TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

On Stationary Subspace Analysis

Zur Erlangung des akademischen Grades

Diplom-Ingenieur

Im Rahmen des Studiums

Masterstudium Statistik-Wirtschaftsmathematik

eingereicht von

Léa Flumian

Matrikelnummer 11721374

ausgeführt am Institut für Computational Statistics der Fakultät für Mathematik und Geoinformation der Technischen Universität Wien

Betreuer: Associate Prof. Klaus Nordhausen

Wien, October 24, 2019

Unterschrift Verfasser *Unterschrift Betreuer*

Abstract

Multivariate times series occur in many application areas and are challenging to model. They are used in very various areas such as signal processing, pattern recognition, econometric and mathematical finance, weather forecasting or electroencephalography.
A common approach is therefore to assume that the observed time series can be decomposed into latent components with different exploitable properties. Popular models for time series modelling are for example moving average or autoregressive models. In some of these models especially non-stationary components are of interest.
In the following we will consider in more details the so called Stationary Subspace Analysis approach (SSA). This work considers two questions. On the one hand, how to separate non-stationary components from stationary components within a multivariate time series? On the other hand how to identify the right number of non-stationary components within a multivariate time series? In order to answer these questions we fill first introduce some definitions and properties needed to develop the differents methods that once can implement to perform SSA. Once we have explained these methods we will demonstrate them in a simulation study.

Acknowledgements

Thanks for the patience and the support to Pr. Nordhausen.
Thanks also to my family and my friends.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The definitions and properties presented in this chapter are based mainly on [16] for time series concepts and on [1] for bootstrap notions.
Here we will introduce some definitions and properties needed to understand what we will develop later on. We will first define a probability space and a stochastic process and we will use this concept to define time series. Among time series we will distinguish between stationary and nonstationary time series and within stationary time series we will differentiate between weak and strong stationarity. In order to define stationarity we will see that we need to define some statistical moments. After defining time series, we will present some classical models for univariate time series.
The second part of this introduction will focus on bootstrap and how one can use this method in the context of time series. We will present the general principles of bootstrap and then explain how they can be adapted to fit the specificities of time series.
In this work vectors and multivariate quantities will be written in bold to help differentiate them from scalars.

1.1 Time series

First we define the concept of stochastic process and how it is linked to time series. To introduce this concept we need to introduce the notion of probability space. Then we also define the first and second order moments of a time series as well as its autocovariance. In Section 1.1.2 we will see how we can use these properties to characterise the stationarity or nonstationarity of a time series.

1.1.1 Definitions

We first need to begin with the notion of probability space which is the space where our time series will then be defined. In order to properly construct a probability space we need to introduce σ-algebras and probability measures.
We start with the definition of a σ-algebra.

Definition 1 (σ-algebra)**.** *Let E be some set and $\mathcal{P}(E)$ its power set (e.g. the set of all subset of E including the empty set and E itself). A subset $A \in \mathcal{P}(E)$ is called a σ-algebra if:*

- *$E \in A$.*
- *A is closed under complementation: $\forall B \in A$, $\bar{B} \in A$.*
- *A is closed under countable unions: $\forall p \in \mathbb{N}$, $p \geq 2$ and p finite, if $A_1 ... A_p \in A$ then $A_1 \bigcup, ..., \bigcup A_p \in A$.*

A probability space is composed of two important parts: a σ-algebra and a probability measure which is a measure with some specific additional properties.

Definition 2 (Probability measure)**.** *A function $\mathbb{P}$ is called a probability measure on a σ-algebra $\mathcal{A}$ if:*

- $\forall A \in \mathcal{A}$, $\mathbb{P}(A) \in [0,1]$.
- $\mathbb{P}(\emptyset) = 0$ *and* $\mathbb{P}(\mathcal{A}) = 1$.
- *P satisfies the countable additive property: for any finite set A_1,...,A_p of elements of $\mathcal{A}$ pairwise disjoint it holds that $\mathbb{P}(\bigcup_{i=1}^{p} A_i) = \sum_{i=1}^{p} \mathbb{P}(A_i)$.*

Eventually we use the definitions of σ-algebra and probability measure as basis to build probability spaces.

Definition 3 (Probability Space)**.** *A probability space consists of three parts:*

1. *A sample space, Ω , which is the set of all possible outcomes.*
2. *A set of events $\mathcal{A}$, where each event is a set containing zero or more outcomes.*
3. *The assignment of a probability measure $\mathbb{P}$ to the events.*

The next definition justifies the introduction of the construct of probability space. Probability spaces are spaces on which stochastic processes can be build and we will see that to understand time series it is easier to be familiar with stochastic processes.

Definition 4 (Stochastic Process)**.** *A stochastic process $(X_t | t \in \mathbb{T})$ is a collection of real value random variables defined on a common probability space $(\Omega, \mathcal{A}, \mathbb{P})$ where Ω is a sample space, $\mathcal{A}$ is a σ-algebra and $\mathbb{P}$ is a probability measure. In this definition $\mathbb{T}$ designates the set of time on which the stochastic process is defined. It can be either discrete or continuous.*

Ultimately we come to the definition that might be of highest interest in this work: what is a time series.

Definition 5 ((univariate) Time series)**.** *A time series is a series of data points indexed in time. Most commonly a time series is a sequence taken at successive equally spaced point in time. Thus it is a sequence of discrete-time data.*
It is usually denoted as:

$$x_t, t \in 1, ..., T.$$

A time series is thus a realisation of a stochastic process (X_t).

Note that a time series can be univariate (at each time point we consider a single value) or multivariate (for each time point we then consider a vector instead of a single value). In the univariate case the time series will be represented as a vector whereas in the multivariate case the time series will be represented through a matrix where each line corresponds to a different time point.
In this thesis we will only consider discrete processes e.g. processes where $\mathbb{T} \subseteq \mathbb{Z}$ or $\mathbb{T} \subseteq \mathbb{N}$.
After defining what a time series is, a legitimate question arises: how to categorise or differentiate between different time series? We will come to that with the notions of stationarity and nonstationarity but first we define the first and second statistical moment of a time series as well as its covariance. Later on we will develop why these statistical measures are of importance.

Definition 6 (Expected value)**.** *The expected value of a univariate sotchastic process X on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ is defined through:*

$$\mathbb{E}(X) = \int_{\Omega} X d\mathbb{P} = \int_{\mathbb{R}} x dX \mathbb{P}(x),$$

if this integral is absolutely convergent i.e $\int_{\Omega} |X| d\mathbb{P} < +\infty$.
For a time series we write $\mathbb{E}(X_t) = \mu(t) = \int_{\Omega} X_t$.

The mean measures the location of a random variable: it roughly tells where the majority of the values are located. Next, we also define the variance.

Definition 7 (Variance)**.** *The variance of a univariate stochastic process X is the expected value of the squared deviation from μ, the mean of X:*

$$Var(X) = \mathbb{E}[(X - \mu)^2].$$

The variance is a dispersion measure: it indicates how spread are the values around the mean.

Definition 8 (Covariance)**.** *Let X and Y be two univariate stochastic processes then the covariance of X, Y is defined as:*

$$Cov(X,Y) = \mathbb{E}[(X - \mathbb{E}(X))(Y - \mathbb{E}(Y))].$$

Covariance can be used to measure the joint variability of two stochastic processes.

1.1.2 Properties

After defining time series and their first and second order moments we will see how these are related to the notion of stationarity. Among stationarity one has to separate strong from weak stationarity. Eventually if we talk about stationarity we also have to define the concept of nonstationarity.
As in this work we will mainly deal with mulitvariate time series, the definitions and properties that follow are written for multivariate time series.

Definition 9 (Strong Stationarity)**.** *A stochastic process $(\boldsymbol{x}_t | t \in \mathbb{T})$ is called strong stationary if the distribution of $(\boldsymbol{x}_{t_1+k}, ..., \boldsymbol{x}_{t_s+k})$, for all finite subsets $\{t_1, ..., t_s\} \in \mathbb{Z}$ and for every $k \in \mathbb{Z}$ is independent of k.*

Because the above definition can be very restrictive statisticians also work with the concept of weak stationarity.

Definition 10 (Weak Stationarity)**.** *A stochastic process $(\boldsymbol{x}_t | t \in \mathbb{T})$ is called weak stationary if $\forall s, t \in \mathbb{Z}$*

1. $\mathbb{E}(\boldsymbol{x}_t \boldsymbol{x}_t^t) = \boldsymbol{\Sigma} < +\infty$.
2. $\mathbb{E}(\boldsymbol{x}_s) = \mathbb{E}(\boldsymbol{x}_0)$.
3. $\mathbb{E}(\boldsymbol{x}_t \boldsymbol{x}_s^t) = \mathbb{E}(\boldsymbol{x}_{t-s} \boldsymbol{x}_0^u)$.

The last condition is equivalent to $\boldsymbol{Cov}(\boldsymbol{x}_t, \boldsymbol{x}_s) = \boldsymbol{Cov}(\boldsymbol{x}_{t-s}, \boldsymbol{x}_0)$. It is to say that the covariance of the process only depends on the distance between two time points.
With the concept of stationarity we can introduce a new statistical measure.

Definition 11 (Autocovariance)**.** *If the stochastic process $(\boldsymbol{x}_t | t \in \mathbb{T})$ is squared integrable i.e $\mathbb{E}({x^2}_{i,j}) < +\infty \ \forall \ i \in \mathbb{N}^*$, $\forall \ j \in \mathbb{Z}$ and (weak) stationary then we define the autocovariance of the stochastic process $(\boldsymbol{x}_t | t \in \mathbb{T})$ as:*

$$\boldsymbol{Cov}(\boldsymbol{x}_t, \boldsymbol{x}_s) = \mathbb{E}[(\boldsymbol{x}_t - \boldsymbol{\mu}(t))(\boldsymbol{x}_s - \boldsymbol{\mu}(s))^t].$$

Definition 12 (Autocovariance function)**.** *The autocovariance matrix of a (weak) stationary process $(\boldsymbol{x_t}, t \in \mathbb{Z})$ is defined as:*

$$\boldsymbol{\gamma} : \mathbb{Z} \to \mathbb{R}^{p \times p}$$
$$k \mapsto \boldsymbol{\gamma}(k) = \boldsymbol{Cov}(\boldsymbol{x}_{t+k}, \boldsymbol{x}_t).$$

A nonstationary time series is then a time series for which one of the above criteria does not hold. In case the first condition is violated we then talk of nonstationarity in variance; if the second condition is disturbed, the time series exhibits nonstationarity in mean; and if the third condition is not met we qualify the time series as nonstationary in covariance. Common examples of nonstationarity in mean are for example trend, seasonal or cyclic behaviour.
One other statistical measure of interest that one can use to characterize a time series is its autocorrelation function. This measure is based on the autocovariance.

Definition 13 (Autocorrelation function)**.** *The autocorrelation matrix of a (weak) stationary process $(\boldsymbol{x}_t, t \in \mathbb{Z})$ is defined as:*

$$\boldsymbol{\rho} : \mathbb{Z} \to \mathbb{R}^{p \times p}$$
$$k \mapsto \boldsymbol{\rho}(k) = \boldsymbol{Corr}(\boldsymbol{x}_{t+k}, \boldsymbol{x}_t).$$

The (i,j)-th element of the matrix $\boldsymbol{\rho}(k)$ is the correlation between the variable $x_{i,t+k}$ and $x_{j,t}$ defined as,

$$\rho_{i,j}(k) = \frac{\gamma_{i,j}(k)}{\sqrt{\gamma_{i,i}(0)\gamma_{j,j}(0)}}.$$

This function tells us how strongly the different terms of the time series are correlated with each other and how far in the past or future this correlation goes.

Property 1. *For the autocorrelation γ function of a stochastic process $(\boldsymbol{x}_t | t \in \mathbb{T})$ it holds:*

$$\gamma(k) = \gamma(-k), \forall k \in \mathbb{Z}.$$

1.1.3 Popular time series model

There exist several classical models to simulate or fit univariate time series. We will use these models later on in our simulations.

Definition 14 (Moving Average Process)**.** *A (univariate) time series (x_t) follows a moving average model of order q - MA(q) - when,*

$$x_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q},$$

where the innovations ϵ_i are independent identically distributed white noise and $\theta_1, ..., \theta_q$ are the parameters of the model.

If one takes for example a MA(q) process it means that the output variable e.g.the value at time t depends linearly on the current and past values of the stochastic process (ϵ). In this model the elements of the stochastic process (ϵ) can be interpreted as random shocks. The definition of the process shows that these shocks are directly propagated to future values of the time series. Within a MA(q) process, a shock affects the current value and the next q future values of the process: in the process definition the terms ϵ_t to ϵ_{t-q} are taken into consideration.

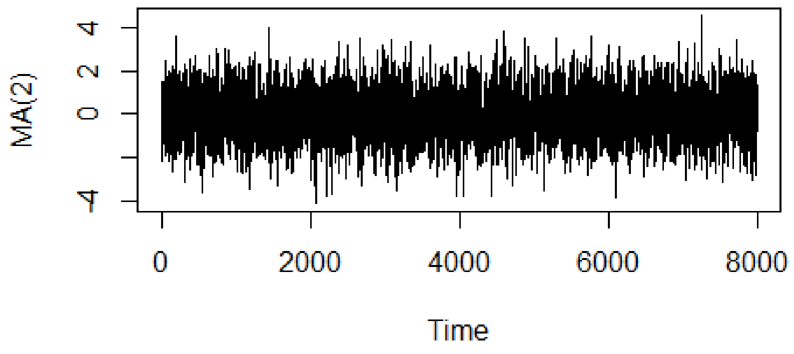

Figure 1.1: Moving Average Process of order 2.

In Figure 1.1 we show an example of a (univariate) moving average process of order 2 with coefficients $0.38, 0.22$ and innovations drawn from a normal distribution $\mathcal{N}(0, 1)$.

Definition 15 (Autoregressive Process)**.** *A (univariate) time series (x_t) follows an autoregressive model of order p - AR(p) - when,*

$$x_t = \mu + \phi_1 \boldsymbol{x}_{t-1} + ...\phi_p x_{t-p} + \epsilon_t,$$

where the innovations ϵ_i are independent identically distributed white noise and $\phi_1, ..., \phi_q$ are the parameters of the model.

In an AR(p) process a shock affects the values of the process infinitely far into the future. In the equation defining the process at state t appears only the shock at time t e.g. the term ϵ_t but as we are also looking at terms x_{t-1} to term x_{t-p} we are then also considering the effects of shocks ϵ_{t-1} to ϵ_{t-p}. If the process is stationary the effects of the shocks diminish until becoming zero in the limit. An example of an (univariate) AR(2) process is presented in Figure 1.2 below. The coefficients are $0.70, 0.25$ and innovations are drawn from a normal distribution $\mathcal{N}(0, 1)$.

