat{\mathbf{U}}^{(2)}]$ and $\widehat{\mathbf{V}} = [\widehat{\mathbf{V}}^{(1)} : \widehat{\mathbf{V}}^{(2)}]$, to achieve the $(\mathbf{L}^{(k)}, \mathbf{R}^{(k)})$-biorthogonality.

The convergence of $\widetilde{\mathbb{Y}}^{(1,k)}$ and $\widetilde{\mathbb{Y}}^{(2,k)}$ to a proper decomposition is not proved theoretically. However, looking at the construction scheme, which resembles the alternating projections, we do expect this convergence, at least if the case chosen is not too unusual. Numerical experiments confirm the convergence in the majority of examples. Note also that Iterative O-SSA does not change the separating decomposition; that is, the separating decomposition is a fixed point of the algorithm.

2.4.1.5 Modification with Sigma-Correction

If the initial point $(\mathbf{L}^{(0)}, \mathbf{R}^{(0)})$ for iterations is not far from the separating pair $(\mathbf{L}^{*}, \mathbf{R}^{*})$, then we can expect that the convergence in the algorithm above will take place, since we are close to the fixed-point value and we can expect that $\sigma_i^{(k)}$ in the $(\mathbf{L}^{(k)}, \mathbf{R}^{(k)})$-SVDs $\mathbf{Y} = \sum_{i=1}^{r} \sigma_i^{(k)} P_i^{(k)} (Q_i^{(k)})^{\mathrm{T}}$ are just slightly changed during iterations. In general, however, a possible reordering of the decomposition components between iterations of Iterative O-SSA can interfere with convergence. The case of $J_1 = \{1, \ldots, r_1\}$, when the minimal singular value σ_{r_1} of the first series is kept significantly larger than the maximal singular value σ_{r_1+1} of the second series, would prevent the component reordering and hence improve the convergence.

Let us describe a modification of Iterative O-SSA that provides reordering of the components, moves them apart and thereby relaxes the problem of component mixing. In this modification, an adjustment is made for calculation of $\widehat{\mathbf{U}}^{(2)}$ and $\widehat{\mathbf{V}}^{(2)}$ at Step (C) of iterations.

Let us choose a parameter $\varkappa > 1$. If $\lambda_{r_1}^{(1)} < \varkappa^2 \lambda_1^{(2)}$ at Step (C), then define $\mu = \varkappa\sqrt{\lambda_1^{(2)}/\lambda_{r_1}^{(1)}}$ and change $\widehat{\mathbf{U}}^{(2)} \leftarrow \sqrt{\mu}\widehat{\mathbf{U}}^{(2)}$, $\widehat{\mathbf{V}}^{(2)} \leftarrow \sqrt{\mu}\widehat{\mathbf{V}}^{(2)}$. To be consistent with the reordering, set $J_1 = \{1, \ldots, r_1\}$, $J_2 = \{r_1 + 1, \ldots, r\}$.

Note that if $\lambda_{r_1}^{(1)} < \varkappa^2 \lambda_1^{(2)}$, then the adjustment above makes a change in the order of the matrix components in (2.18), since they are ordered by $\sigma_i^{(k)}$. Hence we force an increase of the matrix components related to the first series component. For explanation of how this sigma-correction works, see Golyandina and Shlemov (2015; Proposition 5).

Remark 2.3 The reordering procedure is made by sequential adjustment of the component weights and therefore depends on the component enumeration.

Summarizing, the described correction can help to improve convergence and to provide strong separability of components in the case when only weak separability takes place.

2.4.2 Separability

The notion of weak and strong $(\mathbf{L}, \mathbf{R})$-separability, which is similar to the conventional separability described in Sect. 2.1.3, can be introduced. Let $\mathbb{X} = \mathbb{X}^{(1)} + \mathbb{X}^{(2)}$, $\mathbf{X}$ be the trajectory matrix of $\mathbb{X}$, $\mathbf{X}^{(m)}$ be the trajectory matrices of the series components, $\mathbf{X}^{(m)} = \sum_{i=1}^{r_m} \sigma_{m,i} P_{m,i} Q_{m,i}^{\mathrm{T}}$ be their $(\mathbf{L}, \mathbf{R})$-SVDs, $m = 1, 2$. We assume that $\mathbf{L}$ and $\mathbf{R}$ are consistent with $\mathbf{X}$, $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$.

Definition 2.6 Let L be fixed. Two series $\mathbb{X}_N^{(1)}$ and $\mathbb{X}_N^{(2)}$ are called *weakly* $(\mathbf{L}, \mathbf{R})$-*separable*, if their column trajectory spaces are $\mathbf{L}$-orthogonal and their row trajectory spaces are $\mathbf{R}$-orthogonal; that is, $(\mathbf{X}^{(1)})^{\mathrm{T}}\mathbf{L}\mathbf{X}^{(2)} = \mathbf{0}_{K,K}$ and $\mathbf{X}^{(1)}\mathbf{R}(\mathbf{X}^{(2)})^{\mathrm{T}} = \mathbf{0}_{L,L}$.

Definition 2.7 Two series $\mathbb{X}_N^{(1)}$ and $\mathbb{X}_N^{(2)}$ are called *strongly* $(\mathbf{L}, \mathbf{R})$-*separable*, if they are weakly $(\mathbf{L}, \mathbf{R})$-separable and $\sigma_{1,i} \neq \sigma_{2,j}$ for any i and j.

The $(\mathbf{L}, \mathbf{R})$-separability of two series components means $\mathbf{L}$-orthogonality of their subseries of length L and $\mathbf{R}$-orthogonality of the subseries of length $K = N - L + 1$. For suitably chosen L and R, the $(\mathbf{L}, \mathbf{R})$-separability is much less restrictive than the ordinary one. Indeed, Theorem 1 from Golyandina and Shlemov (2015) states that for series $\mathbb{X} = \mathbb{X}^{(1)} + \mathbb{X}^{(2)}$ of rank r, where $\mathbb{X}^{(m)}$ is the series of rank r_m, $m = 1, 2$, and $r_1 + r_2 = r$, there exist separating matrices $\mathbf{L} \in \mathsf{R}^{L\times L}$ and $\mathbf{R} \in \mathsf{R}^{K\times K}$ of rank r such that the series $\mathbb{X}^{(1)}$ and $\mathbb{X}^{(2)}$ are strongly $(\mathbf{L}, \mathbf{R})$-separable. Moreover, the separating matrices $\mathbf{L}$ and $\mathbf{R}$ can be explicitly written down.

Denote by $\{P_i^{(m)}\}_{i=1}^{r_m}$ a basis of the column space of $\mathbf{X}^{(m)}$ and by $\{Q_i^{(m)}\}_{i=1}^{r_m}$ a basis of the row space of $\mathbf{X}^{(m)}$, $m = 1, 2$; e.g., $P_i^{(m)} = P_{m,i} \in \mathsf{R}^L$, $Q_i^{(m)} = Q_{m,i} \in \mathsf{R}^K$. Define

$$\mathbf{P} = [P_1^{(1)} : \ldots : P_{r_1}^{(1)} : P_1^{(2)} : \ldots : P_{r_2}^{(2)}],$$
$$\mathbf{Q} = [Q_1^{(1)} : \ldots : Q_{r_1}^{(1)} : Q_1^{(2)} : \ldots : Q_{r_2}^{(2)}].$$

Then the separating matrices have the form $\mathbf{L} = (\mathbf{P}^{\dagger})^{\mathrm{T}}\mathbf{P}^{\dagger}$ and $\mathbf{R} = (\mathbf{Q}^{\dagger})^{\mathrm{T}}\mathbf{Q}^{\dagger}$ (compare with Step (D) of the algorithm scheme in Sect. 2.4.1.4).

The condition $r_1 + r_2 = r$ can be expressed in terms of the characteristic roots. This condition is satisfied if the sets of the characteristic roots of the series are disjoint.

Thus, any two times series governed by LRRs with different characteristic roots can be separated by some $(\mathbf{L}, \mathbf{R})$-SVD for sufficiently large series and window lengths. This statement is not constructive, since the trajectory spaces of the separated series should be known for exact separation. However, we can try to estimate these spaces and therefore improve the separability.

We have already explained how to achieve weak separability. Proposition 5 from Golyandina and Shlemov (2015) shows how to correct the decomposition to get strong separability. Denote by I the set of decomposition components corresponding to $\mathbb{X}^{(1)}$ in a separating $(\mathbf{L}, \mathbf{R})$-SVD

$$\mathbf{Y} = \sum_i \sigma_i P_i Q_i^{\mathrm{T}} = \sum_{i \in I} \sigma_i P_i Q_i^{\mathrm{T}} + \sum_{i \notin I} \sigma_i P_i Q_i^{\mathrm{T}}. \tag{2.16}$$

If in the group I there is a σ_i, which coincides with a σ_j from the residual group, then the SVD decomposition is not unique and therefore the calculated SVD can differ from the separating SVD (2.16). This situation can be easily avoided as follows: let us take $\widetilde{P}_i = \mu_i P_i$ and $\widetilde{Q}_i = \nu_i Q_i$ for some μ_i and ν_i, then the $(\widetilde{\mathbf{L}}, \widetilde{\mathbf{R}})$-SVD

$$\mathbf{Y} = \sum_i \widetilde{\sigma}_i \widetilde{P}_i \widetilde{Q}_i^{\mathrm{T}}$$

for $\widetilde{\mathbf{L}} = (\widetilde{\mathbf{P}}^{\dagger})^{\mathrm{T}} \widetilde{\mathbf{P}}^{\dagger}$ and $\widetilde{\mathbf{R}} = (\widetilde{\mathbf{Q}}^{\dagger})^{\mathrm{T}} \widetilde{\mathbf{Q}}^{\dagger}$ will still be separating; however, $\widetilde{\sigma}_i = \sigma_i / (\mu_i \nu_i)$ can be made equal to any given numbers by choosing appropriate μ_i and ν_i.

Measures of oblique separability. If Oblique SSA does not separate the components exactly, a measure of separability is necessary. As stated in Sect. 1.3, the main measure of separability in Basic SSA is the $\mathbf{w}$-correlation between two time series: $\rho_{\mathbf{w}}(\mathbb{X}, \mathbb{Y}) = \langle \mathbf{X}, \mathbf{Y} \rangle_{\mathrm{F}} / \left(\|\mathbf{X}\|_{\mathrm{F}} \|\mathbf{Y}\|_{\mathrm{F}} \right)$, where $\mathbf{X}$ and $\mathbf{Y}$ are the trajectory matrices of the series.

In Oblique SSA with $(\mathbf{L}, \mathbf{R})$-SVD we then naturally consider $\rho_{\mathbf{L},\mathbf{R}}$, which is similar to $\rho_{\mathbf{w}}$ and defines the inner product by

$$\langle \mathbf{X}, \mathbf{Y} \rangle_{(\mathbf{L},\mathbf{R})} = \mathrm{trace}(\mathbf{L}\mathbf{X}\mathbf{R}\mathbf{Y}^{\mathrm{T}}).$$

Note that if the matrices $\mathbf{L}$ and $\mathbf{R}$ are not consistent with $\mathbf{X}$ and $\mathbf{Y}$, then $\rho_{\mathbf{L},\mathbf{R}}$ takes into consideration only projections of their columns and rows on the column spaces of $\mathbf{L}$ and $\mathbf{R}$. This means that $\rho_{\mathbf{L},\mathbf{R}}$ can underestimate the separability inaccuracy.

For Oblique SSA, when only one of two coordinate systems (left or right) is oblique, the conventional $\mathbf{w}$-correlations between series are more appropriate measures of separability, since in the case of exact oblique separability we have both $\rho_{\mathbf{w}}$ and $\rho_{\mathbf{L},\mathbf{R}}$ equal to zero.

Another important measure of separability is the closeness of the reconstructed series components to set of time series of finite rank. This can be measured by the contribution of the leading $r_m = |I_m|$ eigentriples into the SVD of the trajectory matrix $\widetilde{\mathbf{X}}^{(m)}$ of the mth reconstructed series component $\widetilde{\mathbb{X}}^{(m)}$. If we denote $\widetilde{\lambda}_{m,i}$ the squared singular values in the ordinary SVD of $\widetilde{\mathbf{X}}^{(m)}$, then

$$\tau_{r_m}(\widetilde{\mathbb{X}}^{(m)}) = 1 - \sum_{i=1}^{r_m} \widetilde{\lambda}_{m,i} / \|\widetilde{\mathbf{X}}^{(m)}\|^2 \tag{2.17}$$

can be considered as a characteristic of closeness of the mth series to the set of series of rank r_m.

2.4.3 Algorithms

Let us present the method described in Sect. 2.4 in the form of algorithms. The method consists of different parts and therefore we describe it as several algorithms.

Let us start with a general algorithm demonstrating how Oblique SVD of a matrix $\mathbf{Z}$ can be reduced to an ordinary SVD.

Algorithm 2.6 $(\mathbf{L}, \mathbf{R})$-SVD

```
Input: Matrix $\mathbf{Z} \in \mathsf{R}^{L \times K}$ to decompose and matrices $\mathbf{L} \in \mathsf{R}^{L \times L}$, and $\mathbf{R} \in \mathsf{R}^{K \times K}$ of rank $r$, where
    $(\mathbf{L}, \mathbf{R})$ is consistent with $\mathbf{Z}$.
Output: The $(\mathbf{L}, \mathbf{R})$-SVD in the form $\mathbf{Z} = \sum_{i=1}^{r} \sigma_i P_i Q_i^{\mathrm{T}}$.
1: Find $\mathbf{O_L} \in \mathsf{R}^{r \times L}$ and $\mathbf{O_R} \in \mathsf{R}^{r \times K}$ such that $\mathbf{O}_\mathbf{L}^{\mathrm{T}} \mathbf{O_L} = \mathbf{L}$ and $\mathbf{O}_\mathbf{R}^{\mathrm{T}} \mathbf{O_R} = \mathbf{R}$.
2: Calculate $\mathbf{O_L} \mathbf{Z} \mathbf{O}_\mathbf{R}^{\mathrm{T}}$.
3: Compute the ordinary SVD decomposition $\mathbf{O_L} \mathbf{Z} \mathbf{O}_\mathbf{R}^{\mathrm{T}} = \sum_{i=1}^{r} \sqrt{\lambda_i} U_i V_i^{\mathrm{T}}$.
4: Set $\sigma_i = \sqrt{\lambda_i}$, $P_i = \mathbf{O}_\mathbf{L}^{\dagger} U_i$ and $Q_i = \mathbf{O}_\mathbf{R}^{\dagger} V_i$, where $\dagger$ denotes pseudo-inverse.
```

Now we formulate the Iterative O-SSA algorithm. Denote $\mathbf{Y} = \mathbf{X}_I$, $r = \operatorname{rank} \mathbf{Y}$, $\mathbb{Y} = \mathcal{T}^{-1} \circ \Pi_{\mathcal{H}}(\mathbf{Y})$ the series obtained from $\mathbf{Y}$ by the diagonal averaging.

We separate the whole algorithm into two parts. Algorithm 2.7 shows a general scheme of Iterative O-SSA, but it does not show how to calculate the pair of matrices $(\mathbf{L}^{(k)}, \mathbf{R}^{(k)})$ at each iteration. Algorithm 2.8 covers this gap.

Algorithm 2.7 Iterative O-SSA

```
Input: Decomposition of the $L$-trajectory matrix $\mathbf{X} = \sum_{i=1}^{d} \sigma_i P_i Q_i^{\mathrm{T}}$ of the series $\mathbb{X}$; disjoint sets
    of indices $\widetilde{J}_1$ and $\widetilde{J}_2$ from $\{1, \ldots, d\}$; the accuracy tolerance $\varepsilon$; function $\rho$ for calculating the
    accuracy; the maximal number of iterations $M$; initial matrices $(\mathbf{L}^{(0)}, \mathbf{R}^{(0)})$ consistent with
    $\mathbf{Y} = \mathbf{X}_I$. The set $I = \{i_1, \ldots, i_r\}$ is defined as $I = \widetilde{J}_1 \sqcup \widetilde{J}_2$, $r_m = |\widetilde{J}_m|$, $r = |I| = r_1 + r_2$, the
    sets $J_1$ and $J_2$ are defined in Remark 2.2. This partition produces the decompositions for the
    matrices and series: $\mathbf{Y} = \mathbf{Y}_{J_1}^{(0)} + \mathbf{Y}_{J_2}^{(0)}$ and $\mathbb{Y} = \widetilde{\mathbb{Y}}^{(1,0)} + \widetilde{\mathbb{Y}}^{(2,0)}$.
Output: $\mathbb{Y} = \widetilde{\mathbb{Y}}^{(1)} + \widetilde{\mathbb{Y}}^{(2)}$.
1: Set $k = 1$.
2: Call Algorithm 2.8 for calculation of $(\mathbf{L}^{(k)}, \mathbf{R}^{(k)})$ consistent with $\mathbf{Y}$.
3: Compute the $(\mathbf{L}^{(k)}, \mathbf{R}^{(k)})$-SVD of $\mathbf{Y}$ by Algorithm 2.6:
       $$\mathbf{Y} = \sum_{i=1}^{r} \sigma_i^{(k)} P_i^{(k)} (Q_i^{(k)})^{\mathrm{T}} = \mathbf{Y}_{J_1}^{(k)} + \mathbf{Y}_{J_2}^{(k)}. \qquad (2.18)$$
4: Obtain the decomposition of the series $\mathbb{Y} = \widetilde{\mathbb{Y}}^{(1,k)} + \widetilde{\mathbb{Y}}^{(2,k)}$, where $\widetilde{\mathbb{Y}}^{(m,k)} = \mathcal{T}^{-1} \circ \Pi_{\mathcal{H}}(\mathbf{Y}_{J_m}^{(k)})$,
    $m = 1, 2$.
5: If $k \geq M$ or $\max\big(\rho(\widetilde{\mathbb{Y}}^{(m,k)} - \widetilde{\mathbb{Y}}^{(m,k-1)}), m = 1, 2\big) < \varepsilon$, then $\widetilde{\mathbb{Y}}^{(m)} \leftarrow \widetilde{\mathbb{Y}}^{(m,k)}$, $m = 1, 2$, and
    STOP; else $k \leftarrow k + 1$ and go to Step 2.
```

Algorithm 2.8 presents the iteration itself, including the sigma-correction, which may be useful for achieving the strong separability.

Algorithm 2.8 Calculation of $(\mathbf{L}^{(k)}, \mathbf{R}^{(k)})$

```
Input: Partition $\{1,\ldots,r\} = J_1 \sqcup J_2$; $r_m = |J_m|$; pair of matrices $(\mathbf{L}^{(k-1)}, \mathbf{R}^{(k-1)})$; parameter for
    sigma-correction $\varkappa > 1$ (if $\varkappa = 0$, then the sigma-correction is not performed).
Output: Pair of matrices $(\mathbf{L}^{(k)}, \mathbf{R}^{(k)})$ for kth iteration.
1: Calculate $\widetilde{\mathbf{Y}}_m = \Pi_{\mathcal{H}} \mathbf{Y}_{J_m}^{(k-1)}$, $m = 1, 2$.
2: Construct the ordinary SVDs:
       $$\widetilde{\mathbf{Y}}_m = \sum_{i=1}^{d_m} \sqrt{\lambda_i^{(m)}} U_i^{(m)} (V_i^{(m)})^{\mathrm{T}}, \ m = 1, 2,$$
   (we need the first $r_m$ terms only).
3: Sigma-correction (if $\varkappa \neq 0$): If $\lambda_{r_1}^{(1)} < \varkappa^2 \lambda_1^{(2)}$, then define $\mu = \varkappa\sqrt{\lambda_1^{(2)}/\lambda_{r_1}^{(1)}}$ and change
   $\widehat{\mathbf{U}}^{(2)} \leftarrow \sqrt{\mu}\widehat{\mathbf{U}}^{(2)}$, $\widehat{\mathbf{V}}^{(2)} \leftarrow \sqrt{\mu}\widehat{\mathbf{V}}^{(2)}$. In view of reordering, set $J_1 = \{1,\ldots,r_1\}$, $J_2 = \{r_1 +$
   $1,\ldots,r\}$.
4: Find the projections $\widehat{U}_i^{(m)} = \Pi_{\mathrm{col}} U_i^{(m)}$ and $\widehat{V}_i^{(m)} = \Pi_{\mathrm{row}} V_i^{(m)}$ for $i = 1,\ldots,r_m$, $m = 1, 2$.
   Denote
       $$\widehat{\mathbf{U}}^{(m)} = [\widehat{U}_1^{(m)} : \ldots : \widehat{U}_{r_m}^{(m)}], \ \widehat{\mathbf{V}}^{(m)} = [\widehat{V}_1^{(m)} : \ldots : \widehat{V}_{r_m}^{(m)}].$$
5: Calculate $\mathbf{L}^{(k)} = (\widehat{\mathbf{U}}^{\dagger})^{\mathrm{T}} \widehat{\mathbf{U}}^{\dagger}$ and $\mathbf{R}^{(k)} = (\widehat{\mathbf{V}}^{\dagger})^{\mathrm{T}} \widehat{\mathbf{V}}^{\dagger}$, where $\widehat{\mathbf{U}} = [\widehat{\mathbf{U}}^{(1)} : \widehat{\mathbf{U}}^{(2)}]$ and $\widehat{\mathbf{V}} = [\widehat{\mathbf{V}}^{(1)} :$
   $\widehat{\mathbf{V}}^{(2)}]$.
```

Remark 2.4 Algorithm 2.7, which uses the sigma-correction, may change the groups of indices. The new groups in Algorithm 2.8 are constructed in such a way that J_1 and J_2 partition the set $\{1,\ldots,r\}$. The new partition of I is obtained as $\widetilde{J}_1 = \{i_k \in I : k \in J_1\}$ and $\widetilde{J}_2 = \{i_k \in I : k \in J_2\}$.

