NTNU

Norwegian University of
Science and Technology

Statistical Analysis of Metal Processes

Andreas Strand

Master of Science in Physics and Mathematics
Submission date: June 2017
Supervisor: John Sølve Tyssedal, IMF

Norwegian University of Science and Technology
Department of Mathematical Sciences

Preface

This is a master thesis in statistics at NTNU carried out in the spring semester of 2017 as a part of the study in Industrial Mathematics. It is a contribution to a project named Electrical Conditions and their Process Interactions in High Temperature Metallurgical Reactors (ElMet) initialized in 2015 and ending in 2019. ElMet is a collaboration between universities and the industry. The full list of partners is given in the introduction.

This work describes methods of analyzing data from metallurgical reactors. The reader is assumed to have some experience with data analysis.

Trondheim, June 27, 2017

Andreas Strand

Acknowledgment

I would like to thank my supervisor Professor John Sølve Tyssedal for excellent guidance on this work. Furthermore, I would like to thank Teknova and Eramet Norway for a great collaboration.

A.S.

Abstract

Electric smelting furnaces are complex systems that are not easily explained by a physical or chemical model. However, a statistical model based on observed quantities may be just as good. Among many attemps of modelling metal processes, few are statistical. A smelting furnace runs without stop and the process varibles are changed on the fly if necessary. There is a lag from a change is made until the process adjusts accordingly. Hence, the statistical model used should incorporate the possibility of lagged relationships.

This thesis describes systems with one ouput variable and several input variables. Two main approaches for describing a metal process are suggested. Both models are based on correlation between observed variables. The dependence structure in the input variables and the lag in the model decides which model is preferred. The prediction accuracy is assessed for different numbers of variables, number of observations and levels of noise. A real life example is also included. Both methods explains more than half of the variation in the output variable.

Sammendrag

Elektriske smelteovner er sammensatte systemer som ikke enkelt lar seg beskrive av en fysisk eller kjemisk modell. Det er derfor nærliggende å undersøke om en statistisk modell basert på målinger vil fungere. Det er gjort mange forsøk på å beskrive metallprosesser, men få av disse bruker statistikk. En smelteovn drives uten stopp og endringer i prosessvariabler gjøres mens prosessen går. Det er en forsinkelse fra en endring blir gjort til prosessen er forandret. Derfor må modellen vi bruker tillate muligheten for forsinkelse mellom variabler.

Denne oppgaven beskriver systemer men én responsvariabel og flere forklaringsvariabler. To hovedmetoder for å beskrive metallprosesser blir lagt frem. Begge baserer seg på korrelasjonen mellom variablene. Avhengighetsstrukturen i forklaringsvariablene og forsinkelsen i modellen avgjør hvilken metode som er foretrukket. Treffsikkerheten i prediksjonen av responsvariabelen er beregnet for foskjellige verdier for antall variabler, antall målinger og mengde støy. En anvendelse med faktiske målinger fra en smelteovn er også inkludert i oppgava. Begge metodene forklarer mer enn halvparten av variasjonen i responsvariabelen.

Contents

Chapter 1

Introduction

This work is asscociated with a project on Electrical Conditions and their Process Interactions in High Temperature Metallurgical Reactors (ElMet), running from 2015 to 2019. The goal of the project is to understand how the conditions in electric furnaces behave with three phase alternating currents. The project owner is Teknova, a Norwegian non-profit research institute. Collaborators are University of Oxford, NTNU and Universidade de Santiago de Compostela, as well as the companies Eramet, Alcoa and Elkem. The Research Council of Norway provides funding equal to eighty percent of the ElMet budget.

Eramet Norway Sauda (ENS) runs two 40 MW furnaces and a refinery for production of ferromanganese, FeMn. The facility is located in the southwest of Norway and is the largest FeMn production site in Northern Europe. ENS has collected data from the furnace operation at their facility. This study of ferromanganese production is based on daily measurements in periods from 2012 to 2016.

Eramet has previously pointed at key figures in their production, which we can treat as ouput in the analysis. Furthermore, Eramet has pointed at variables that are thought to affect each ouput variable. The main motivation for this paper is to develop methods for testing these hypotheses. There are many approaches for explaining the furnace process, such as simulations and physical or chemical models. However, the approach discussed here is purely statistical. The resulting models were presented at a workshop in Kristiansand on May 11. Various industry partners and researchers presented their work on metallurgical reactors as a part of the ElMet project. This

statistical approach was encouraged and we might see extensions of it in the future.

Due to confidensiality of measured data, no variables are named. Furthermore, physical interpretations of relationships between variables are left out. Focus will be on explaining mathematical relationships. Chapter 2 presents the data and methods for data cleansing. Theoretical background is provided in chapter 3, and the statistical methods are given in chapter 4. Results are presented in chapter 5 and includes general performance of the methods and applications to real data.

Chapter 2

Data

There are 852 observations of 75 variables, of which about 0.4% are missing. Only a selection of variables are used in the analysis. The presented data will be normalized in order to provide anonymity to the variables and to make them easier to compare. A sketch of an electric furnace used in ferromanganese production is provided in figure 2.1. The variables used for analysis in this paper describe important features of the production.

Before using observations in statistical methods, we should assess their quality. That is considering whether the reported measurements appear reasonable, and potentially remove those that do not. A data cleansing approach involves removing certain entries or rows to obtain high-quality data for statistical inference. We want to remove entries that are considered out of range or do not fulfill other requirements. The acceptance range of each variable is determined by physical limitations or standard production levels. Values outside the range are called missing values.

Ideally, we want to detect all erronous observations, also those inside the acceptance range. Doing so is not straightforward. Errors may be caused by malfunction of instruments or poor calibration. Other reasons are inconsistent measurement procedures, typographical errors or other entry errors. Some of these sources of errors may corrupt data across variables and time. If systematic errors are detected frequently in a period of time, we remove all observations in order to increase validity.

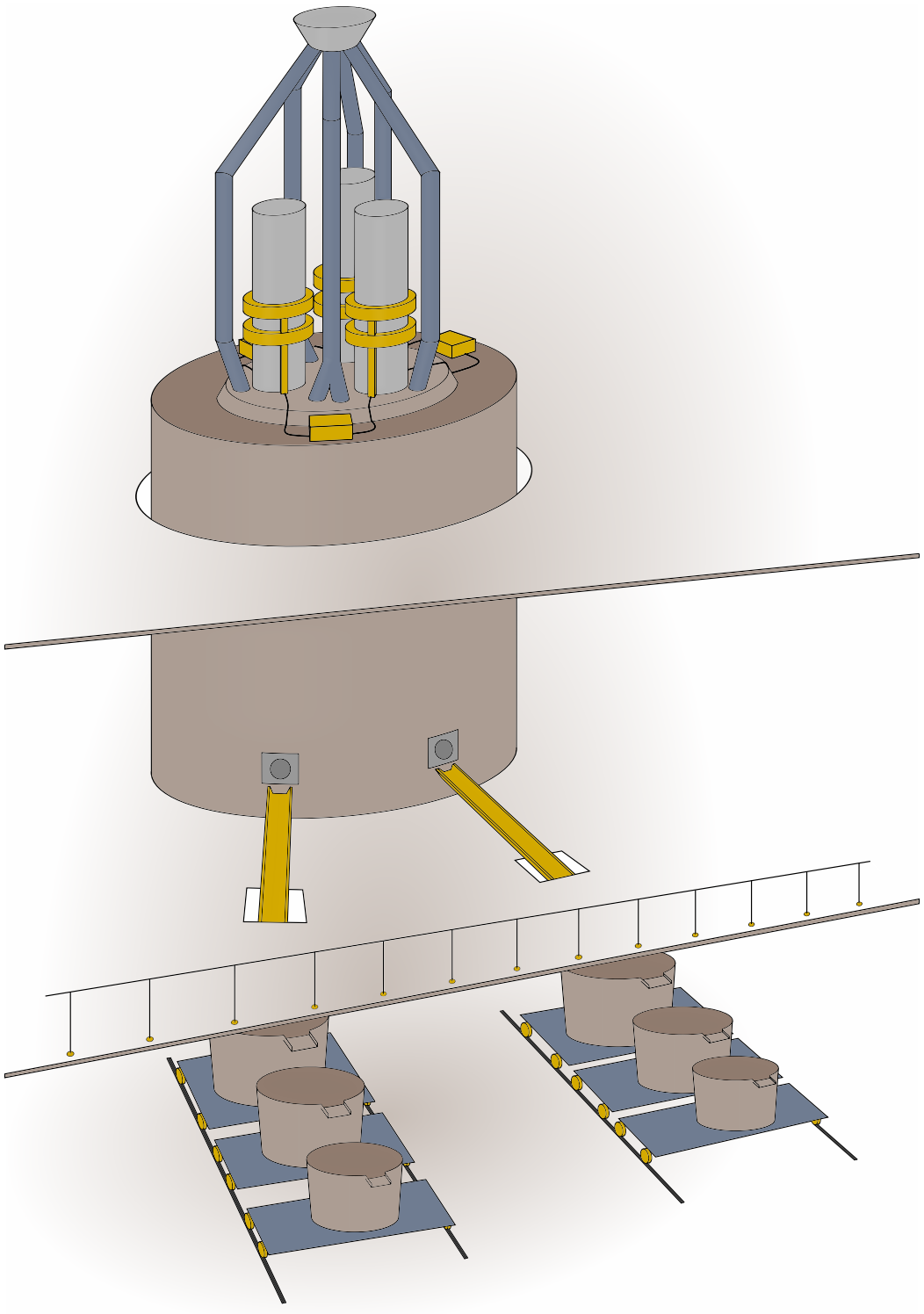

Figure 2.1: A standard electric furnace used for production of ferromanganese. The metal is extracted from ore at high temperatures inside the furnace. Three electrodes provide the necessary energy. A mixture of several substances is provided through the tubes entering the roof of furnace, and the products are retrieved through taps in the bottom part. The products are then guided down the chutes and fill in the ladles at the lower floor.

Chapter 3

Time Series Model

I Time series

A time series is a sequence of observations. The sequence is most commonly listed in time order, as the name suggest. However, it can be taken through any dimension. Some time series can be recorded continously, such as outdoor temperature measured by a thermometer. We call them continous. On the contrary, when observations are taken at intervals, they are said to be discrete. As is the case for daily sales or stock prices. The most common type of time series are succesive observations equally spaced in time. Henceforth, these are the time series in question. It is convinient to define a stochastic process when discussing time series. Time series can be regarded as a realizaion of a stochastic process.

Definition 1 - Stochastic process

Let ω belong to a sample set and t belong to a index set. The indexed random variables $z(\omega, t)$ is a stochastic process.

There are numerous mathematical tools in the scope of time series. Including methods for understanding the process that have generated your observations. Furthermore, predictions of future values may be computed. Forecasts are valuable in many fields, such as meteorology, real estate and ecology to mention a few. We want the model of the time series to be flexible, yet

simple. Satisfying both criteria and frequently used, are the ARIMA models. Note that formulas derived in this chapter assume zero-mean processes. However, all results can be generalized by reading $z_t - \mathrm{E}[z_t]$ where is says z_t.