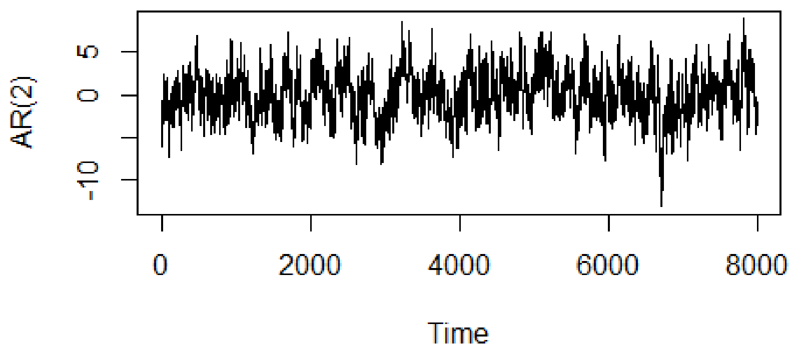

Figure 1.2: Autoregressive Process of order 2.

Definition 16 (Autoregressive Moving Average Process)**.** *A (univariate) time series* (x_t) *follows an Autoregressive Moving Average model of order* (p, q) *- ARMA(p,q) - when it is a combination of two: an autoregressive process of order* p *and a moving average process of order* q*.*

We illustrate this definition by displaying the plot of an ARMA(2,2) process in Figure 1.3 below. The coefficients for the moving average and the autoregressive process are the same as in the example above: the signal plotted here is the exact association of the MA(2) and AR(2) processes displayed in Figure 1.1 and Figure 1.2. Innovations are still drawn from a normal distribution $\mathcal{N}(0, 1)$.

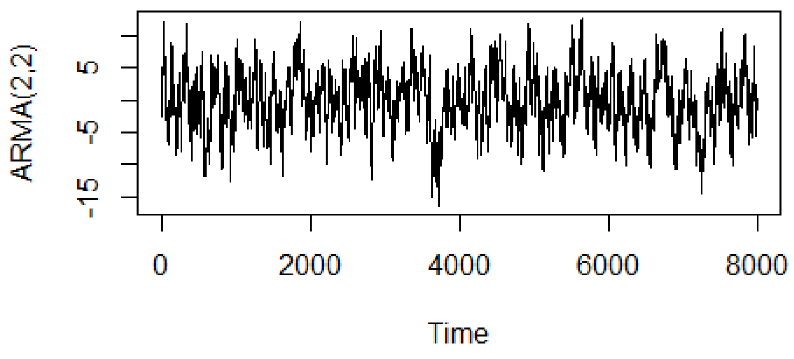

Figure 1.3: ARMA Process of order (2,2).

For all three models conditions on the parameters θ and ϕ are made to guarantee stationarity. Usually the innovations are considered to be normally distributed.

1.2 Bootstrap

This section is not about time series themselves but rather on how to process them. We will explain the well known bootstrap method and then show how it can be specifically adapted to time series.

1.2.1 General Principles

Traditional bootstrap is a resampling method by independently resampling with replacement and using the resampled data to perform inference. If one wants to determine characteristics of a population, often one cannot test or compute statistics over the population. But one can select one original sample and the additional samples, also referred to as bootstrap samples, are drawn from this original sample. One can think of these as outcomes of independent and identically distributed random variables with a certain probability density function f and a certain cumulative distribution function denoted by F. The samples are then used to make inference about a chosen population characteristic.
There are two main strategies when bootstrapping a set of data: either parametric or non-parametric bootstrap.
In the parametric case the cumulative distribution function of the data is fully specified by a finite set of parameters $\boldsymbol{\Psi}$ and we aim to make inference about a population characteristic $\boldsymbol{\theta}$ which can be a component of $\boldsymbol{\Psi}$ or a function of $\boldsymbol{\Psi}$. We then estimate the parameter(s) of interest and use them to get a fitted model and then we draw samples from this fitted model.
In case of a non-parametric model for the population we use non-parametric bootstrap. In this case we sample with replacement from the empirical distribution function.

1.2.2 Bootstrapping times series

For bootstrapping times series one has to take into account the order within the data. Independent bootstrap is not meaningful in this case. Two main strategies exist for bootstrapping time series. Additional information on bootstrapping time series can be found in [17].

Residual based bootstrap

If a model for a time series has innovations assumed to be independent and identically distributed (iid) then it is possible to perform classical bootstrap directly on those residuals. Fit the model based on the available data and then bootstrap the so obtained residuals to generate new time series.

Example Consider an AR(p) model:

1. Estimate the parameters $\phi_1, ..., \phi_p$ for the centered time series $(x_t - \bar{x_t})$.
2. Estimate the residuals $\hat{\epsilon_j} = x_j - \sum_{i=1}^{p} \hat{\phi}_i x_{j-i}, j = p+1, ..., T$.
3. Center the residuals $\hat{\epsilon_j^c} = \hat{\epsilon} - \bar{\hat{\epsilon}}$.
4. Sample with replacement: ϵ^* from $\hat{\epsilon_j^c}$ T+1 bootstrap innovations.
5. Construct the bootstrap time series $x_t^* = \sum_{i=1}^{p} \hat{\phi}_i x_{t-i}^* + \epsilon_t^*$ and add the mean once again.

Block bootstrap

If we want to preserve some of the serial dependence within the data we can bundle the observations in blocks and then sample from the blocks.
For a time series $\boldsymbol{x}$ of length n we assume n can be written $n = bl$ where b is the number of blocks and l is the average block length. Then,

$$\boldsymbol{z}_1 = (\boldsymbol{x}_1, ..., \boldsymbol{x}_l), \boldsymbol{z}_2 = (\boldsymbol{x}_{l+1}, ..., \boldsymbol{x}_{2l}), ...$$

We sample from the blocks $\boldsymbol{z}_1$,...,$\boldsymbol{z}_b$ with replacement. This is called non-overlapping block bootstrap. The idea is that if the block length is long enough then enough of the dependence structure will be kept in the bootstrap times series and the approximation, such as test statistics, made on the bootstrap times series will still hold.
There exists an overlapping version of the block bootstrap. Different weights are then given to the edges of the data which are then often considered to be circular. The observations then appear equally often within the blocks.
Note that the strategies presented above are nonstationary. However if we chose blocks with different lengths coming from a geometrical distribution with mean l then we can overcome this

nonstationarity issue. We need to generate enough block lengths so that the sum exceeds the time series length. Then we create non-overlapping blocks and sample directly from them.

Semiparametric non overlapping block bootstrap of hypothesis testing

In Chapter 4 we will use a semiparametric model for hypothesis testing. In hypothesis testing, the null hypothesis often suggests some of the parameters of the model. Assuming we are working with a sample of size n yielding the test statistic T_0, then the semiparametric bootstrap strategy is:

1. Sample using the null model m samples of size n.
2. Compute the corresponding m test statistics T^*.
3. The bootstrap p-value is then defined as:
$$\frac{\#(T^* > T_0) + 1}{m + 1}.$$

The R-code used to perform our bootstrap strategy is available in Appendix in Section D.

Chapter 2

Stationary Subspace Analysis

The approach we present below is detailed in [21], [20].

2.1 Principles

The main idea of Stationary Subspace Analysis (SSA) is to decomposed a p-variate time series $(\boldsymbol{x}_t)$ into a stationary part and a nonstationary part:

$$\boldsymbol{x}_t = \boldsymbol{A}\boldsymbol{z}_t = [\boldsymbol{A}_n \boldsymbol{A}_s] \begin{pmatrix} \boldsymbol{n}_t \\ \boldsymbol{s}_t \end{pmatrix}. \tag{2.1}$$

In the previous equation $\boldsymbol{x_t}$ designates an observed p-variate time series with several components among which it is difficult to distinguish stationary and nonstationary elements. The stationarity or nonstationarity of the components could be detected in the unobservable latent sources $\boldsymbol{z}_t$. The sources are composed of a signal part corresponding to the nonstationary components and a noise part corresponding to the stationary components. We are interested in the signal part. The link between the observed time series $\boldsymbol{x}_t$ and the sources $\boldsymbol{z}_t$ is established through the usually unknown $p \times p$ full rank mixing matrix $\boldsymbol{A}$. The mixing matrix and the the sources can be decomposed into a nonstationary part $\boldsymbol{A}_n$ and $\boldsymbol{n}_t$ and a stationary part $\boldsymbol{A}_s$ and $\boldsymbol{s}_t$. We assume in the following that $\boldsymbol{n_t}$ is composed of independent components. Latent refers here to the fact that the sources are not directly observable. One can only observe the time series $\boldsymbol{x}_t$ and from there infer the sources: if the mixing matrix is known - which is often not the case - then by multiplying the signal by the inverse of the mixing matrix, $\boldsymbol{W}$ (defined as $\boldsymbol{W} = \boldsymbol{A}^{-1}$) then we could theoretically access the sources where it is observable which component is stationary or nonstationary. Examples of this duality are given later on when we discuss the chosen settings for our simulations in Chapter 4.
We would like to be able to access these latent components to focus the analysis on the nonstationary part of the process. Separating the subspaces of $\boldsymbol{n}_t$ and $\boldsymbol{s}_t$ is then the goal of stationary subspace analysis. From now on we denote k as the number of nonstationary components. Note that we can estimate only the subspaces corresponding to $\boldsymbol{n}_t$ and $\boldsymbol{s}_t$ and those subspaces can only be estimated up to a rotation. To solve this non uniqueness issue we will transform our matrices into orthogonal projection matrices. This way we will define unique linear projection subspaces.
Starting from the matrix $p \times k$ $\boldsymbol{B}$ with linear independent columns we can define an orthogonal projection matrix $\boldsymbol{P}$ by setting,

$$\boldsymbol{P} = \boldsymbol{B}(\boldsymbol{B}\boldsymbol{B}^t)^{-1}\boldsymbol{B}^t.$$

In our case the matrix $\boldsymbol{B}$ will be either the inverse of $\boldsymbol{W}_n = \boldsymbol{A}_n^{-1}$ if we are interested in finding the projection to the nonstationary subspace or the inverse of $\boldsymbol{W}_s = \boldsymbol{A}_s^{-1}$ if we are interested in projecting to the stationary subspace.
Now in order to detect the nonstationary components, we divide the original time series $\boldsymbol{x}_t$ into K intervals and compute for each interval first and second order statistics. We then compare these statistics and use these comparisons to determine the stationarity or nonstationarity of a component. The idea behind this decomposition is that if the time series is stationary in mean then the mean will be constant over all the K intervals. However if the time series is not stationary in mean then the mean will take different values depending on the intervals. Same idea can be applied for the variance (or any other statistics).

Issue with this approach is that the comparison focuses only on deviations in mean and variance but not in autocorrelation and we do not investigate which specific type of nonstationarity is at play (for example for the mean we would like to know if the nonstationarity is coming from a trend or from a seasonality pattern within the data).

2.2 Dimension Reduction

In dimension reduction methods like Principal Component Analysis (PCA) the idea is to compute the eigenvalue decomposition of the covariance of a time series to isolate the principal dimensions. For definitions regarding PCA we use as reference [18] and [8].
The principal components are defined as the directions with the largest possible variance e.g. the direction accounting for as much of the variability in the data as possible and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components.
In SSA one looks for a matrix $\boldsymbol{M}$ summarising the stationarity state for some statistics (mean, variance, covariance or mixed nonstationarities) and then computes the eigenvalue decomposition of this matrix $\boldsymbol{M}$. The eigenvectors whose corresponding eigenvalues are 0 (or statistically close to 0) are then the ones corresponding to the stationary components. These components do not contain a lot of information or variation as they are equivalent to noise. We can write this decomposition:

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^t,$$

where the matrix $\boldsymbol{U}$ contains as columns the eigenvectors of the matrix $\boldsymbol{M}$ and the matrix $\boldsymbol{D}$ is a diagonal matrix whose diagonals elements are the eigenvalues of $\boldsymbol{M}$ ordered in decreasing order. We then select the k non-zero eigenvalues in $\boldsymbol{D}$ representing the nonstationary components and write $\begin{pmatrix}\boldsymbol{U}_n \\ \boldsymbol{U}_s\end{pmatrix}$ where $\boldsymbol{U}_n$ is a $k \times p$ matrix whose rows contain the eigenvectors belonging to the non-zero eigenvalues and $\boldsymbol{U}_s$ is a $(p-k) \times p$ matrix whose rows contain the remaining eigenvectors.

2.3 Assumptions

Assume that $(\boldsymbol{x}_t)$ is the p-variate time series observed at time t, $t \in \{1, ..., T\}$ and let $T_1 ... T_K$ be K disjoint intervals used for the comparison of the statistics of interest (mean, variance, covariance). We do not detail here the choice of K but we specify that K has to be large enough to capture statistical changes and narrow enough to detect slight variations within the observed time series. We would like each interval to contain sufficient time points for a good estimation of the properties of interest. Optimally the cut points of the intervals should allow the quantities to be as different as possible. This is difficult to chose but at least we try to have intervals of different length so that seasonality effects are easier to detect. In all of our computations we will set $K = 11$.
We define then:

$$\boldsymbol{m}_{T_i} = \frac{1}{|T_i|} \sum_{t \in T_i} (\boldsymbol{x}_t), \tag{2.2}$$

and,

$$\boldsymbol{S}_{\tau,T_i} = \frac{1}{|T_i| - \tau} \sum_{t \in T_i} (\boldsymbol{x}_t - \boldsymbol{m}_{T_i})(\boldsymbol{x}_{t+\tau} - \boldsymbol{m}_{T_i})^t. \tag{2.3}$$

Note that the covariance of the time series $(\boldsymbol{x}_t)$ is equal to $\boldsymbol{S}_{0,T}$.
We will use these as measures to detect nonstationarity in mean or variance between the K intervals.
We also make the following assumptions:

(A1) $\boldsymbol{cov}(\boldsymbol{s}_t, \boldsymbol{n}_{t+\tau}) = 0$, $\forall \tau \geq 0$.

(A2) $\mathbb{E}(\boldsymbol{s}_t) = \boldsymbol{0}$ and $\boldsymbol{cov}(\boldsymbol{s}_t) = \boldsymbol{I}_{p-k}$.

(A3) $\boldsymbol{cov}(\boldsymbol{n}_t) = \boldsymbol{D}_t$ where $\boldsymbol{D}_t$ is a diagonal matrix with positive diagonal elements.

The first assumption states that the nonstationary and stationary part are uncorrelated throughout time and the second assumptions fixes the location and covariance of the stationary part. This is done only for convenience. The third assumptions states that the nonstationary components are

uncorrelated.
As in dimension reduction to allow the comparisons between the different components we work with a whitened time series. We introduce:

$$\boldsymbol{y}_t = \boldsymbol{S}_{0,T}(\boldsymbol{x}_t)^{-\frac{1}{2}}(\boldsymbol{x}_t - \boldsymbol{m}_T).$$

Based on this transformation we extract $\boldsymbol{U}$ the matrix of eigenvectors of the separation method matrix $\boldsymbol{M}$. This matrix changes depending on the type of nonstationarity studied. It measures the deviation of a statistic within the intervals to the global statistic. The matrix $\boldsymbol{M}$ is computed based on the time series $(\boldsymbol{y}_t)$ In Chapter 3 we will develop more detailed definitions of $\boldsymbol{M}$. We can then compute the transformation matrix to the nonstationary subspace:

$$\boldsymbol{W}_n = \boldsymbol{U}_n\boldsymbol{S}_{0,T}(\boldsymbol{x})^{-\frac{1}{2}},$$

and the transformation matrix to the stationary subspace,

$$\boldsymbol{W}_s = \boldsymbol{U}_s\boldsymbol{S}_{0,T}(\boldsymbol{x})^{-\frac{1}{2}}.$$

Interested readers in theoretical backgrounds and theories underlying SSA can refer to [20], [21], [7], [4].
In Chapter 3 we describe and explain which methods can be implemented to perform SSA.