Remark 2.5 Algorithm 2.7 describes a refined decomposition of the matrix $\mathbf{X}_I$. However, we can consider Iterative O-SSA as an algorithm, where the full decomposition of the trajectory matrix $\mathbf{X}$ of an original series $\mathbb{X}$ is used (which changes components from the group I). The result would also be a full decomposition.

2.4.4 *Iterative O-SSA in* Rssa

2.4.4.1 Description of Functions

Since Iterative O-SSA is a nested method, the `ssa` function can be called for obtaining an `ssa` object `s`, see "Description of Function" in Sects. 2.1–2.3, 2.6. For Iterative O-SSA itself, the function `iossa` is used. Since the result of `iossa` is also an `ssa` object, which contains the full decomposition, the `iossa` function can be applied to the result of another application of `iossa`.

Let us outline the main arguments of `iossa` in a typical function call:

```
ios <- iossa(s, nested.groups = list(c(1,4),7:10), trace = FALSE,
             tol = 1e-5, maxiter = 100,
             norm = function(x) sqrt(mean(x^2)),
             kappa = 2)
```

Arguments:

`s` is an `ssa` object holding the full one-dimensional SSA (Basic SSA, Toeplitz SSA, Basic SSA with projections, Shaped SSA) decomposition.

`nested.groups` is a list of groups of eigentriples from the full decomposition `s`; the list gives the initial grouping for Iterative O-SSA iterations.

`tol`, `maxiter`, `norm` are related to the convergence of iterations: tolerance with respect to the indicated norm and the number of iterations. Function `norm` calculates a norm of a vector; this norm is applied to the difference between the reconstructed series at sequential iterations and is used for convergence detection. If this norm is smaller than `tol`, then iterations stop.

`trace` indicates whether the convergence process should be traced.

`kappa` is the "kappa"-parameter for the sigma-correction procedure. If `kappa = NULL`, the sigma-correction is not applied.

Note that only `s` and `nested.groups` should be set if the default values are appropriate.

The returning value of the function is an object of `ossa` class. In addition to usual `ssa` object fields, it also contains the following fields:

`iossa.result` is an object of `iossa.result` class, a list which contains algorithm parameters, condition numbers, separability measures, the number of iterations, and the convergence status.

`iossa.groups` is a list of groups within the nested decomposition; indices of components correspond to their indices in the full decomposition.

`iossa.groups.all` is a list, which describes the cumulative grouping after sequential Iterative O-SSA decompositions in the case of non-intersecting groups given by `nested.groups`. Otherwise, the list `iossa.groups.all` coincides with `iossa.groups`.

`ossa.set` is a vector of indices of elementary components used in Iterative O-SSA (that is, used in `nested.groups`).

To look at weighted oblique correlations of the obtained elementary components, one can call `owcor(ios, groups = ios$ossa.set)`.

The `reconstruct` function performs reconstruction as usual. A possible question is how to set the groups, which are a part of the result of `iossa`. A typical call is

```
r.ios <- reconstruct(ios, groups = ios$iossa.groups)
```

2.4.4.2 Typical Code

Fragment 2.4.1 demonstrates the method on a simulated example. Since Iterative O-SSA is designed to separate non-orthogonal series components, let us consider a noisy sum of three sine waves with two of them having close frequencies, see Fig. 2.14. For achieving separability from noise we assume that the level of noise is low.

First, we apply Basic SSA. In Fig. 2.15 we see that the signal is contained in components 1–6 and is separated from noise. Weighted correlations do not show any

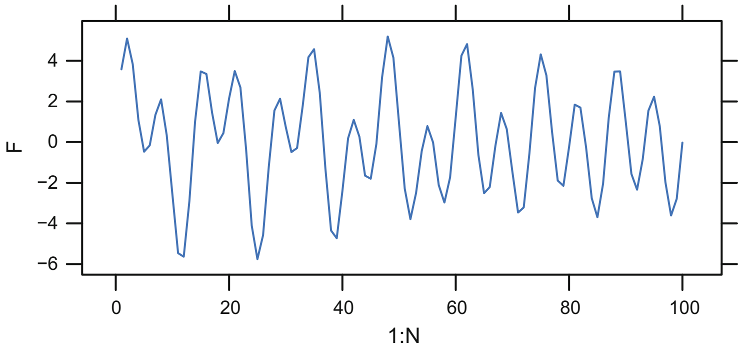


Fig. 2.14 Noisy sum of three sinusoids: The original series

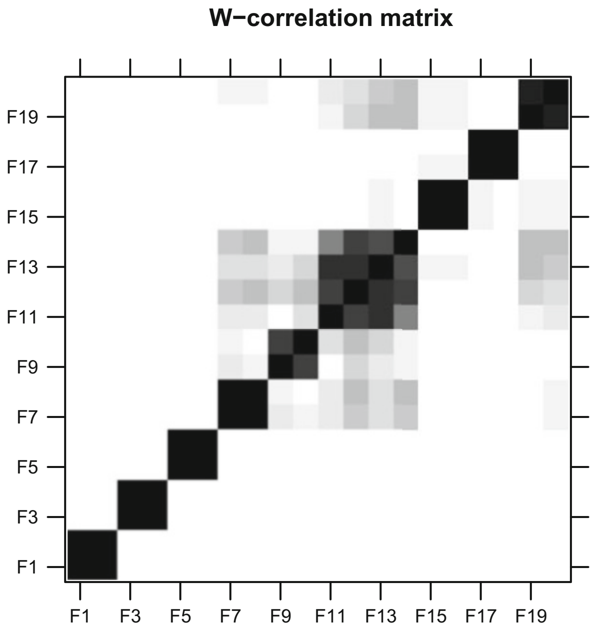


Fig. 2.15 Noisy sum of three sinusoids, Basic SSA: **w**-Correlation matrix

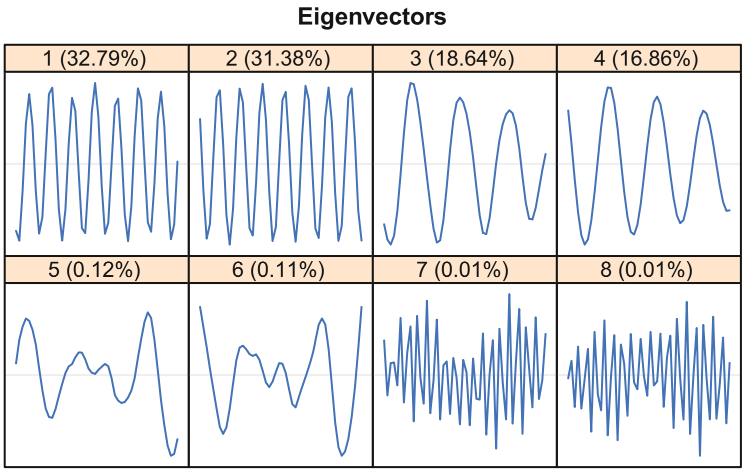


Fig. 2.16 Noisy sum of three sinusoids, Basic SSA: Eigenvectors

problem with separability of three groups of components, 1–2, 3–4, 5–6. However the graph with eigenvectors (Fig. 2.16) shows that the pairs 3–4 and 5–6 are most likely mixed within the signal. This means that Iterative O-SSA may help.

Fragment 2.4.1 (Noisy Sum of Three Sinusoids: Iterative O-SSA)

```
> N <- 100
> L <- 50
> omega1 <- 0.07
> omega2 <- 0.065
> omega3 <- 0.15
> sigma <- 0.1
> set.seed(3)
> F <- 2 * sin(2 * pi * omega1 * (1:N)) +
+   sin(2 * pi * omega2 * (1:N)) +
+   3 * sin(2 * pi * omega3 * (1:N)) + sigma * rnorm(N)
> xyplot(F ~ 1:N, type = "l")
> s <- ssa(F, L)
> plot(s, type = "vectors", idx = 1:8, layout = c(4, 2))
> plot(wcor(s, groups = 1:20), scales = list(at = seq(1,20,2)))
> ios <- iossa(s, nested.groups = list(3:4, 5:6), maxiter = 1000)
> plot(ios, type = "vectors", idx = 1:8, layout = c(4, 2))
> ior <- reconstruct(ios, groups = c(list(1:2), ios$iossa.groups))
> plot(ior, plot.method = "xyplot", add.original = FALSE,
+      add.residuals = FALSE)
```

Indeed, the reconstruction by Basic SSA has failed, while the nested reconstruction of the signal components by Iterative O-SSA is successful, see Figs. 2.17 and 2.18. We can apply Iterative O-SSA to the whole signal but the separability is better if we take only the mixed components 3–6. We choose initial grouping

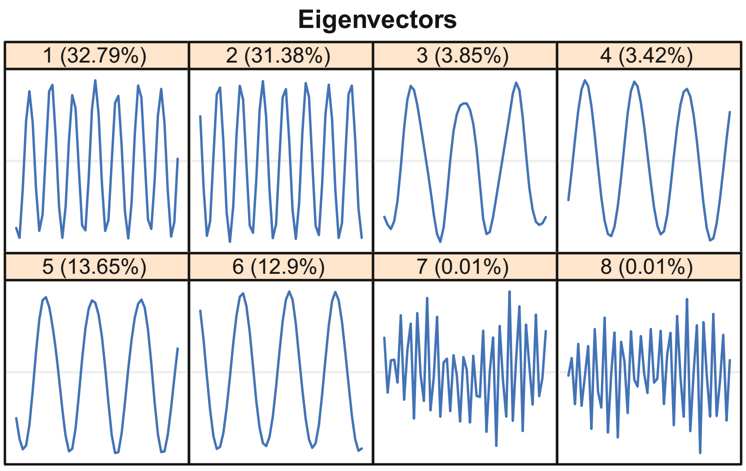


Fig. 2.17 Noisy sum of three sinusoids, Iterative O-SSA: Eigenvectors

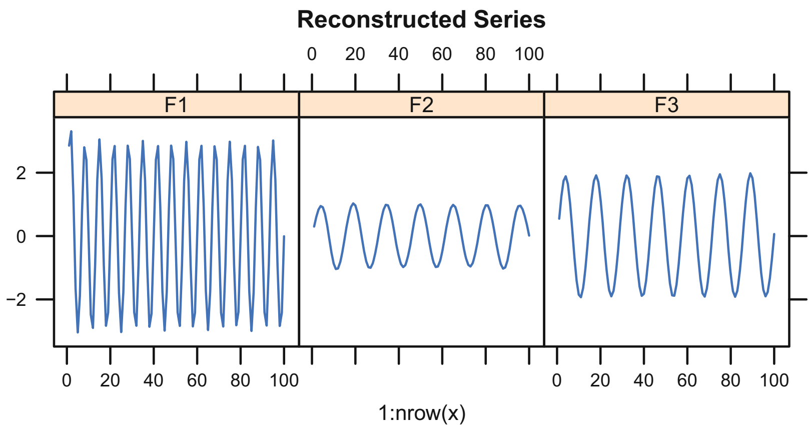


Fig. 2.18 Noisy sum of three sinusoids, Iterative O-SSA: Reconstruction

`list(3:4, 5:6)` since these groups look like corrupted pairs; we could have also chosen `list(3:6, 4:5)`.

For Iterative O-SSA, it is convenient to use automatically generated groups `ios$iossa.groups` for grouping within the refined decomposition. The groups returned by `ios$iossa.groups` can differ from the groups which were set initially, especially if the sigma-correction is used (`kappa` is not zero). In the considered example, the resulting groups after sigma-correction will be `list(3:4, 5:6)`.

If the problem of lack of conventional weak separability is supplemented by the problem of lack of strong separability (in the considered example, when the amplitudes of sinusoids coincide or are close to each other), the use of `kappa` can still allow us to achieve the right decomposition.

Fragment 2.4.2 contains characteristics of the resultant decomposition by Iterative O-SSA. The summary contains measures of quality of the initial and the final decompositions. One can see that the iterations converged according to a default tolerance (`tol`) in 243 iterations. The measure τ defined in (2.17) as a measure of rank-deficiency of trajectory matrices of the found components has decreased considerably. Standard **w**-correlations appear inappropriate as measures of separability.

Fragment 2.4.2 (Noisy Sum of Three Sinusoids: Iterative O-SSA, Summary)

```
> print(ios$iossa.groups)
[[1]]
[1] 3 4
[[2]]
[1] 5 6
> summary(ios)
Call:
iossa.ssa(x = s, nested.groups = list(3:4, 5:6), maxiter = 1000)
Series length: 100,Window length: 50,        SVD method: eigen
Special triples:  0
Computed:
Eigenvalues: 50,        Eigenvectors: 50,      Factor vectors: 6
Precached: 0 elementary series (0 MiB)
Overall memory consumption (estimate): 0.0352 MiB
Iterative O-SSA result:
        Converged:              yes
        Iterations:             243
        Initial mean(tau):      0.1032
        Initial tau:            0.0007976, 0.2055299
        I. O-SSA mean(tau):     0.0004452
        I. O-SSA tau:           0.0006709, 0.0002196
        Initial max wcor:       0.02442
        I. O-SSA max wcor:      0.06986
        I. O-SSA max owcor:     0.0732
```

2.4.4.3 Simulated Example: Separability of Sine Waves

Let us add noise to the sum of two sinusoids and take

$$x_n = \sin(2\pi\omega_1 n) + A\sin(2\pi\omega_2 n) + \delta\varepsilon_n$$

with close frequencies $\omega_1 = 0.07$ and $\omega_2 = 0.06$ and unequal amplitudes, 1 and $A = 1.2$. Here ε_n is white Gaussian noise with variance 1, $\delta = 1$. Let $N = 150$, $L = 70$.

Basic SSA separates well the sinusoids from noise, but cannot separate these sinusoids from each other. Thus, Iterative O-SSA, applied to the estimated signal subspace, should be used. We use the sigma-correction with $\varkappa = 2$, since the difference between amplitudes, 1 and 1.2, appears to be small for achieving strong separability in the presence of noise. We set the initial grouping ET1–2 and ET3–4.

Let us investigate the dependence of number of iterations on ω_1 with fixed $\omega_2 = 0.06$. We change ω_1 from 0.03 to 0.059 and from 0.061 to 0.1. Fragment 2.4.3 depicts the results.

Fragment 2.4.3 (Dependence of `iossa` Error on Difference Between Frequencies)

```
> rowMeansQuantile <- function(x, level = 0.05) {
+   apply(x, 1,
+         function(x) {
+           q <- quantile(x, c(level / 2, 1 - level / 2))
+           x[x < q[1]] <- q[1]
+           x[x > q[2]] <- q[2]
+
+           mean(x)
+         })
+ }
> data("iossa.err", package = "ssabook")
> lseq <- c(seq(0.03, 0.058, 0.002), seq(0.062, 0.1, 0.002))
> iter.real <- rowMeansQuantile(iossa.err$iter.real)
> iter.est <- iossa.err$iter.est
> err1 <- sqrt(rowMeansQuantile(iossa.err$err1))
> err2 <- sqrt(rowMeansQuantile(iossa.err$err2))
> xlab <- expression(omega[1])
> ylab1 <- expression(N[plain(iter)])
> ylab2 <- expression(RMSE)
> p1 <- xyplot(iter.real + iter.est ~ lseq,
+               type = "l", ylab = ylab1, xlab = xlab)
> p2 <- xyplot(err1 + err2 ~ lseq,
+               type = "l", ylab = ylab2, xlab = xlab)
> print(p1, split = c(1, 1, 1, 2), more = TRUE)
> print(p2, split = c(1, 2, 1, 2), more = FALSE)
```

Figure 2.19 (top) shows the number of iterations for noiseless signal (blue line) and the estimated mean number of iterations for the noisy signal (red line); the number of repetitions equals 1000, 5% winsorized estimates of means were calculated. Note that the number of iterations was limited by 200, although for the pure signal the convergence was achieved for each ω_1 from the considered set. A surprisingly small number of iterations for the noisy signal and close frequencies is explained by the convergence to a wrong limit, see Fig. 2.19 (bottom) with root mean square errors of LS-ESPRIT estimates for ω_1 and ω_2 based on the subspaces spanned by eigenvectors from ET1–2 and ET3–4 (see Algorithm 3.3 for the ESPRIT algorithms). Since we use the nested decomposition, the noise slightly influences the reconstruction accuracy for the frequencies that are quite different (ω_1 smaller than 0.048 and larger than 0.072).

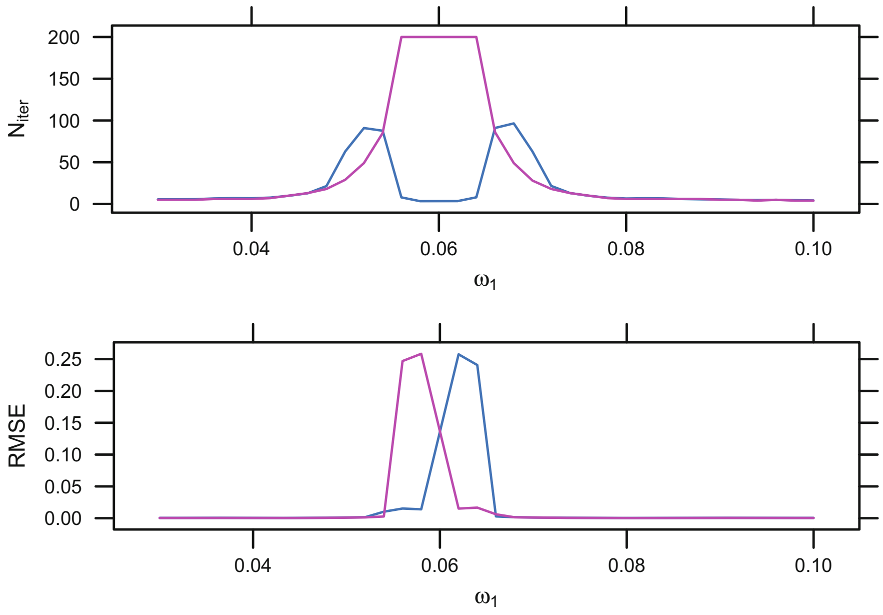


Fig. 2.19 Dependence of number of iterations (top) and RMSE errors of frequency estimations (bottom) on ω_1 for $\omega_2 = 0.6$

2.5 Filter-Adjusted O-SSA and SSA with Derivatives

2.5.1 Method

In this section we describe further variations of SSA that help to overcome the problem of lack of strong separability of components if weak separability holds.