II Introduction to ARMA and ARIMA models

An autoregressive moving average (ARMA) model is fitted to a time series for understanding and predicting its behavior. Each observation is explained by a sum of various terms. The type of each term is either autoregressive (AR) or moving average (MA). An AR term is the value of an earlier observation multiplied by some coefficient. An MA term is white noise associated with a previous observation. There is some variation in the notation used, but the following is common. Denote the time series $(z_1, z_2, \dots z_n)^\mathsf{T}$ and the corresponding white noise terms $(\epsilon_1, \epsilon_2, \dots \epsilon_n)^\mathsf{T}$. If we include p AR coefficients, $\phi_1, \phi_2, \dots, \phi_p$, and q MA coefficients, $\theta_1, \theta_2, \dots, \theta_q$, we will have an ARMA$(p, q)$ model, which we may write as

$$\begin{aligned} z_t = \phi_1 z_{t-1} + \phi_2 z_{t-2} + \ldots + \phi_p z_{t-p} + \epsilon_t - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} - \ldots - \theta_q \epsilon_{t-q}, \\ \epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2). \end{aligned} \tag{3.1}$$

Suppose the coefficients are non-zero. Then the model suggest that an observation depends on the p previous observations and the noise in the q previous observations. Also note that each observation z_t has noise ϵ_t, no matter the values of the AR and MA coefficients. When z_t is not white noise, the observations are correlated. However, the errors, ϵ_t, are independent from each other.

An ARMA(p, q) model of a real life problem typically does not include many terms, i.e. the values of p and q are small. In some cases either p or q is zero. If both are zero, the resulting model is white noise. When q is zero, the model has no MA terms and the model is purely autoregressive. A purely autoregressive model is denoted AR(p). For instance we write the AR(2) model

$$z_t = \phi_1 z_{t-1} + \phi_2 z_{t-2} + \epsilon_t. \tag{3.2}$$

Similarily, when p is zero, the model has no AR terms and the model is a pure moving average, MA(q). For example, we write the MA(1) model as

$$z_t = \epsilon_t - \theta_1 \epsilon_{t-1}. \tag{3.3}$$

Time series simulated from (3.2) and (3.3) are plotted in figure 3.1. Two hundred subsequent values are computed, and the coefficients used are $\phi_1 = 0.8$, $\phi_2 = -0.8$ and $\theta_1 = 0.8$. There are no repeating patterns in the time series, as we would expect due to noise. Still, there are clear differences betweeen the two. The variance of the AR(2) is about 3.1, while the variance of the MA(1) is about 1.7. Hence the variance of the AR sequence is much higher. At the same time, the corners of the AR series appear to be more rounded off and has some periodicities. Both sequences remains close to zero. The ARMA model can capture quite different behaviours, but there are some drawbacks.

The model (3.1) is only good for *stationary* time series. Roughly speaking a time series is stationary if it seem to fluctuate around some equilibrium in the same fashion for any time interval. Stationarity referes in this case to covariance stationarity.

Definition 2 - Covariance stationarity

A stochastic process $\{z_t\}$ *is covariance stationary if and only if*

1. $\mathrm{E}[z_t] = \mathrm{E}[z_s] \quad \forall\ t, s \quad$ *and*
2. $\mathrm{Cov}(z_t, z_s) = \mathrm{Cov}(z_{t+\tau}, z_{s+\tau}) \quad \forall\ t, s, \tau.$

Stationarity of a time series can be determined by inspection of the ARMA representation of the series. This will be covered later. When a time series is not stationary in the mean, we may still use the framework above, if we first do *differencing*. When differencing, we first compute the difference, $w_t = z_t - z_{t-1}$, between subsequent terms of the time series. Next we should investigate whether w_t is stationary. If this is the case, we may fit an ARMA model to w_t. If w_t is not stationary in the mean, differencing again one or more times may work. The enhanced model is the ARIMA(p, d, q) model. The parameter d is the number of differentiations, p is the number of AR coefficients and q is the number of MA coefficients.

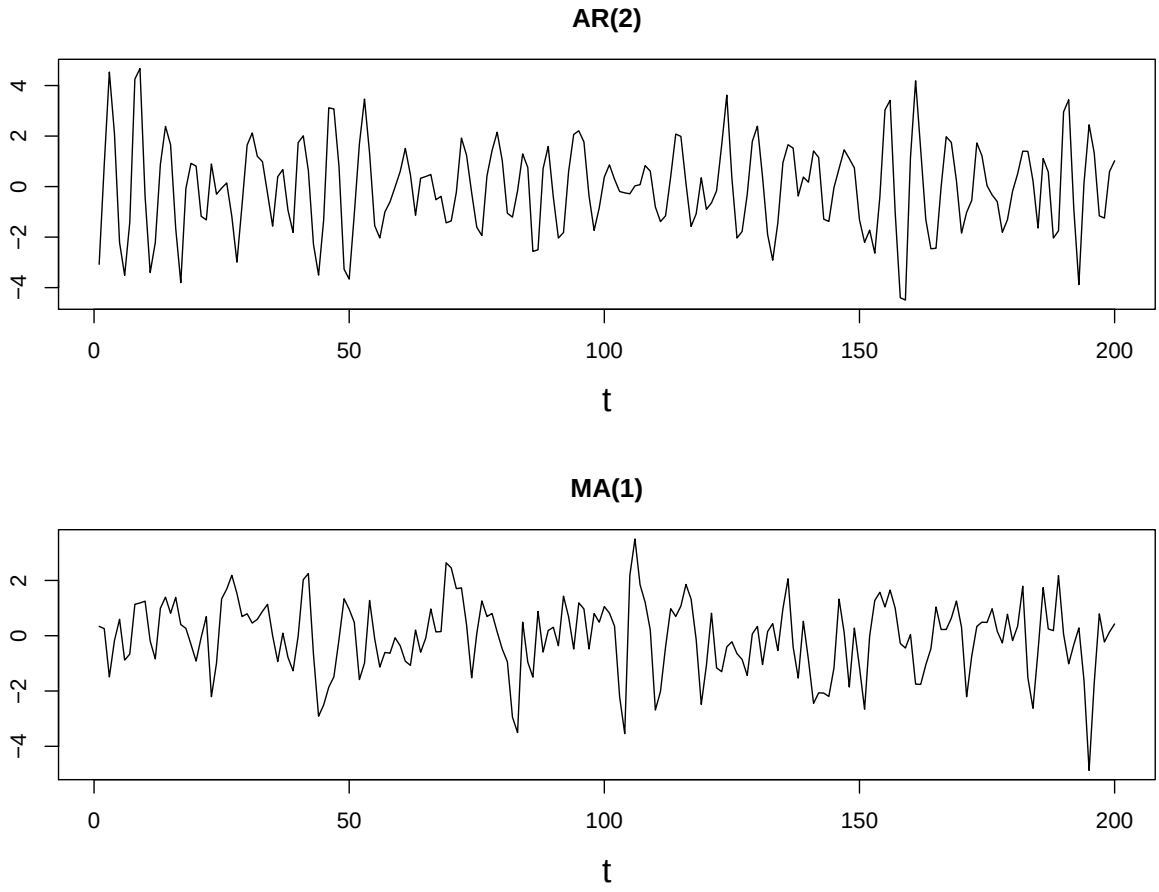

Figure 3.1: The top frame shows an AR(2) process with coefficients $\phi_1 = 0.8$ and $\phi_2 = -0.8$. In the lower frame is an MA(2) process with $\theta_1 = 0.8$.

ARIMA models are often written using the backshift operator B, defined $Bz_t = z_{t-1}$. Consequently $B^2 z_t = Bz_{t-1} = z_{t-2}$. Now, we can write the differencing in a more compact manner. The difference between subsequent terms is $w_t = z_t - z_{t-1} = (1-B)z_t$. Hence, differencing z_t once is the same as applying the operator $(1-B)$ to z_t. Differencing d times is the same as using the operator $(1-B)^d$ on z_t. The difference operator, $(1-B)^d$, is a polynomial in B of order d.

The "I" in ARIMA stands for *integrated*. In order to understand why the model is called integrated, we can look at the case where $d = 1$. Then the process

$$w_t = (1-B)z_t$$

is stationary. The observed series can then be expressed as

$$z_t = \frac{1}{1-B} w_t.$$

Taylor expansion (Rottmann, 2011) of $f(B) = (1-B)^{-1}$ at $B = 0$ gives

$$\begin{aligned} z_t &= \left(f(0) + \frac{f'(0)}{1!} B + \frac{f''(0)}{2!} B^2 + \ldots \right) w_t \\ &= (1 + B + B^2 + \ldots) w_t \\ &= w_t + w_{t-1} + w_{t-2} + \ldots. \end{aligned}$$

Hence the observed series, z_t, can be interpreted as a cumulative sum "integrating" the stationary series, w_t . In general, this "integration" takes place d times.

We previously saw that the operation of differencing in an ARIMA model can be represented as a polynomial in B. Also the AR part and the MA part of the ARIMA model can be expressed using polynomials in B. The model (3.1) can be expressed as

$$\begin{aligned} & z_t - \phi_1 z_{t-1} - \phi_2 z_{t-2} - \ldots - \phi_p z_{t-p} = \epsilon_t - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} - \ldots - \theta_q \epsilon_{t-q} \\ \implies \quad & z_t - \phi_1 B z_t - \phi_2 B^2 z_t - \ldots - \phi_p B^p z_t = \epsilon_t - \theta_1 B \epsilon_t - \theta_2 B^2 \epsilon_t - \ldots - \theta_q B^q \epsilon_t \\ \implies \quad & (1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p) z_t = (1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q) \epsilon_t \end{aligned}$$

Denote the AR polynomial $\phi(B) = (1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p)$. Similarily, denote the MA polynomial $\theta(B) = (1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q)$. The AR polynomial accesses previous values of the time series, and thus operates on z_t. The MA polynomial operates on the i.i.d. noise terms ϵ_t. We may now obtain the common form of the ARIMA(p, d, q) model by combining the AR, MA

and difference part, i.e.

$$\phi(B)(1-B)^d z_t = \theta(B)\epsilon_t$$

or

$$(1-\phi_1 B-\phi_2 B^2-\ldots-\phi_p B^p)(1-B)^d z_t = (1-\theta_1 B-\theta_2 B^2-\ldots-\theta_p B^p)\epsilon_t.$$

A more general model includes a *deterministic trend*, θ_0. The resulting model is

$$\phi(B)(1-B)^d z_t = \theta_0 + \theta(B)\epsilon_t$$

The deterministic trend is a constant, but is not to be confused with the average of z_t. When $d=0$, we have $\theta_0 = (1-\phi_1-\ldots-\phi_p)\mathrm{E}[z_t]$. However, when $d=1$, we should rather interpret θ_0 as a proportionality constant of the time series.