Chapter 3

Separation Methods

Within a time series there can be different types of nonstationarities. Different methods exist to detect these non-stationarities: specific methods build for specific nonstationarities (for example in mean, variance or in the covariance strucutre) or combination methods efficient with mixed non-stationarities. The idea of the separation does not depend of the type of nonstationarity. First, one defines a square separation matrix $\boldsymbol{M}$ which reflects the nonstationary behaviour. Then one takes the eigenvalue decomposition of $\boldsymbol{M}$ (which exists because $\boldsymbol{M}$ is always defined to be a squared matrix):

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^t.$$

where $\boldsymbol{U}$ is a square orthogonal matrix containing the eigenvectors of $\boldsymbol{M}$ ordered according to the decreasing order of the eigenvalues of $\boldsymbol{M}$ contained in the square diagonal matrix $\boldsymbol{D}$. To allow the comparisons as introduced in the following subsections we define the time series $\boldsymbol{y}$:

$$\boldsymbol{y_t} = \boldsymbol{S}_{0,T}^{-\frac{1}{2}}(\boldsymbol{x_t} - \boldsymbol{m}_T(\boldsymbol{x})).$$

The first two methods introduced below are inspired from [13] for the first one (SIR) and [19] for the second one (SAVE).

3.1 Non stationarity in mean - SIR

If the time series exhibits mean nonstationary behaviour then the idea is to slice it into intervals where one can differentiate between parts with non stationary mean and parts with stationary mean. With this in mind we define the square matrix $\boldsymbol{M}$ such that:

$$\boldsymbol{M}_{SIR} = \sum_{i=1}^{K} \frac{|T_i|}{T} \boldsymbol{m}_{T_i}(\boldsymbol{y}_t) m_{T_i}(\boldsymbol{y}_t)^t.$$

Nonstationarity in location can be due to trend, cyclic or seasonal patterns, structural breaks or mean shift. Seasonal patterns are patterns that repeat with a fixed period of time.
$\boldsymbol{M}$ is then the covariance between the means of the different intervals weighted by the interval length. It is easy to see that for all components with a stationary mean the eigenvalue of $\boldsymbol{M}_{SIR}$ should be zero and those components with non-zero eigenvalues then correspond to nonstationary components. If we then write the eigenvalue decomposition:

$$\boldsymbol{M}_{SIR} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^t,$$

where $\boldsymbol{U} = \begin{pmatrix} \boldsymbol{U}_n \\ \boldsymbol{U}_s \end{pmatrix}$ are the eigenvectors of $\boldsymbol{M}$ arranged such that $\boldsymbol{U}_n$ is the $k \times p$ matrix containing as rows the eigenvectors belonging to the non-zero eigenvalues and U_s contains the remainings eigenvectors. The final unmixing matrices are then:

$$\boldsymbol{W}_n = \boldsymbol{U}_n \boldsymbol{S}_{0,T}^{-\frac{1}{2}}(\boldsymbol{x}_t),$$

and,

$$\boldsymbol{W}_s = \boldsymbol{U}_s \boldsymbol{S}_{0,T}^{-\frac{1}{2}}(\boldsymbol{x}_t).$$

The matrices $\boldsymbol{W}_n$ and $\boldsymbol{W}_s$ respectively project the observed time series to the subspace of components with nonstationary mean and the subspace of components with stationary mean.

3.2 Non stationarity in variance - SAVE

If we now consider nonstationarity in variance then we can define the square matrix:

$$\boldsymbol{M}_{SAVE} = \sum_{i=1}^{K} \frac{|T_i|}{T} (\boldsymbol{I}_p - \boldsymbol{S}_{0,T_i}(\boldsymbol{y}_t))^2.$$

This matrix evaluates the distance between the intervals variance and the global variance.
The separation process is then the same as in the SIR method. We compute the eigenvalue decomposition of the matrix $\boldsymbol{M}_{SAVE}$ and then deduce the projection matrices:

$$\boldsymbol{W}_n = \boldsymbol{U}_n \boldsymbol{S}_{0,T}^{-\frac{1}{2}}(\boldsymbol{x}_t),$$

and,

$$\boldsymbol{W}_s = \boldsymbol{U}_s \boldsymbol{S}_{0,T}^{-\frac{1}{2}}(\boldsymbol{x}_t).$$

Additional robust methods for time series with nonstationarity in variance are developed in details in [11].

3.3 Non stationarity in the covariance structure

To detect nonstationarity in the covariance structure, one should use the following matrix:

$$\boldsymbol{M}_{cov,\tau} = \sum_{i=1}^{K} \frac{|T_i|}{T} (\boldsymbol{S}_{\tau,T}(\boldsymbol{y}_t) - \boldsymbol{S}_{\tau,T_i}(\boldsymbol{y}_t)).$$

As the matrix $\boldsymbol{S}_{\tau,T} - \boldsymbol{S}_{\tau,T_i}$ evaluates the deviation between the global covariance and the lagged covariance within interval T_i this separation matrix $\boldsymbol{M}$ enables us to detect the changes in the time series covariance structure.
We compute the eigenvalue decomposition of the matrix $\boldsymbol{M}$ and deduce then the matrices $\boldsymbol{W}_n$ and $\boldsymbol{W}_s$ (the computation steps are identical to the steps in the SIR and SAVE methods).

3.4 Mixed non stationarities

In the previous methods detecting either nonstationarity in mean (SIR), variance (SAVE) or in the covariance structure, the separation matrix $\boldsymbol{M}$ targets only one specific kind of nonstationarity. If the exact nature of the nonstationarities remains unknown or if we know that we have more than one type of nonstationarity then it may be useful to resort to combination methods.

3.4.1 Joint Diagonalisation

The ideas developed in this subsection are inspired from [10]. Goal is to find a matrix $\boldsymbol{M}$ that gives the best diagonalized matrix approximation of the following matrices $\boldsymbol{M}_{SIR} = \boldsymbol{M}_1$, $\boldsymbol{M}_{SAVE} = \boldsymbol{M}_2$, $\boldsymbol{M}_{cov,\tau} = \boldsymbol{M}_3$. We search for the diagonal matrix $\boldsymbol{U}$ that maximizes:

$$\sum_{i=1}^{3} ||diag(\boldsymbol{U}\boldsymbol{M}_i\boldsymbol{U}^t)||^2.$$

Only a subset of these matrices or some natural lags could be used but the principle remains the same. Several algorithms exist for joint diagonalization see for example [10] for details. To perform the joint diagonalization of the three separation matrices we will use in this work the R-Package JADE which applies Jacobi angles (see [3]).
Based on $\boldsymbol{U}$ one computes then $\boldsymbol{D}_i = diag(d_{i,1}, ..., d_{i,p})$ and then $\boldsymbol{U}_n$ collects the nonstationary components as all the columns of U such that $\sum_i d_{i,j} \neq 0$. As before $\boldsymbol{D}_i$ is the matrix containing the eigenvalues of the matrix $\boldsymbol{M}_i$ $(i = 1, 2, 3)$ in decreasing order.
Note that in this method the three matrices that are jointly diagonalized measure nonstationarity in mean, variance and covariance. Thus this method can detect nonstationarity in mean, variance and covariance structure.

3.4.2 Analytic Stationary Subspace Analysis (ASSA)

For this separation method the idea is again the same as for the SIR, SAVE methods but the separation matrix $\boldsymbol{M}_{ASSA}$ is defined to detect nonstationarity in both mean and variance. Details can be found in [20]. We define:

$$\boldsymbol{M}_{ASSA} = \frac{1}{T}\sum_{i=1}^{K}\{\boldsymbol{m}_{T_i}(\boldsymbol{y}_t)\boldsymbol{m}_{T_i}(\boldsymbol{y}_t)^t + \frac{1}{2}\boldsymbol{S}_{0,T_i}(\boldsymbol{y}_t)\boldsymbol{S}_{0,T_i}(\boldsymbol{y}_t)^t\} - \frac{1}{2}\boldsymbol{I}_p.$$

Then again the projections matrices as obtained as in the SIR and SAVE methods.
Drawbacks of this method are that it cannot not detect nonstationarity in the covariance strucutre and is also not able to detect the precise type of nonstationarity the components exhibit: for example in case of nonstationarity in mean the ASSA method will detect the nonstationarity but not if it is a due to a trend, a cyclic or seasonal component.

3.4.3 Average orthogonal projection - AOP

The ideas presented in this section are all developed in [6].
If one suspect nonstationarity in mean, variance and covariance - or just two of these different types - one might want to try to find a subspace that is the closest possible to the corresponding subspaces for each type of nonstationarity. This is the idea of the AOP method. If one consider the AOP in the case of nonstationary in mean, variance and covariance then we can say that the projector corresponding to the AOP is the orthogonal projection projecting to the subspace closest to the subspace of nonstationarity in mean, the subspace of nonstationarity in variance as well as the subspace of nonstationarity in covariance.

Definition 17 (Crone and Crosby distance, see [15] (1995))**.** *For two orthogonal projectors $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ that have equal rank, the Crone and Crosby Distance is defined as:*

$$D_w^2(\mathbf{P}_1, \mathbf{P}_2) = 1 - \frac{tr(\mathbf{P}_1, \mathbf{P}_2)}{\sqrt{(tr(\mathbf{P}_1)tr(\mathbf{P}_2))}}.$$

If the subspaces are not of equal dimensions then we define a new distance.

Definition 18 (Distance between subspaces in case of arbitrary ranks)**.** *For two projection matrices $\mathbf{P}_1$, $\mathbf{P}_2$ with respective ranks k_1 and k_2, we define the weighted distance:*

$$D_w^2(\mathbf{P}_1, \mathbf{P}_2) = \frac{1}{2}||w(k_1)\mathbf{P}_1 - w(k_2)\mathbf{P}_2||.$$

Weights are used in order to make the projection matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ which have different ranks more comparable. Some interesting choices of the weights are, for $k > 0$:

- $w_a(k) = 1$,
- $w_b(k) = \frac{1}{k}$,
- $w_c(k) = \frac{1}{\sqrt{k}}$.

In general if we compare subspaces of equal rank then we use the Crone and Crosby distance but if we compare subspaces of different ranks then we will have to use the second definition. In the following, as the number of non stationary components will be known, we will know that the subspaces being compared have equal rank. We will then use the Crone and Crosby distance which is equivalent to using the second definition with weights set to $w(k_1) = w(k_2) = 1$. When we refer to subspaces being close to each other it means in our context that the Crone and Crosby distance or the generalised distance between the two subspaces is small.
The idea between the average orthogonal projection is following: we consider different orthogonal projections $\mathbf{P}_1, ..., \mathbf{P}_n$ and combine them to find the average orthogonal projection $\mathbf{P}$ based on the weights $w_1, ..., w_n$ as the projection that minimizes the objective function:

$$\sigma_w^2(\mathbf{P}) = \frac{1}{n}\sum_{i=1}^{n} D_w^2(\mathbf{P_i}, \mathbf{P})$$

where the distance function D is either the Crone and Crosby distance in case of equal ranks between the matrices $\mathbf{P}_1, ..., \mathbf{P}_n$ or the generalised distance in case the n matrices have different ranks.
To apply this method we take as original matrices the same as in the joint diagonalization algorithm: $\boldsymbol{M}_{SIR} = \boldsymbol{M}_1$, $\boldsymbol{M}_{SAVE} = \boldsymbol{M}_2$, $\boldsymbol{M}_{cov,\tau} = \boldsymbol{M}_3$. For each of this separation matrix we compute the associated projection matrix to the nonstationary subspace (the same can be done for the stationary subspace). Then we use this projection matrices as $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ and then perform AOP on this set of matrices.
This method detects different nonstationarities but we cannot say exactly if we are confronted to nonstationarity in mean, variance or covariance. As in the joint diagonalization method if we want to change the kind of nonstationarities we are mixing we can define new separation matrices and perform AOP on the new set of projection matrices. Also we do not have to limit ourself to three type of nonstationarities we could chose more.
Now that we have an overview of the different separation methods we will compare them.

Chapter 4

Evaluation

4.1 Evaluation of the separation methods performances

4.1.1 Settings

In order to test the performances of the separation methods we evaluate the distance between the true and estimated nonstationary and stationary subspaces. The six methods presented in Chapter 3 have been compared for different times series lengths.
For all settings under consideration we compare the methods for seven different times series lengths namely: 1000, 2000, 4000, 8000, 16000, 32000, 64000. We consider 8-variate times series. According to the notation in Chapter 2.1 it is equivalent to $p = 8$. We will set the number of nonstationary components to $k = 3$. This means we will be working with 8-variate time series composed of 3 nonstationary components and 5 stationary components. Note that in this first part we assume the the dimensions of the two parts are known.
We chose for the mixing matrix an orthogonal mixing matrix. Knowing the mixing matrix allows us to compute the true nonstationary and stationary subspaces. We use the distance defined in Chapter 3 to compute the distance between true and estimated subspaces. As we know that the different subspaces are of equal rank, 3 for the true and estimated nonstationary subspaces and 5 for the stationary ones, we can directly use the Crone and Crosby distance. For each method we follow Algorithm 1.
We apply Algorithm 1 for each separation method. The settings will be defined later on. We have defined three different settings: the first one is an 8-variate time series exhibiting nonstationarity in mean, the second one is an 8-variate time series whose nonstationary components are nonstationary in variance and the third setting is an 8-variate time series with nonstationarity in the covariance structure. Through these settings we want to see the performances of the algorithms defined in Chapter 3.
The R-Code for all the simulation and evaluation functions in available in Appendix in Section D. All the computations were carried out in R (version 3.6.0). We used the following R-packages JADE [12], LDRTools [14], pracma [2] and BSSasymp [9].
For all the simulation we have used an orthogonal mixing matrix. We use this type of mixing matrix because it ensures that the distance between true and estimated subspaces does not depend on the mixing matrix. This result is proven in Appendix in the Section A. More on affine invariance can be read in [5].

```
Input: Mixing matrix $\boldsymbol{A}$, separation method, time series $\boldsymbol{x}$, number of nonstationary
       components $k$.
Compute $\boldsymbol{W}^{true} = \boldsymbol{A}^{-1}$.
Use $k$, the predefined number of nonstationary components, to compute the matrices
$\boldsymbol{W}_n^{true}$ and $\boldsymbol{W}_s^{true}$.
Compute the orthogonal projection matrices $\boldsymbol{P}_n^{true}$ and $\boldsymbol{P}_s^{true}$.
Initialisation: i = 1.
for $i \leq 1000$ do
    1. Use the simulation function defined in Input to simulate one time series $\boldsymbol{x}$.
    2. Based on $\boldsymbol{x}$ compute the separation matrix as defined in the separation method chosen in
       Input.
    3. Compute $\boldsymbol{W}$ based on $\boldsymbol{M}$ which is the separation matrix obtained through the separation
       method defined in the Input (see Step 2).
    4. Use $k$, the predefined number of nonstationary components, to compute the matrices $\boldsymbol{W}_n$
       and $\boldsymbol{W}_s$.
    5. Compute the orthogonal projection matrices $\boldsymbol{P}_n$ and $\boldsymbol{P}_s$ based on $\boldsymbol{W}_n$ and $\boldsymbol{W}_s$.
    6. Compute $D_n^i = D^2(\boldsymbol{P}_n^{true}, \boldsymbol{P}_n)$ which is the squared distance between true and estimated
       nonstationary subspaces and $D_s^i = D^2(\boldsymbol{P}_s^{true}, \boldsymbol{P}_s)$ the squared distance between true and
       estimated stationary subspaces.
    7. $i = i + 1$
end
Compute the average distance between true and estimated nonstationary subspaces:
$D_n = \frac{1}{1000} \sum_{i=1}^{1000} D_n^i$.
Compute the average distance between true and estimated stationary subspaces:
$D_s = \frac{1}{1000} \sum_{i=1}^{1000} D_s^i$.
```

Algorithm 1: How to evaluate the performance of one separation method.