Recall that the lack of strong separability of two series components is caused by equal singular values in the sets of the singular values generated by the two components. In turn, the singular values depend on the coefficients A_1 and A_2 in the representation of the signal as the sum $s_n = A_1 s_n^{(1)} + A_2 s_n^{(2)}$. The question is how to change the coefficients A_1 and A_2 with unknown $s_n^{(1)}$ and $s_n^{(2)}$ to make the singular values different.

It appears that it could be advantageous to use the derivative of the time series in order to change the coefficients without changing the component subspaces. For example, if $x_n = A\sin(2\pi\omega n + \phi)$, then $x'_n = 2\pi\omega A\cos(2\pi\omega n + \phi)$; that is, the new coefficient is $A' = 2\pi\omega A$. For two sinusoids with different frequencies, derivatives change their amplitudes differently. The derivative of $x_n = Ae^{\alpha n}$ also changes the coefficient before the exponential, since $x'_n = \alpha Ae^{\alpha n}$, preserving the rate. For most of the series of finite rank, the derivative subspace coincides with the series subspace. The exception is the polynomial series, when the derivative subspace is a subset of the initial subspace.

As we deal with discrete time, we consider the differences $\varphi_n(\mathbb{X}) = x_{n+1} - x_n$ instead of derivatives, but this is still an approach that seems to be working well. For example, for the series $\mathbb{X} = \mathbb{X}_N$ of length N with $x_n = A\sin(2\pi\omega n + \phi)$, the differences give us the series $\varPhi_{N-1}(\mathbb{X}) = (\varphi_1(\mathbb{X}), \ldots, \varphi_{N-1}(\mathbb{X}))$ of length $N-1$ with $\varphi_n(\mathbb{X}) = 2\sin(\pi\omega)A\cos(2\pi\omega n + \pi\omega + \phi)$; for $x_n = Ae^{\alpha n}$, we obtain $\varphi_n(\mathbb{X}) = (e^\alpha - 1)Ae^{\alpha n}$.

Thus, we can combine the initial series and the series of its differences to change the balance for the component contributions and therefore to reach strong separability. For sinusoids with small periods, an increase of the sinusoid amplitudes is large. Therefore, taking derivatives (or differences) increases the contribution of high frequencies. This can also increase the level of the noise component, if the series is corrupted by a high-frequency noise. Hence, the nested version of the method implementation should be employed; in particular, the noise component should be removed by Basic SSA first.

The approach involving derivatives (that is, sequential differences) can be naturally extended to considering an arbitrary linear filtration φ instead of taking simple (sequential) differences. We start with the version with derivatives (that is, differences), since this particular case is simple and has very useful applications.

2.5.1.1 Nested O-SSA with Derivatives (DerivSSA)

Taking the sequential differences changes contributions of the components. Therefore, the method is inappropriate as an approximation method for signal extraction. Thus, the suggested method should be applied in a nested manner (see Sect. 2.4.1.2).

Let us formulate the nested version of O-SSA with derivatives called DerivSSA (Golyandina and Shlemov 2015). As well as in Sect. 2.4.1.2, let L be the window length, $K = N - L + 1$, and $\mathbf{Y} = \mathbf{X}_I \in \mathsf{R}^{L\times K}$ be one of the matrices in the decomposition $\mathbf{X} = \mathbf{X}_{I_1} + \ldots + \mathbf{X}_{I_p}$ obtained at Grouping step of Basic SSA; each group corresponds to a separated time series component and we want to construct a refined decomposition of $\mathbf{Y}$. As before, denote $r = \operatorname{rank}\mathbf{Y}$, $\mathbb{Y} = \mathcal{T}^{-1} \circ \varPi_{\mathcal{H}}(\mathbf{Y})$.

DerivSSA is similar to Basic SSA. Since DerivSSA is applied in a nested manner, the window length is already chosen. Therefore, DerivSSA adds only one additional parameter γ, which regulates the contribution of derivatives.

The DerivSSA method consists of decomposition and reconstruction. First we take sequential differences for each row of $\mathbf{Y} = [Y_1 : \ldots : Y_K]$ and hence compute the matrices $\varPhi(\mathbf{Y}) = [Y_2 - Y_1 : \ldots : Y_K - Y_{K-1}] \in \mathsf{R}^{L\times(K-1)}$ and $\mathbf{Z} = [\mathbf{Y} : \gamma\varPhi(\mathbf{Y})]$. Then DerivSSA works almost exactly as Basic SSA but it uses $\mathbf{Z}$ instead of the conventional trajectory matrix.

After Decomposition step, we obtain the SVD of $\mathbf{Z}$ in the form $\mathbf{Z} = \sum_{i=1}^{r}\sqrt{\lambda_i}U_i V_i^{\mathrm{T}}$. We are interested only in the first K columns in this matrix decomposition. Since the column space of $\mathbf{Z}$ coincides with the column space of

$\mathbf{Y}$ and therefore $\{U_i\}_{i=1}^r$ is a basis of the column space of $\mathbf{Y}$, we can rewrite the decomposition as $\mathbf{Y} = \sum_{i=1}^r U_i U_i^{\mathrm{T}} \mathbf{Y}$.

The rest of the method coincides with Reconstruction step of Basic SSA. After Grouping step, we obtain the decomposition $\mathbf{Y} = \mathbf{Y}_{J_1} + \ldots + \mathbf{Y}_{J_l}$ and then, as a result, the refined series decomposition $\mathbb{Y} = \widetilde{\mathbb{Y}}^{(1)} + \ldots + \widetilde{\mathbb{Y}}^{(l)}$, where $\widetilde{\mathbb{Y}}^{(m)} = \mathcal{T}^{-1} \circ \varPi_{\mathcal{H}}(\mathbf{Y}_{J_m})$, $m = 1, \ldots, l$.

The following proposition shows that DerivSSA is a version of Oblique SSA with a specific pair of matrices $(\mathbf{L}, \mathbf{R})$, where $P_i = U_i$ and Q_i are normalized vectors $\mathbf{Y}^{\mathrm{T}} U_i$ in (2.15). Denote $\mathbf{I}_M$ the $M \times M$ identity matrix.

Proposition 2.1 (Golyandina and Shlemov (2015)) *The left singular vectors of the ordinary SVD of* $\mathbf{Z}$ *coincide with the left singular vectors of the* $(\mathbf{I}_L, \mathbf{R})$*-SVD of the input matrix* $\mathbf{Y}$*, where* $\mathbf{R}$ *is defined by the equality* $\mathbf{R} = \mathbf{I}_K + \gamma^2 \mathbf{F}^{\mathrm{T}} \mathbf{F}$ *and*

$$\mathbf{F} = \begin{pmatrix} -1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 1 & 0 \\ 0 & \cdots & 0 & 0 & -1 & 1 \end{pmatrix} \in \mathsf{R}^{(K-1)\times K}.$$

2.5.1.2 Filter-Adjusted O-SSA

Note that sequential differences, which are taken for each row of the matrix $\mathbf{Y}$, can be extended to an arbitrary linear filter of the rows. That is, we can choose coefficients of a linear filter $A = (a_1, \ldots, a_t)^{\mathrm{T}}$ and define $\varPhi(\mathbf{Y}) = [Y_1^*, \ldots, Y_{K-t+1}^*]$, where $Y_i^* = a_1 Y_i + \ldots + a_t Y_{i+t-1}$. The rest of the respective version of SSA is the same as DerivSSA. DerivSSA corresponds to $t = 2$ and $A = (-1, 1)^{\mathrm{T}}$.

2.5.2 Separability

Let $\mathbb{X}_N = \mathbb{X}_N^{(1)} + \mathbb{X}_N^{(2)}$ and $\mathbb{X}_N^{(1)}$ and $\mathbb{X}_N^{(2)}$ be of finite rank and approximately weakly separable which implies that their row and column trajectory spaces are approximately orthogonal. The same is then true for $\varPhi_{N-1}(\mathbb{X}^{(1)})$ and $\varPhi_{N-1}(\mathbb{X}^{(2)})$, due to the fact that their column spaces belong to the column spaces of $\mathbb{X}_N^{(1)}$ and $\mathbb{X}_N^{(2)}$, while their row spaces are spanned by the vectors of the same structure that the vectors constituting bases of the row spaces of $\mathbb{X}_N^{(1)}$ and $\mathbb{X}_N^{(2)}$, except that these basis vectors have length $K - 1$ and not K. Therefore, approximate orthogonality still holds. Since $\varPhi_{N-1}(\mathbb{X}) = \varPhi_{N-1}(\mathbb{X}^{(1)}) + \varPhi_{N-1}(\mathbb{X}^{(2)})$, DerivSSA applied to $(\mathbb{X}_N, \gamma\varPhi_{N-1}(\mathbb{X}))$ will approximately separate the time series $\mathbb{X}_N^{(1)}$ and $\mathbb{X}_N^{(2)}$.

Thus, DerivSSA does not worsen weak separability and can achieve strong separability. It is important to always keep in mind that DerivSSA increases the contribution of high-frequency components and decreases that for low-frequency components.

2.5.3 Algorithm

The general algorithm of Filter-adjusted O-SSA is described in two equivalent forms in Algorithms 2.9 and 2.10. Algorithm 2.9 directly follows the description of the method given in Sect. 2.5.1, while Algorithm 2.10 is more appropriate for an effective implementation and for a modification implemented in Algorithm 2.11.

Algorithm 2.9 Filter-adjusted O-SSA: decomposition

```
Input: Decomposition of the L-trajectory matrix $\mathbf{X} = \sum \sigma_i P_i Q_i^{\mathrm{T}}$, group of components $I$, $|I| = r$,
    filter coefficients $(a_1, \ldots, a_t)$, weight $\gamma > 0$.
Output: Decomposition of $\mathbf{Y} = \mathbf{X}_I$ on elementary matrices $\mathbf{Y} = \mathbf{Y}_1 + \ldots + \mathbf{Y}_r$, where $\mathbf{Y}_i = \sigma_i' P_i' (Q_i')^{\mathrm{T}}$.
1: Form the matrix $\mathbf{Y} = \mathbf{X}_I = \sum_{i \in I} \sigma_i P_i Q_i^{\mathrm{T}}$.
2: Denote $\Phi(\mathbf{Y}) = [Y_1^*, \ldots, Y_{K-t+1}^*]$, where $Y_i^* = a_1 Y_i + \ldots + a_t Y_{i+t-1}$. Construct the matrix
    $\mathbf{Z} = [\mathbf{Y} : \gamma \Phi(\mathbf{Y})]$.
3: Compute the SVD of $\mathbf{Z}$: $\mathbf{Z} = \sum_{i=1}^{r} \sqrt{\lambda_i} U_i V_i^{\mathrm{T}}$.
4: Construct the following decomposition of $\mathbf{Y} = \mathbf{X}_I$ into a sum of elementary matrices: $\mathbf{Y} = \sum_{i=1}^{r} U_i U_i^{\mathrm{T}} \mathbf{Y}$.
5: Obtain the decomposition $\mathbf{Y} = \sum_{i=1}^{r} \sigma_i' P_i' (Q_i')^{\mathrm{T}}$, where $\sigma_i' = \|U_i^{\mathrm{T}} \mathbf{Y}\|$, $P_i' = U_i$, $Q_i' = U_i^{\mathrm{T}} \mathbf{Y} / \sigma_i'$.
```

Algorithm 2.10 Filter-adjusted O-SSA: decomposition (equivalent)

```
Input: Decomposition of the trajectory matrix $\mathbf{X} = \sum \sigma_i P_i Q_i^{\mathrm{T}}$, group of components $I$, $|I| = r$,
    filter coefficients $(a_1, \ldots, a_t)$, weight $\gamma$.
Output: Decomposition of $\mathbf{Y} = \mathbf{X}_I$ on elementary matrices $\mathbf{Y} = \mathbf{Y}_1 + \ldots + \mathbf{Y}_r$, where $\mathbf{Y}_i = \sigma_i' P_i' (Q_i')^{\mathrm{T}}$.
1: Form the matrix $\mathbf{Y} = \mathbf{X}_I = \sum_{i \in I} \sigma_i P_i Q_i^{\mathrm{T}}$ and compute its thin SVD $\mathbf{Y} = \mathbf{U}_r \Lambda_r^{1/2} \mathbf{V}_r^{\mathrm{T}}$. Set
    $\mathbf{S} = [S_1 : \ldots : S_K] = \Lambda_r^{1/2} \mathbf{V}_r^{\mathrm{T}} \in \mathsf{R}^{r \times K}$.
2: Denote $\Phi(\mathbf{S}) = [S_1^*, \ldots, S_{K-t+1}^*]$, where $S_i^* = a_1 S_i + \ldots + a_t S_{i+t-1}$. Construct the matrix
    $\mathbf{Z} = [\mathbf{S} : \gamma \Phi(\mathbf{S})] \in \mathsf{R}^{r \times (2K-t+1)}$.
3: Calculate the rotation matrix $\widetilde{\mathbf{U}} \in \mathsf{R}^{r \times r}$ consisting of the eigenvectors of $\mathbf{Z}\mathbf{Z}^{\mathrm{T}}$ as columns.
4: Set $\widetilde{\mathbf{P}} = [\widetilde{P}_1 : \ldots : \widetilde{P}_r] = \mathbf{U}_r \widetilde{\mathbf{U}}$ and $\widetilde{\mathbf{Q}} = [\widetilde{Q}_1 : \ldots : \widetilde{Q}_r] = \mathbf{S}^{\mathrm{T}} \widetilde{\mathbf{U}}$.
5: Obtain the decomposition $\mathbf{Y} = \sum_{i=1}^{r} \sigma_i' P_i' (Q_i')^{\mathrm{T}}$, where $\sigma_i' = \|\widetilde{Q}_i\|$, $P_i' = \widetilde{P}_i$, $Q_i' = \widetilde{Q}_i / \sigma_i'$.
```

The method introduced in Sect. 2.5.1 has a modification implemented in RSSA, which can slightly worsen the separability but has an advantage that it orders the eigentriples corresponding to sine-waves exactly by the decrease of their frequencies, independently of the values of the sine-wave amplitudes. We will call this modification “Filter-adjusted O-SSA with normalization.” The main difference between Algorithms 2.10 and 2.11 is in the construction of the matrix **S** at step 1.

Algorithm 2.11 Filter-adjusted O-SSA with normalization: decomposition

```
Input: Decomposition of the trajectory matrix $\mathbf{X} = \sum \sigma_i P_i Q_i^{\mathrm{T}}$, group of components $I$, $|I| = r$,
    filter coefficients $(a_1, \ldots, a_t)$, weight $\gamma$.
Output: Decomposition of $\mathbf{Y} = \mathbf{X}_I$ on elementary matrices $\mathbf{Y} = \mathbf{Y}_1 + \ldots + \mathbf{Y}_r$, where $\mathbf{Y}_i = \sigma_i' P_i' (Q_i')^{\mathrm{T}}$.
1: Form the matrix $\mathbf{Y} = \mathbf{X}_I = \sum_{i \in I} \sigma_i P_i Q_i^{\mathrm{T}}$ and construct its thin SVD $\mathbf{Y} = \mathbf{U}_r \Lambda_r^{1/2} \mathbf{V}_r^{\mathrm{T}}$. Set
    $\mathbf{S} = [S_1 : \ldots : S_K] = \mathbf{V}_r^{\mathrm{T}} \in \mathsf{R}^{r \times K}$.
2: Denote $\Phi(\mathbf{S}) = [S_1^*, \ldots, S_{K-t+1}^*]$, where $S_i^* = a_1 S_i + \ldots + a_t S_{i+t-1}$. Construct the matrix
    $\mathbf{Z} = [\mathbf{S} : \gamma \Phi(\mathbf{S})] \in \mathsf{R}^{r \times (2K-t+1)}$.
3: Calculate the rotation matrix $\widetilde{\mathbf{U}} \in \mathsf{R}^{r \times r}$ consisting of the eigenvectors of $\mathbf{Z}\mathbf{Z}^{\mathrm{T}}$ as columns.
4: Set $\widetilde{\mathbf{P}} = [\widetilde{P}_1 : \ldots : \widetilde{P}_r] = (\mathbf{U}_r \Lambda_r^{1/2}) \widetilde{\mathbf{U}}$ and $\widetilde{\mathbf{Q}} = [\widetilde{Q}_1 : \ldots : \widetilde{Q}_r] = \mathbf{S}^{\mathrm{T}} \widetilde{\mathbf{U}}$.
5: Obtain the decomposition $\mathbf{Y} = \sum_{i=1}^{r} \sigma_i' P_i' (Q_i')^{\mathrm{T}}$, where $\sigma_i' = \|\widetilde{P}_i\|$, $P_i' = \widetilde{P}_i / \sigma_i'$, $Q_i' = \widetilde{Q}_i$.
```

Remark 2.6 Algorithms 2.9–2.11 can be extended to the case when several filters in a stacked manner are applied. For example, if filters Φ_1 and Φ_2 are given, then the matrix **Z** at Step 2 has the forms $\mathbf{Z} = [\mathbf{Y} : \gamma \Phi_1(\mathbf{Y}) : \gamma \Phi_2(\mathbf{Y})]$ and $\mathbf{Z} = [\mathbf{S} : \gamma \Phi_1(\mathbf{S}) : \gamma \Phi_2(\mathbf{S})]$, respectively.

The reconstruction algorithm is the same as for most versions of SSA. Since Filter-adjusted O-SSA (as well as DerivSSA) is a nested method, the result is a decomposition of a chosen series component rather than a decomposition of the original series.

Algorithm 2.12 Filter-adjusted O-SSA: reconstruction

```
Input: Decomposition $\mathbf{Y} = \sum_{i=1}^{r} \mathbf{Y}_i$, where $\mathbf{Y}_i = \sigma_i' P_i' (Q_i')^{\mathrm{T}}$, grouping $I = \bigsqcup_{k=1}^{l} J_k$.
Output: Refined series decomposition $\mathbb{Y} = \widetilde{\mathbb{Y}}^{(1)} + \ldots + \widetilde{\mathbb{Y}}^{(l)}$.
1: Obtain the grouped matrix decomposition $\mathbf{Y} = \mathbf{Y}_{J_1} + \ldots + \mathbf{Y}_{J_l}$, where $\mathbf{Y}_J = \sum_{j \in J} \mathbf{Y}_j$.
2: Obtain a refined series decomposition $\mathbb{Y} = \widetilde{\mathbb{Y}}^{(1)} + \ldots + \widetilde{\mathbb{Y}}^{(l)}$, where $\widetilde{\mathbb{Y}}^{(m)} = \mathcal{T}^{-1} \circ \Pi_{\mathcal{H}}(\mathbf{Y}_{J_m})$,
    $m = 1, \ldots, l$.
```

Remark 2.7 Algorithm 2.12 describes a refined decomposition of the matrix $\mathbf{X}_I$. However, we can consider Filter-adjusted O-SSA as an algorithm, which we apply to the full decomposition of the trajectory matrix **X** of an original series $\mathbb{X}$, which changes components from the group I. Then the result is also a full decomposition of **X**.

2.5.4 Filter-Adjusted O-SSA in Rssa

2.5.4.1 Description of Functions

As in the case of Iterative O-SSA, since Filter-adjusted O-SSA is a nested method, the `ssa` function must be called prior to this in order to obtain an `ssa` object `s`, see "Description of function" in Sects. 2.1–2.3, 2.6. For Filter-adjusted O-SSA itself, the function `fossa` is used. Since the result of `fossa` is also an `ssa` object, which contains the full decomposition, the `fossa` and `iossa` functions can be applied to the result of another application of `fossa`.

Let us outline the main arguments of `fossa` in a typical function call:

```
fos <- fossa(s, nested.groups = list(2:3, 6:7),
             filter = c(-1,1), gamma = 1, normalize = TRUE)
```

Arguments:

`s` is an `ssa` object holding the full one-dimensional SSA (Basic SSA, Toeplitz SSA, Basic SSA with projections, Shaped SSA) decomposition.

`nested.groups` is a vector of indices of eigentriples from the full decomposition for the nested decomposition. The argument is coerced to a vector, if necessary.

`filter` is a list of numeric vectors of reverse impulse response coefficients for filter adjustment. Value by default `c(-1,1)` corresponds to DerivSSA, which is the most common case.