III Alternative representation

In this section we will go further with ARIMA models. We will see alternative models that are useful when we later will talk about forecasting of a time series. Recall that the integration "I" part of an ARIMA model simply is a transformation of the observed time series for obtaining stationarity. Hence, for most purposes it is sufficient to study the ARMA model, i.e.

$$\phi(B)z_t = \theta(B)\epsilon_t. \tag{3.4}$$

Here, an AR part and an MA part is combined for describing the process. Actually, AR and MA terms are really two sides of the same coin. For certain processes there exist multiple equivalent representations, either with only AR terms, only MA terms or a combination. We will now dig into which processes that is. It is natural to proceed with an essential theorem.

Theorem 1 - Wold representation theorem

Let z_t be any zero mean stationary process. We may then write it as

$$z_t = \sum_{j=0}^{\infty} \psi_j \epsilon_{t-j}, \tag{3.5}$$

where ϵ_t is an uncorrelated sequence of white noise. The weights ψ_j are possibly infinite in number, but but restricted by $\sum_{j=0}^{\infty} \psi_j^2 < \infty$.

For proof and details about this brilliant result, (Wold, 1939) can be consulted. Note that the random variables ϵ_t do not need to be independent of each other, only uncorrelated. The format (3.5) is called the moving average representation of a process. Furthermore, it is convenient to introduce the infinite MA polynomial $\psi(B) = \sum_{j=0}^{\infty} \psi_j B^j$. Recall that B is the backshift operator. It is conventional to define $\psi_0 = 1$. The variance of a stationary process is finite, and can easily be expressed using the infinite MA coefficients. Now we may write

$$\begin{aligned} z_t &= \psi(B)\epsilon_t, \\ \text{Var}(z_t) &= \sigma_\epsilon^2 \sum_{j=0}^{\infty} \psi_j^2 < \infty. \end{aligned} \tag{3.6}$$

Similarily, for certain processes there exist a representation with only AR terms. For this representation we will use the infinite AR polynomial $\pi(B) = \sum_{j=0}^{\infty} \pi_j B^j$, with $\pi_0 = 1$. Such that the process becomes

$$\begin{aligned} \pi(B) z_t &= \epsilon_t, \\ \sum_{j=0}^{\infty} \pi_j^2 &< \infty. \end{aligned} \tag{3.7}$$

In (Box and Jenkins, 1976), a process is called *invertible* if it can be written like this, i.e. a pure autoregression. The process (3.4) is invertible if $\theta(B)$ can be inverted to an AR polynomial, such that we can write $\pi(B) = \phi(B)/\theta(B)$. In the same fashion, a process is stationary if we can write $\psi(B) = \theta(B)/\phi(B)$. Furthermore, the inverse of the polynomial exist if its roots lie outside the unit circle in the complex plane. We can express this by introducing new notation. Let $|\cdot|$ be the standard euclidian metric. Let the roots of $\theta(B)$ be denoted B_i, and the roots of $\phi(B)$ be denoted

B_j. Finally,

$$\phi(B)z_t = \theta(B)\epsilon_t \quad \text{invertible} \iff |B_i| > 1 \quad \forall i \text{ s.t. } \theta(B_i) = 0,$$
$$\phi(B)z_t = \theta(B)\epsilon_t \quad \text{stationary} \iff |B_j| > 1 \quad \forall i \text{ s.t. } \phi(B_j) = 0.$$

A stationary process does not have to be invertible, and an invertible process is not necessarily stationary.

IV Modelling

There are a variety of powerful tools for modelling time series. We are here concered with ARIMA modelling. First, we will see some key quantities describing the structure of a time series. From section II we know that a covariance between two terms in stationary stochastic process only depends on the time difference between them. This covariance measure is commonly called the autocovariance.

Definition 3 - Autocovariance

Consider a stationary stochastic process z_t. The autocovariance is defined as the covariance between z_t and z_{t+k},

$$\gamma_k = \text{Cov}(z_t, z_{t+k}).$$

The lag k can be both positive, zero and negative. Since the covariance is symmetric in its arguments, it follows immediately that $\gamma_k = \gamma_{-k}$. When the lag is zero, the autocovariance coincides with the variance of the process, i.e. $\gamma_0 = \text{Cov}(z_t, z_t) = \text{Var}(z_t)$. Since γ_0 is a constant and not a function of t, the variance of the process is constant. Note that for a zero-mean process, the autocovariance simplifies to

$$\gamma_k = \text{E}(z_{t-k} z_t). \tag{3.8}$$

Furthermore, it is useful to define the autocorrelation function (ACF) and the partial autocorrelation function (PACF).

Definition 4 - ACF and PACF

The autocorrelation is the autocovariance scaled by the variance γ_0 of the stochastic process, i.e.

$$\rho_k = \frac{\text{Cov}(z_t, z_{t+k})}{\sqrt{\text{Var}(z_t)}\sqrt{\text{Var}(z_{t+k})}} = \frac{\gamma_k}{\gamma_0}.$$

The partial autocorrelation is the correlation of z_t and z_{t+k} after removing their dependence on intermediate variables, i.e.

$$\phi_{kk} = \frac{\text{Cov}(z_t, z_{t+k}|z_{t+1}, z_{t+2}, \dots z_{t+k-1})}{\sqrt{\text{Var}(z_t|z_{t+1}, \dots z_{t+k-1})}\sqrt{\text{Var}(z_{t+k}|z_{t+1}, , \dots z_{t+k-1})}}.$$

The double subscript kk is introduced because ϕ_{kk} is the kth element of the vector ϕ_k defined later. The autocorrelation describes the similarity between lagged elements of a process. This measure disregard any dependence structure, but simply tells how similar the elements are. The partial autocorrelation, on the other hand, describes the direct relationship between the elements. The autocorrelation and partial autocorrelation may also be regarded as linear regressions on the population. The ACF at lag k is the coefficient in a regression with z_t and z_{t-k}. The PACF is the coeffiecient of z_{t-k} in a regression with $z_t, z_{t-1}, \dots, z_{t-k-1}, z_{t-k}$. In other words, the PACF controls for the effects of $z_{t-1}, \dots, z_{t-k-1}$. There are several properties of the ACF and the PACF worth noticing. For a stationary process,

$$\begin{aligned} \rho_0 &= \phi_{00} = 1, \\ \rho_1 &= \phi_{11}, \\ |\rho_k| &\leq 1, \\ |\phi_{kk}| &\leq 1. \end{aligned}$$

The ACF and the PACF of a stationary process are useful for determining the orders of the ARMA(p,q) model of that process. When fitting an ARMA(p,q) model to a time series we will estimate these correlations. For a finite time series with n elements, unbiased estimators for the

autocovariance and the autocorrelation are

$$\hat{\gamma}_k = \frac{1}{n}\sum_{t=1}^{n-|k|}(z_{t+|k|} - \bar{z})(z_t - \bar{z}), \quad -n < k < n, \tag{3.9}$$
$$\hat{\rho}_k = \frac{\hat{\gamma}_k}{\hat{\gamma}_0}, \quad -n < k < n.$$

Obtaining a good estimate for the PACF is less straightforward. We may express the PACF using the Durbin-Levinson algorithm (Durbin, 1960). The algorithm computes the PACF iteratively using the autocovariances $\gamma_0, \ldots \gamma_k$. Set the intial values in the recursion to $\phi_1 = \phi_{11} = \gamma(1)/\gamma(0)$. Define the vectors

$$\begin{aligned} \phi_k &= (\phi_{k1}, \ldots, \phi_{kk})^\mathsf{T}, & \tilde{\phi}_k &= (\phi_{kk}, \ldots, \phi_{k1})^\mathsf{T}, \\ \gamma_k &= (\gamma(1), \ldots, \gamma(k))^\mathsf{T}, & \tilde{\gamma}_k &= (\gamma(k), \ldots, \gamma(1))^\mathsf{T}. \end{aligned} \tag{3.10}$$

Then, the iteration scheme is given by

$$\phi_{kk} = \frac{\gamma(k) - \phi_{k-1}^\mathsf{T}\tilde{\gamma}_{k-1}}{\gamma(0) - \phi_{k-1}^\mathsf{T}\gamma_{k-1}}, \quad \phi_k = \begin{pmatrix} \phi_{k-1} - \phi_{kk}\tilde{\phi}_{k-1} \\ \phi_{kk} \end{pmatrix}.$$

A sample estimate, $\hat{\phi}_{kk}$, for the PACF is obtained by the above recursions with the autocovariance vectors γ_k and $\tilde{\gamma}_k$ replaced by estimates from (3.9). The total time of computing the PACF for $k = 1, \ldots, n$ is $\mathcal{O}(n^2)$. The Durbin-Levinson appears to be fastest alternative, but it is not the most intuitive. As mentioned previously, the PACF can be expressed as a population regression. For a sample of n, the regression model is

$$z_t = \phi_{k1} z_{t-1} + \ldots + \phi_{kk} z_{t-k} + e_t, \tag{3.11}$$

where the e_t is a zero-mean Gaussian error with a variance of $\frac{1}{n}$ (Kendall et al., 1968). The regression will have k dependent variables and $n - k$ statistical units. We can write the reponse

vector z and design matrix X as

$$z = \begin{pmatrix} z_n \\ z_{n-1} \\ \vdots \\ z_{k+1} \end{pmatrix}, \qquad X = \begin{pmatrix} z_{n-1} & \cdots & z_{n-k} \\ z_{n-2} & \cdots & z_{n-k-1} \\ \vdots & \ddots & \vdots \\ z_k & \cdots & z_1 \end{pmatrix}.$$

Using the definition of ϕ_k from (3.10), we can write the ordinary least squares (OLS) estimate of ϕ_k as

$$\hat{\phi}_k = (X^\mathsf{T} X)^{-1} X^\mathsf{T} z.$$

The last element of $\hat{\phi}_k$ is an estimate $\hat{\phi}_{kk}$. The estimates for the coefficients in the regression are exactly the same as those in the Durbin-Levinson iterations. However, the least squares approach is much slower. Computing $(X^\mathsf{T} X)$ takes $\mathcal{O}(k^2 n)$ time and the complexity of $(X^\mathsf{T} z)$ is $\mathcal{O}(nk)$. Factorizing and computing the final product is $\mathcal{O}(k^3)$. Note that $n > k$, otherwise $(X^\mathsf{T} X)$ is singular. Hence, the complexicity of a single OLS is $\mathcal{O}(k^2 n)$. If we want to compute the PACF for multiple lags, we need to compute the OLS for each. Hence, the complexity of computing $\hat{\phi}_{kk}$ for $k = 1, \ldots, n$ is $\mathcal{O}(n^4)$.