Setting 1: nonstationarity in mean

We describe here the simulation setting to simulate a time series whose first three components are non stationary in mean. With this simulation function we would expect the SIR function to perform better than the others. The settings we used to simulate the sources are:

- Season: MA(1) with coefficient (0.7) + c where c is a constant whose value is -1.52 for the first half of the time then 1.38 for the second half. The idea is here to have a stationary process whose mean varies depending on seasons (here 2 seasons corresponding to 2 different values). This seasonality affects the mean but not the variance of the time series. When computing the covariance between the means of the different slicing intervals we expect to see some variations indicating nonstationarity in mean.
- Season: MA(1) with coefficient (0.5) + c where c is a constant whose value is -0.75 for the first third of the time then 0.84 for the second third, -0.45 for the last third.
- Season: MA(1) with coefficient (0.3) + c where c is a constant whose value is 1 for the first quarter of the time then 2 for the second quarter, 3 for the last quarter, and eventually $c = 4$ for the last quarter.

For the five stationary components below the innovations are drawn from a normal distribution $\mathcal{N}(0, 1)$.

- z_4 = MA(2) with coefficients $(0.72, 0.24)$.
- z_5 = AR(3) with coefficients $(0.34, 0.27, 0.18)$.
- z_6 = ARMA(3,2) with coefficients $(0.34, 0.27, 0.18)$. and $(0.72, 0.15)$.

- z_7 = AR(2) with coefficients $(0.11, 0.58)$.
- z_8 = MA(1) with coefficients (0.78).

Below the first figure, Figure 4.1, represents the observed time series corresponding to the Setting 1. The second figure, Figure 4.2, represents the sources deriving from Setting 1. We see that in the time series we cannot distinguish between stationary and nonstationary components but in the sources we can already see which components belong to the signal and which ones are noises.

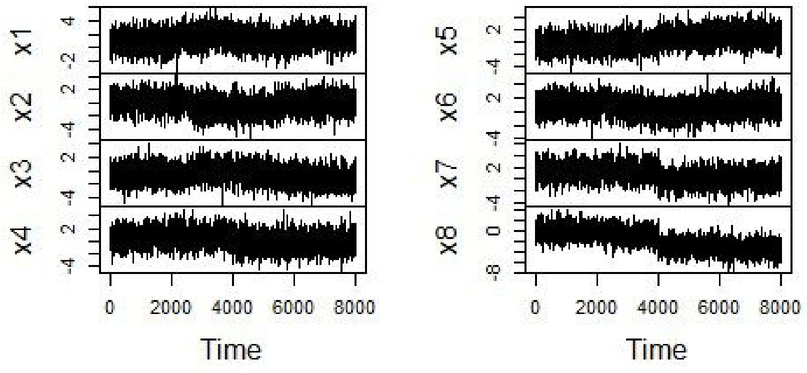

Figure 4.1: Observed time series for Setting 1 with time series length T = 8000.

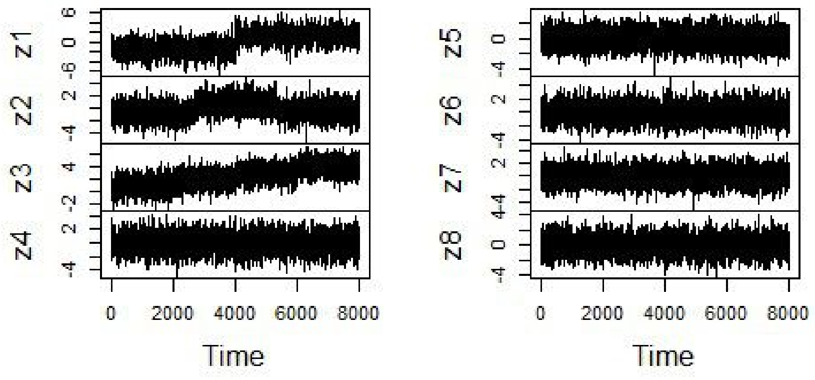

Figure 4.2: Sources for Setting 1 with time series length T = 8000.

Next we will carry out similar simulation for nonstationarity in variance and in covariance. For these simulations we will keep the same 5 stationary components and just adapt the first 3 nonstationary components so they exhibit the right kind of nonstationarity. All the plots are displayed for $\mathbb{T} = \{1, ..., 8000\}$.

Setting 2: nonstationarity in variance

We describe here the simulation setting to simulate a time series whose first three components are non stationary in variance. As mentioned before the 5 stationary components remain the same as in Setting 1.

- $z_1 = \epsilon + \alpha$ where ϵ is randomly drawn from a normal distribution $\mathcal{N}(0, 1)$ and α is a function of t defined as $\alpha(t) = 3 \times sin(\frac{t}{6 \times \pi})$ for the first third of the time interval, then for the second third $\alpha(t) = \frac{(cos(t)+sin(t))}{sin(2 \times t)}$ and eventually for the last third $\alpha(t) = 10 \times tanh(0.0001 \times t)$; then we filter the values so that all the values stay in the interval $[-30; 30]$.

- z_2 random walk: we take a series of independent random variables where each variable is either 1 or -1 with a probability of $\frac{1}{2}$ for either value and then we take the sum of these variables. The sum defines then a new series of random variables. The random walk corresponds to this newly created series. As each random variables constituting the sum has expectation 0 and variance 1 the sum has expectation 0 and is nonstationary variance. If we consider the n-th term of the series of sums it has variance n. This random walk exhibits then stationarity in mean but nonstationarity in variance.

- z_3 it is the concatenation of four normal distribution with same mean but different variances $\mathcal{N}(0,1)$, $\mathcal{N}(0,2)$, $\mathcal{N}(0,4)$ and $\mathcal{N}(0,8)$. Each block as length one quarter of the total time series length.

For the five stationary components below the innovations are drawn from a normal distribution $\mathcal{N}(0,1)$.

- z_4 = MA(2) with coefficients $(0.72, 0.24)$.
- z_5 = AR(3) with coefficients $(0.34, 0.27, 0.18)$.
- z_6 = ARMA(3,2) with coefficients $(0.34, 0.27, 0.18)$ and $(0.72, 0.15)$.
- z_7 = AR(2) with coefficients $(0.11, 0.58)$.
- z_8 = MA(1) with coefficients (0.78).

Below in Figure 4.3 and Figure 4.4 we show one realisation of Setting 2 for $\mathbb{T} = 8000$. As before the first figure, Figure 4.3, corresponds to the observed time series and it is not possible to determine which components are stationary or nonstationary in this representation. However we can see in Figure 4.4 corresponding to the sources that the first three components are nonstationary in variance. In the first component the variance depends on which third of the time interval we are located in. The second component is the random walk. Eventually in the third component the means remains constant but the variance is increasing with time.

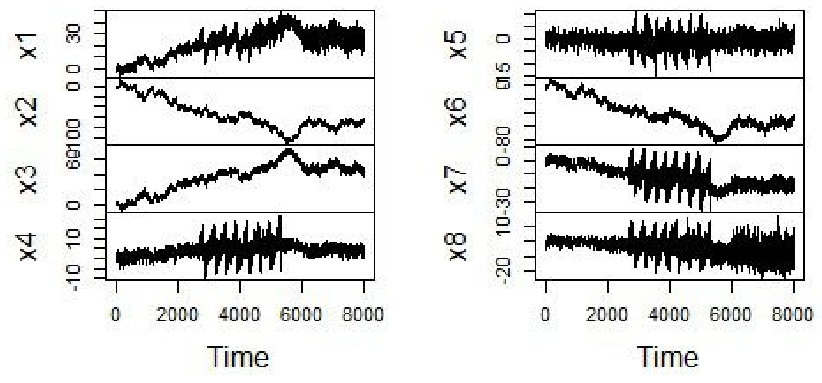

Figure 4.3: Observed time series for Setting 2 with time series length T = 8000.

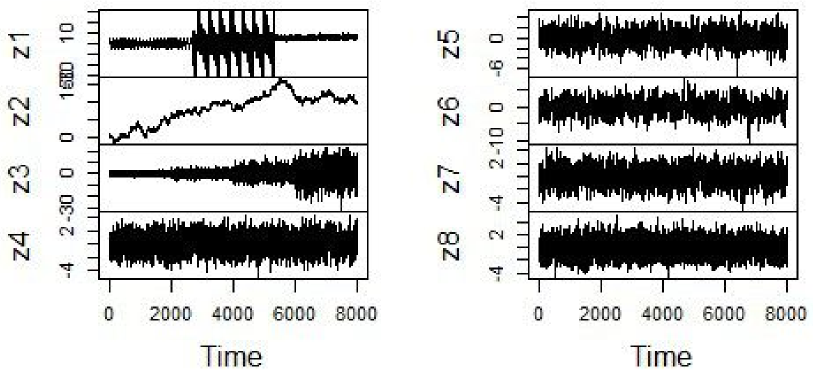

Figure 4.4: Sources for Setting 2 with time series length T = 8000.

Setting 3: nonstationarity in the covariance structure

For this next setting the idea is to simulate some changes in the covariance structure of the time series. To this extend we simulate two components z_2 and z_3. We cut z_2 into 2 parts and z_3 into 3 parts. Within a component the different parts have equal mean and variance but there are differences between the covariance structures. We get the following setting:

- $z_1 = \epsilon + 10 \times tanh(0.0001 \times t))$ where ϵ is randomly drawn from a normal distribution $\mathcal{N}(0, 1)$
- z_2 is a superposition of three different settings: for the first third AR(1) with coefficient (0.5) and innovations from $\mathcal{N}(0, 1)$, AR(1) for the second third with coefficient (0.2) and innovations from $\mathcal{N}(0, 1.28)$ and for the third part AR(1) with coefficient (0.8) and innovations from $\mathcal{N}(0, 0.48)$
- z_3 is composed of two components: MA(1) with coefficient (0.5) and innovations from $\mathcal{N}(0, 1)$ for the first part and MA(2) with coefficient $(0.9, 0.17)$ and innovations from $\mathcal{N}(0, 0.68)$ for the second part.

To compute the variances in z_2 and z_3 we used the known formula for variance in case of AR(1) or MA(1), MA(2) processes (results are written for univariate processes as one component is univariate and we assume that the stationarity conditions are fulfilled for the AR(1) process e.g. $|\theta_1| < 1$):

- MA(1): $x_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1}$ where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ then $Var(x_t) = \sigma^2(1 + \theta_1^2)$
- MA(2): $x_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$ where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ then $Var(x_t) = \sigma^2(1 + \theta_1^2 + \theta_2^2)$
- AR(1): $x_t = \mu + \theta_1 x_{t-1} + \epsilon_t$ where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ then $Var(x_t) = \frac{\sigma^2}{(1+\theta_1^2)}$

For the five stationary components below the innovations are drawn from a normal distribution $\mathcal{N}(0, 1)$.

- z_4 = MA(2) with coefficients $(0.72, 0.24)$
- z_5 = AR(3) with coefficients $(0.34, 0.27, 0.18)$
- z_6 = ARMA(3,2) with coefficients $(0.34, 0.27, 0.18)$ and $(0.72, 0.15)$
- z_7 = AR(2) with coefficients $(0.11, 0.58)$
- z_8 = MA(1) with coefficients (0.78)

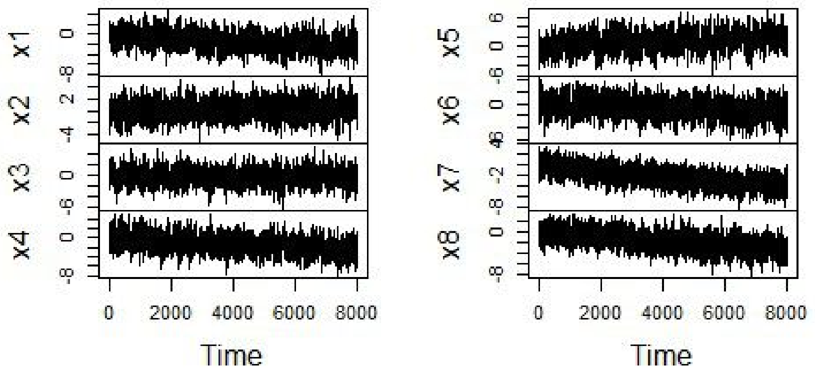

Figure 4.5: Observed time series for Setting 3 with time series length T = 8000.

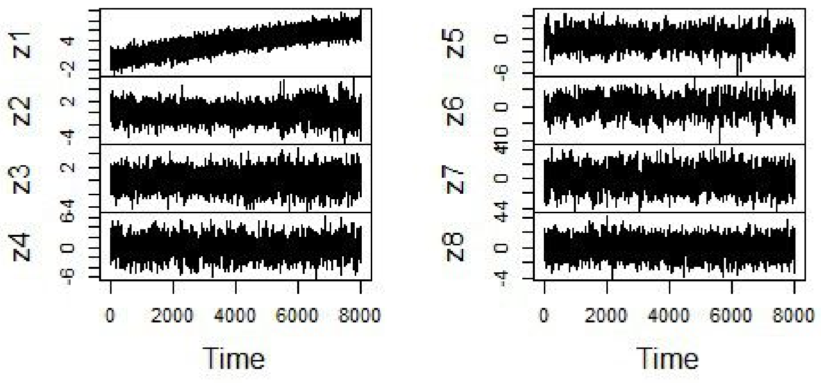

Figure 4.6: Sources for Setting 3 with time series length T = 8000.