`gamma` is the weight of filter adjustment; the value `Inf` corresponds to the removal of the first part of the matrix $\mathbf{Z}$: $\mathbf{Z} = \Phi(\mathbf{S})$.

`normalize` indicates if the modification with normalization is used (see Algorithm 2.11).

Note that for Filter-adjusted SSA, only a group I should be given; the partition is performed later at Reconstruction step. Therefore, the list of groups, which are given for the parameter `nested.groups`, is transformed to a single vector of indices composing I (compare with `iossa`). That is, in the considered function call, $I = \{2, 3, 5, 6\}$.

The return value is an object of class `ossa`. The field `ossa.set` contains the vector of indices of elementary components used in Filter-adjusted O-SSA (that is, used in `nested.groups`, which is in fact just I). For example, to look at weighted oblique correlations of the obtained elementary components, one can call `owcor(fos, groups = fos$ossa.set)`.

2.5.4.2 Typical Code

This example demonstrates the difference between Filter-adjusted O-SSA and Iterative O-SSA with sigma-correction. Let us consider a noisy sum of two sinusoids with different and not close frequencies (see Fragment 2.5.1). These sinusoids are approximately separable. However, since the sinusoid amplitudes are equal, there is no strong separability and therefore after application of Basic SSA we obtain an unsatisfactory decomposition, an arbitrary mixture of the sinusoids (the top pictures of Fig. 2.21 with eigenvectors).

Fragment 2.5.1 (Separation of Two Sine Waves with Equal Amplitudes)

```
> N <- 100
> L <- 50
> omega1 <- 0.03
> omega2 <- 0.06
> sigma <- 0.1
> set.seed(3)
> F <- sin(2 * pi * omega1 * (1:N)) +
+   sin(2 * pi * omega2 * (1:N)) +
+   sigma * rnorm(N)
> s <- ssa(F, L = L, neig = min(L, N - L + 1)) #full decomposition
> plot(s)
> p1 <- plot(s, type = "vectors", idx = 1:4, layout = c(4, 1),
+            main = "Eigenvectors, Basic SSA")
> fos <- fossa(s, nested.groups = list(1:2, 3:4), gamma = 10,
+              normalize = FALSE)
> # The total percent is equal to 100%
> print(sum(fos$sigma^2) / sum(s$sigma^2) * 100)
[1] 100
> p2 <- plot(fos, type = "vectors", idx = 1:4, layout = c(4, 1),
+            main = "Eigenvectors, SSA with derivatives")
> ios1 <- iossa(s, nested.groups = list(1:2, 3:4), maxiter = 1)
> # The total percent is not equal to 100%
> print(sum(ios1$sigma^2) / sum(s$sigma^2) * 100)
[1] 99.62939
> p3 <- plot(ios1, type = "vectors", idx = 1:4, layout = c(4, 1),
+            main = "Eigenvectors, Iterative O-SSA, 1 iter")
> ios2 <- iossa(ios1, nested.groups = list(1:2, 3:4), maxiter = 1)
> # The total percent is not equal to 100%
> print(sum(ios2$sigma^2) / sum(s$sigma^2) * 100)
[1] 101.7544
> p4 <- plot(ios2, type = "vectors", idx = 1:4, layout = c(4, 1),
+            main = "Eigenvectors, Iterative O-SSA, 2 iter")
> plot(p1, split = c(1, 1, 1, 4), more = TRUE)
> plot(p2, split = c(1, 2, 1, 4), more = TRUE)
> plot(p3, split = c(1, 3, 1, 4), more = TRUE)
> plot(p4, split = c(1, 4, 1, 4), more = FALSE)
> fo.rec <- reconstruct(fos, groups = list(1:2, 3:4))
> pr1 <- plot(fo.rec, plot.method = "xyplot",
+             main = "SSA with derivatives", xlab = "")
> io.rec <- reconstruct(ios2, groups = ios2$iossa.groups)
> pr2 <- plot(io.rec, plot.method = "xyplot",
+             main = "Iterative O-SSA", xlab = "")
> plot(pr1, split = c(1, 1, 1, 2), more = TRUE)
> plot(pr2, split = c(1, 2, 1, 2), more = FALSE)
```

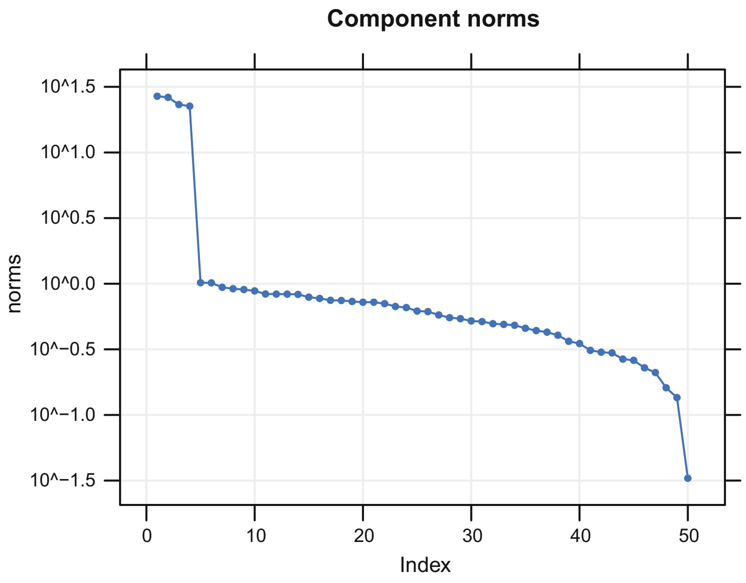


Fig. 2.20 Noisy sum of sinusoids: Graph of eigenvalues for Basic SSA

To apply a nested version of Oblique SSA, we should start with the signal subspace extraction. Figure 2.20 confirms that the signal is contained in eigentriples 1–4, since the corresponding eigenvalues are considerably larger than the eigenvalues of the residual components.

Thus, we apply SSA with derivatives to the group ET1–4, taking a large enough $\gamma = 10$. The eigenvectors become regular (Fig. 2.21, top) and the reconstruction is accurate (Fig. 2.22, top).

The Iterative O-SSA algorithm is designed to improve weak separability. Let us also use its ability to change component contribution during iterations by means of the sigma-correction. One can see that one iteration slightly improves the decomposition, while the second iteration provides almost ideal decomposition (see Fig. 2.21 with eigenvectors and Fig. 2.22 with reconstruction).

Comparing the results in this example, we see that among the versions of the two methods with the same computational cost, DerivSSA is better; however, one more iteration in Iterative O-SSA makes Iterative O-SSA advantageous to DerivSSA.

Let us draw attention to different order of the sinusoids in the reconstructions (Fig. 2.22). The order of components produced by DerivSSA is explained by an increase of contribution of high frequencies due to taking the differences, while the order of components in Iterative O-SSA is more or less random in this example.

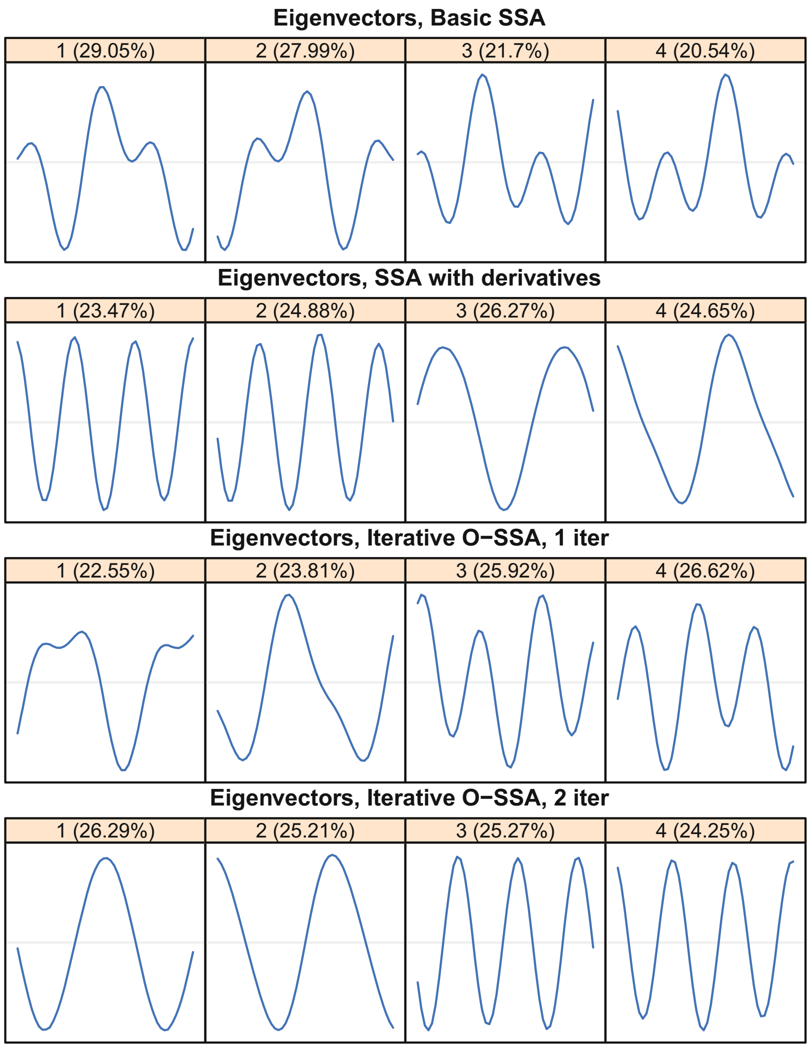
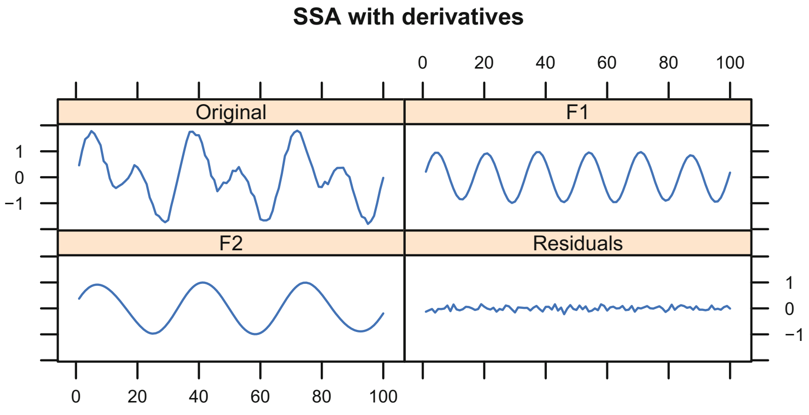


Fig. 2.21 Noisy sum of sinusoids: 1D graphs of eigenvectors for Basic SSA (top), DerivSSA (middle) and Iterative O-SSA, 1 iteration (second from bottom) and 2 iterations (bottom)

For DerivSSA, the decomposition is F-orthogonal and therefore the contributions of series components are correct. Iterative O-SSA may have the sum of contributions different from 100%. This is exactly the case in our example, see the last warning message:

```
   In .contribution(x, idx, ...): Elementary matrices are not
F-orthogonal (max F-cor is -0.016). Contributions can be
irrelevant.
```



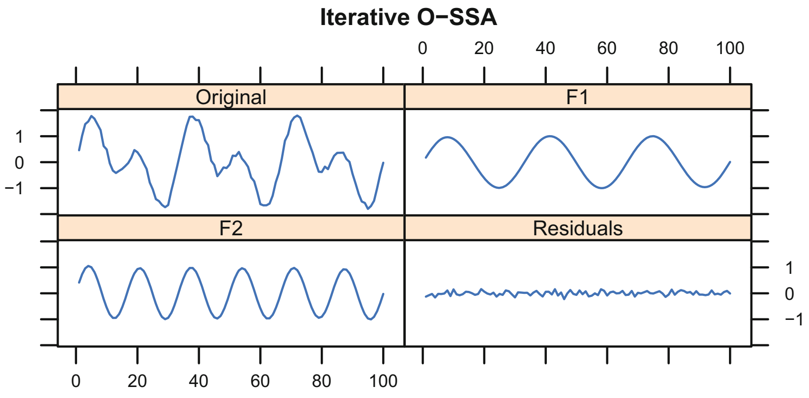


Fig. 2.22 Noisy sum of sinusoids: Reconstructions for DerivSSA (top) and Iterative O-SSA, 2 iterations (bottom)

2.6 Shaped 1D-SSA

2.6.1 Method

Shaped SSA is a very general SSA method. Formally, any other SSA method can be considered as a particular case of Shaped SSA. For multivariate extensions, the versions of Shaped SSA, which cannot be reduced to conventional SSA, have many different applications. For one-dimensional time series, Shaped SSA is particularly useful when the time series contains gaps as in this case the versions of 1D-SSA considered above are not directly applicable.

The specificity of Shaped SSA is in the construction of the L-trajectory matrix, which we denote $\widetilde{\mathbf{X}} = \mathcal{T}_{\mathrm{shSSA}}(\mathbb{X})$. This matrix is constructed so that its columns are the complete L-lagged vectors. Any incomplete lagged vectors containing missing values are not included into $\mathcal{T}_{\mathrm{shSSA}}(\mathbb{X})$.

Denote the set of series elements, which are presented in the trajectory matrix $\widetilde{\mathbf{X}}$, as $\mathfrak{N}$; that is, $\mathfrak{N}$ is the set of non-missed elements of $\mathbb{X}$, which are covered by windows of length L. The operator $\mathcal{T}_{\mathrm{shSSA}}$ makes a one-to-one correspondence between a restriction of the series to $\mathfrak{N}$ and the set of trajectory matrices, if the location of the missing data is fixed.

The SSA decomposition is performed by any technique (excluding Toeplitz SSA) described in Sects. 2.1–2.5 including nested Iterative O-SSA and Filter-adjusted O-SSA. All these SSA decompositions, except for Toeplitz SSA, are eligible tools since construction of the trajectory matrix and a tool for SSA decomposition do not affect one another. Thus, we can naturally define shaped Basic SSA, shaped SSA with projection, and so on.

For Toeplitz SSA, the decomposition is performed in a very specific way, which is not directly based on the trajectory matrix; thereby the shaped version of Toeplitz SSA does not make much sense and hence it is not implemented in the current version of RSSA.

After decomposition of the trajectory matrix into a sum of elementary matrices and then into a sum of grouped matrices, we need to obtain the series decomposition. Generally speaking, the trajectory matrix is not Hankel in view of the gaps. Therefore, we need a more general procedure than the hankelization and diagonal averaging. This procedure is determined by the operator $\Pi_{\mathcal{H},\mathrm{sh}}(\cdot)$ which is defined as follows.

For ith term of the series, where $i \in \mathfrak{N}$, denote $\mathbb{E}_i$ the series with zeros everywhere except ith term, which is equal to 1, $\mathbf{B}_i = \mathcal{T}_{\mathrm{shSSA}}(\mathbb{E}_i)$. For a given matrix $\mathbf{Y}$, define a series $\widetilde{\mathbb{Y}}$ by $\widetilde{y}_i = \langle \mathbf{Y}, \mathbf{B}_i \rangle_{\mathrm{F}} / \|\mathbf{B}_i\|^2$ which gives

$$\Pi_{\mathcal{H},\mathrm{sh}}(\mathbf{Y}) = \mathcal{T}_{\mathrm{shSSA}}(\widetilde{\mathbb{Y}}).$$

2.6.2 *Separability*

Definition and conditions of separability for Shaped SSA are the same as for underlying modifications of SSA (see, e.g., Sect. 2.1.3). However, the separability accuracy is naturally worse when there are gaps in the series. Moreover, there are extreme cases where ranks of series can be corrupted by gaps and where the conditions of separability cannot be satisfied.

2.6.3 *Algorithm*

The SSA modifications described in Sects. 2.2–2.5 differ from Basic SSA (Sect. 2.1) by Decomposition step, while Shaped SSA differs from these modifications by Embedding step, which depends on the window shape and the shape of the analyzed object. Therefore, we consider Shaped SSA as an extension of SSA. Let us describe the Shaped SSA algorithm for the analysis of time series with gaps.

Algorithm 2.13 Shaped SSA: decomposition

```
Input: Time series $\mathbb{X}$ of length $N$ with missing data, window length $L$, an SSA modification for making a decomposition.
Output: Decomposition of the trajectory matrix on elementary matrices $\widetilde{\mathbf{X}} = \widetilde{\mathbf{X}}_1 + \ldots + \widetilde{\mathbf{X}}_d$, where $\widetilde{\mathbf{X}}_i = \sigma_i P_i Q_i^{\mathrm{T}}$.
1: Construct the trajectory matrix $\widetilde{\mathbf{X}} = \mathcal{T}_{\mathrm{shSSA}}(\mathbb{X})$.
2: Obtain the decomposition $\widetilde{\mathbf{X}} = \sum_{i=1}^{d} \widetilde{\mathbf{X}}_i$, where $\widetilde{\mathbf{X}}_i = \sigma_i P_i Q_i^{\mathrm{T}}$, by means of Decomposition step of the chosen SSA modification.
```

Since Decomposition stage differs by Embedding step, which transfers a time series with gaps into a trajectory matrix, Reconstruction stage differs by the way a matrix is transferred to a time series with gaps.

Algorithm 2.14 Shaped SSA: reconstruction

```
Input: Decomposition $\widetilde{\mathbf{X}} = \widetilde{\mathbf{X}}_1 + \ldots + \widetilde{\mathbf{X}}_d$, $\widetilde{\mathbf{X}}_i = \sigma_i P_i Q_i^{\mathrm{T}}$, grouping $\{1, \ldots, d\} = \bigsqcup_{j=1}^{m} I_j$.
Output: Decomposition of the time series on identifiable components $\mathbb{X} = \widetilde{\mathbb{X}}_1 + \ldots + \widetilde{\mathbb{X}}_m$.
1: Construct the grouped matrix decomposition $\widetilde{\mathbf{X}} = \widetilde{\mathbf{X}}_{I_1} + \ldots + \widetilde{\mathbf{X}}_{I_m}$, where $\widetilde{\mathbf{X}}_I = \sum_{i \in I} \widetilde{\mathbf{X}}_i$.
2: $\mathbb{X} = \widetilde{\mathbb{X}}_1 + \ldots + \widetilde{\mathbb{X}}_m$, where $\widetilde{\mathbb{X}}_j = \mathcal{T}_{\mathrm{shSSA}}^{-1} \circ \Pi_{\mathcal{H},\mathrm{sh}}(\widetilde{\mathbf{X}}_{I_j})$, $j = 1, \ldots, m$.
```

We have assumed here that all series points can be covered by the window of the chosen length. If it is not so, then we obtain the reconstruction given only on the covered points.

2.6.4 *Shaped SSA in* RSSA

2.6.4.1 Description of Functions

There is no specific form of the `ssa` function for Shaped SSA. If the series has gaps (that is, it contains `NA` values), then the shaped version of Basic SSA is applied by default.

In the 1D case as well as in the case of 2D-SSA, specific masks for the window shape can be done. For example, the window can be not a whole interval but an interval with a gap. Since windows with gaps do not have much sense in the one-dimensional case we do not discuss general shaped window in this chapter; see Sect. 5.2 on how to set shaped windows for 2D-data.

2.6.4.2 Typical Code

Let us consider the time series “CO2” and set up several artificial missing values.

Fragment 2.6.1 (Decomposition for Series with a Gap)

```
> F <- co2; F[100:200] <- NA
> # Prompt for the choice of window length
> clplot(F)
> # Perform shaped SSA
> s1 <- ssa(F, L = 72)
> plot(s1, type = "vectors", idx = 1:12)
> plot(s1, type = "series", groups = 1:6, layout = c(2, 3))
> plot(wcor(s1, groups = 1:20), scales = list(at = seq(1,20,2)))
> plot(reconstruct(s1, groups = list(c(1, 4, 7))),
+       add.residuals = FALSE,
+       plot.method = "xyplot", superpose = TRUE)
```

Fragment 2.6.1 demonstrates that the code looks similar to the code for Basic SSA. However, there is a specificity. First, the choice of the window length should take into consideration the number of complete vectors. The function `clplot` (Fig. 2.23) shows the proportions of complete vectors; this proportion is equal to 1 only if there are no gaps. Eigenvectors serve for component identification in the same way as for the series with no gaps (Fig. 2.24). Note that plotting the factor vectors can be misleading, since factor vectors are depicted point-by-point and therefore the gaps are not visible. The reconstructed series, however, are drawn correctly, with the gaps (Figs. 2.25, 2.27). Weighted correlations, as usual, help for component identification (Fig. 2.26).