For the confidence limits of the correlation functions, we may utilize that the estimates for the ACF and the PACF are asymptotically normally distributed with mean zero and variance $\mathrm{Var}(\hat{\rho}_k) \approx \mathrm{Var}(\hat{\phi}_{kk}) \approx \frac{1}{n}$ (Kendall et al., 1968). The variance is asymtotically the same for all estimators mentioned above. Let z_p be the p-quantile of the standard normal distribution. The true variance is lag dependent, but roughly, values outside $\left[-\frac{z_{\alpha/2}}{\sqrt{n}}, \frac{z_{\alpha/2}}{\sqrt{n}}\right]$ are considered significant on an α-level.

When identifying an ARMA(p, q) model, the standard approach is plotting $\hat{\rho}_k$ and $\hat{\phi}_{kk}$ by the lag k, and then determine which entries are significantly different from zero. This will help identify both which $z_{t-1}, z_{t-2}, \ldots$ and $\epsilon_{t-1}, \epsilon_{t-2}, \ldots$, that z_t depends on. The relevant feature of each plot, is not the value of each correlation but the qualitative shape of the ACF and the PACF. The shape helps identify the dynamics of the time series.

In an AR(p) process, an observation depends on prior observations, which in turn depends on prior observations, and so on. The contributions decays exponentially with increasing lag due to a weight less than unit with each step. Thus, the ACF decays exponentially. The PACF shows which terms an observation depends on directly. Hence, the PACF of an AR(p) will be truncated at lag p.

For a pure MA(q) process, observations depend only on the random errors, back to lag q. Hence, the ACF will shut off after lag q. The PACF exhibits a smooth decay. The explanation for this is less straightforward, but an MA(1) example may provide intuition. We can write the MA(1) process as

$$z_t = \epsilon_t + \theta_1 \epsilon_{t-1}.$$

Rewriting and shifting the lag by various amounts gives,

$$\begin{aligned} \epsilon_{t-1} &= z_{t-1} - \theta_1 \epsilon_{t-2}, \\ \epsilon_{t-2} &= z_{t-2} - \theta_1 \epsilon_{t-3}, \\ &\vdots \qquad\qquad . \end{aligned}$$

Combining the above equations provides the infinite AR representation

$$z_t = \epsilon_t + \theta_1 z_{t-1} - \theta_1^2 z_{t-2} + \theta_1^3 z_{t-3} - \ldots.$$

This model is similar to (3.11), but with an infinite number of terms. In the finite AR(k) process, the PACF for lag k is just the kth AR coefficient. In the infinite representation it turns out that PACF is the kth AR coefficient reduced by a factor $\left(\sum_{j=0}^{k} \theta_1^{2j}\right)^{-1}$ according to (Wei, 1994). Hence, the PACF of an MA(1) process is

$$\phi_{kk} = \frac{\theta_1^k (-1)^{k+1}}{\sum_{j=0}^{k} \theta_1^{2j}}.$$

For an invertible MA(1) process, $|\theta_1| < 1$. Thus the PACF of an MA(1) decays gradually for in-

creasing lags. Similar arguments can be made to show that the PACF is decaying gradually for the general MA(q).

For an ARMA(p,q) process, both the ACF and the PACF are described by trigonometric funtions and/or exponential decay. Hence, identification of the parameters p and q, when both are at least one, is somewhat complex. One approach is trying multiple combinations of parameter values, fitting a model for each and then choose the one with the best fit in some sense.

In practice software is used to fit a model. The approach in this work is using the programming software RStudio.[1] The tools used are `auto.arima()` from the *forecast* package and various types of correlation functions from the *stats* package. The function `auto.arima()` fits an ARIMA model to a time series by first performing Kwiatkowski-Phillips-Schmidt-Shin (KPSS) tests for d, then using the corrected Akaike information criterion (AICc) and maximum likelihood estimation (MLE) for determining p, q and the coefficients ϕ_1, ϕ_2, ...,ϕ_p, θ_1, θ_2, ...θ_q (Hyndman and Khandakar, 2007). The AICc is minimized and the likelihood maximized when finding a model.

We have discussed the general method for fitting a model to time series. Next we will look more into a couple of examples.

The AR(1) process

The first-order autoregressive process is

$$(1-\phi_1 B)z_t = \epsilon_t.$$

Each value of the time series is expected to be the previous value times a factor. The only root of $(1-\phi_1 B)$ is $1/\phi_1$. The model is stationary if and only if all roots of the AR polynomial is outside the unit circle, i.e. when $|\phi_1| < 1$. The model is invertible since it is purely autoregressive. Figure 3.2 displays realizations of two AR(1) processes. The left column displays the process $z_t = 0.8z_{t-1} + \epsilon_t$, with ϵ_t standard normal. The coefficient is less than one in absolute value, implying stationary. This is consistent with the fact that a stationary time series always returns

[1] R version 3.4.0 (2017-04-21) on the platform x86_64-pc-linux-gnu (64-bit).

to its mean. The deterministic trend is an exponential decay, but the random shock creates oscillations. The time series in right column is a realization of a different AR(1) process. The process is identical to the first one with exception of the sign of the coefficient. The deterministic trend is an alternating decay. The sample ACFs of both processes are decaying and the sample PACFs cut off at $k = 1$. When ϕ_1 is negative the signs of the correlations are alternating.

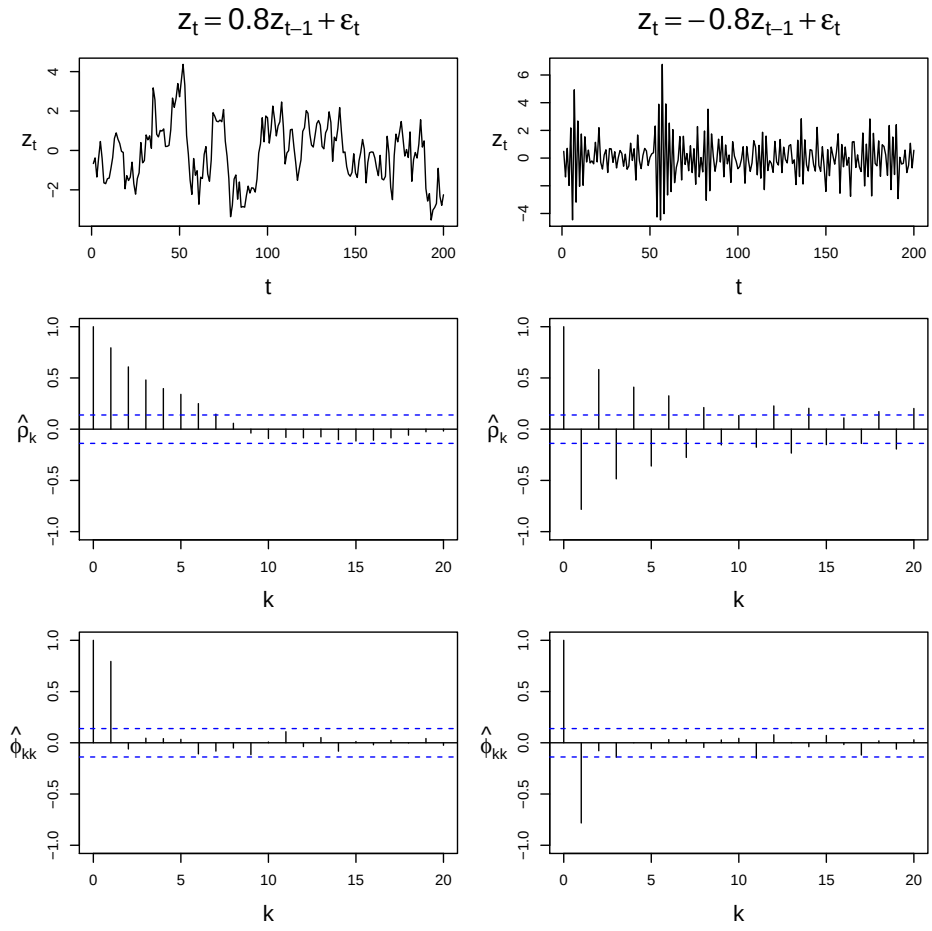

Figure 3.2: Two realization of an AR(1) process. Each column includes a time series along with its sample ACF and sample PACF.

The theoretical ACF is easily derived by substituting for $z_t = \phi_1 z_{t-1} + \epsilon_t$ in (3.8), i.e.

$$\begin{aligned}\gamma_k &= \mathrm{E}(z_{t-k} z_t) \\ &= \mathrm{E}(\phi_1 z_{t-k} z_{t-1}) + \mathrm{E}(z_{t-k}\epsilon_t) \\ &= \phi_1 \mathrm{E}(z_{t-k} z_{t-1}) + 0 \\ &= \phi_1 \gamma_{k-1} \\ &= \phi_1^k \gamma_0.\end{aligned}$$

Dividing both sides by γ_0 gives $\rho_k = \phi_1^k$ for $k \geq 0$. Hence, the ACF experience exponential decays as suggested from the sample estimates in figure 3.2. Recall that the PACF is one at lag zero and always equal the ACF for $k = 1$. Furthermore, the PACF is zero after lag p in an AR(p) process. These results also correspond with the sample estimates.

The AR(2) process

The second order autoregressive process is

$$(1 - \phi_1 B - \phi_2 B^2) z_t = \epsilon_t.$$

As the AR(2) process is close to an AR(1) process when ϕ_2 is close to zero, their behaviours are similar. The process is always invertible. It is stationary if all the roots of $\phi(B)$ are outside the unit circle. This requirement is equivalent to

$$\begin{cases} \phi_2 + \phi_1 < 1, \\ \phi_2 - \phi_1 < 1, \\ -1 < \phi_2 < 1. \end{cases} \tag{3.12}$$

For proof see (Wei, 1994). In other words, in order for an AR(2) process to be stationary, the coefficients (ϕ_1, ϕ_2) have to lie in the triangular region defined by 3.12. The roots are real if $\phi_2 > -\phi_1^2/4$, otherwise complex. These regions are depicted in figure 3.3.