4.1.2 Distance evaluation

We show in Appendix in Section A that the distances between true subspace and estimated subspace are equal in case of trivial mixing (mixing matrix is the identity matrix) and non trivial mixing if and only if the mixing matrix A is orthogonal.
Below we show the results of our distance evaluation. We display two plots for each setting showing the performances for the six different methods presented in Chapter 3. The numerical results are also available in Appendix in Section B. We display two curves pro separation method: one for the evaluation of the distance between true and estimated nonstationary subspace (later we may refer to this distance as nonstationary distance and we denote it by D_n in all the figures below) and the second for the evaluation of the distance between true and estimated stationary subspaces (this distance could also be called stationary distance and, in all the figures below, we refer to this distance as D_s).
The distances are evaluated as explained in Alogrithm 1. In each of the plots below we see that the distance between true and estimated subspaces decreases when the length of the time series increases. It also seems that some methods are always performing best such as the ASSA method. We would like to know when to prefer one method to another. We will show that in case of specific nonstationarity, in mean, variance or covariance, then the corresponding specific algorithm outperforms the other specific algorithm. By specific algorithms we denote algorithms such as SIR, SAVE or the procedure for nonstationarity in the covariance structure that have been specifically designed to detect and treat specific nonstationarities - respectively in mean, variance or covariance. On the other hand by combined algorithm we refer to the three following methods: joint diagonalization, ASSA and AOP that can process more than one type of nonstationarity.

Nonstationarity in mean

In Figure 4.7 and in Figure 4.8 we can see that the best performing specific separation method is the SIR algorithm. Best performing means here that the distance between true and estimated subspaces is the lowest when taking this particular algorithm. The SIR algorithm performed a satisfying separation of nonstationary and stationary components and the corresponding estimated subspaces are quite close to the true subspaces. For the combined algorithms we see they are all very close but it seems that the AOP method performs slightly better than the other one.

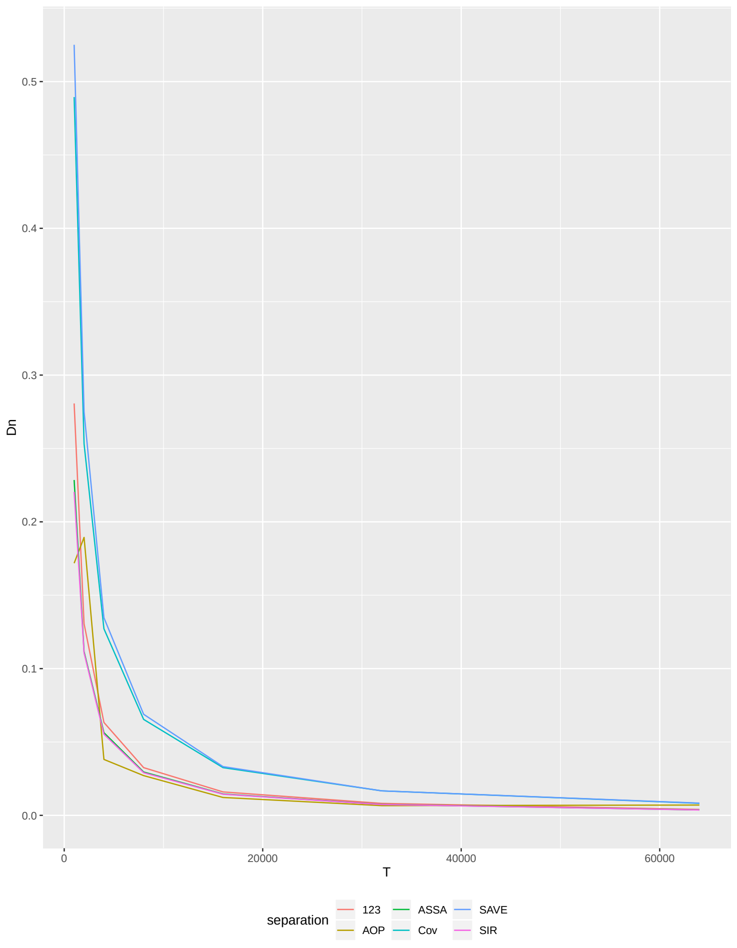

Figure 4.7: Setting 1: Evolution of the nonstationary distance D_n when the time series length increases.

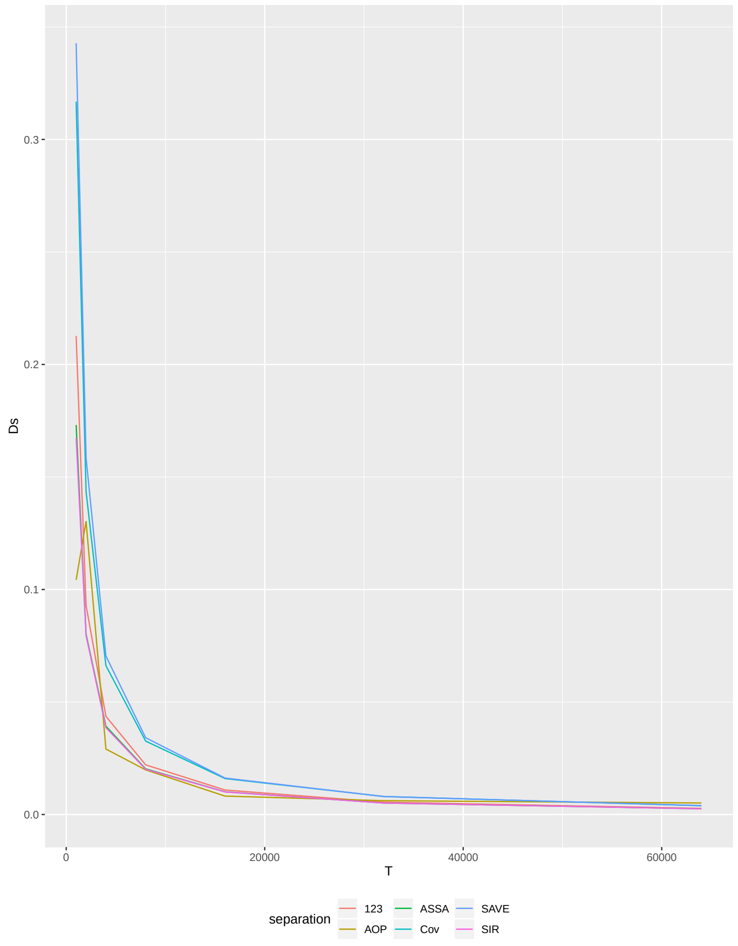

Figure 4.8: Setting 1: Evolution of the stationary distance D_s when the time series length increases.

Nonstationarity in variance

In Figure 4.9 and in Figure 4.10 it appears that the best performing specific method is, as hoped, the SAVE algorithm. Amongst the combined algorithm we can mention the AOP and the ASSA algorithms as appearing more reliable than the joint diagonalization algorithm. It is important here to underline that the SAVE algorithm performs better than the SIR one. It means that the nonstationarity in variance was detected as such.

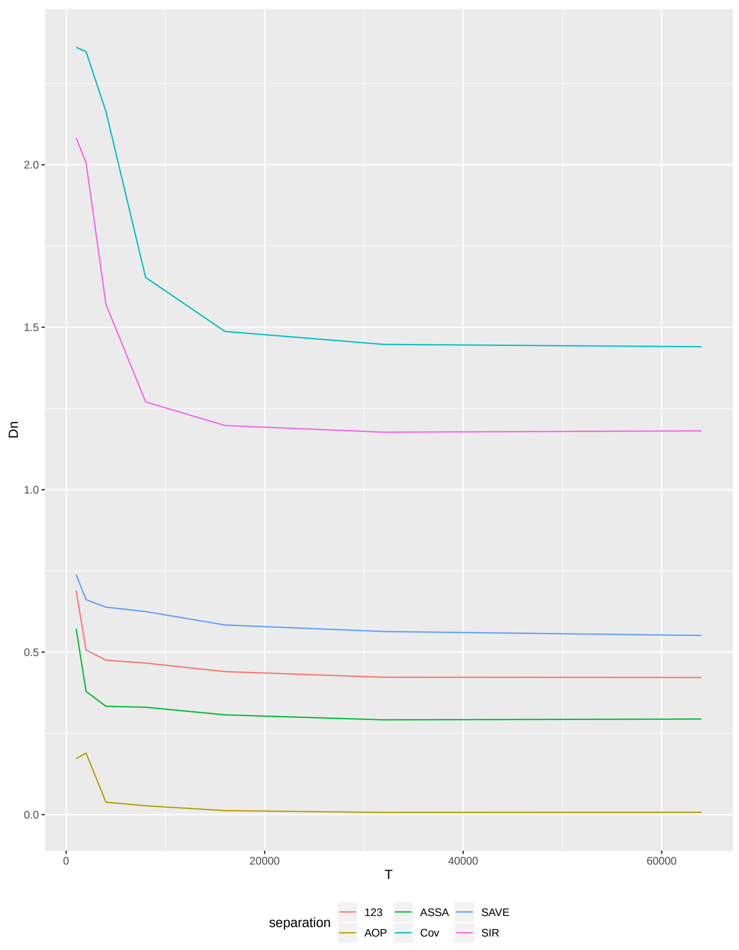

Figure 4.9: Setting 2: Evolution of the nonstationary distance D_n when the time series length increases.

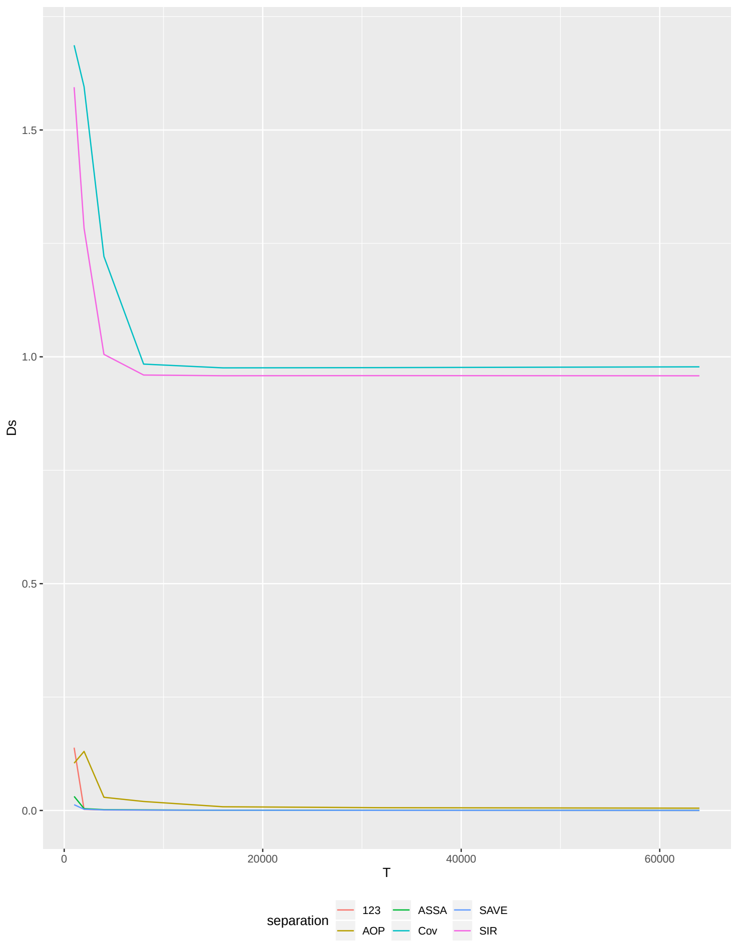

Figure 4.10: Setting 2: Evolution of the stationary distance D_s when the time series length increases.

Nonstationarity in covariance

In Figure 4.11 and in Figure 4.12 the best specific algorithm is in this case the algorithm for nonstationarity in covariance. Note that for this method and setting the results from the SIR and AOP methods are equals. It appears that for smaller time series length (also under 30000) the SAVE algorithm outperforms the algorithm for nonstationarity in the covariance structure. However when the time series length crosses 30000 then the algorithm for nonstationarity in covariance becomes the best specific one. For the combined algorithms notice here that unlike for nonstationarity in mean or in variance the AOP algorithm performs poorly. The best combined algorithm seems to be here the joint diagonalization algorithm. The ASSA method also yields good results but only for time series length above 30000.

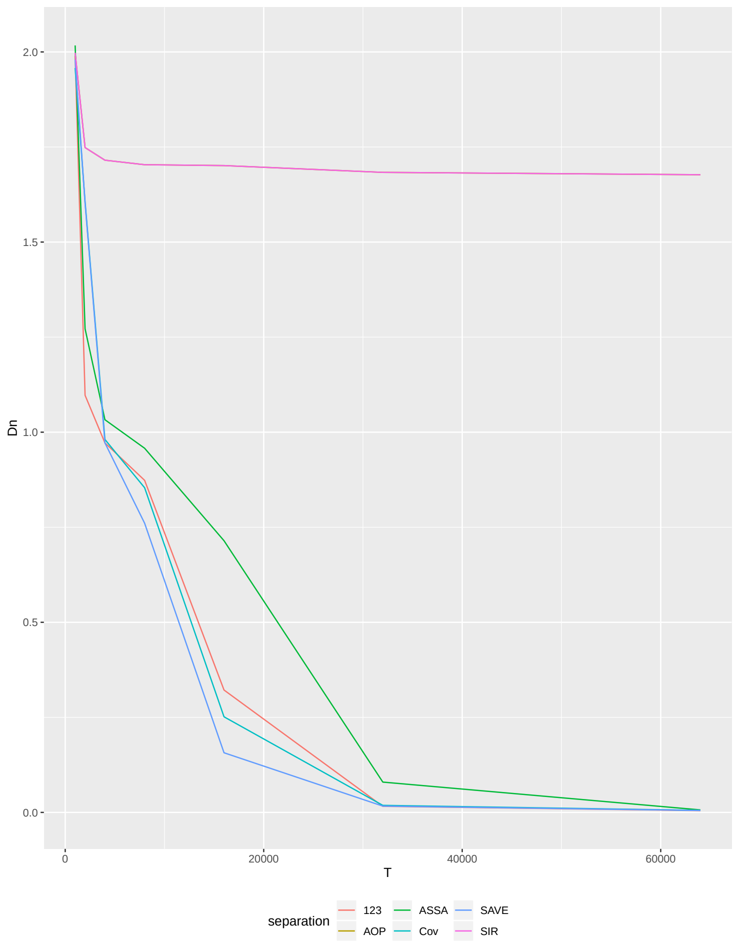

Figure 4.11: Setting 3: Evolution of the nonstationary distance D_n when the time series length increases.