Since in Basic SSA large window lengths can provide better accuracy, Fragment 2.6.2 performs reconstruction with window length $L = 120$. This window

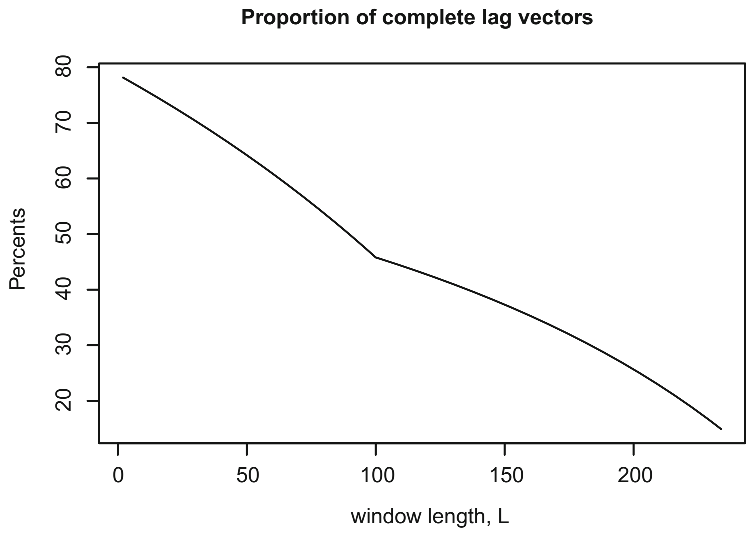


Fig. 2.23 "CO2" with gaps, Shaped SSA: Dependence of proportion of complete vectors on window length

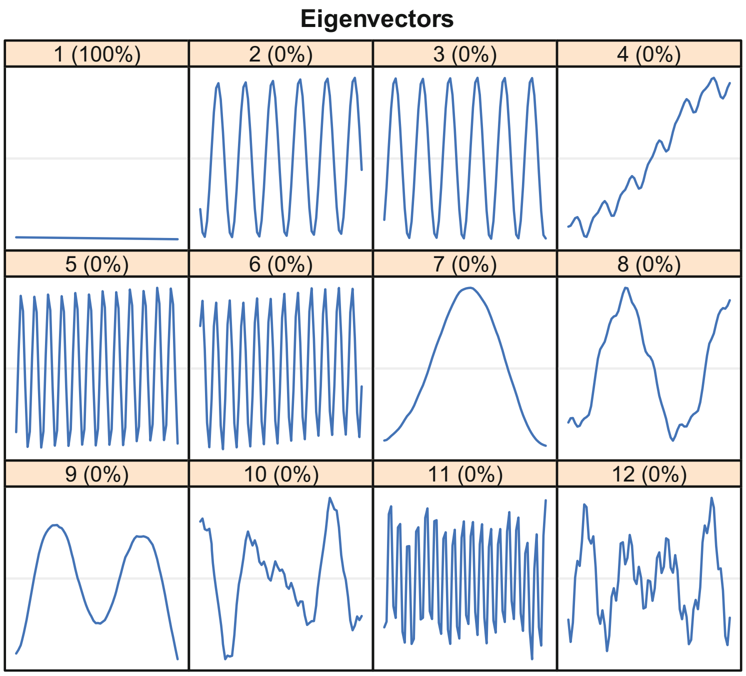


Fig. 2.24 "CO2" with gaps, Shaped SSA: Eigenvectors, $L = 72$

cannot cover the series points on the left from the gap and therefore the left part of the series cannot be reconstructed (Fig. 2.28).

Fragment 2.6.2 (Incomplete Decomposition for a Series with a Gap)

```
> s2 <- ssa(F, L = 120)
> # plot(s2, type = "vectors")
> # plot(wcor(s2, groups = 1:20))
> # plot of reconstruction
> plot(reconstruct(s2, groups = list(c(1, 6, 7))),
+       add.residuals = FALSE,
+       plot.method = "xyplot", superpose = TRUE)
```

2.7 Automatic Grouping in SSA

2.7.1 Methods

While the choice of the window length is well supported by the SSA theory, the procedure for choosing the eigentriples for grouping is much less formal. Several methods for component identification and automatic grouping are described in

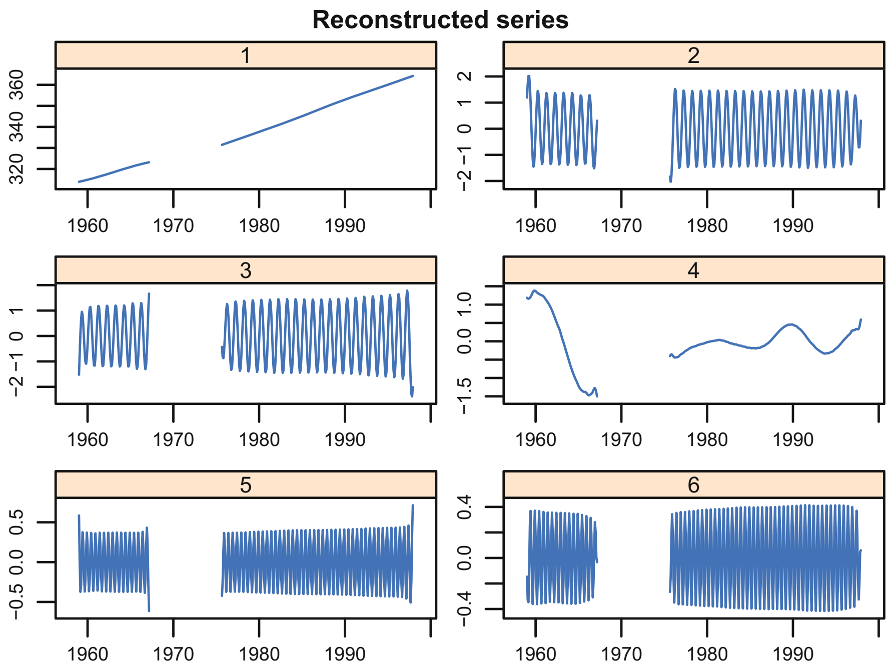


Fig. 2.25 "CO2" with gaps, Shaped SSA: Elementary reconstructed series, $L = 72$

Golyandina and Zhigljavsky (2013; Section 2.4.5). Let us shortly discuss these methods and the basic principles of automatic grouping.

Automatic grouping assumes that the components to be identified are (approximately) separated between themselves and from the residual. Grouping is based on finding common features in the components. The first method measures the communality of components by means of the **w**-correlations $\rho_{ij}^{(\mathbf{w})}$, see (1.7), between them: if a weighted correlation is large, then the corresponding components have similar behavior and should be included into the same group. This approach is very similar to the so-called correlation clustering of variables in multivariate statistics. The input dissimilarity matrix for clustering methods contains values $1 - |\rho_{ij}^{(\mathbf{w})}|$.

The second method is based on finding components with similar frequency characteristics. Contribution of frequencies is defined through the periodogram

$$\Pi_y^M(k/M) = \begin{cases} c_0^2 & \text{for } k = 0, \\ (c_k^2 + s_k^2)/2 & \text{for } 0 < k < M/2, \\ c_{M/2}^2 & \text{for } k = M/2 \text{ if } M \text{ is even,} \end{cases} \tag{2.19}$$

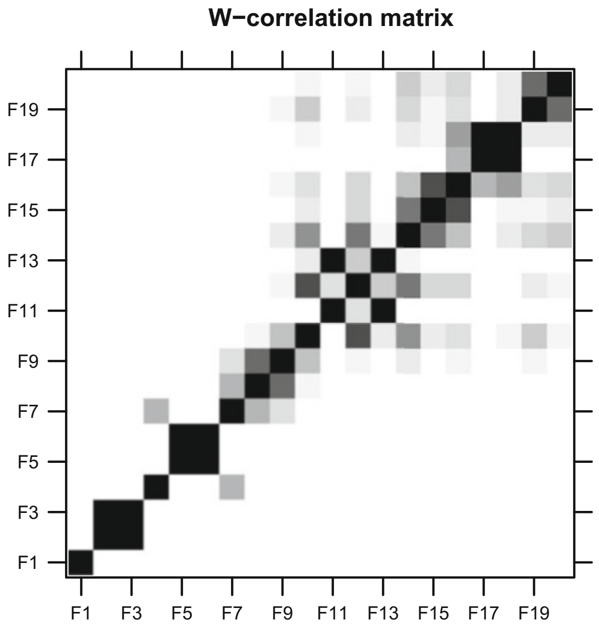


Fig. 2.26 "CO2" with gaps, Shaped SSA: **w**-Correlation matrix, $L = 72$

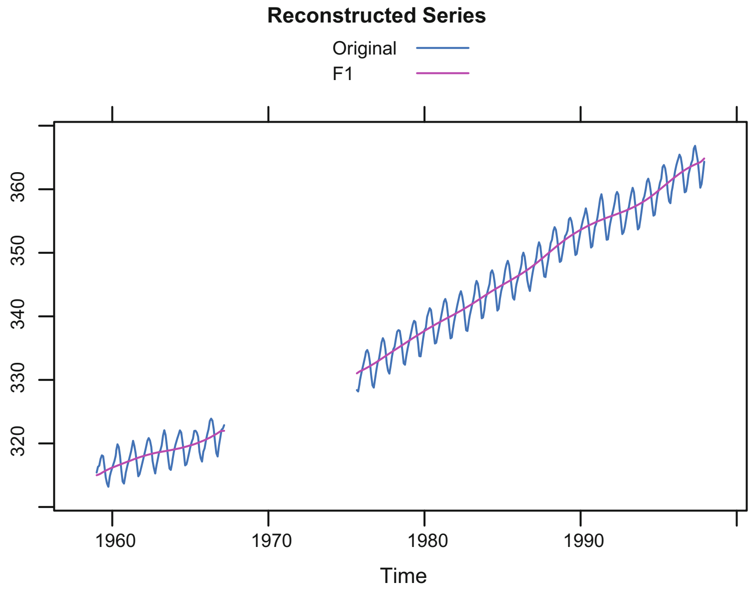


Fig. 2.27 "CO2" with gaps, Shaped SSA: Trend reconstruction, $L = 72$

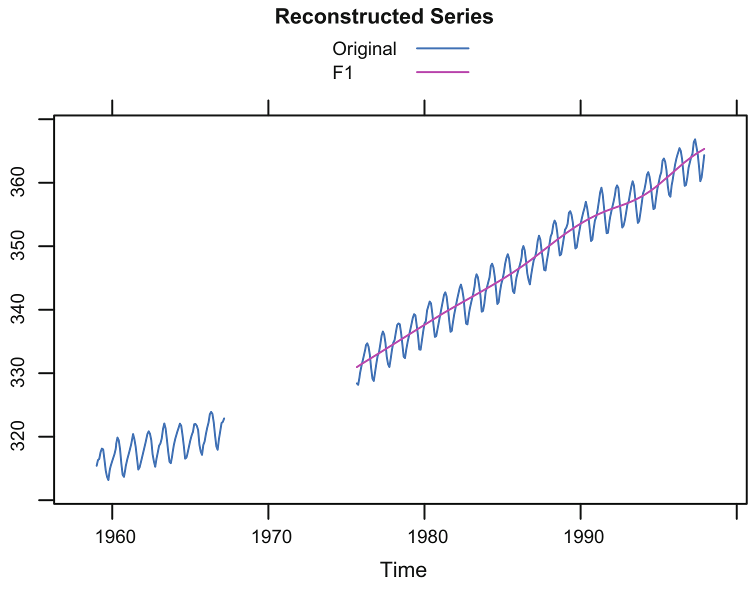


Fig. 2.28 "CO2" with gaps, Shaped SSA: Incomplete trend reconstruction, $L = 120$

where the coefficients c_k and s_k are taken from the Fourier decomposition of $\mathbb{Y} = (y_1, \ldots, y_M)$:

$$y_n = c_0 + \sum_{k=1}^{\lfloor M/2 \rfloor} \Big(c_k \cos(2\pi n\, k/M) + s_k \sin(2\pi n\, k/M) \Big),$$

For a series $\mathbb{Y}$ of length M and for $0 \le \omega_1 \le \omega_2 \le 0.5$, we define

$$T(\mathbb{Y}; \omega_1, \omega_2) = \sum_{k:\omega_1 \le k/M < \omega_2} I_y^M(k/M), \tag{2.20}$$

where $I_y^M(k/M) = M\, \Pi_y^M(k/M)/\|\mathbb{Y}\|^2$, Π_y^M is defined in (2.19). Since $\|\mathbb{Y}\|^2 = M \sum_{k=1}^{[M/2]} \Pi_y^M(k/M)$, the measure $T(\mathbb{Y}; \omega_1, \omega_2)$ can be considered as a proportion of frequencies contained in the frequency bin $[\omega_1, \omega_2)$.

One of the aims in performing grouping is the extraction of a series component with frequency range mostly from the chosen frequency bin. Therefore, it is natural to calculate the value of T for elementary reconstructed components. Moreover, SSA reconstruction can be considered as a linear filter. It appears that the frequency response of the filter generated by the ith eigentriple is almost the same as the periodogram of the corresponding singular vector, see Golyandina and Zhigljavsky (2013; Proposition 3.13). Therefore, it is reasonable to apply T also to singular vectors to reconstruct the series components with the given frequency ranges.

Since the trend of a series can be defined as its slowly varying series component, for extracting a trend a frequency bin in the form $[0, \omega)$ should be chosen. The value of ω reflects the frequency range, which we associated with a trend. For example, if the series has monthly seasonality, ω should be notably smaller than $1/12$. Note that the grouping method does not answer the question whether the extracted component is indeed a deterministic trend or simply a result of smoothing.

We can also consider several frequency bins, perhaps overlapping; in this case, the described method can be applied to each bin separately. A modification of grouping with several bins can be suggested.

Let the whole frequency range be divided into disjoint bins. Then we can refer a component to a frequency bin with the largest proportions of the corresponding frequency range; that is, with the maximal value of T. In this modification, all the considered bins participate simultaneously and are hence dependent. This modification can be used for splitting the series into a set of the components according to the specified frequency ranges.

Values of T for each elementary decomposition component can be used for devising the grouping. To perform an automatic grouping, a threshold T_0, $0 \leq T_0 \leq 1$, should be given. For example, if the value $T(\mathbb{Y}_i; 0, \omega)$ is larger than T_0 for some small ω, where $\mathbb{Y}_i$ is the ith elementary series or ith left/right singular vector, then the corresponding eigentriple can be automatically considered as a part of trend.

2.7.2 Algorithm

The algorithm of auto-grouping, which uses the **w**-correlation matrix, supplements an algorithm which performs clustering based on the (dis)similarity matrix.

Algorithm 2.15 Auto-grouping: Clustering

```
Input:  w-Correlation matrix [$\rho_{ij}^{(\mathbf{w})}$] between reconstructed components, number of groups.
Output: Groups of components.
1: Use a method of cluster analysis to the dissimilarity measure defined by $1 - |\rho_{ij}^{(\mathbf{w})}|$.
2: Obtain the given number of groups from the results of cluster analysis.
```

The algorithm of component identification based on frequency characteristics of the components is implemented in two versions. In the first version, each frequency interval is considered separately and the desired components are selected by comparing the values of (2.20) to a given threshold. For simplicity, we formulate Algorithm 2.16 for one frequency interval only. The second version, Algorithm 2.17, uses the set of frequency intervals simultaneously.

Algorithm 2.16 Auto-grouping: Frequency ranges, by the threshold

```
Input: Frequency range $[\omega_1, \omega_2)$, threshold $T_0$, group $I$, type of series: eigenvectors, factor vectors
    or reconstructed series.
Output: A group of components $J \subset I$.
1: For each series $\mathbb{Y}_i$, $i \in I$, the measure $T(\mathbb{Y}_i; \omega_1, \omega_2)$ given in (2.20) is calculated.
2: The resultant group $J$ consists of indices $i \in I$ such that $T(\mathbb{Y}_i; \omega_1, \omega_2) \geq T_0$.
```

Algorithm 2.17 Auto-grouping: Frequency ranges, by the maximal contribution

```
Input: Set of frequency ranges $[\omega_1^{(m)}, \omega_2^{(m)})$, $m = 1, \ldots, k$ (if the separating points $0 = \omega_0 <$
    $\omega_1 < \omega_2 < \ldots < \omega_k$ for frequencies are given, then $[\omega_1^{(m)}, \omega_2^{(m)}) = [\omega_{m-1}, \omega_m)$), group $I$,
    type of series $\mathbb{Y}_i$: eigenvectors, factor vectors or reconstructed series.
Output: Set of $k$ groups $J_m \subset I$, $I = \bigsqcup_m J_m$.
1: For each series $\mathbb{Y}_i$, $i \in I$, and each frequency interval $[\omega_1^{(m)}, \omega_2^{(m)})$, $m = 1, \ldots, k$, the measure
    $T(\mathbb{Y}_i; \omega_1^{(m)}, \omega_2^{(m)})$ given in (2.20) is calculated.
2: Each index $i$, $i \in I$, is referred to a group $J_{m_0}$ with the maximal value of the measures
    $T(\mathbb{Y}_i; \omega_1^{(m)}, \omega_2^{(m)})$, $m \in \{1, \ldots, k\}$. As a result, $k$ groups are formed.
```

2.7.3 Automatic Grouping in Rssa

2.7.3.1 Description of Functions

Let us outline the main arguments of `grouping.auto` for Algorithm 2.15 in typical function calls (the two grouping calls below are equivalent):

```
g <- grouping.auto(s, nclust = 2, groups = 1:20,
                   method = "complete", grouping.method = "wcor")
g <- grouping.auto.wcor(s, nclust = 2,
                        groups = 1:20, method = "complete")
```

Arguments

`s` is an `ssa` object holding the full one-dimensional SSA (Basic SSA, Toeplitz SSA, SSA with projection, Shaped SSA, DerivSSA, Iterative O-SSA) decomposition.

`grouping.method` is a method for automatic grouping.

`groups` is a list of groups, which is coerced to a vector with component numbers to obtain the elementary reconstructed components and calculate the **w**-correlation matrix for them.

`nclust` is a number of clusters.

`method` determines the way of cluster amalgamation; `method` is a parameter of the `R` function `hclust` from the STATS package, which performs the hierarchical cluster analysis.

The result of the function `grouping.auto.wcor` can be depicted by the function call `plot(g)` in the form of a hierarchical tree.

For application of Algorithms 2.16 and 2.17, the typical call is

```
gp <- grouping.auto(s, groups = 1:20, base = "series",
                    freq.bins = list(0.01,0.02),
                    threshold = 0.8,
                    grouping.method = "pgram")
```

Arguments

`s` is an `ssa` object holding the full one-dimensional SSA (Basic SSA, Toeplitz SSA, SSA with projection, Shaped SSA, DerivSSA, Iterative O-SSA) decomposition.

`grouping.method` is a method for automatic grouping.

`groups` is a list of indices of elementary components for grouping, which is coerced to a vector.

`base` is an input for the periodogram analysis: elementary reconstructed series (`"series"`), eigenvectors (`"eigen"`), or factor vectors (`"factor"`).

`freq.bins` could be: a single integer larger than 1, which defines the number of intervals of equal length dividing the frequency range $[0, 1/2]$; a vector of frequency separating points (of length ≥ 2); a list of frequency ranges. For each range, if only one frequency is indicated, then it will be used as the upper bound, while the lower bound will be zero. If the frequency intervals, given by the parameter `freq.bins`, are named, then the resultant groups will take these names.

`threshold` is a threshold for frequency contributions. The value `threshold`=0 indicates that Algorithm 2.17 will be used.

`method` is a method of interpolation (`"const"` or `"linear"`) of the periodogram values, which are initially given on the regular grid.