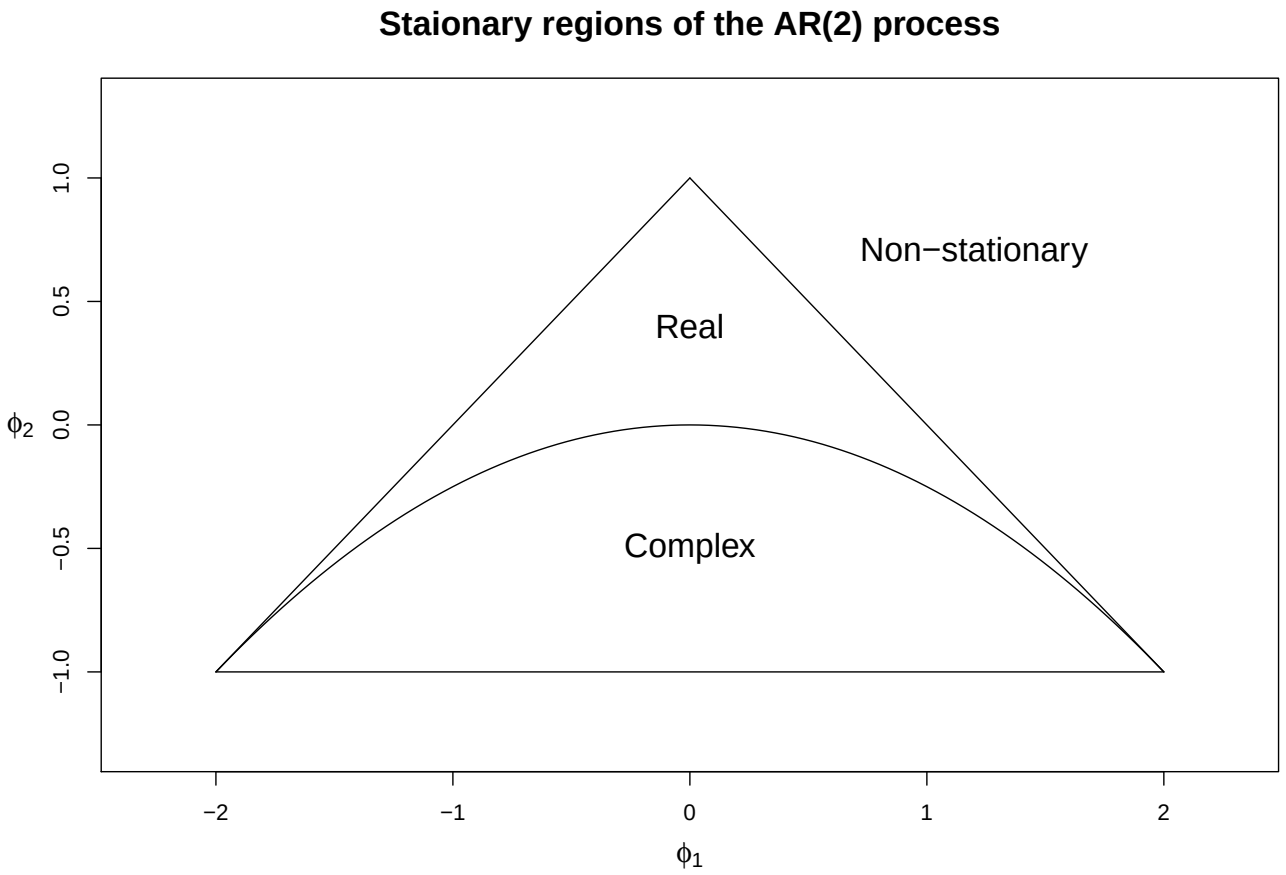

Figure 3.3: The AR(2) process is stationary if the pair of coefficients are located inside the triangle. The roots of $\phi(B)$ are real above the curve, i.e. $\phi_2 > -\phi_1^2/4$.

As for the AR(1), we can compute the autocovoraiance of an AR(2) process using 3.8, i.e.

$$\begin{aligned}\gamma_k &= \mathrm{E}(z_{t-k}z_t) \\ &= \phi_1\mathrm{E}(z_{t-k}z_{t-1}) + \phi_1\mathrm{E}(z_{t-k}z_{t-1}) + \mathrm{E}(z_{t-k}\epsilon_t) \\ &= \phi_1\gamma_{k-1} + \phi_2\gamma_{k-2}.\end{aligned}$$

Dividing by γ_0 gives $\rho_k = \phi_1\rho_{k-1} + \phi_2\rho_{k-2}$. This is a recurrence relation giving the value for subsequent lags as initial values are found. We allready know $\rho_0 = 1$. Inserting for $k = 1$ gives

$\rho_1 = \phi_1/(1-\phi_2)$. Hence, a recursive equation for the ACF is

$$\rho_k = \begin{cases} 1, & k = 0, \\ \frac{\phi_1}{1-\phi_2}, & k = 1, \\ \phi_1\rho_{k-1} + \phi_2\rho_{k-2}, & k \geq 2. \end{cases} \tag{3.13}$$

As $\phi_1 + \phi_2 < 1$, we can see that the ACF is decaying. Furthermore, recall that the PACF at lag k is the correlation with z_{t-k} when controlling for smaller lags. For an AR(p), $\phi_{pp} = \phi_p$, as we control for all other terms. When k is zero or unit there are no terms to control for, implying that the ACF and PACF are equal. For an AR(2), i.e.

$$\phi_{kk} = \begin{cases} 1, & k = 0, \\ \frac{\phi_1}{1-\phi_2}, & k = 1, \\ \phi_2, & k = 2, \\ 0, & k \geq 3. \end{cases}$$

The correlation expressions above tell us a lot about the dynamics of the AR(2) process. For an observed time series, we can plot $\hat{\rho}_k$ and $\hat{\phi}_{kk}$ and see if they resemble the theoretical functions derived. However, there is more to be said about the dynamics of the AR(2) process. The recurrence relation in (3.13) does not say much about in what way the ACF is decaying. We will look at the representation of the ACF in trigonometric form and focus on the case with complex roots. The roots of $\phi(B) = (1 - \phi_1 B - \phi_2 B^2)$ are

$$R_{1,2} = \frac{1}{2}\phi_1 \pm i\frac{1}{2}\sqrt{-\phi_1^2 - 4\phi_2}$$

or

$$R_{1,2} = r(\cos\theta \pm i\sin\theta),$$

$$\begin{cases} r &= \sqrt{-\phi_2}, \\ \theta &= \arccos\left(\frac{\phi_1}{2\sqrt{-\phi_2}}\right). \end{cases} \tag{3.14}$$

The expression for the argument, θ is derived by comparing the real part of the cartesian representation and triginometric representation. We can use these roots to express the ACF. The general solution of (3.13) is

$$\begin{aligned}\rho_k &= A_1 R_1^k + A_2 R_2^k \\ &= r^k \left(A_1[\cos k\theta + i \sin k\theta] + A_2[\cos k\theta + i \sin k\theta]\right)\end{aligned}$$

We can rewrite this to a simpler form by letting $A_{1,2} = k_1 \pm i k_2$. Furthermore, we may use the identity for sine of a sum, $\sin(k\theta + \alpha) = \cos k\theta \sin\alpha + \sin k\theta \cos\alpha$. Then, we may express the autocorrelation function

$$\rho_k = r^k (2k_1 \cos k\theta - 2k_2 \sin k\theta)$$

or

$$\rho_k = A r^k \sin(k\theta + \alpha), \tag{3.15}$$

$$\begin{cases} k_1 &= \frac{1}{2} A \sin\alpha, \\ k_2 &= \frac{1}{2} A \cos\alpha, \\ A &= \frac{1}{2}\sqrt{k_1^2 + k_2^2}, \\ \alpha &= \arctan\frac{k_1}{k_2}. \end{cases}$$

Hence, when the roots of $(1 - \phi_1 B - \phi_2 B^2)$ are complex, the ACF of that AR(2) process is a damped sine wave. The amplitude of the wave is reduced by a factor $r = \sqrt{-\phi_2}$ for each lag. Henceforth, r is referred to as the decay constant. The ACF is pseudo-periodic with period

$$T = \frac{2\pi}{\theta} = 2\pi \left(\arccos\left(\frac{\phi_1}{2\sqrt{-\phi_2}}\right)\right)^{-1}.$$

The period and the decay constant combined gives a good picture of the behavior of the ACF of a complex AR(2). Figure 3.4 shows T and r as a function of ϕ_1 and ϕ_2 when the roots of $\phi(B)$ are complex. We can see that the period increases with ϕ_1 and ϕ_2, and that the decay constant increases with ϕ_2.

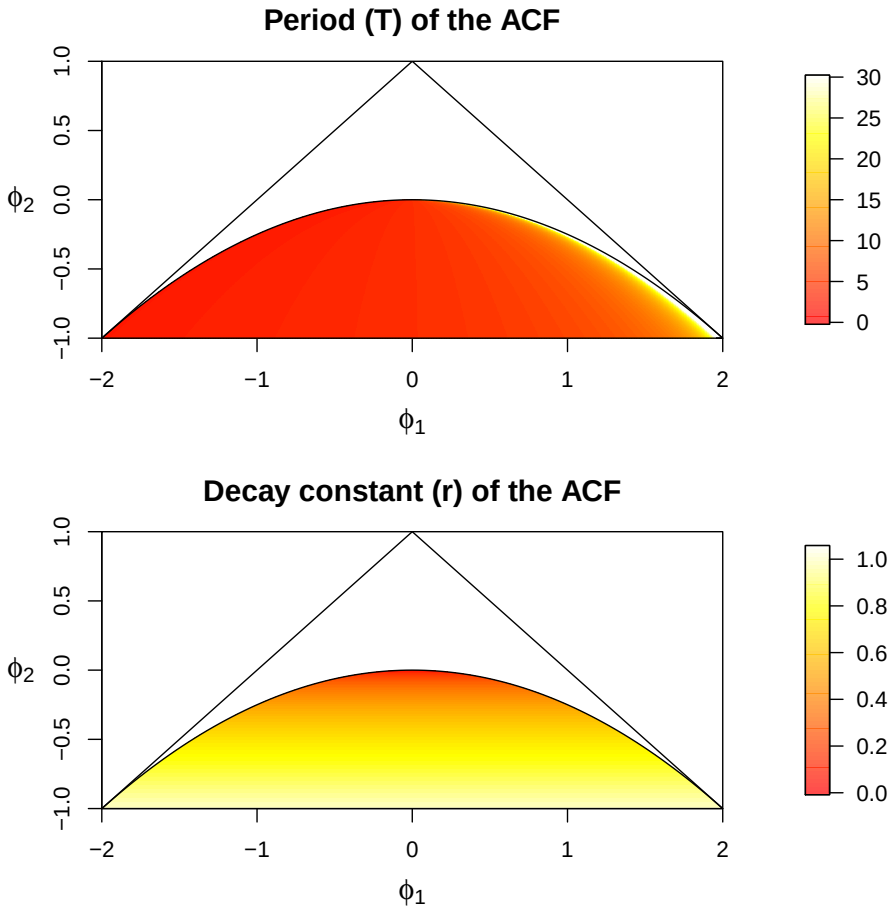

Figure 3.4: The period and decay constant of the ACF of an AR(2) process. The legend on the right of each plot maps color to value.

It is clear that the shape of the ACF varies on a wide spectrum based on ϕ_1 and ϕ_2. Recall that we have focused purely on the cases with complex roots. Some examples of AR(2) processes are provided in figure 3.5. The top panel gives the values of the coefficients. The small plots display the corresponding ACFs. The period and decay constant of the ACF is given in the top-right corner. Six examples are provided. Example A has coefficients $(\phi_1, \phi_2) = (0, 0.05)$. The corresponding ACF is provided in the first sub-plot. The period of this ACF is 4.00 and the decay constant is $r = 0.22$.