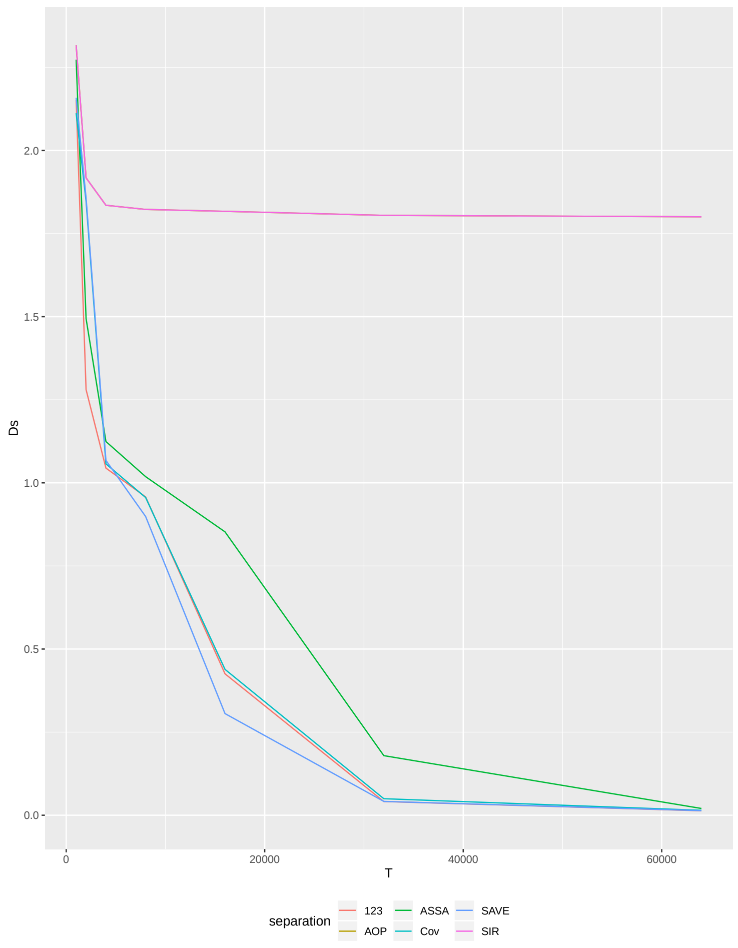

Figure 4.12: Setting 3: Evolution of the stationary distance D_s when the time series length increases.

Conclusion

We see that the performances of the different algorithms vary depending on the settings e.g. on the type of behaviours of the time series. Performances of the separation algorithms also depend on the time series length. It is important to notice that in case we have established a specific nonstationarity, for example in mean, in variance or in covariance then the algorithm corresponding (e.g. SIR in the case of nonstationarity in mean, SAVE for nonstationarity in variance and the algorithm for nonstationarity in the covariance structure) performs better than the other specific algorithms. Regarding the combined algorithm the ASSA method seems to systematically yield satisfying results (at least for time series length above 30000). We can then make the following recommandation: if we know which type of nonstationarity are underlying in our time series then we can apply the corresponding specific algorithm. In case of uncertainty regarding the type of nonstationarity or in case of combined nonstationarity then we should rather apply some combined algorithm like ASSA.

4.2 Estimation of the number of non stationary components

4.2.1 Bootstraping

To apply all the methods presented above we need to know the number of nonstationary components. This parameter is usually not known and if we have a guess regarding its value we would like to be able to test. The hypothesis we would like to test can then be formulated as:

$$H_0 : k_0 = k.$$

The value k designates here the value we think would be the most appropriate for the number of nonstationary components and k_0 designates the true number of nonstationary components.
Now that we have formulated our hypothesis we need to find a statistic to answer our question. As explained in Chapter 2.1 when we take the eigenvalues decomposition of the separation matrix, the eigenvalues corresponding to stationary components are close to 0. Thus if we are working with a p-variate time series we can chose as test statistic the variance of the $p-k$ eigenvalues of the eigenvalues decomposition of the separation matrix. This means we take the last $p-k$ diagonal elements of the matrix $\boldsymbol{D}_0$ which is the diagonal matrix containing all the eigenvalues of the separation matrix $\boldsymbol{M}$. As the $p-k$ are supposed to be stationary according to H_0, we expect this variance to be close to 0. In Algorithm 2 we describe how we test for one possible value of k. Based on one original time series $\boldsymbol{x}_0$, one separation method and the number of nonstationary components k that we wish to test, we implement the following algorithm. We execute $N = 2000$ repetitions of $m = 200$ bootstrap samples each. We use stationary non-overlapping block bootstrap to obtain our 200 bootstrap samples. We set as mean for the underlying geometric distribution $l = 50$ for all of the repetitions. We set our significance level to $\alpha = 0.05$.

```
Input: Number of Monte Carlo simulations $N$, number of bootstrap samples $m$,
       separation method, time series $\boldsymbol{x}$, number of nonstationary components $k$.
Apply the separation algorithm to the original time series $\boldsymbol{x}$ with number of nonstationary
      components $k$.
      Output: $\boldsymbol{W}^0$, $\boldsymbol{D}_0 = diag(\lambda_1, ..., \lambda_p)$.
Compute $t_0 = var(\lambda_{k+1}, ..., \lambda_p)$.
Use $k$ and $\boldsymbol{W}^0$ to compute $\boldsymbol{W}_n^0$ consisting of the first $k$ rows of $\boldsymbol{W}^0$ and $\boldsymbol{W}_s^0$ composed
with the last $p-k$ rows of $\boldsymbol{W}^0$.
Compute $\boldsymbol{z}_n^0 = (\boldsymbol{W}_n^0)^{-1}\boldsymbol{x}$ and $\boldsymbol{z}_s^0 = (\boldsymbol{W}_s^0)^{-1}\boldsymbol{x}$ respectively corresponding to the signal and
the noise.
Initialisation: i = 1.
for $i \leq N = 2000$ do
    for $j \leq m = 200$ do
        1. Compute the $j$-th stationary bootstrap sample $\boldsymbol{z}_s^j$ by applying stationary non overlapping
           block bootstrap to $\boldsymbol{z}_s^0$.
        2. Build the $j$-th bootstrap sample $\boldsymbol{x}^j$ by assembling $\boldsymbol{z}_n^0$ and $\boldsymbol{z}_s^j$.
        3. Apply the separation algorithm to the $j$-th bootstrap sample $\boldsymbol{x}^j$ with number of
           nonstationary components $k$ and store $\boldsymbol{D}^j = diag(\lambda_1^j, ..., \lambda_p^j)$ the matrix containing
           eigenvalues of the corresponding separation matrix.
        4. Compute $t^j = var(\lambda_{k+1}^j, ..., \lambda_p^j)$.
    end
    Compute $p^i = \frac{\#(t>t^0)+1}{m+1}$ for $t \in t^1...t^{2000}$.
end
Count how many p values are under the significance level $\alpha = 0.05$ e.g. how many times
$H_0$ was rejected.
```

Algorithm 2: Semiparametric bootstrap test for $H_0 : k = k_0$.

The R-Code corresponding to Algorithm 2 as well as the code used for step 1 within the second loop are available in Appendix in Section D.
We expect that for the true number of nonstationary components ($k = k_0$) e.g. under H_0 then only 5% of the p-values are under α; for $k > k_0$ then the majority of the p-values should be below the α-level; and for $k < k_0$ we hope the p-values to be quite close to 1 (the test rejects quite systematically H_0). In this last case all the p-value should be at least above the α-level.

4.2.2 Results

For hypothesis testing we chose the following settings: we use Setting 1 and the SIR algorithm. Note that if we chose a poor separation algorithm (e.g. the difference between the eigenvalues of the separation matrix corresponding to the last nonstationary component and the first stationary component, in our case the difference between the third and the fourth eigenvalues, is too small) then the test will perform poorly and results will not meet the chosen significance level.
We present here the results for the SIR method. For the other methods the analysis would be the same. The code for this simulation is also available in Appendix in Section D.
The simulation was conducted with a 8-variate time series exhibiting nonstationarity in mean simulated as described in Setting 1 and 3 nonstationary components. We made 2000 repetitions and for each of this repetition 200 bootstrap samples were computed. During the whole procedure the mean for the geometric distribution used for block bootstrap is $l = 50$. We chose as α-level $\alpha = 0.05$. The process was repeated for five different time series length: 1000, 2000, 4000, 8000, 16000. Below, in Figure 4.13 we present the results for this procedure.

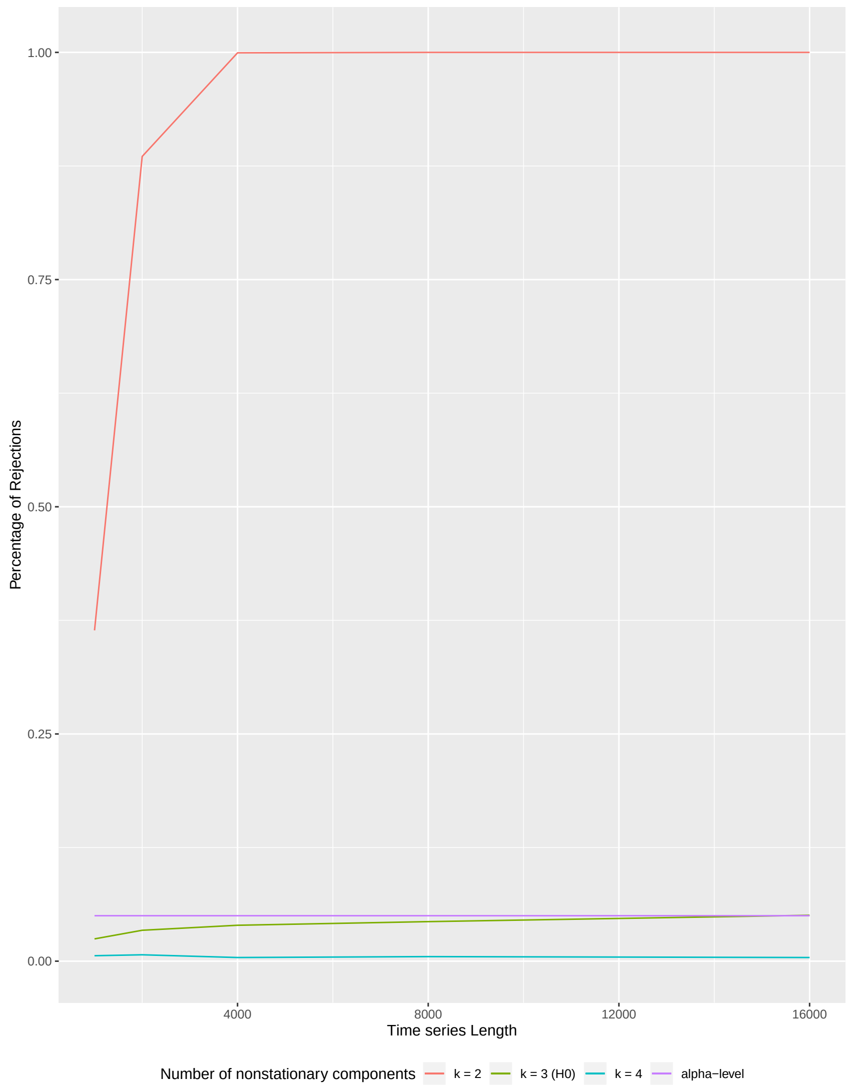

Figure 4.13: Rejection frequencies the SIR methods over 2000 repetitions with 200 bootstrap samples each.

In Figure 4.13 the α-level is represented through the purple horizontal line set to 0.05. The red line correspond to the case $k = 2$ and we see that as hoped the percentage of rejections for this case is often close to 1 and always well above the α-level. This case demonstrates the power of our test: the probability to reject H_0 is high when H_0 is false. The green line displays the results for the case corresponding to H_0: $k = 3$. We see that the number of rejections is very close to the

α-level as it should be. The blue line illustrates the results for $k = 4$. In this case we are always under the α-level and as expected also always under the number of rejection for the case $k = 3$. The numerical results are available in Appendix in Section C.
For other simulation settings or other methods the methodology would be the same and we hope that the results would also be very similar.

Conclusion

On the one hand we see that under the null-hypothesis, corresponding to the settings where $k = 3$ the number of rejections by the test is well around the α-level set to 0.05%. In case where the H_0-hypothesis is not respected, e.g. in case where $k = 2$ we see that the percentage of rejections is very close to or equal to 1. Through this case we can determine the power of our test which is the probability of rejecting the H_0-hypothesis knowing that it is incorrect. For the third and last case $k = 4$ we have shown that this case was somewhat included into the H_0-hypothesis: when we take 4 eigenvalues instead of 3 as we have really only three nonstationary components then the fourth eigenvalue will statistically be very close to 0 so will not contribute much to the variance.
Knowing how this test performs we can also use it if we do not know what the number of non-stationary components might be. We test for all the possible values and then chose the values at which the test start rejecting less. For example in our case the switch would happen between $k = 2$ and $k = 3$ so we would probably chose $k = 3$.

Chapter 5

Conclusion

We showed in this work that we dispose of at least six different separation methods to separate nonstationary components from stationary components within a time series. We evaluated these methods and established that one of them, namely ASSA seems to be performing better than the others no matter what the setting is. As we do not always know with which kind of nonstationarity we are dealing we would then recommend using this method in case of uncertainty. In case we are dealing with specific nonstationarity in mean, variance or covariance, we have shown that the specific algorithms yield very satisfying results.

But before separating the components we also need to know how many components we need to isolate. For this we came up with a statistical test: if we assume that the p-variate time series we are studying have k nonstationary components (H_0: the number of nonstationary components is k) then based on the variance of the $p-k$ last eigenvalues of the separation matrix as test statistic we can test whether the data support this hypothesis or not.

Further steps would be to apply the semiparametric bootstrap testing to other methods. We only tested it for the SIR method here but it should also work for the other separation methods presented in this work. The principle would stay the same but it could be interesting to explicitly see the numerical results and performances for these methods.

One other question one could ask is how to detect more in details the types of nonstationarity observed. For example if our data are exhibiting nonstationarity in mean then it would be nice to know whether it is a trend, a cyclic or seasonal behaviour. Eventually we could think of some enhanced way to detect combined nonstationarities and to disclose the processes behind these nonstationarities.

At the end of the day we tested our separation algorithms and test process on simulated data but we could also try to implemented them for real life phenomena.

Chapter 6

Appendix

A Demonstations

We prove here that the distance in case of non-trivial and trivial mixing matrices are the same if and only if the mixing matrix is orthogonal.
Let $\boldsymbol{A}$ be the mixing matrix, $\boldsymbol{x}_t$ the observed signal at time t and $\boldsymbol{z}_t$ the corresponding sources.
To prove that if the distances are equal then the mixing matrix is orthogonal we use the contra-positive. The computation shows that if the mixing matrix is not orthogonal then the distances are not equal. Just take any random invertible non orthogonal non mixing matrix and try to compute the distance between true and estimated projector in case of non-trivial mixing (through the selected mixing matrix) and in case of trivial mixing. They will not be equal.
Now let assume that the mixing matrix $\boldsymbol{A}$ is orthogonal. We denote by $\boldsymbol{x}_t$ the signal obtained through the orthogonal mixing matrix $\boldsymbol{A}$ and by $\boldsymbol{x}_t^*$ the corresponding signal but without the mixing. We can write:

$$[B2P(\boldsymbol{W}\hat{(}\boldsymbol{x}_t)) - B2P(\boldsymbol{W}(\boldsymbol{x}_t))][B2P(\boldsymbol{W}\hat{(}\boldsymbol{x}_t)) - B2P(\boldsymbol{W}(\boldsymbol{x}_t))]^t$$

$$= ((\boldsymbol{A}^t)^{\frac{1}{2}})^{-1}[B2P(\boldsymbol{W}\hat{(}\boldsymbol{x}_t^*)) - B2P(\boldsymbol{W}(\boldsymbol{x}_t^*))][B2P(\boldsymbol{W}\hat{(}\boldsymbol{x}_t^*)) - B2P(\boldsymbol{W}(\boldsymbol{x}_t^*))]^t](\boldsymbol{A}^t)^{\frac{1}{2}}.$$

Then if we take the trace of each side of the equation they are equal and we get that

$$||B2P(\boldsymbol{W}(\boldsymbol{x}_t))][B2P(\boldsymbol{W}\hat{(}\boldsymbol{x}_t)) - B2P(\boldsymbol{W}(\boldsymbol{x}_t))||_F = ||B2P(\boldsymbol{W}\hat{(}\boldsymbol{x}_t^*)) - B2P(\boldsymbol{W}(\boldsymbol{x}_t^*))||_F.$$

B Distance Evaluation Numerical Results

In the three tables below, Table 6.1, 6.2 , 6.2, we denote by D_n the distance between true and estimated non-stationary subspaces and by D_s the distance between true and estimated stationary subspaces. The distances are computed according to Chapter 3. T designates here the time series length.