The result of `grouping.auto.pgram` (that is, of `grouping.auto`, where the parameter `grouping.method = "pgram"`) can be depicted in the form of component contributions T by the call

```
plot(gp, superpose = TRUE, order = TRUE)
```

Here `superpose` is logical and indicates whether to plot contributions for all intervals on one panel. If the parameter `order` is `TRUE`, then the depicted component contributions are ordered by their values.

2.7.3.2 Typical Code

Let us demonstrate how to replicate the examples of automatic grouping taken from Golyandina and Zhigljavsky (2013; Section 2.4.5) by means of the RSSA package.

Grouping Based on **w**-Correlations

Let us consider the "White dwarf" data and apply clustering to the corresponding **w**-correlation matrix for window length $L = 100$. To do that we use Fragment 2.7.1.

Fragment 2.7.1 ("White dwarf": Auto Grouping by Clustering)

```
> data("dwarfst", package = "ssabook")
> s <- ssa(dwarfst, L = 100)
> g <- grouping.auto(s, grouping.method = "wcor",
+                    method = "average", nclust = 2)
> print(g[[1]])
 [1]  1  2  3  4  5  6  7  8  9 10 11
> plot(wcor(s, groups = 1:30), scales = list(at = c(1, 11, 30)))
> plot(reconstruct(s, groups = g),
+      add.residuals = FALSE,
+      plot.method = "xyplot", superpose = FALSE)
```

The **w**-correlations between the 30 leading elementary reconstructed components are depicted in Fig. 2.29. We can deduce from this figure that the components can be partitioned into two groups, signal (ET1–11) and noise (ET12–100). Hierarchical clustering with average linkage into two groups provides a proper split into two clusters with the first cluster consisting exactly of ET1–11. Reconstruction with automatic grouping is presented in Fig. 2.30.

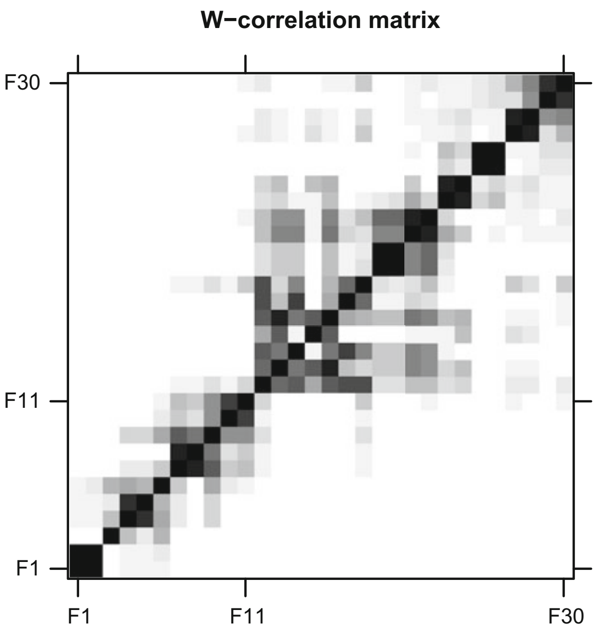


Fig. 2.29 "White dwarf": **w**-Correlation matrix, $L = 100$

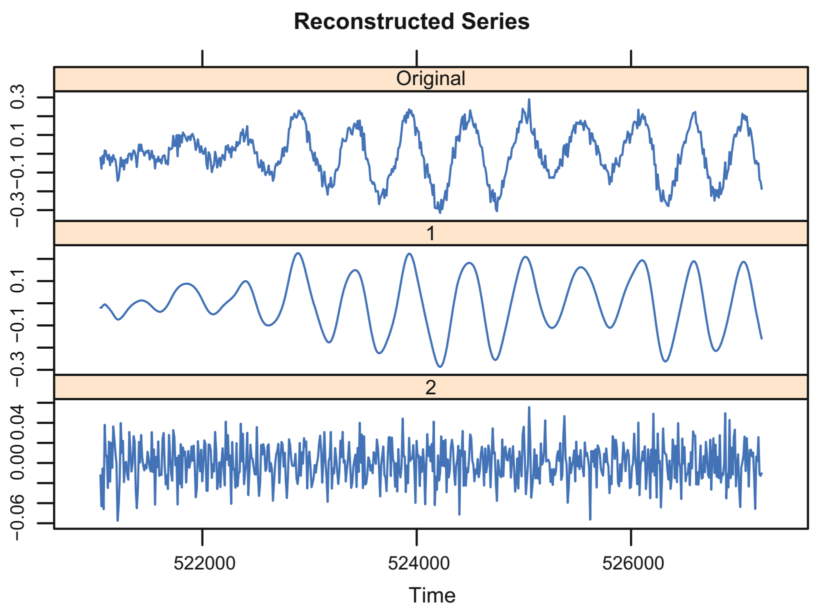


Fig. 2.30 “White dwarf”: Decomposition with automatic grouping performed by clustering

Identification of Trend

Let us consider the “Production” example. Fragment 2.7.2 demonstrates how to choose the threshold and how to extract trends of different forms by means of the frequency approach. We consider two frequency ranges, [0, 1/240] and [0, 1/24]. To understand what is a reasonable value of the threshold, we first choose an arbitrary small threshold to draw the plot of component contributions in the chosen frequency ranges; we reorder the components by their contributions. Figure 2.31 shows that the threshold should be between the contributions of the 9th and 10th components. We choose the contribution of the 9-th component, which is approximately equal to 0.89, as a new threshold.

Fragment 2.7.2 (“Production”: Auto Grouping by Frequency Analysis)

```
> data("oilproduction", package = "ssabook")
> s <- ssa(oilproduction, L = 120)
> plot(s, type = "vectors", vectors = "factor", idx = 1:12)
> g0 <- grouping.auto(s, base = "series",
+                     freq.bins = list(Tendency = 1/240,
+                                      Trend = 1/24),
+                     threshold = 0.1)
> plot(g0, order = TRUE, type = "b")
> contrib <- attr(g0, "contributions")[, 2]
> print(thr <- sort(contrib, decreasing = TRUE)[9])
      8
```

```
0.861955
> g <- grouping.auto(s, base = "series",
+                    freq.bins = list(Tendency = 1/240,
+                                     Trend = 1/24),
+                    threshold = thr)
> print(g[[1]])
[1] 1 2
> print(g[[2]])
[1]  1  2  3  6  8 11 12 17 18
> plot(reconstruct(s, groups = g),
+      add.residuals = FALSE,
+      plot.method = "xyplot", superpose = TRUE)
```

If we choose $\omega_0 = 1/24$ and $T_0 = 0.89$, then the described procedure identifies ET1–3,6,8,11,12,17,18; a rough trend is thus identified accurately enough, see Fig. 2.33, red line. Figure 2.32 with factor vectors explains the result. Indeed, the detected factor vectors are slowly varying. The function `grouping.auto` allows to consider several frequency intervals. In this case, one should set a threshold for each frequency bin. In the code of Fragment 2.7.2, by rules of `R`, `threshold = thr` is equivalent to `threshold = list(thr, thr)`. For convenience, the implementation of `grouping.auto` allows to write `freq.bins = list(1/240,1/24)` instead of `freq.bins = list(c(0,1/240),c(0,1/24))`. Figure 2.33 shows two trends of different forms obtained by means of different frequency intervals.

If we were interested in the general tendency only, then the measure T with $\omega_0 = 1/240$ and the threshold $T_0 = 0.89$ identifying one leading component would be sufficient.

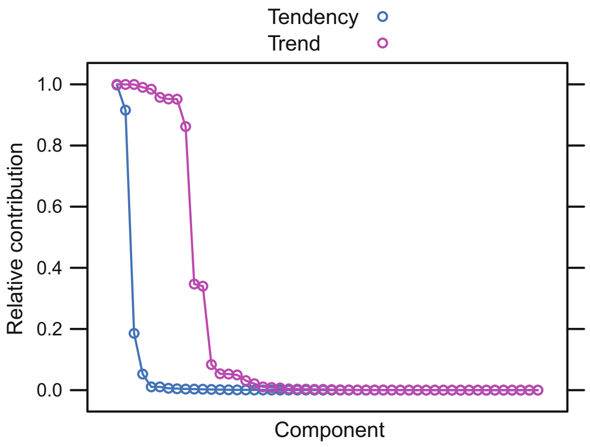


Fig. 2.31 “Production”: Ordered frequency contributions of factor vectors, $L = 120$

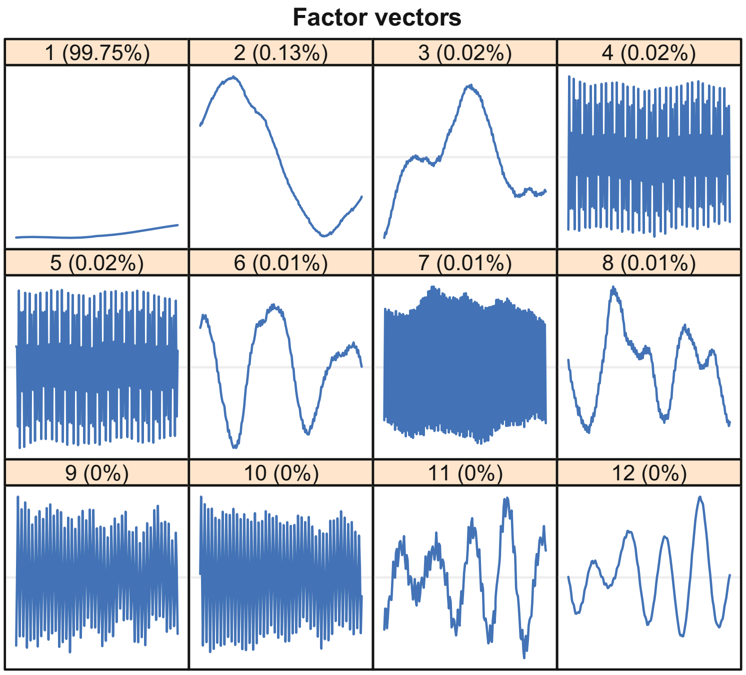


Fig. 2.32 "Production": Factor vectors, $L = 120$

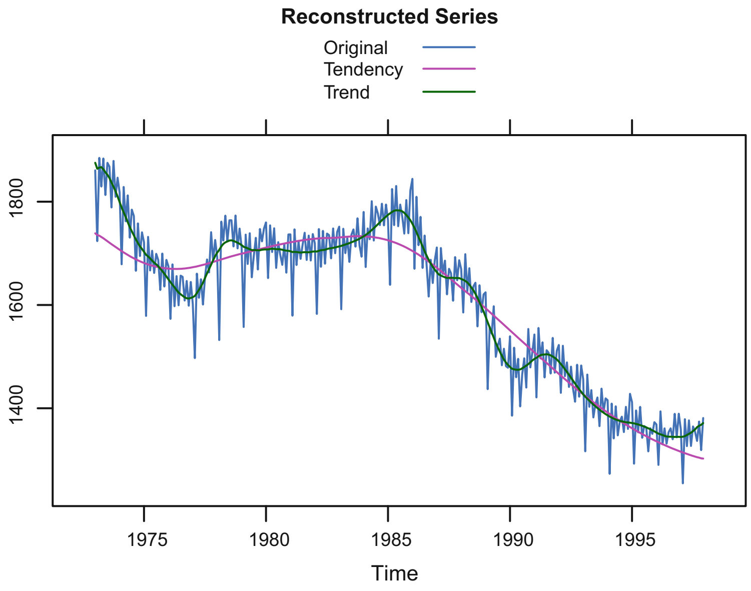


Fig. 2.33 "Production": Two extracted trends of different resolution, automatic grouping by frequencies

2.8 Case Studies

2.8.1 Extraction of Trend and Oscillations by Frequency Ranges

A decomposition on interpretable series components may differ from a decomposition on components with different frequency ranges but sometimes these decompositions can be similar. For example, extraction of a trend can be considered as a smoothing, i.e., extraction of a slowly-varying series component with a frequency range close to zero.

In Sect. 2.1.5.3, a typical decomposition of the series of sales of fortified wines in Australia into a sum of a trend, a seasonal component and a noise is shown.

Let us introduce an example of frequency decomposition. Consider the series "Tree rings" (tree ring width, annual, 1282–1950).

Fragment 2.8.1 makes a decomposition of the series "Tree rings" into components from the following frequency ranges: $[0, 0.1)$, $[0.1, 0.2)$, $[0.2, 0.3)$, $[0.3, 0.4)$, and $[0.4, 0.5]$. In the code, the last frequency is depicted as `+Inf` but in fact the upper bound is 0.5.

Fragment 2.8.1 ("Tree rings": Frequency Decomposition)

```
> data("dftreering", package = "ssabook")
> L <- 300
> s.tree <- ssa(dftreering, L = L, neig = L)
> g.tree <- grouping.auto(s.tree, base = "series",
+                       freq.bins = c(0.1, 0.2, 0.3, 0.4, +Inf))
> r.tree <- reconstruct(s.tree, groups = g.tree)
> plot(r.tree, add.residuals = FALSE, add.original = TRUE,
+      plot.method = "xyplot")
> specs <-
+   lapply(r.tree, function(x) spectrum(x, plot = FALSE)$spec)
> w.tree <- seq(0, length.out = length(specs$F1),
+               by = 1/length(dftreering))
> xyplot(F1 + F2 + F3 + F4 + F5 ~ w.tree, data = specs,
+        superpose = FALSE, type = "l", xlab = NULL, ylab = NULL,
+        auto.key = list(lines = TRUE, points = FALSE,
+                        column = 5))
```

The resultant decomposition is depicted in Fig. 2.34. Since the given frequency ranges split the whole range $[0, 0.5]$, we obtain a full decomposition of the original series.

Figure 2.35 shows spectrums of the series components depicted in Fig. 2.34. It can be seen that the frequency ranges of the series components are almost disjoint. Since the window length L makes an influence on the resolution of the method (see, e.g., Golyandina and Zhigljavsky (2013; Section 2.9)), the intersection of frequency ranges increases for small window lengths.

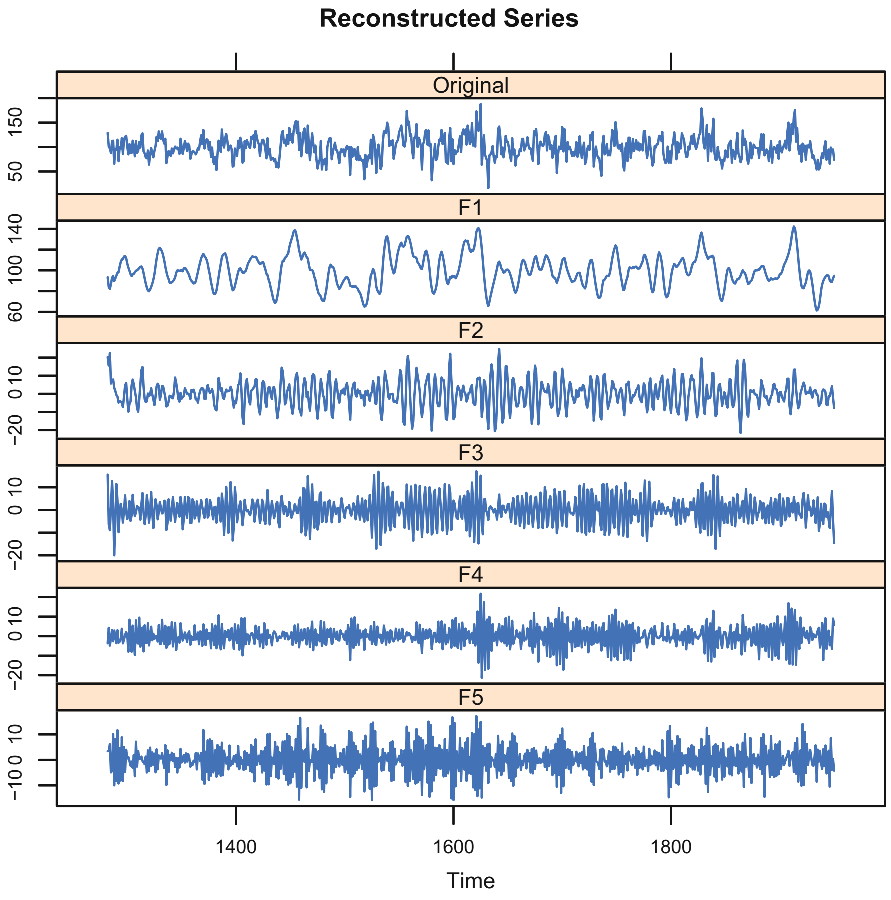


Fig. 2.34 "Tree rings": Frequency decomposition

2.8.2 *Trends in Short Series*

Let us consider the series "FORT" from the dataset "Australian Wines" with monthly sales. The first 120 points of the series are depicted in Fig. 2.36.

The series length is long enough to obtain weak separability; therefore, we will consider short subseries to demonstrate the ability of Iterative O-SSA to improve separability.

We choose the window length $L = 18$ to make the difference between Basic SSA and Iterative O-SSA clearly visible on the figures, although the relation between accuracies of the considered methods is very similar for other choices of the window length. Let us consider the subseries consisting of the points from 30th to 72th.

Let us start with Basic SSA. ET1 is identified as corresponding to trend; other components are produced by seasonality and noise (we do not include their plots).

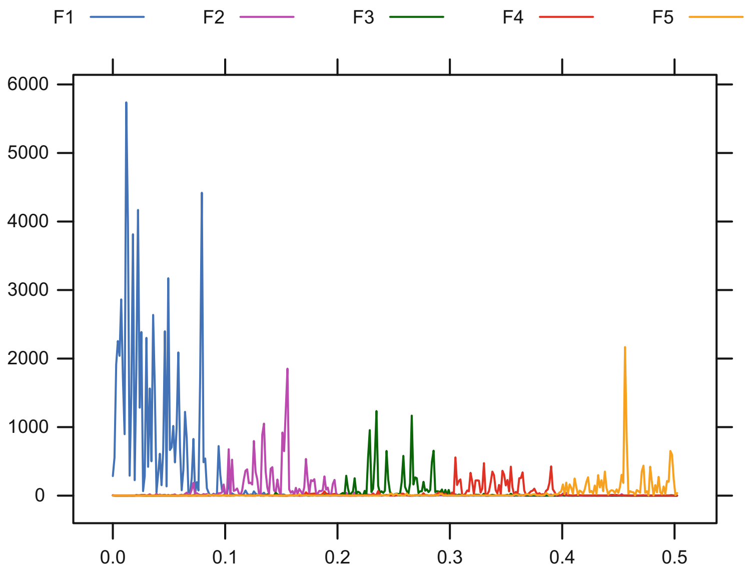


Fig. 2.35 “Tree rings”: Periodograms of the series components

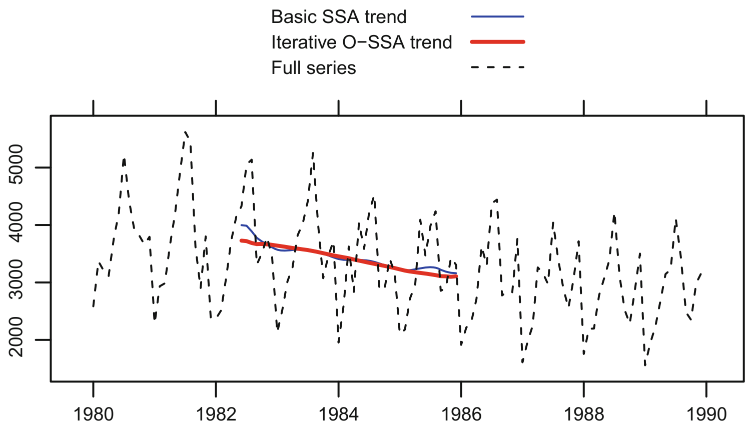


Fig. 2.36 “FORT”: Trend reconstruction by Iterative O-SSA for the subseries consisting of points 30–72

One can see in Fig. 2.36 that the reconstructed trend is slightly mixed with the seasonality and follows the seasonal component at the ends of the series.