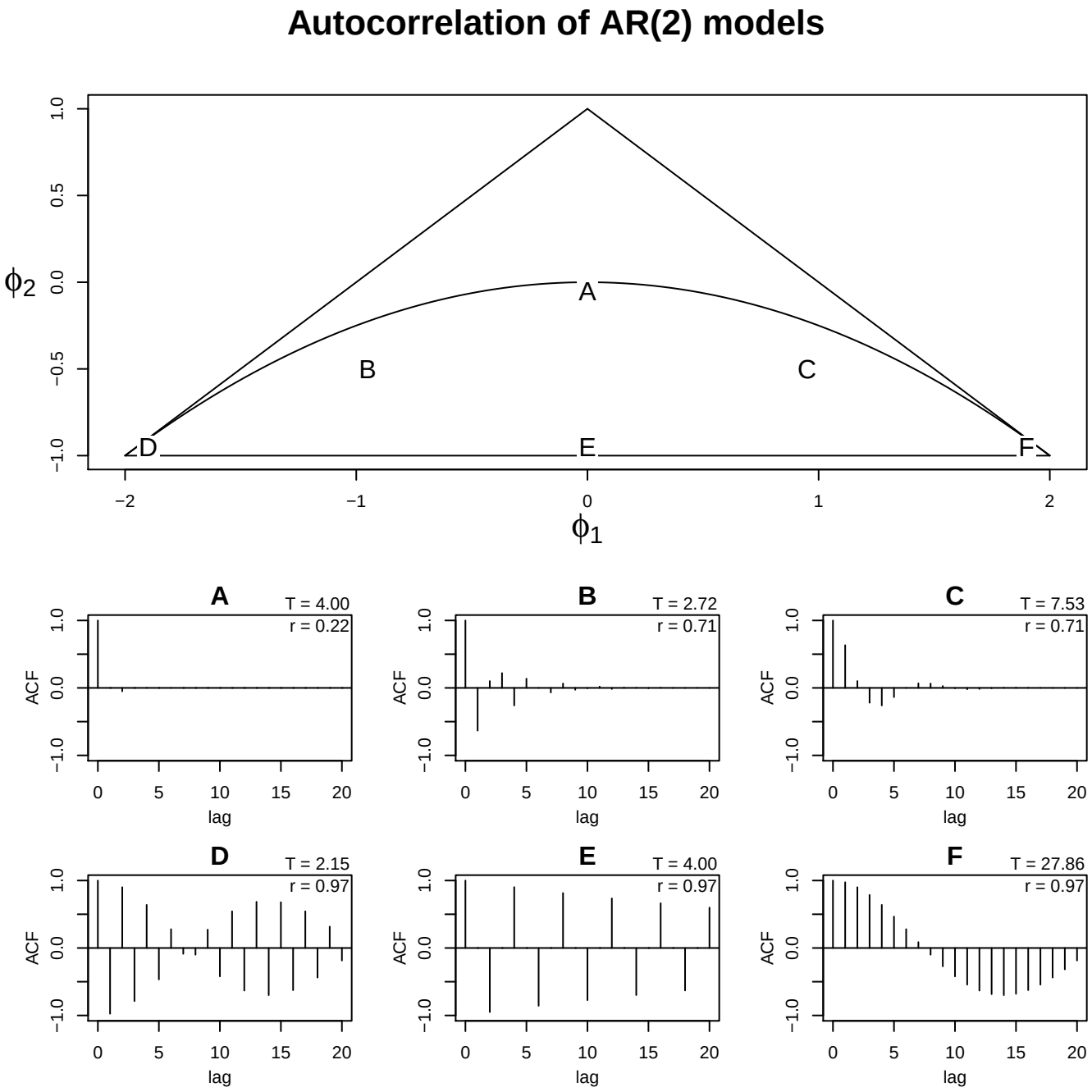

Figure 3.5: Six examples of AR(2) processes, named a, b, c, d, e and f. The coefficients of each process is provided in the top panel. The six small panels display the ACFs for each example. The period and decay constant of each ACF is provided in the top-right corner of the plot.

V Forecasting

As a model is established for a time series, it may be of interest to make a guess of future values of that sequence. The ARIMA framework provides an intuitive and accurate way of doing so. Say, we have observed a time series $z_1, \ldots, z_n$. A forecast is the expectation of the next value of the time series given what we have observed, i.e. $\mathrm{E}[z_{n+1}|z_n, \ldots, z_1]$. Thus, we can only forecast processes where the next value can be represented by previous values, i.e. invertible processes. The approach in this section assumes stationary and invertible processes.

Recall that we can express the latest value of a time series as a regression with previous entries, as in (3.11). Shifting indices and taking expectation results in the forecast

$$\begin{aligned} \mathrm{E}[z_{n+1}|z_n, \ldots, z_1] &= \mathrm{E}[\phi_{n1} z_n + \phi_{n2} z_{n-1} + \ldots + \phi_{nn} z_1 + e_{20}|z_n, \ldots, z_1] \\ &= \hat{\phi}_{n1} z_n + \hat{\phi}_{n2} z_{n-1} + \ldots + \hat{\phi}_{nn} z_1. \end{aligned} \tag{3.16}$$

The weights $\hat{\phi}_{n1}, \hat{\phi}_{n2}, \ldots, \hat{\phi}_{nn}$ can be obtained by the Durbin-Levinson recursions in (3.10), with estimates from (3.9). From (Brockwell and Davis, 2013) we know that the forecast error variance v_n can be computed iteratively. The forecast error is distributed as

$$\begin{aligned} &e_n \sim \mathcal{N}(0, v_n), \\ &\begin{cases} v_0 &= \gamma_0, \\ v_{j+1} &= v_j(1 - \phi^2_{j+1,j+1}). \end{cases} \end{aligned}$$

Imputing missing values

The method of imputing missing values suggested in this section requires a time series centered at zero. For a series not centered at zero, subtract the mean before commencing, and later add the same value to the resulting time series.

For some zero-mean time series z_t values can be missing for certain indices. Denote the number of missing indices by k and the array of such indices $M = m_1, m_2, \ldots, m_k$. In order to easily make inference on the data set, we may substitute missing values by an imputed value. This

process does not neccessarily provide more information, but may be done due to practical regards. Assume the first missing value is not at $t = 1$, that is $m_1 > 1$. A way of imputing missing values is to do it from the beginning to the end. First impute z_{m_1}, then z_{m_2}, for every value until z_{m_k}. An imputed value is immediately substituted for the missing value and used for imputing the remaining missing values. We can use the forecast in (3.16) for imputation. Let $\tilde{z}_t = (z_t, z_{t-1}, \ldots, z_1)$. Then we may write a forecast as

$$\begin{aligned}\hat{z}^*_{m_i} &= z_{m_i-1}\hat{\phi}_{m_i-1,1} + z_{m_i-2}\hat{\phi}_{m_i-1,2} + \ldots z_1\hat{\phi}_{m_i-1,m_i-1} \\ &= \tilde{z}_{m_i-1}\hat{\phi}_{m_i-1}. \end{aligned} \tag{3.17}$$

However, the last elements of ϕ_{m_i-1} will typically be small for large m_i. An option is only looking at terms with recent values of the time series.

The forecasted value $\hat{z}^*_t$ in (3.17) depends exclusively on previous observations of the time series. A better description of the time series may be to consider the above imputation as the expected value of the forecast, and also include noise. The idea is to simulate the process generating the time series. The noise, ϵ_t, of the time series is normally distributed. I suggest estimating the noise variance σ^2_ϵ by a mean square error,

$$\begin{aligned}\epsilon_t &\sim \mathcal{N}(0, \sigma^2_\epsilon), \\ \hat{\sigma}^2_\epsilon &= \frac{\sum_{t \notin M} (z_t - \hat{z}^*_t)^2}{t-k}, \end{aligned} \tag{3.18}$$

where $t - k$ is the number of non-missing values. Now, we can write the forecast as

$$\hat{z}_{m_i} = \hat{z}^*_{m_i} + \epsilon_t, \tag{3.19}$$

where ϵ_t are drawn from the distribution in (3.18). This way of imputing missing values will only work when z_1 is not misssing. In the case where it is missing, other methods will be neccessary to first impute this value.

Chapter 4

Regression Model

I Model

This chapter presents a model for ferromanganese production. The main process of creating the alloy takes place within a furnace like the one sketched in figure 2.1. Measurements of process variables are the basis for stating a model. The suggested approach is general and may be used in other fields of study.

From the pool of variables we choose one output variable, also called a response variable. This is a key variable, typically related to the profit of the process. Next, we state variables that we think might affect the output. These are called input variables or explanatory variables. If we have all the relevant data and the statistical method is reasonable, the resulting model will give the real picture of the process.

However, there are complications. The measurements will always be inaccurate to some degree. This is discussed in chapter 2. Furthermore, there may be variables not measured affecting the output. If so, some variation in the output remains unexplained. We will perceive this as noise. Still, we can define measures of the performance of the model. If the performance is satisfactory, we need not worry about excluded variables or poor measurements.

The process of creating ferromanganese from other materials is not instant. It can be days from a unit enter the furnace until it leaves. Hence, observations from subsequent days can all help

describe the same unit. Furthermore, the measurements do in general describre the process at difference stages of the process. Thus, when describing the output one day, we should look at input from multiple days. This is a key factor in the modelling approach.

The model immediately becomes complex as we allow one variable to affect another variable on a different day. A realization of a variable will in general also affect future realizations of the same variable. We will use time series in this approach as these includes framework for describing a dependence structure with time lag. Before describing the model, it is convenient to introduce some notation. Denote the days $t = 1,2,\ldots,n$. We can write the observed output sequence as y, where y_t refers to a specific entry. Let p be the number of input variables, each consisting of n observations. Denote the input sequences $x_1, x_2, \ldots, x_p$ and let X be the vector of input sequences. We can write

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \qquad X^* = (x_1, x_2, \ldots, x_p) = \begin{pmatrix} x_{11} & x_{21} & \cdots & x_{p1} \\ x_{12} & x_{22} & \cdots & x_{p2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1n} & x_{2n} & \cdots & x_{pn} \end{pmatrix}.$$

We want to express the output precisely as a function f of input values. If we evaluate the function with our observed input, the result should be close to the observed output. Let ϵ_t be the deviation from y_t. Then we may write

$$y_t = f(X^*) + \epsilon_t.$$

In order to determine f, we need to formally state a criteria for a good model. I suggest minimizing the error sum of squares, i.e. minimizing $\sum_{t=1}^{n} \epsilon_t^2$. Furthermore, let $\mathrm{E}[\epsilon_t] = 0$. It is necessary to put further restrictions on f in order to make inference. A linear regression model is a resonable proposal. The linear regression explains output y_t as a linear combination of regressors $x_{1t}, \ldots, x_{rt}$ and noise ϵ_t. Let the coefficients on the regressors be denoted $\beta_1, \ldots, \beta_r$ and let β_0 be

an intercept. Then we may express the regression model as

$$y_t = \beta_0 + \sum_{i=1}^{r} \beta_i x_{it} + \epsilon_t.$$

We assume no serial correlation in the output or the noise. Moreover, we assume the error terms to be of mean zero and with constant variance. The regressors can either be assumed to be random variables or constants. If they are constants, then $\mathrm{E}[y_t] = \beta_0 + \sum_{i=1}^{r} \beta_i x_{it}$. If the regressors are stochastic we instead write $\mathrm{E}[y_t | x_{1t}, \ldots, x_{rt}] = \beta_0 + \sum_{i=1}^{r} \beta_i x_{it}$. Henceforth, we will treat the regressors as constants.