Separation Method	T	D_n	D_s
SIR	1000	0,2204	0,1673
	2000	0,1106	0,0795
	4000	0,0553	0,0386
	8000	0,0291	0,0201
	16000	0,0144	0,01
	32000	0,0074	0,0051
	64000	0,0037	0,0025
SAVE	1000	0,5251	0,3428
	2000	0,275	0,1585
	4000	0,1347	0,0704
	8000	0,0689	0,0341
	16000	0,0332	0,0162
	32000	0,0167	0,008
	64000	0,0083	0,004
Covariance	1000	0,4894	0,3168
	2000	0,2532	0,1435
	4000	0,1271	0,0662
	8000	0,0654	0,0326
	16000	0,0325	0,016
	32000	0,0167	0,008
	64000	0,0082	0,0039
Joint Diagonalization	1000	0,2807	0,2127
	2000	0,1305	0,0929
	4000	0,0633	0,0436
	8000	0,0325	0,0221
	16000	0,016	0,0109
	32000	0,0081	0,0055
	64000	0,004	0,0027
ASSA	1000	0,2285	0,1731
	2000	0,1115	0,0801
	4000	0,0563	0,0392
	8000	0,0295	0,0203
	16000	0,0146	0,0101
	32000	0,0075	0,0051
	64000	0,0037	0,0025
AOP	1000	0,1718	0,1043
	2000	0,1894	0,1302
	4000	0,0381	0,0291
	8000	0,0271	0,0198
	16000	0,0122	0,0082
	32000	0,0066	0,0061
	64000	0,007	0,0051

Table 6.1: Distance evaluation for Setting 1.

Separation Method	T	D_n	D_s
SIR	1000	2,0829	1,5943
	2000	2,0065	1,2834
	4000	1,5707	1,0054
	8000	1,2702	0,9599
	16000	1,1975	0,9583
	32000	1,177	0,9586
	64000	1,181	0,9582
SAVE	1000	0,7388	0,0127
	2000	0,6614	0,0028
	4000	0,6384	0,0012
	8000	0,6249	0,0008
	16000	0,5835	0,0005
	32000	0,5634	0,0002
	64000	0,5513	0,0001
Dovariance	1000	2,3618	1,6868
	2000	2,3481	1,5954
	4000	2,1642	1,2212
	8000	1,6533	0,9839
	16000	1,487	0,9756
	32000	1,4472	0,9762
	64000	1,44	0,9779
Joint Diagonalization	1000	0,6894	0,1384
	2000	0,5066	0,0023
	4000	0,4754	0,0011
	8000	0,4662	0,0009
	16000	0,4401	0,0005
	32000	0,4227	0,0002
	64000	0,4219	0,0001
ASSA	1000	0,5719	0,031
	2000	0,379	0,0041
	4000	0,3333	0,0018
	8000	0,3304	0,0013
	16000	0,3069	0,0006
	32000	0,2916	0,0003
	64000	0,294	0,0001
AOP	1000	0,1718	0,1043
	2000	0,1894	0,1302
	4000	0,0381	0,0291
	8000	0,0271	0,0198
	16000	0,0122	0,0082
	32000	0,0066	0,0061
	64000	0,007	0,0051

Table 6.2: Distance evaluation for Setting 2.

Separation Method	T	D_n	D_s
SIR	1000	1,9977	2,3168
	2000	1,7485	1,9176
	4000	1,7153	1,8352
	8000	1,7034	1,8227
	16000	1,7011	1,817
	32000	1,6833	1,8048
	64000	1,6771	1,8005
SAVE	1000	1,9796	2,1585
	2000	1,6024	1,8579
	4000	0,9719	1,0676
	8000	0,7605	0,8988
	16000	0,1569	0,3058
	32000	0,0166	0,0416
	64000	0,0051	0,0138
Covariance	1000	1,9578	2,1126
	2000	1,6063	1,8474
	4000	0,9809	1,0572
	8000	0,8539	0,9559
	16000	0,2515	0,4388
	32000	0,0186	0,0497
	64000	0,0054	0,0148
Joint Diagonalization	1000	1,9896	2,1516
	2000	1,097	1,2805
	4000	0,972	1,0444
	8000	0,8733	0,9577
	16000	0,3219	0,4254
	32000	0,0161	0,0412
	64000	0,0048	0,0134
ASSA	1000	2,0171	2,2732
	2000	1,2719	1,4942
	4000	1,0328	1,1244
	8000	0,9578	1,0187
	16000	0,7145	0,8526
	32000	0,0798	0,1793
	64000	0,0066	0,0203
AOP	1000	1,9977	2,3168
	2000	1,7485	1,9176
	4000	1,7153	1,8352
	8000	1,7034	1,8227
	16000	1,7011	1,817
	32000	1,6833	1,8048
	64000	1,6771	1,8005

Table 6.3: Distance evaluation for Setting 3.

C Bootstrap Hypothesis Testing Numerical Results

In Table 6.4 we present the rejection frequencies of our semiparametric bootstrap test to determine whether the number of non-stationary components was correctly determined. Our null-hypothesis is: $H_0 : k = k_0$.

T	Number of Rejections		
	$k = 2$	$k = 3 = k_0\ (H_0)$	$k = 4$
1000	0.364	0.0245	0.006
2000	0.8855	0.034	0.007
4000	0.9995	0.0395	0.004
8000	1 00	0.0435	0.005
16000	1.00	0.0505	0.004

Table 6.4: Percentage of Rejection of $H_0 : k = k_0$.

D R-Code

Bootstrap Strategy

In our simulation we will use the following function to perform non-stationary non-overlapping block bootstrap from our times series. $\boldsymbol{y}$ is the input time series we want to bootstrap from and $\frac{1}{l}$ is the mean of the geometric distribution we sample the block lengths from.

Listing 6.1: R code – Bootstrap Sampling

```
nos1 <- function(y, l) {
 NR <- nrow(y)
 NC <- ncol(y)
 ybt <- rep(0, NC)
 L <- numeric()
 j <- 1
 # create the block lengths
 while(sum(L)<=(NR+1)) {
   L[j] <- rgeom(1,1/l)
   j <- j+1
   }
 nbBlocks <- length(L)
 # sample from the blocks
 # cntRows <- c(nrow(ybt))
 cntRows <- c(NULL)
 while (sum(cntRows) < NR+1) {
   a <- sample(c(1:nbBlocks), 1)
   if (a>1) {
     L_aux <- L[1:(a-1)]
     s <- cumsum(L_aux)[a-1]
     } else {
     L_aux <- L[1]
     s <- 0
     }
   ind_start <- s+1
   ind_end <- s+L[a]
   if(ind_end <= NR) {
     ybt <- rbind(ybt, y[ind_start:ind_end,])
     } else {
     if (ind_start > NR) {
       ind_start <- NR
       } else {
       ybt <- rbind(ybt, y[ind_start:NR,])
       }
     }
   cntRows <- c(nrow(ybt))
   }
```

```
  ybt <- as.ts(ybt[2:(NR+1),])
  return(list(boot_sample = ybt, blocks_lenghts = L))
  }
```

The function below computes the test statistics of the bootstrap samples (created with the above function) as well as the p-value for testing H_0: the number of non-stationary component is k | H_1: the number of non-stationary component is not k For a p-variate time series the test statistic used is the variance of the $k+1$ to p eigenvalues of the eigenvalue decomposition of the separation matrix. It takes as input:

Mix	Mixing matrix to be used in the simulation function sim_fun
sim_fun	simulation function to simulate the time serie
n	length of the simulated time serie
k	number of non-stationary components in the simulated time-serie according to H_0
m	number of bootstrap repetitions
l	$\frac{1}{l}$ is the mean of the geometric distribution used to sample the block lengths
bt_fun	bootstrap function used to create the bootstrap samples
sep_fun	separation function
K, tau, $n.cuts$	parameters of the separation function

Listing 6.2: R code – Bootsrap Hypothesis Testing

```
boot_fun <- function(Mix, n, k, m, l, sim_fun, bt_fun, sep_fun, K, tau, n.cuts) {
  x0 <- sim_fun(n, Mix)$xnt
  p <- dim(x0)[2]
  MEAN0 <- colMeans(x0)
  xc <- sweep(x0, 2, MEAN0, "-")
  S0 <- sep_fun(x0, K, tau, n.cuts)
  D0 <- S0$eval
  W0 <- S0$W
  W0inv <- solve(W0)
  z0 <- tcrossprod(xc, W0)
  t0 <- var(D0[-(1:k)])
  z0n <- z0[,0:k, drop=FALSE]
  z0s <- z0[,-(0:k), drop=FALSE]
  t <- numeric(m)
  for (i in 1:m) {
    zs_sample <- bt_fun(z0s, l)$boot_sample
    z_bt <- as.ts(cbind(z0n, zs_sample))
    x_bt <- tcrossprod(z_bt, W0inv)
    x_bt <- sweep(x_bt, 2, MEAN0, "+")
    D_bt <- sep_fun(x_bt, K, tau, n.cuts)$eval
    t[i] <- var(D_bt[-(1:k)])
    }
  p <- (sum(t>t0)+1)/(m+1)
  return(list(t0 = t0, t=t, p=p))
  }
```

The function returns the test statistic $t0$ of the original time series, the vector of the bootstrap statistics t and the p-value p.

Separation Functions

Listing 6.3: R code – SIR method - Non stationarity in mean

```
MSIR <- function(X, K, tau, n.cuts) {
  MEAN <- colMeans(X)
  p <- ncol(X)
  COV <- cov(X)
  EVD <- eigen(COV, symmetric =TRUE)
  COV.sqrt.i <- EVD$vectors %*% (diag(EVD$values^(-0.5))) %*% t(EVD$vectors)
  X.C <- sweep(X, 2, MEAN, "-")
  Y <- tcrossprod(X.C, COV.sqrt.i)
  n <- nrow(Y)
  if (is.null(n.cuts)) n.cuts <- ceiling(seq(1, n, length = K + 1))
  N.cuts <- n.cuts + c(rep(0, K), 1)
  COV <- array(0, dim = c(p, p, K))
  for (i in 1:K) {
    YK <- Y[N.cuts[i]:(N.cuts[i + 1] - 1),]
    slicemean <- colMeans(YK)
    COV[, , i] <- nrow(YK)*tcrossprod(slicemean)/n
    }
  M <- apply(COV, 1:2, sum)
  EVD <- eigen(M, symmetric = TRUE)
  W <- crossprod(EVD$vectors, COV.sqrt.i)
  S <- tcrossprod(X.C, W)
  S <- ts(S, names = paste("Series", 1:p))
  RES <- list(M = M, eval = EVD$values, W = W)
  RES
  }
```

Listing 6.4: R code – SAVE method - Non-stationarity in variance

```
MSAVE <- function(X, K, tau, n.cuts) {
  MEAN <- colMeans(X)
  p <- ncol(X)
  COV <- cov(X)
  EVD <- eigen(COV, symmetric =TRUE)
  COV.sqrt.i <- EVD$vectors %*% (diag(EVD$values^(-0.5))) %*% t(EVD$vectors)
  X.C <- sweep(X, 2, MEAN, "-")
  Y <- tcrossprod(X.C, COV.sqrt.i)
  n <- nrow(Y)
  p <- ncol(Y)
  if (is.null(n.cuts)) n.cuts <- ceiling(seq(1, n, length = K + 1))
  N.cuts <- n.cuts + c(rep(0, K), 1)
  COV <- array(0, dim = c(p, p, K))
  for (i in 1:K) {
    YK <- Y[N.cuts[i]:(N.cuts[i + 1] - 1),]
    slicevar <- crossprod(YK)/nrow(YK)
    COV[, , i] <- nrow(YK)*(diag(p) - slicevar)%*%t(diag(p) - slicevar)/n
    }
  M <- apply(COV, 1:2, sum)
  EVD <- eigen(M, symmetric = TRUE)
  W <- crossprod(EVD$vectors, COV.sqrt.i)
  S <- tcrossprod(X.C, W)
  S <- ts(S, names = paste("Series", 1:p))
  RES <- list(M = M, eval = EVD$values, W = W)
  RES
  }
```

Listing 6.5: R code – Non-stationarity in the covariance structure

```
actau <- function(Y, tau) {
  n <- nrow(Y)
  Yt <- Y[1:(n - tau), ]
  Yti <- Y[(1 + tau):n, ]
  crossprod(Yt, Yti)/nrow(Yt)
  }

Mcovtau <- function(X, K, tau, n.cuts) {
  MEAN <- colMeans(X)
  p <- ncol(X)
  COV <- cov(X)
  EVD <- eigen(COV, symmetric =TRUE)
  COV.sqrt.i <- EVD$vectors %*% (diag(EVD$values^(-0.5))) %*% t(EVD$vectors)
  X.C <- sweep(X, 2, MEAN, "-")
  Y <- tcrossprod(X.C, COV.sqrt.i)
  n <- nrow(Y)
  p <- ncol(Y)
  if (is.null(n.cuts)) n.cuts <- ceiling(seq(1, n, length = K + 1))
  N.cuts <- n.cuts + c(rep(0, K), 1)
  S <- array(0, dim = c(p, p, K))
  Sall <- actau(Y, tau = tau)
  for (i in 1:K) {
    slice <- Y[N.cuts[i]:(N.cuts[i + 1] - 1), ]
    sli.length <- nrow(slice)
    S[, , i] <- sli.length*(Sall - actau(slice, tau = tau))
            %*%t(Sall - actau(slice, tau = tau))/n
    }
  M <- apply(S, 1:2, sum)
  EVD <- eigen(M, symmetric = TRUE)
  W <- crossprod(EVD$vectors, COV.sqrt.i)
  S <- tcrossprod(X.C, W)
  S <- ts(S, names = paste("Series", 1:p))
  RES <- list(M = M, eval = EVD$values, W = W)
  RES
  }
```