To apply Iterative O-SSA, we should choose a group of elementary components containing the trend components and approximately separated from the residual. Let it be ET1–7. Thus, we apply one iteration of O-SSA to the refined groups ET1 and ET2–7. Since the trend has much larger contribution than the residual, we consider Iterative O-SSA with no sigma-correction. The result of reconstruction is much more relevant, see Fig. 2.36. This reconstruction is obtained by means of the code of Fragment 2.8.2.

Fragment 2.8.2 ("FORT": Basic SSA and Iterative O-SSA Trends)

```
> data("AustralianWine", package = "Rssa")
> Nfull <- 120
> wine <- window(AustralianWine,
+                end = time(AustralianWine)[Nfull])
> fort_sh <- window(wine[, "Fortified"],
+                   start = c(1982, 6), end = c(1985, 12))
> ss_sh <- ssa(fort_sh, L = 18)
> res_ssa_sh <- reconstruct(ss_sh, groups = list(1, 2:7))
> iss_sh <- iossa(ss_sh, nested.groups = list(1, 2:7),
+                 kappa = 0, maxiter = 1, tol = 1e-5)
> res_issa_sh <- reconstruct(iss_sh, groups = iss_sh$iossa.groups)
> theme <- simpleTheme(col = c("blue", "red", "black"),
+                      lwd = c(1, 2, 1),
+                      lty = c("solid", "solid", "dashed"))
> xyplot(cbind(res_ssa_sh$F1, res_issa_sh$F1, wine[, "Fortified"]),
+        superpose = TRUE,
+        xlab = "", ylab = "", type = "l", lwd = c(1, 2, 1),
+        col = c("blue", "red", "black"),
+        auto.key = list(text = c("Basic SSA trend",
+                                 "Iterative O-SSA trend",
+                                 "Full series"),
+                        type = c("l", "l", "l"),
+                        lines = TRUE, points = FALSE),
+        par.settings = theme)
```

2.8.3 *Trend and Seasonality of Complex Form*

Let us analyze the time series "MotorVehicle" which contains monthly data of total domestic and foreign car sales in the USA, from 1967 to 2012, January. The total series length is 541. This time series was investigated in Golyandina and Korobeynikov (2013) by means of Sequential SSA.

Figure 2.37 shows that the shape of the trend is complex. From the viewpoint of SSA, complexity of a trend means that it can only be approximated by a time series of a large rank and therefore it is decomposed into a large number of elementary components, if a large enough window length was chosen. Therefore, there are high odds that the trend decomposition components are going to be mixed with seasonal components in Basic SSA.

Since seasonal components are approximately orthogonal to slowly-varying components, we can consider the problem of mixing as a problem of lack of strong separability.

RSSA offers several options helping to avoid mixing. The first option is to use Sequential SSA as done in Golyandina and Korobeynikov (2013) (see Sect. 2.1.3.2, where the idea of Sequential SSA is briefly described).

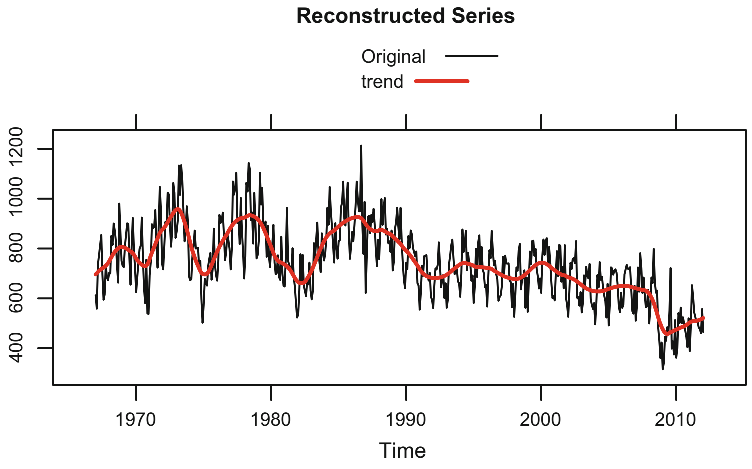


Fig. 2.37 "MotorVehicle": Trend extracted by Basic SSA with small window length $L = 12$

First, let us extract a trend. Since for a trend of such a difficult shape its extraction is similar to smoothing, we start with choosing a minimally possible window length, which in this case is $L = 12$. The reason for this choice of window length is similar to that in the moving averaging procedure: for smoothing a time series containing a periodic component, the window length should be divisible by the period. Then, the residual is decomposed with a large window length 264 to extract the seasonality, since the seasonal component can be considered as a sum of exponentially-modulated harmonics and therefore has a rank not exceeding 11.

Fragment 2.8.3 shows how Sequential SSA can be performed (see the explanation of the component choice in Golyandina and Korobeynikov (2013)). Figure 2.37 shows the resultant decomposition.

Fragment 2.8.3 ("MotorVehicle": Decomposition by Sequential SSA)

```
> data("MotorVehicle", package = "Rssa")
> s1 <- ssa(MotorVehicle, L = 12)
> res1 <- reconstruct(s1, groups = list(trend = 1))
> trend <- res1$trend
> plot(res1, add.residuals = FALSE, plot.type = "single",
+      col = c("black", "red"), lwd = c(1, 2),
+      plot.method = "xyplot", superpose = TRUE)
> res.trend <- residuals(res1)
> s2 <- ssa(res.trend, L = 264)
> res2 <- reconstruct(s2, groups = list(seasonality = 1:10))
> seasonality <- res2$seasonality
> res <- residuals(res2)
> # The resultant decomposition consists of
> # trend, seasonality and residual
```

Sequential SSA consists of a repeated application of one of the SSA methods, for example, Basic SSA. Let us demonstrate the use of another two-step approach, DerivSSA of Sect. 2.5, using a nested decomposition. In DerivSSA, a signal subspace should be estimated at the first step and then an additional rotation is performed in the signal subspace to avoid a mixture. Fragment 2.8.4 shows how DerivSSA can be applied. The signal subspace was detected by the analysis of eigenvectors and the **w**-correlation matrix. Identification of components of the refined decomposition, which was obtained by means of DerivSSA, is performed in the same way as it is done in Basic SSA.

Fragment 2.8.4 ("MotorVehicle": Decomposition by DerivSSA)

```
> data("MotorVehicle", package = "Rssa")
> s <- ssa(MotorVehicle, L = 264)
> sf <- fossa(s, nested.groups = 1:19)
> rf <- reconstruct(sf, groups =
+                     list(seasonality = 1:10, trend = 11:19))
> plot(rf, plot.method = "xyplot", superpose = TRUE,
+      add.residuals = FALSE,
+      col = c("black", "darkgreen", "red"), lwd = c(1, 1, 2))
> p<- parestimate(sf, groups = list(1:10),
+                 method = "esprit")
> print(p$period[seq(1, 10, 2)], digits = 3)
[1]  3.00 12.01  2.40  5.99  4.02
```

The decomposition results are similar. We present the results of full DerivSSA decomposition (we have used the version with normalization, see Algorithm 2.11) into trend, seasonal component and a noise in Fig. 2.38.

2.8.4 *Finding Noise Envelope*

Here we demonstrate how to estimate the variance of a heterogeneous noise. The procedure is based on the following two observations: first, the variance of noise is equal to the expectation of the squared noise values, and second, for a stochastic process the trend is its expectation. Therefore, the variance can be estimated as the trend of squared residuals. This trend can be extracted by SSA with a small window length and reconstructed by the leading eigentriple. The choice of the window length influences the level of detail with which we see the extracted trend. For the "MotorVehicle" data, window length $L = 30$ provides an appropriate trend. The result of Fragment 2.8.5 is depicted in Fig. 2.39 which shows the residuals and the standard deviation bounds.

Fragment 2.8.5 ("MotorVehicle": Finding Noise Envelope)

```
> resf <- residuals(rf)
> s.env <- ssa(resf^2, L = 30)
> rsd <- sqrt(reconstruct(s.env, groups = list(1))$F1)
> xyplot(resf + rsd + (-rsd) ~ time(resf), type = "l")
```

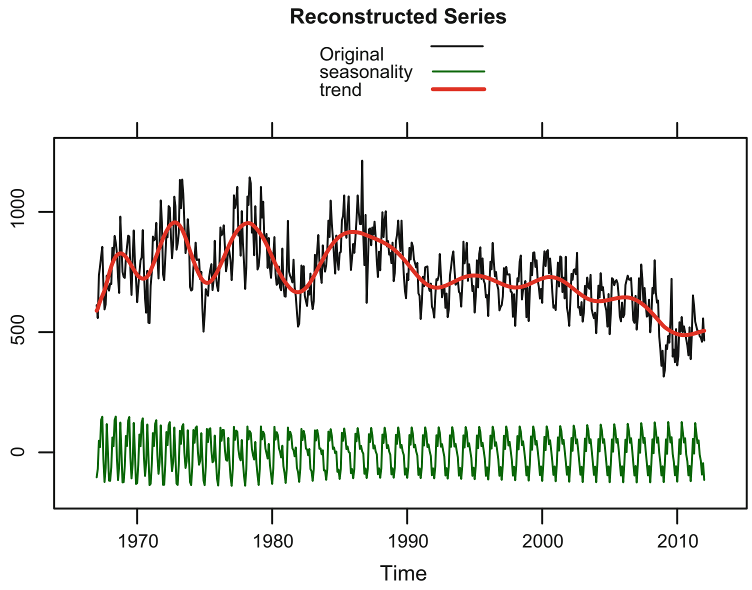


Fig. 2.38 “MotorVehicle”: Decomposition by DerivSSA with $L = 264$

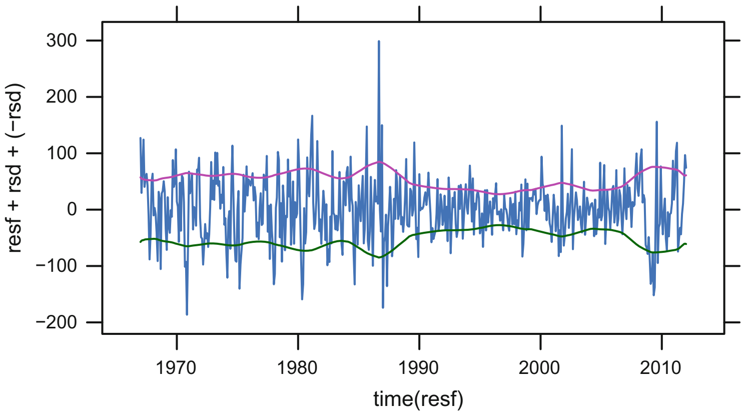


Fig. 2.39 “MotorVehicle”: Residuals with envelopes

2.8.5 Elimination of Edge Effects

Let us consider “US Unemployment” data (monthly, 1948–1981, thousands) for male (20 years and over). The series length is $N = 408$. Since the series is long, we can expect weak separability between the trend and seasonality. For better weak separability we choose the window length equal to $L = N/2 = 204$, which is divisible by 12. Fragment 2.8.6 demonstrates how DerivSSA allows to improve the decomposition.

Fragment 2.8.6 ("US unemployment": Improvement by DerivSSA)

```
> data("USUnemployment", package = "Rssa")
> ser <- USUnemployment[, "MALE"]
> Time <- time(ser)
> L = 204
> ss <- ssa(ser, L = L, svd.method = "eigen")
> res<- reconstruct(ss, groups =
+                      list(c(1:4, 7:11), c(5, 6, 12, 13)))
> trend <- res$F1
> seasonality <- res$F2
> w1 <- wcor(ss, groups = 1:30)
> fss <- fossa(ss, nested.groups =
+                 list(c(1:4, 7:11), c(5, 6, 12, 13)),
+               gamma = Inf)
> fres <- reconstruct(fss, groups = list(5:13, 1:4))
> ftrend <- fres$F1
> fseasonality <- fres$F2
> theme1 <- simpleTheme(col = c("grey", "blue","red"),
+                       lwd = c(2, 1, 1),
+                       lty = c("solid", "solid", "solid"))
> theme2 <- simpleTheme(col = c("blue", "red"), lwd = c(1, 1),
+                       lty = c("solid", "solid"))
> p1 <- xyplot(ser + trend + ftrend ~ Time,
+        xlab = "", ylab = "", type = "l", lwd = c(2, 1, 1),
+        col = c("grey", "blue","red"),
+        auto.key = list(text = c("Full series",
+                                 "Basic SSA trend",
+                                 "DerivSSA trend"),
+                        type = c("l", "l", "l"),
+                        lines = TRUE, points = FALSE),
+        par.settings = theme1)
> p2 <- xyplot(seasonality + fseasonality ~ Time,
+        xlab = "", ylab = "", type = "l", lwd = c(2, 1),
+        col = c("blue", "red"),
+        auto.key = list(text = c("Basic SSA seasonality",
+                                 "DerivSSA seasonality"),
+                        type = c("l", "l"),
+                        lines = TRUE, points = FALSE),
+        par.settings = theme2)
> plot(p1, split = c(1, 1, 1, 2), more = TRUE)
> plot(p2, split = c(1, 2, 1, 2), more = FALSE)
```

Basic SSA does not separate the trend and seasonality for this time series. It is another very typical situation that if trend has a complex form, then trend components are mixed with the seasonality components and therefore the so-called Sequential SSA was recommended (Golyandina et al. 2001; Section 1.7.3). However, this is also the case when DerivSSA is able to help.

We apply DerivSSA (the version with normalization) to the group ET1–13 that can be related to the signal. DerivSSA separates different frequencies so that components with higher frequencies become the leading ones. Since the low-frequency components in the considered series have large contribution, the weight

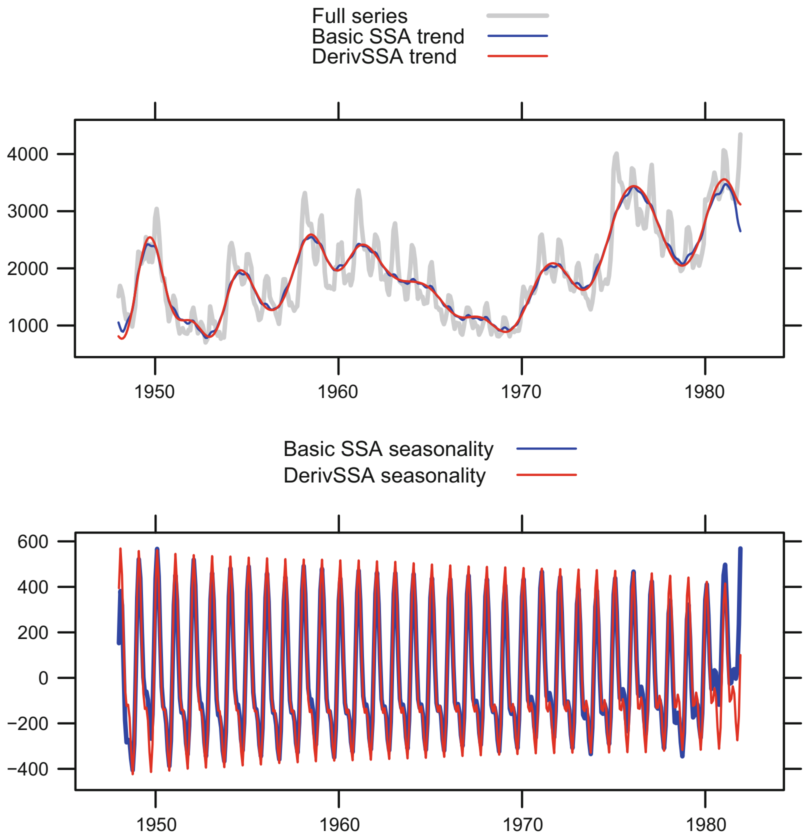


Fig. 2.40 "US unemployment": Decompositions by Basic SSA and DerivSSA

of derivatives should be large in order to make the seasonal components leading; we take `gamma = Inf` to exclude non-derivative part from the decomposed matrix.

Figure 2.40 depicting the DerivSSA reconstructions of the trend and the seasonality confirms that DerivSSA visibly improves the reconstruction accuracy, especially at both ends of the series. It is shown in Golyandina and Shlemov (2015) how to obtain a similar effect by means of Iterative O-SSA.

2.8.6 Extraction of Linear Trends

Here we consider the example "Hotel" following Golyandina et al. (2001; Section 1.7.1). We extract trend from a short subseries of length n. Then we compare predictions by linear regressions constructed from the series itself and constructed from the trends extracted by SSA. More detailed comparison of SSA and regression for simulated examples can be found in Golyandina and Shlemov (2017).

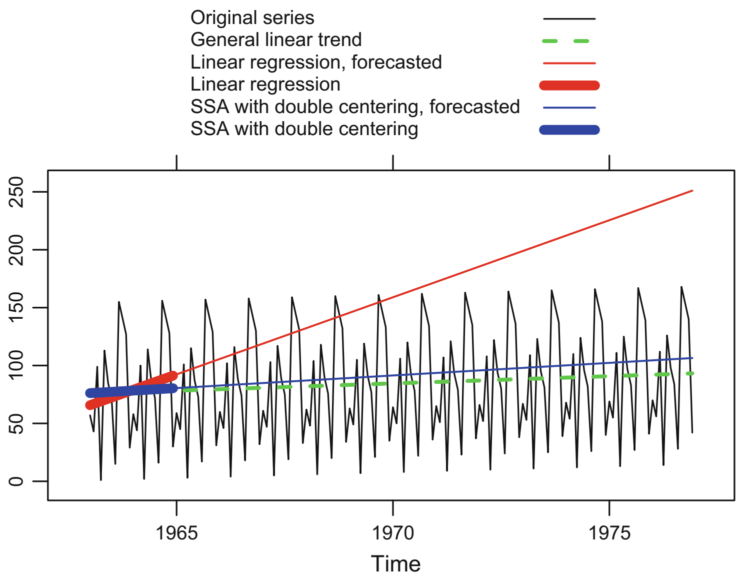


Fig. 2.41 "Hotel": SSA with projection, linear trend detection

Consider two cases: $n = 24$ (Fragment 2.8.7) and $n = 30$ (Fragment 2.8.8). For $n = 24$, the best separability of the linear trend from the residual is achieved by SSA with double centering (a particular case of SSA with projection). This is because we can choose L and K approximately divisible by the period 12. One can see in Fig. 2.41 that SSA linear trend (blue) is very close to a linear trend constructed by the whole long time series (green). Linear regression line (red) gives an approximation of the trend which is much worse than for SSA.

However, for $n = 30$, there is no appropriate window length providing desired orthogonalities. Therefore, direct SSA with double centering is not so successful in this case but to improve separability we can apply Iterative O-SSA. The result shown in Fig. 2.42 is rather good.