We assume that the model will only have the input variables at various lags as regressors. This excludes cross-terms, powers or other non-trivial terms. Including to many terms will result in overfitting. Specifically, we should make sure that the number of regressors is lower than the number of observations. A possible model is

$$\begin{aligned} y_t =& \beta_{1,0} x_{1,t} + \beta_{1,1} x_{1,t-1} + \beta_{1,2} x_{1,t-2} + \ldots \\ &+ \beta_{2,0} x_{2,t} + \beta_{2,1} x_{2,t-1} + \beta_{2,2} x_{2,t-2} + \ldots \\ &\vdots \\ &+ \beta_{p,0} x_{p,t} + \beta_{p,1} x_{p,t-1} + \beta_{p,2} x_{p,t-2} + \ldots + \epsilon_t, \end{aligned} \tag{4.1}$$

where $\beta_{j,k}$ is a coefficient corresponding to $x_{j,t-k}$. Henceforth, the term regressor refers to $x_{j,t-k}$. The subscript $j = 1, \ldots, p$ refers to the input variable x_j. The subscript k denote the lag between input and output. Assume that the coefficients $\beta_{j,k}$ are constant in t, such that the dynamic relationship between input and output is time-invariant. Then, (4.1) explains output as a linear combination of the input and an error.

The lags in (4.1) are non-negative, i.e. $k \geq 0$. Obviously, the output one day does not depend on future input. We also need to set an upper limit, K, for k. The limit would be the greatest lag at which an input affect the output. Furthermore, as we explain the output y_t using input at time $t - K$ and later, we restrict the time variable in the model to $t = K + 1, \ldots, n$. We can write the

model as

$$y_t = \sum_{j=1}^{p} \sum_{k=0}^{K} \beta_{j,k} x_{j,t-k} + \epsilon_t, \quad t = K+1, \ldots, n. \tag{4.2}$$

The coefficient $\beta_{j,k}$ can be any real number. Next, we will look at ways of fitting this model. This is determining the coefficients such that the MCE is minimized.

II Reducing the model

The approach used for fitting the regression model in (4.1) is ordinary least squares (OLS) estimation. However, there is potentially a large number of coefficients to estimate, such that OLS estimation is computationally expensive. Recall that the complexity is $\mathcal{O}(r^2 n)$, where r is the number of regressors and n is the number of observations. Reducing the number of regressors beforehand, can speed up the algorithm drastically.

We can remove $x_{j,t-k}$ from the regression if it is independent of y. Removing it is equivalent of putting $\beta_{j,k} = 0$. However, we should not remove a regressor just because it is uncorrelated with the output. The regressor can still have a significant contribution to the output through other regressors. Still, when choosing which regressors to keep, a reasonable guess are those which are highly correlated with the output. The correlation between y and $x_{j,t-k}$ is called the cross-correlation for lag k.

Definition 5 - Cross-correlation function (CCF)

Let x_j and y be stochastic processes with standard deviations σ_{x_j} and σ_y respectively. The cross-correlation for lag k is defined as

$$\rho_{x_j y}(k) = \frac{1}{\sigma_{x_j} \sigma_y} \mathrm{E}[(x_{j,t-k} - E[x_j])(y_t - E[y])].$$

The CCF gives the correlation between two processes shifted by k steps. For a time series y, we can estimate the mean as $\bar{y} = \frac{1}{n} \sum_{t=1}^{n} y_t$ and the standard deviation as $\hat{\sigma}_y = \sqrt{\frac{1}{n-1} \sum_{t=1}^{n} (y_t - \bar{y})^2}$.

The natural estimator for the cross-correlation between the time series x_j and y is

$$r_{x_j y}(k) = \frac{1}{n\hat{\sigma}_{x_j}\hat{\sigma}_y} \sum_{t=k+1}^{n} (x_{j,t-k} - \bar{x}_j)(y_t - \bar{y}), \quad k \geq 0. \tag{4.3}$$

The sample CCF between an input sequence and the output sequence indicate the lags at which they are correlated. Specifically, there is a direct connection between the $r_{x_j y}(k)$ and $\beta_{j,k}$. When the regressor is correlated with output, we suspect the correponding coefficient to be significant. However, we cannot simply put $\beta_{j,k}$ equal to $r_{x_j y}(k)$. One reason for this is that $r_{x_j y}(k)$ is influenced by the autocorrelation in x_j. This will be explained in detail later in this chapter.

The first step in the suggested estimation algorithm is to compute the sample CCF. For each variable x_j, we compute $r_{x_j y}(k)$. It seems unnecessary to compute all cross-correlations up to the highest possible lag, $n-1$. Our knowledge about the process may allow us to set an upper limit $K^* < n-1$ for the lag of the most lagged relationship. Then, it will suffice to compute $r_{x_j y}(k)$ for $k = 1, \ldots, K^*$. A rough estimate for the standard error of the cross-correlation estimate is $1/\sqrt{n}$ (Bisgaard and Kulahci, 2011). Comparing the sample CCF with $\pm 1.96/\sqrt{n}$ will help us to see which cross-correlations are significantly different from zero.

As the maximum lag K in (4.2), we may use k of the most lagged sample CCF significantly different from zero. Furthermore, for each input variable we define a vector of lags at which the correlation to output is signicant, i.e.

$$\omega_j = \left\{ k \in \mathbb{N} : \left| r_{x_j y}(k) \right| > \frac{1.96}{\sqrt{n}} \right\}.$$

Next, we will discuss an example. Say we have observed input time series x_1 and x_2 and an output time series y. We have a hundred observations of each variable, i.e. $n = 100$. The sample CCFs are given in figure 4.1. The significance limits are included as dotted lines. For the input x_1, the sample cross-correlation to y is significant at lags $\omega_1 = \{3, 4, 5, 7\}$. From the right panel, we can see that $\omega_2 = \{0, 1\}$.

We have reason to believe that y is dependent of x_1 on lags ω_1, and dependent of x_2 on lags ω_2. That is, the coefficients $\beta_{j,k} \neq 0$ for $k \in \omega_j$. Furthermore, note that the max lag is $K = 7$ and that

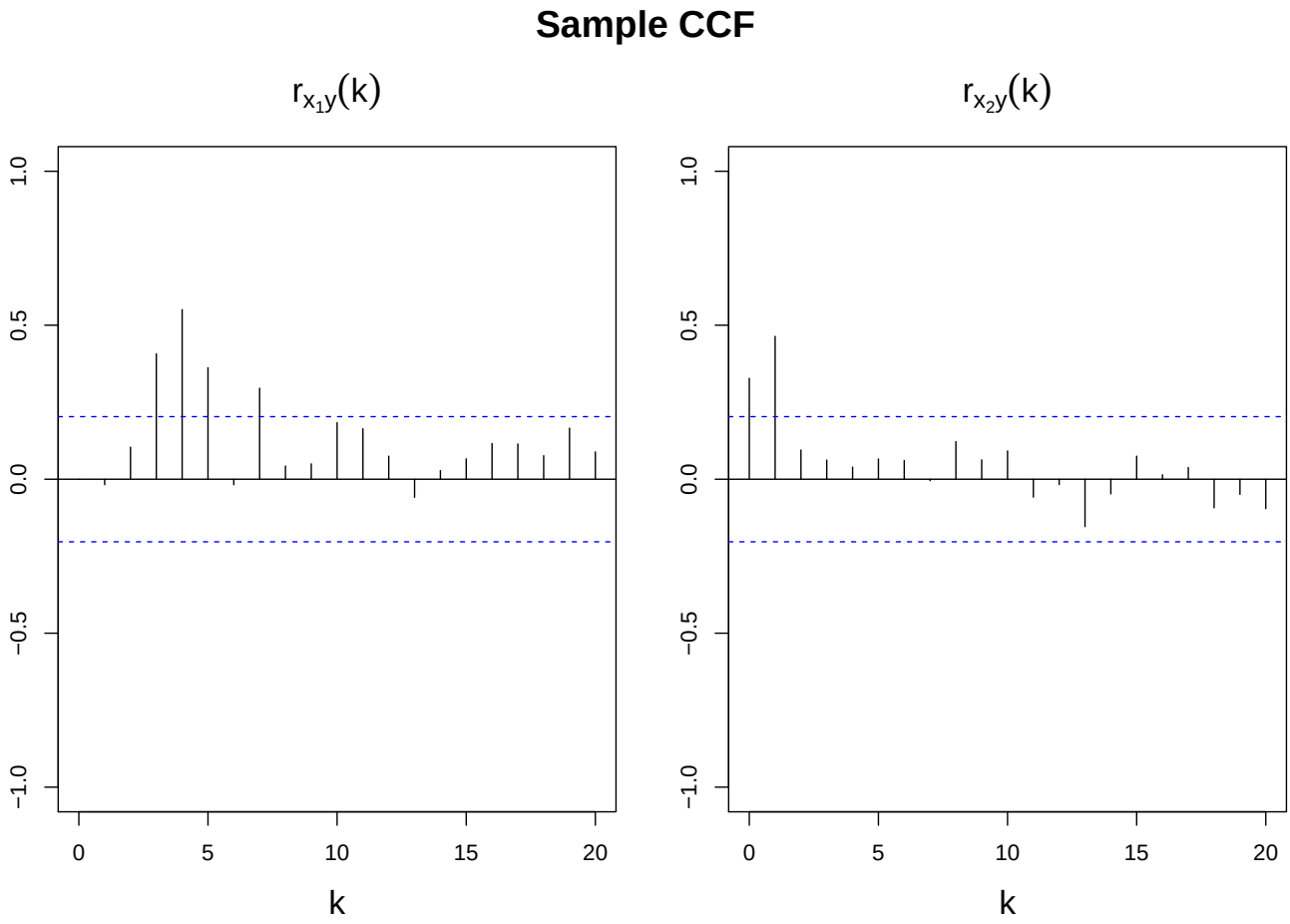

Figure 4.1: Sample cross-correlations based on observed input x_1 and x_2 and output y. On the left is the sample CCF between x_1 and y. On the right is the same for x_2 and y. The blue dotted lines are significance limits with the significance level set to 0.05.

the number of observations is $n = 100$. In this example, the model in (4.2) reduces to

$$\begin{aligned} y_t &= \sum_{j=1}^{2} \sum_{k \in \omega_j} \beta_{j,k} x_{j,t-k} + \epsilon_t \\ &= \beta_{1,3} x_{1,t-3} + \beta_{1,4} x_{1,t-4} + \beta_{1,5} x_{1,t-5} + \beta_{1,7} x_{1,t-7} \\ &\quad + \beta_{2,0} x_{2,t} + \beta_{2,1} x_{2,t-1} + \epsilon_t, \qquad t = 8, \ldots, 100. \end{aligned} \tag{4.4}$$

It is straightforward to fit this model with ordinary least squares. The OLS will provide estimates for $\beta_{1,3}$, $\beta_{1,4}$, $\beta_{1,5}$, $\beta_{1,7}$, $\beta_{2,0}$, $\beta_{2,1}$. Based on the sample CCF, the remaining $\beta_{j,k}$ are deemed equal to zero. However, it might not be this simple. The sample CCF should be used with care. In fact

we have no guarantee that a regressor can be dropped even if it is weakly correlated with output. The approach in the next section deals with this problem.