Listing 6.6: R code – ASSA method - Mixed non-stationarities in mean and variance

```
ASSA <- function (X, K, tau, n.cuts) {
  n <- nrow(X)
  MEAN <- colMeans(X)
  COV <- cov(X)
  EVD <- eigen(COV, symmetric = TRUE)
  COV.sqrt.inv <- EVD$vectors %*% (diag(EVD$values^(-0.5))) %*% t(EVD$vectors)
  X.C <- sweep(X, 2, MEAN, "-")
  Y <- tcrossprod(X.C, COV.sqrt.inv)
  p <- ncol(X)
  if (is.null(n.cuts)) {
    n.cuts <- ceiling(seq(1, n, length = K + 1))
    } else {
    K <- length(n.cuts) - 1
    }
  SStmats <- array(0, dim = c(p, p, K))
  MMtmats <- array(0, dim = c(p, p, K))
  N.cuts <- n.cuts + c(rep(0, K), 1)
  for (i in 1:K) {
    Yint <- Y[N.cuts[i]:(N.cuts[i + 1] - 1),]
    nint <- nrow(Yint)
    MMtmats[, , i] <- tcrossprod(colMeans(Yint))
    Smati <- crossprod(Yint)/nint
    SStmats[, , i] <- 0.5 * tcrossprod(Smati)
    }
  M <- (apply(MMtmats, 1:2, sum) + apply(SStmats, 1:2, sum))/K - 0.5*diag(p)
  Meigen <- eigen(M, symmetric = TRUE)
  W <- crossprod(Meigen$vectors, COV.sqrt.inv)
  S <- tcrossprod(X.C, W)
  S <- ts(S, names = paste("Series", 1:p))
  RES <- list(W = W, eval = Meigen$values, n.cut = n.cuts, K = K, S = S)
  RES
  }
```

Listing 6.7: R code – Joint Diagonalization - Mixed non-stationarities mean

```
M123 <- function (X, K, tau, n.cuts) {
  eps = 1e-6
  maxiter = 2000
  n <- nrow(X)
MEAN <- colMeans(X)
COV <- cov(X)
EVD <- eigen(COV, symmetric = TRUE)
COV.sqrt.i <- EVD$vectors %*% (diag(EVD$values^(-0.5))) %*% t(EVD$vectors)
X.C <- sweep(X, 2, MEAN, "-")
Y <- tcrossprod(X.C, COV.sqrt.i)
p <- ncol(X)
R <- array(0, dim = c(p, p, 3))
R[, , 1] <- MSIR(X, K, tau, n.cuts)$M
R[, , 2] <- MSAVE(X, K, tau, n.cuts)$M
R[, , 3] <- Mcovtau(X, K, tau, n.cuts)$M
JD <- frjd(R, eps = eps, maxiter = maxiter)
D <- JD$D
sumdg <- diag(apply(D, 1:2, sum))
ord <- order(sumdg, decreasing = TRUE)
V <- JD$V[,ord]
W <- crossprod(V, COV.sqrt.i)
S <- tcrossprod(X.C, W)
S <- ts(S, names = paste("Series", 1:p))
RES <- list(W = W, eval = D, V = V, tau = tau, R = R)
RES
}
```

Distance Computation

Listing 6.8: R code – Simulation function for separation methods other than AOP

```
M_sim <- function(n, m, seed1, sim, k, sep_fun, Mix) {
K <- 11
tau <- 1
n.cuts <- NULL
D <- matrix(0, nrow = m, ncol = 2)
UnMix <- solve(Mix)
Wtrue_ns <- UnMix[1:k, ]
Ptrue_ns <- B2P(t(Wtrue_ns))
Wtrue_s <- UnMix[-(1:k),]
Ptrue_s <- B2P(t(Wtrue_s))
set.seed(seed1)
for (i in 1:m) {
  X <- sim(n, Mix)
  S <- sep_fun(X, K, tau, n.cuts)
  W <- S$W
  Wn <- W[1:k,]
  Ws <- W[-(1:k),]
  Pn <- B2P(t(Wn))
  Ps <- B2P(t(Ws))
  dn <- Pdist(list(Ptrue_ns, Pn), weights = "constant")
  ds <- Pdist(list(Ptrue_s, Ps), weights = "constant")
  D[i,] <- c(dn, ds)
  }
colnames(D) <- c("non␣stationary", "stationary")
MEAN <- colMeans(D)
#return(list(D=D, MEAN=MEAN))
return(MEAN)
}
```

Listing 6.9: R code – Simulation function for AOP

```
AOP_sim <- function(n, m, seed1, sim, k, Mix) {
K <- 11
tau <- 1
n.cuts <- NULL
```

```
  D <- matrix(0, nrow = m, ncol = 2)
  UnMix <- solve(Mix)
  Wtrue_ns <- UnMix[1:k, ]
  Ptrue_ns <- B2P(t(Wtrue_ns))
  Wtrue_s <- UnMix[-(1:k),]
  Ptrue_s <- B2P(t(Wtrue_s))
  set.seed(seed1)
  for (i in 1:m) {
    # simulation
    X <- sim(n, Mix)
    # separation
    W1 <- MSIR(X, K, tau, n.cuts)$W
    W2 <- MSAVE(X, K, tau, n.cuts)$W
    W3 <- Mcovtau(X, K, tau, n.cuts)$W
    W1_ns <- W1[1:k,]
    W2_ns <- W2[1:k,]
    W3_ns <- W3[1:k,]
    W1_s <- W1[-(1:k),]
    W2_s <- W2[-(1:k),]
    W3_s <- W3[-(1:k),]
    # orthogonal projections
    P1_ns <- B2P(t(W1_ns))
    P2_ns <- B2P(t(W2_ns))
    P3_ns <- B2P(t(W3_ns))
    P1_s <- B2P(t(W1_s))
    P2_s <- B2P(t(W2_s))
    P3_s <- B2P(t(W3_s))
    # AOP
    Pn <- AOP(list(P1_ns, P2_ns, P3_ns), "constant")$P
    Ps <- AOP(list(P1_s, P2_s, P3_s), "constant")$P
    dn <- Pdist(list(Ptrue_ns, Pn), weights = "constant")
    ds <- Pdist(list(Ptrue_s, Ps), weights = "constant")
    D[i,] <- c(dn, ds)
    }
  colnames(D) <- c("non␣stationary", "stationary")
  MEAN <- colMeans(D)
  #return(list(D=D,MEAN=MEAN))
  return(MEAN)
  }
```

Simulation Functions

The functions below are used to simulate times series with specific types of non-stationarities. They take as arugment the time serie length n and a mixing matrix A. They return the time serie without mixing (equivalent to the case where the mixing matrix would be the identity matrix) and the time serie with mixing. All the times series are 8-variates times series.

Listing 6.10: R code – Simulation function for a time serie with non-stationarity in mean

```
simSIR <- function(n, A) {
  z1 <- arima.sim(n, model = list(ar = 0.7)) +
  rep(c(-1.52, 1.38),c(floor(n*0.5), floor(n*0.5)))
  z2 <- arima.sim(n, model = list(ar = 0.5)) +
  rep(c(-0.75, 0.84, -0.45), c(floor(n/3), floor(n/3), n-2*floor(n/3)))
  z3 <- arima.sim(n, model = list(ar = 0.3)) +
  rep(c(1, 2, 3, 4), c(floor(n/4), floor(n/4), floor(n/4), floor(n/4)))
  z4 <- arima.sim(n, model = list(ma = c(0.72, 0.24)))
  z5 <- arima.sim(n, model = list(ar = c(0.34, 0.27, 0.18)))
  z6 <- arima.sim(n, model = list(ar = c(0.34, 0.27, 0.18), ma = c(0.72, 0.15)))
  z7 <- arima.sim(n, model = list(ar = c(0.11, 0.58)))
  z8 <- arima.sim(n, model = list(ma = 0.78))
  z <- cbind(z1,z2, z3, z4, z5, z6, z7, z8)
  xnt <- tcrossprod(z, A)
  xt <- z
  return(list(xnt = as.ts(xnt), xt = as.ts(xt)))
  }
```

Listing 6.11: R code – Simulation function for a time serie with non-stationarity in variance

```
simSAVE <- function(n, A) {
  a1 <- 3*(sin((1:floor(n/3))/(6*pi)))
  a2 <- (cos(2*((1+floor(n/3)):(2*floor(n/3)))) +
  sin((1+floor(n/3)):(2*floor(n/3))))/sin(2*((1+floor(n/3)):(2*floor(n/3))))
  a3 <- 10*tanh(0.0001*((2*floor(n/3)+1):n))
  z1 <- rnorm(n) + c(a1, a2, a3)
  z1 <- sapply(z1, function(y) min(max(y,-30),30))
  z2 <- rw_fun(n)
  z3 <- c(rnorm(floor(n/4)),
        rnorm(floor(n/4), 0, 2),
        rnorm(floor(n/4), 0, 4),
        rnorm((n-3*floor(n/4)), 0, 8))
  z4 <- arima.sim(n, model = list(ma = c(0.72, 0.24)))
  z5 <- arima.sim(n, model = list(ar = c(0.34, 0.27, 0.18)))
  z6 <- arima.sim(n, model = list(ar = c(0.34, 0.27, 0.18), ma = c(0.72, 0.15)))
  z7 <- arima.sim(n, model = list(ar = c(0.11, 0.58)))
  z8 <- arima.sim(n, model = list(ma = 0.78))
  z <- cbind(z1,z2, z3, z4, z5, z6, z7, z8)
  xnt <- tcrossprod(z, A)
  xt <- z
  return(list(xnt = as.ts(xnt), xt = as.ts(xt)))
  }
```

Listing 6.12: R code – Simulation function for a time serie with non-stationarity in autocorrelation

```
sim123 <- function(n, A) {
  z1 <- rnorm(n) + 10*tanh(0.0001*(1:n))
  z2a <- arima.sim(floor(n/3), model = list(ar = c(0.5),
  innov=c(rnorm(floor(n/3),0,1))))
  z2b <- arima.sim(floor(n/3), model = list(ar = c(0.2),
  innov=c(rnorm(floor(n/3),0,1.28))))
  z2c <- arima.sim(n-2*floor(n/3), model = list(ar = c(0.8),
  innov=c(rnorm(n-2*floor(n/3),0,0.48))))
  z2 <- c(z2a, z2b, z2c)
  z3a <- arima.sim(floor(n/2), model = list(ma = c(0.5), innov=c(rnorm(floor(n/2),0,1))))
  z3b <- arima.sim(floor(n/2), model = list(ma = c(0.9, 0.17),
  innov=c(rnorm(n-floor(n/2),0,0.68))))
  z3 <- c(z3a, z3b)
  z4 <- arima.sim(n, model = list(ma = c(0.72, 0.24), ar = c(0.14, 0.45)))
  z5 <- arima.sim(n, model = list(ar = c(0.34, 0.27, 0.18)))
  z6 <- arima.sim(n, model = list(ar = c(0.34, 0.27, 0.18), ma = c(0.72, 0.15)))
  z7 <- arima.sim(n, model = list(ar = c(0.11, 0.58)))
  z8 <- arima.sim(n, model = list(ar = c(0.10, 0.10, 0.10, 0.10, 0.10)))
  z <- cbind(z1,z2, z3, z4, z5, z6, z7, z8)
  xnt <- tcrossprod(z, A)
  xt <- z
  return(list(xnt = as.ts(xnt), xt = as.ts(xt)))
  }
```

Bibliography

[1] Davison A.C and Hinkley D.V. *Bootstrap Methods and their application.* Cambridge University Press, Cambridge, United Kingdom, 1997.

[2] Hans W. Borchers. *pracma: Practical Numerical Math Functions*, 2019. R package version 3.1.0.

[3] J.-F. Cardoso and Souloumiac. Jacobi angles for simultaneous diagonalization. *SIAM J. Mat. Anal. Appl.*, Vol. 17:pp. 161–164, 1996.

[4] Blythe D.A.J., von Bünau P., Meinecke F.C., and Müller K.-R. Feature extraction for change-point detection using stationarity supbspace analyis. *IEEE Transaction on Neural Networks and Learning Systems*, Vol. 23:pp. 631–643, 2012.

[5] Begelfor E. and Werman M. Affine invariance revisited. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2006.

[6] Liski E., Nordhausen K. Oja H., and Ruiz-Gazen A. *Combining Linear Dimension Reduction Estimates. In Agostinelli, C., Basu, A., Filzmoser, P. and Mukherje, D. (editors) Recent Advances in Robust Statistics: Theory and Applications, 151-167.* Springer, India, New Delhi, 2016.

[7] Nieto F.H., Peña D., and Dagoberto Saboyá D. Common seasonality in multivariate time series. *Statistica Sinica*, Vol. 26:1389–1410, 2016.

[8] James G., Witten D., Hastie T., and Tibshirani R. *An Introduction to Statistical Learning.* Springer, New-York, United States of America, 2013.

[9] Miettinen J., Nordhausen K., Oja H., and Taskinen S. *BSSasymp: Asymptotic Covariance Matrices of Some BSS Mixing and Unmixing Matrix Estimates*, 2017. R package version 1.0.2.

[10] Miettinen J. Nordhausen K. and Taskinen S. Blind source separation based on joint diagonalization in R: The package JADE and BSSasymp. *Journal of Statistical Software*, Volume 76(Issue 2), 2017.

[11] Nordhausen K. On robustifying some second order blind source separation methods for non stationary times series. *StatPaper*, 2014.

[12] Nordhausen K., Cardoso J-F., Miettinen M., Hannu Oja H., Esa Ollila E., and Taskinen S. *JADE: Blind Source Separation Methods Based on Joint Diagonalization and Some BSS Performance Criteria*, 2019.

[13] Li K.-C. Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association*, Vol. 86:pp. 316–327, 1991.

[14] Oja H. Ruiz-Gazen A. Liski E., Nordhausen K. *LDRTools: Tools for Linear Dimension Reduction*, 2018. R package version 3.2.2.

[15] Crone L.J. and Crosby D.S. Statistical applications of a metric on subspaces to satellite meteorology. *Technometrics*, Vol. 37:pp. 324–328, 1995.

[16] Deistler M. and Scherrer W. *Modellen der Zeitreihenanalyse.* Birkhäuser, Cham, Switzerland, 2018.

[17] Matilainen M., Nordhausen K., and Virta J. *On the Number of Signals in Multivariate Time Series." In Deville, Y., Gannot, S., Mason, R., Plumbley, M.D. and Ward, D. (editors) "International Conference on Latent Variable Analysis and Signal Separation", LNCS 10891, 248-258.* Springer, Cham, Switzerland, 2018.

[18] Filzmoser P., Hron K., and Reimann C. Principal component analysis for compositional data with outliers. *Special Issue: The 18th TIES Conference: Computational Environmetrics: Protection of Renewable Environment and Human and Ecosystem Health*, Vol. 20:pp. 621–632, 2009.

[19] Cook R.D. A method for dimension reduction and graphics in regression. *Communication in Statistics - Theory and Methods*, Vol. 29:pp. 2019–2121, 2000.

[20] Hara S., Kawahara Y., Washi T., and von Bünau P. *Stationary Subspace Analysis as a Generalized Eigenvalue Problem. In: Wong K.W., Mendis B.S.U., Bouzerdoum A. (eds) Neural Information Processing. Theory and Algorithms. ICONIP 2010. Lecture Notes in Computer Science, vol 6443.* Springer, Berlin, Germany, 2010.

[21] von Bünau P., Meinecke F.C., Királi F.C., and Müller K.-R . Finding stationary subspaces in multivariate time series. *Physical Review Letters*, Vol. 103:pp. 214–101, 2010.