Fragment 2.8.7 ("Hotel": SSA with Projection, Linear Trend Detection)

```
> data("hotel", package = "ssabook")
> len <- length(hotel)
> n <- 24
> hotel.2years <- window(hotel, end = time(hotel)[n])
> sp <- ssa(hotel.2years, L = 12,
+           row.projector = "center",
+           column.projector = "center")
> r <- reconstruct(sp, groups = list(trend = 1:2))
> hotel.2years.data <- data.frame(x = 1:n, y = hotel.2years)
> fit.2years <- lm(y ~ x, data = hotel.2years.data)
> fit.2years.continued <- predict(fit.2years,
+                                 newdata = data.frame(x = 1:len))
```

```
> hotel.data <- data.frame(x = 1:len, y = hotel)
> fit <- lm(y ~ x, data = hotel.data)
> fit.rec <- lm(r$trend ~ x, data = hotel.2years.data)
> fit.rec.continued <- predict(fit.rec,
+                              newdata = data.frame(x = 1:len))
> xyplot(cbind(hotel,
+              predict(fit),
+              fit.2years.continued,
+              ts(predict(fit.2years),
+                 start = c(1963, 1), freq = 12),
+              fit.rec.continued,
+              ts(predict(fit.rec),
+                 start = c(1963, 1), freq = 12)),
+        superpose = TRUE,
+        type = "l", ylab = "",
+        lty = c(1, 2, 1, 1, 1, 1),
+        lwd = c(1, 2, 1, 5, 1, 5),
+        col = c("black", "green", "red", "red",
+                "blue", "blue"),
+        auto.key =
+          list(text = c("Original series",
+                        "General linear trend",
+                        "Linear regression, forecasted",
+                        "Linear regression",
+                        "SSA with double centering, forecasted",
+                        "SSA with double centering"),
+               type = c("l", "l", "l", "l", "l", "l"),
+               lines = TRUE, points = FALSE))
```

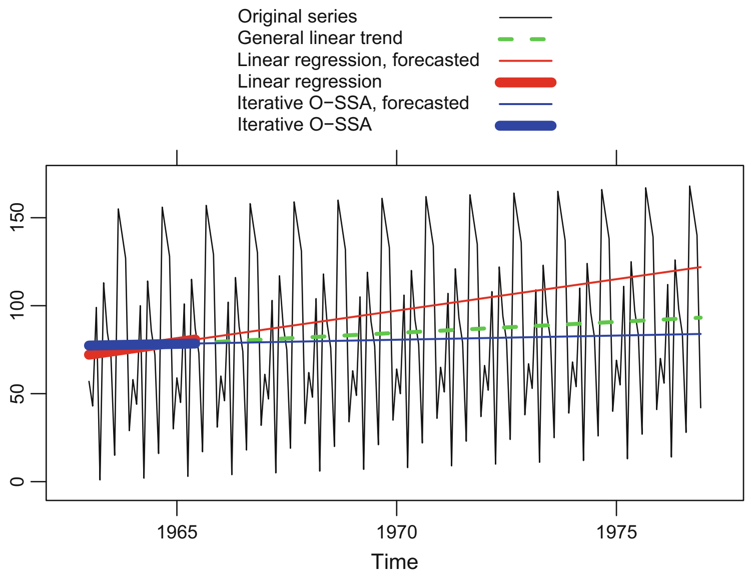


Fig. 2.42 "Hotel": Iterative O-SSA, linear trend detection

Fragment 2.8.8 (“Hotel”: Iterative O-SSA, Linear Trend Detection)

```
> n <- 30
> hotel.2years <- window(hotel, end = time(hotel)[n])
> s <- ssa(hotel.2years, L = 12)
> ios <- iossa(s, nested.groups = list(1, 2:5))
> r <- reconstruct(ios, groups = list(trend = 1))
> hotel.2years.data <- data.frame(x = 1:n, y = hotel.2years)
> fit.2years <- lm(y ~ x, data = hotel.2years.data)
> fit.2years.continued <- predict(fit.2years,
+                                 newdata = data.frame(x = 1:len))
> hotel.data <- data.frame(x = 1:len, y = hotel)
> fit <- lm(y ~ x, data = hotel.data)
> fit.rec <- lm(r$trend ~ x, data = hotel.2years.data)
> fit.rec.continued <- predict(fit.rec,
+                              newdata = data.frame(x = 1:len))
> xyplot(cbind(hotel,
+              predict(fit),
+              fit.2years.continued,
+              ts(predict(fit.2years),
+                 start = c(1963, 1), freq = 12),
+              fit.rec.continued,
+              ts(predict(fit.rec),
+                 start = c(1963, 1), freq = 12)),
+        superpose = TRUE,
+        type = "l", ylab = "",
+        lty = c(1, 2, 1, 1, 1, 1),
+        lwd = c(1, 2, 1, 5, 1, 5),
+        col = c("black", "green", "red", "red",
+                "blue", "blue"),
+        auto.key =
+            list(text = c("Original series",
+                          "General linear trend",
+                          "Linear regression, forecasted",
+                          "Linear regression",
+                          "Iterative O-SSA, forecasted",
+                          "Iterative O-SSA",
+                 type = c("l", "l", "l", "l", "l", "l"),
+                 lines = TRUE, points = FALSE))
```

2.8.7 Automatic Decomposition

Let us consider an automatic identification based on frequency characteristics of the elementary series components (Fragment 2.8.9). This way of identification is very well suited for the problem of trend extraction. For the extraction of periodicities, we would recommend a more sophisticated approach described in Alexandrov and Golyandina (2005) and based on an idea from Vautard et al. (1992).

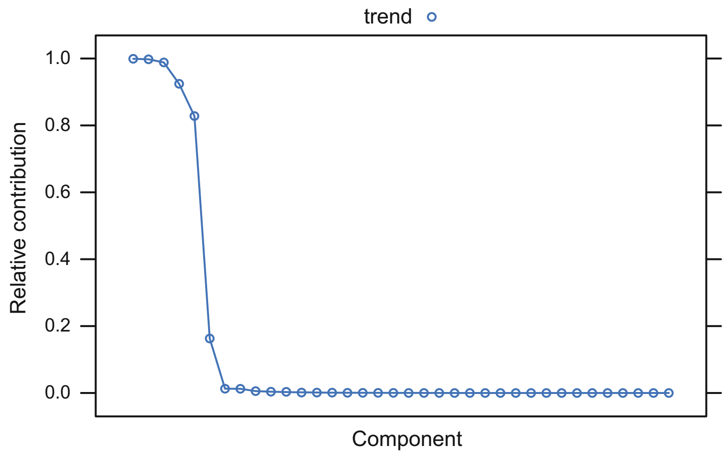


Fig. 2.43 "PayNSA": Contributions of trend components

The data "PayNSA" contains monthly numbers of all employees, total nonfarm payrolls, thousands of persons. Since the trend is complex, we will use Sequential SSA.

To extract a trend we take a small window length $L = 36$ and a frequency range $[0, 0.06]$ noting that the frequency $1/12 \approx 0.0833$ is related to seasonality. Figure 2.43 shows ordered contributions of these frequency ranges. A sharp drop after the fifth ordered component is clearly seen. Therefore, we choose the threshold equal to 0.7. The extracted trend is depicted in Fig. 2.44 together with the original series.

After performing trend extraction, let us investigate the residual. To extract the seasonal component, we choose the frequency range consisting of small intervals containing the frequencies $1/12$, $2/12$, $3/12$, $4/12$, $5/12$, and $6/12$. Figure 2.45 shows the contributions of the components in the initial order (right) and the contributions ordered by the magnitude of their values for each frequency range (left). In the right figure, one can see that the components with large contributions come in pairs with each pair corresponding to one frequency, except for a single component for the frequency $1/2$. The extracted seasonal component is presented in Fig. 2.46.

Note that if we subtract the seasonal component from the original time series, then we obtain the so-called seasonally adjusted component. To confirm that the seasonal component was really extracted, we depict the log-spectra of the original series and seasonally-adjusted series together (Fig. 2.47).

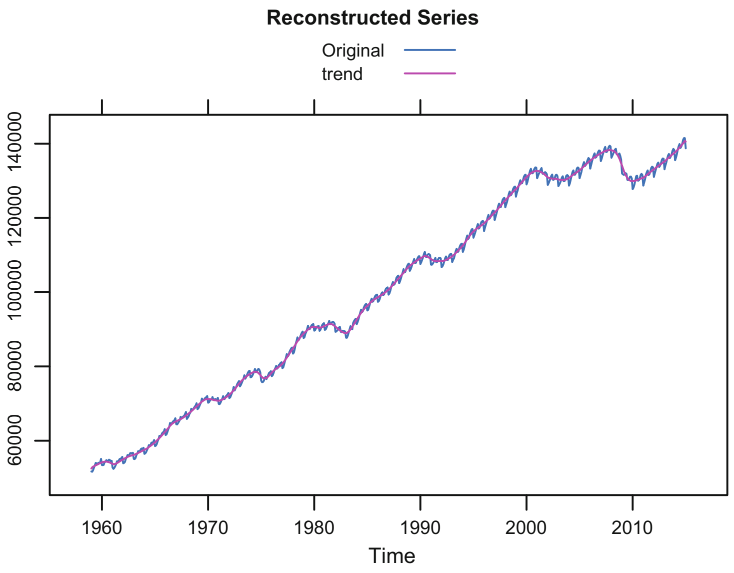


Fig. 2.44 “PayNSA”: Automatically identified trend

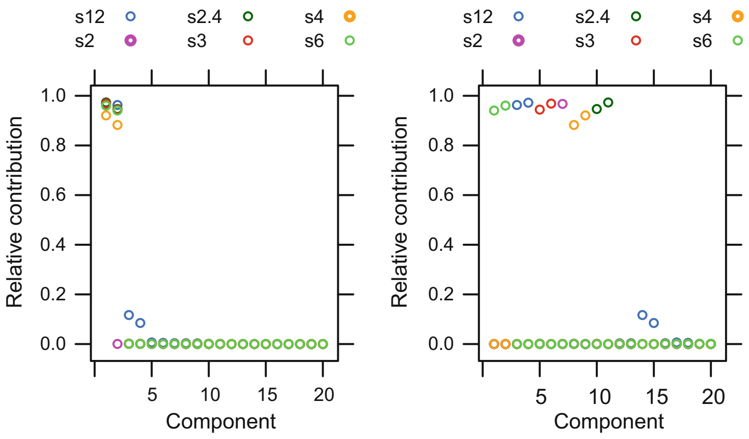


Fig. 2.45 “PayNSA”: Contributions of seasonal components, ordered by their values (left) and ordered by component numbers (right) for different frequency ranges

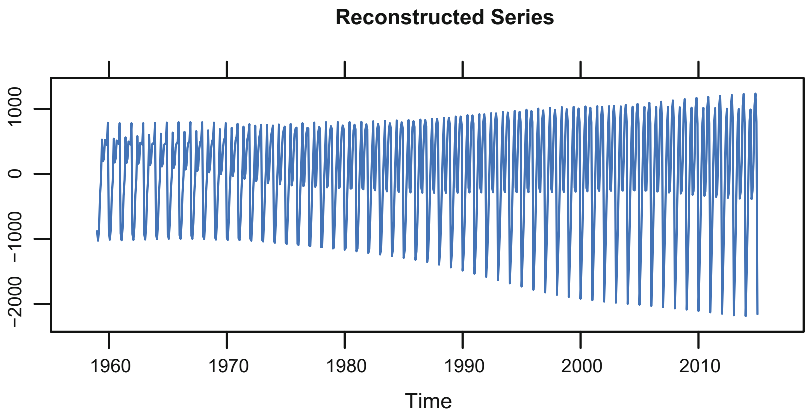


Fig. 2.46 "PayNSA": Automatically identified seasonal component

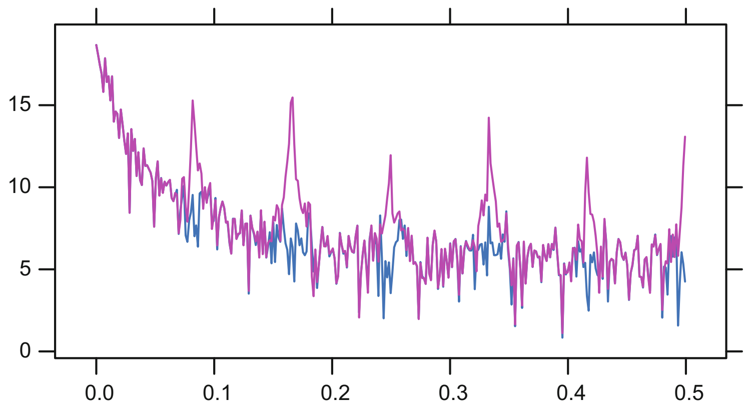


Fig. 2.47 "PayNSA": Log-periodogram of original and seasonally-adjusted series

Fragment 2.8.9 ("PayNSA": Automatically Identified Trend)

```
> data("paynsa", package = "ssabook")
> n <- 241
> pay <- window(paynsa, start = time(paynsa)[n])
> s <- ssa(pay, L = 36)
> g1 <- grouping.auto(s, base = "series",
+                     freq.bins = list(trend = 0.06),
+                     threshold = 0.7)
> print(g1$trend)
[1]  1  2  3  8 12
> plot(g1, order = TRUE, type = "b")
> r1 <- reconstruct(s, g1)
> plot(r1, plot.method = "xyplot", superpose = TRUE,
+      add.residuals = FALSE)
> s1 <- ssa(pay - r1$trend, L = 120)
> coef <- c(1 - 0.02, 1 + 0.02)
> freq.bins.seas = list(s12 = 1/12 * coef, s6 = 1/6 * coef,
+                       s4 = 1/4 * coef, s3 = 1/3 * coef,
+                       s2.4 = 1/2.4 * coef, s2 = 1/2 * coef)
> g3 <- grouping.auto(s1, base = "series", groups = 1:20,
+                     freq.bins = freq.bins.seas,
+                     threshold = list(0.6))
> p1 <- plot(g3, order = TRUE, scales = NULL,
+            auto.key = list(columns = 3))
> p2 <- plot(g3, order = FALSE, scales = NULL,
+            auto.key = list(columns = 3))
> plot(p1, split = c(1, 1, 2, 1), more = TRUE)
> plot(p2, split = c(2, 1, 2, 1), more = FALSE)
> r3 <- reconstruct(s1, groups = list(unlist(g3)))
> plot(r3, plot.method = "xyplot", add.residuals = FALSE,
+      add.original = FALSE)
> specNSA <- spectrum(pay - r3$F1, plot = FALSE)
> specSA <- spectrum(pay, plot = FALSE)
> w.pay <- seq(0, length.out = length(specNSA$spec),
+              by = 1/length(pay))
> xyplot(log(specNSA$spec) + log(specSA$spec) ~ w.pay,
+        type = "l", xlab = NULL, ylab = NULL)
```

2.8.8 Log-Transformation

As mentioned in Golyandina and Zhigljavsky (2013; Section 2.3.1.3), any multiplicative model can be considered as an additive model:

$$x_n = t_n(1 + s_n)(1 + r_n) = t_n + t_n s_n + (t_n + t_n s_n)r_n, \tag{2.21}$$

where t_n is a trend, s_n consists of regular oscillations, and r_n is a homogeneous noise. Since $t_n s_n$ can be considered as modulated regular oscillations and $(t_n + t_n s_n)r_n$ is a heterogeneous noise, SSA is able to perform such a decomposition.

On the other hand, one can consider the log-transformed series $\widetilde{x}_n = \ln x_n = \widetilde{t}_n + \widetilde{s}_n + \widetilde{r}_n$, where $\widetilde{t}_n = \log(t_n)$ is a trend, $\widetilde{s}_n = \log(1+s_n)$, and $\widetilde{r}_n = \log(1+r_n) \approx r_n$ is noise; if s_n is small, then $\widetilde{s}_n = \log(1 + s_n) \approx s_n$ can still be treated as oscillations.

Thus, if a series follows the multiplicative model (2.21), then SSA-family methods can be applied both to the initial series and to the log-transformed data to obtain a decomposition into a sum of a trend, oscillations, and noise.

If the log-transformation makes the structure of the series components simpler, then it can be recommended. For example, if the model is multiplicative and the time series trend has a complex form, then the log-transformation may eliminate the modulation. Note that if the trend is exponential, then the log-transformed trend becomes linear and therefore SSA with double centering can be recommended for its extraction.

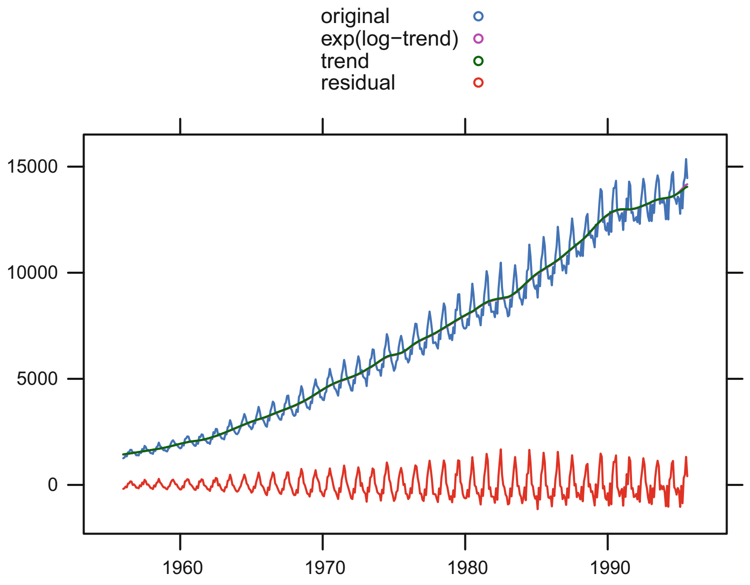


Fig. 2.48 “Elec”: Decomposition for initial and log-transformed data

Let us consider the series “Elec” of Australian monthly electricity production (Jan 1956 – Aug 1995).

In Fragment 2.8.10, the trend is estimated by the decomposition of the original data and of the log-transformed data. Figure 2.48 shows that the trend estimations almost coincide. This is typical in cases when the model is not purely multiplicative. In the present example, the multiplicativity breaks down after 1980 when the range of oscillations slightly decreases.

The log-transformation can only be applied when the original data is positive. After making the log-transformation we have to apply the exponential transformation to the series, which we obtain from the SSA analysis, in order to return to the initial scale. This makes the resulted data positive; this is a very attractive feature of the log-transformation in many applications.

Fragment 2.8.10 (“Elec”: Log-Transformation)

```
> data("elec", package = "fma")
> elec.log <- log(elec)
> Time <- time(elec)
> s <- ssa(elec, L = 12)
> r <- reconstruct(s, groups = list(trend = c(1)))
> sl <- ssa(elec.log, L = 12)
> rl <- reconstruct(sl, groups = list(trend = c(1)))
> xyplot(elec + exp(as.vector(rl$trend)) + r$trend +
+          (elec - r$trend) ~ Time,
+        superpose = TRUE, type ="l", ylab = NULL, xlab = NULL,
+        auto.key = list(text = c("original", "exp(log-trend)",
+                                 "trend", "residual")))
```

References

Alexandrov T, Golyandina N (2005) Automatic extraction and forecast of time series cyclic components within the framework of SSA. In: Proceedings of the 5th St.Petersburg workshop on simulation. St. Petersburg State University, pp 45–50

Andrew AL (1973) Eigenvectors of certain matrices. Linear Algebra Appl 7(2):151–162

Anderson E, Bai Z, Bischof C, Blackford L, Demmel J, Dongarra J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Sorensen D (1999) LAPACK Users' guide, 3rd edn. Society for Industrial and Applied Mathematics

De Moor BLR, Golub GH (1991) The restricted singular value decomposition: Properties and applications. SIAM J Matrix Anal Appl 12(3):401–425

Ghil M, Allen RM, Dettinger MD, Ide K, Kondrashov D, Mann ME, Robertson A, Saunders A, Tian Y, Varadi F, Yiou P (2002) Advanced spectral methods for climatic time series. Rev Geophys 40(1):1–41

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, Lomtev M (2016) Improvement of separability of time series in singular spectrum analysis using the method of independent component analysis. Vestnik St. Petersburg Univ Math 49(1):9–17

Golyandina N, Shlemov A (2015) Variations of singular spectrum analysis for separability improvement: Non-orthogonal decompositions of time series. Stat Interface 8(3):277–294

Golyandina N, Shlemov A (2017) Semi-nonparametric singular spectrum analysis with projection. Stat Interface 10(1):47–57

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, Pepelyshev A, Steland A (2012a) New approaches to nonparametric density estimation and selection of smoothing parameters. Comput Stat Data Anal 56(7):2206–2218

Harris T, Yan H (2010) Filtering and frequency interpretations of singular spectrum analysis. Physica D 239:1958–1967

Korobeynikov A, Larsen RM, Wu KJ, Yamazaki I (2016) SVD: Interfaces to various state-of-art SVD and eigensolvers. URL http://CRAN.R-project.org/package=svd, R package version 0.4

Larsen RM (1998) Efficient algorithms for helioseismic inversion. PhD thesis, University of Aarhus, Denmark

Roy R, Kailath T (1989) ESPRIT: estimation of signal parameters via rotational invariance techniques. IEEE Trans Acoust 37:984–995

Vautard R, Yiou P, Ghil M (1992) Singular-spectrum analysis: A toolkit for short, noisy chaotic signals. Physica D 58:95–126

Yamazaki I, Bai Z, Simon H, Wang LW, Wu K (2008) Adaptive projection subspace dimension for the thick-restart Lanczos method. Tech. rep., Lawrence Berkeley National Laboratory, University of California, One Cyclotron road, Berkeley, California 94720