III Cross-correlation Selection Estimation (CSE)

This section presents a teqnique for modelling output in a lagged process. We will call it Cross-correlation Selection Estimation (CSE). The sample CCF is used to pick a pool of regressors to explain the output with. Then the best subset of regressors are chosen, and finally OLS is used to estimate the regression coefficients. First, we will discuss how the sample CCF can be used for selection of regressors.

For a large process like a furnace, we expect any change in the dynamics of the process to be slow and continuous. This is due to complexity and size. We have seen that the output depends on each input variable on potentially multiple lags. In the previous section, relevant lags were extracted using the sample CCF. However, the model may be improved if we use our a priori knowledge about the process. As the overall process exhibits continuity in some sense, we expect the same for the dynamic relationship between y and each x_j. Specifically, it is reasonable that x_j affect y through all lags in some interval, rather than every other lag for example.

In the example introduced in the previous section we saw that the sample cross-correlation between x_1 and y were significant at lags $\omega_1 = \{3,4,5,7\}$. Furthermore, the proposed model suggested that the coefficients on all other lags were zero, that is $\beta_{1,k} = 0$ for $k \notin \omega_1$. However, based on the discussion in this section, we should reconsider whether $\beta_{1,6} = 0$. When all neighbouring coefficients are non-zero, it could easily be the same for $k = 6$ based on our asssumption of continuity. Let $m_j = \min \omega_j$ and $M_j = \min \omega_j$. Then we can construct a continous extension of ω_j defined as

$$\Omega_j = \left\{ k \in \mathbb{N} : \, m_j \leq k \leq M_j \right\}. \tag{4.5}$$

In words, this is all lags from the smallest lag to the greatest lag at which the sample CCF is significant. From figure 4.1, we can see that $\Omega_1 = \{3,4,5,6,7\}$ and $\Omega_2 = \{0,1\}$. Furthermore, we

can do a regression with Ω_j instead of ω_j. In general, that is

$$y_t = \sum_{j=1}^{p} \sum_{k \in \Omega_j} \beta_{j,k} x_{j,t-k} + \epsilon_t, \qquad t = K+1, \ldots, n. \tag{4.6}$$

In our example, the model in (4.4) will have an extra term $\beta_{1,6} x_{1,t-6}$ added to it. Whether this term is important, will be apparent when comparing the model fit. Maybe it turns out that $x_{1,t-6}$ is not important, such that we should consider dropping it after all.

Including larger sets of regressors Ω_j is an attempt at making the method robust. We do not want to drop terms that are significant in the regression. The idea is to initially rather include too many terms than too few. Then we will fit the regression model in (4.6) and see whether we should reduce the number of regressors. It will be clear why we will need to drop regressors from (4.6) for certain cases. We can use figure 4.1 for an example. Imagine that $r_{x_1 y}(19)$, by chance were slightly larger, such that ω_1 included 19. Then, Ω_1 would be all lags from 3 to 19. The resulting model could be an overfit as we have 19 regressors with only 100 observations.

It is practical to write the regression model in (4.6) in matrix form when later discussion estimators. Including an intercept β_0 makes these estimators unbiased. There are $r = \sum_{j=1}^{p} \left(M_j - m_j + 1\right)$ regressors excluding the intercept. Let the vector of regression coefficients be $\beta = \left(\beta_0, \beta_1, \ldots, \beta_r\right)^\mathsf{T}$. The first column of the design matrix is all ones since we have an intercept. This is a column vector of $n - K$ ones denoted $\mathbb{1}$. Then, the model can be written as

$$y = \beta X + \epsilon, \tag{4.7}$$

where

$$\begin{aligned} y &= \left(y_n, y_{n-1}, \cdots, y_{K+1}\right)^\mathsf{T}, \\ \epsilon &= \left(\epsilon_n, \epsilon_{n-1} \ldots, \epsilon_{K+1}\right)^\mathsf{T} \\ X &= \left(\mathbb{1} \;\; X_1 \;\; X_2 \;\; \cdots \;\; X_p\right), \end{aligned}$$

$$X_j = \begin{pmatrix} x_{j,n-m_j} & x_{j,n-m_j-1} & \cdots & x_{j,n-M_j} \\ x_{j,n-m_j-1} & x_{j,n-m_j-2} & \cdots & x_{j,n-M_j-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{j,K-m_j} & x_{j,K-m_j-1} & \cdots & x_{j,K-M_j} \end{pmatrix}.$$

There are a total of $r+1$ regressors and $n-K$ statistical units. Thus the design matrix, X, is $(n-k)\times(r+1)$. The coefficient vector, β, has dimensions $(r+1)\times 1$ and y is $(n-K)\times 1$. The ordinary least squares estimator is

$$\hat{\beta} = (X^\mathsf{T} X)^{-1} X^\mathsf{T} y. \tag{4.8}$$

This estimator is unbiased (Hayashi, 2000), i.e. $\mathrm{E}[\hat{\beta}] = \beta$. Hence, the modeled value of output is $\hat{y} = \hat{\beta} X$ and the residuals are $\hat{\epsilon} = y - \hat{y}$. The stated model is only useful if it explains how the output changes with input. However, there is a limit to how well we can predict a variable based on data. There will always be some variability in the output that is unexplainable. Still, we can measure the amount of variation in the output that is explained by the model. This is a number between zero and one called R-squared. Unfortunately, the R-squared will typically be close to one both for correct models and overcomplicated models. If we keep adding regressors to a model, increasingly more of the variation in the observed output will be explained. Eventually, we begin modeling the random noise in the data. This is overfitting, and it is not penalized by R-squared. For this reason (Theil, 1961) introduced the adjusted R-squared, defined as

$$\bar{R}^2 = 1 - \frac{\hat{\epsilon}^\mathsf{T}\hat{\epsilon}}{(y-\bar{y})^\mathsf{T}(y-\bar{y})} \cdot \frac{n-1}{n-r-1}.$$

Here, the nominator $\hat{\epsilon}^\mathsf{T}\hat{\epsilon}$ is an estimator of the population variance of ϵ_t with $n-r-1$ degrees of freedom. The denominator $(y-\bar{y})^\mathsf{T}(y-\bar{y})$ is an estimator of the population variance of y_t with $n-1$ degrees of freedom. The R-squared is defined similarly, but without the fraction of the degrees of freedom. The adjusted R-squared does well in comparing models with different numbers of regressors.

We can use $\bar{R}^2$ to see whether regressors should be dropped from the model in (4.7). Each regressor is represented as a column in the design matrix, that is $X = (\mathbb{1}, x^2, \ldots, x^{r+1})$. Let a sub-

set of the column indices be $I = \{i_1, i_2, \ldots, i_N\} \subset \{1, 2, \ldots, r+1\}$. Then, the design matrix with regressors I retained is $X_I = (x^{i_1}, x^{i_2}, \ldots x^{i_N})$, where the order of the columns is preserved, i.e. $2 \leq i_1 < i_2 < \cdots < i_N \leq r+1$. Note that the intercept is not dropped. We express the corresponding coefficients as $\beta_I = (\beta_0, \beta_{i_1}, \beta_{i_2}, \ldots, \beta_{i_N})^\mathsf{T}$. The reduced model and its OLS estimator are

$$\begin{aligned} y &= \beta_I X_I + \epsilon_I, \\ \hat{\beta}_I &= (X_I^\mathsf{T} X_I)^{-1} X_I^\mathsf{T} y. \end{aligned} \tag{4.9}$$

There are $N+1$ estimated coefficients. Furthermore, the residuals of this model is $\hat{\epsilon}_I = y - \hat{\beta}_I X_I$. Then, the adjusted R-squared for the subset I can be expressed as

$$\bar{R}_I^2 = 1 - \frac{\hat{\epsilon}_I^\mathsf{T} \hat{\epsilon}_I}{(y-\bar{y})^\mathsf{T}(y-\bar{y})} \cdot \frac{n-1}{n-N-1}. \tag{4.10}$$

The computed $\bar{R}^2$ is not a measure of model fit. Thus, we need to be careful using it. However, $\bar{R}^2$ may work well for variable selection. Consequently, we will use $\bar{R}^2$ in variable selection and then look at diagnostics to see whether the resulting model is adequate. A candidate model is

$$y = \beta_{\hat{I}} X_{\hat{I}} + \epsilon_{\hat{I}}, \qquad \hat{I} = \underset{I}{\operatorname{argmax}}\, \bar{R}_I^2. \tag{4.11}$$

This is the model that maximizes $\bar{R}^2$ among all submodels of (4.7). An approach for finding the best subset, is simply computing $\bar{R}^2$ for all. Implementing such a method is straightforward, but it can be computationally expensive as the number of observations and variables increase. For each subset I, we have to compute least squares estimates, which is approximately nN^2 operations. When summing over all subsets there is a total of $n \cdot \sum_{N=1}^{r+1} \binom{r+1}{N} N^2$ operations. By the binomial theorem we obtain

$$\sum_{N=1}^{r+1} \binom{r+1}{N} x^N = (1+x)^{r+1}.$$

Taking the derivative in x on both sides, multiplying by x and differentiating again results in an

expression that when inserting for $x = 1$ results in

$$\sum_{N=1}^{r+1} \binom{r+1}{N} N^2 = (r+1)\left(\tfrac{r}{2}+1\right)2^r$$
$$= \mathcal{O}(r^2 2^r).$$

Thus, the asymptotic complexity of the exhaustive subset selection is $\mathcal{O}(r^2 2^r n)$. Consequently, this is extremely computationally expensive already as r approaches one hundred. An alternative implementation for large r should be considered. Adding a ceiling for the size of the subset is one option.

Recall that we wish to find the model (4.11). It might not be necessary to compute every subset, but rather do some iterative optimization. An idea is to start with the full set of regressors, $I = \{1, 2, \ldots, r+1\}$. Then we drop a regressor and see how $\bar{R}_I^2$ changes. Regressors are dropped one at a time as long as $\bar{R}_I^2$ increases or remains about the same. Which variable to drop is decided by the p-values of the coefficients in the regression.

Formally, a two-sided t-tests for zero effect is performed for each regressor x^i in the current regression model. Let the coefficient estimate be denoted $\hat{\beta}_i$. The t-distributed test statistic is

$$t_i = \hat{\beta}_i \sqrt{(n-K-2)\frac{(x^i - \bar{x}^i)^\mathsf{T}(x^i - \bar{x}^i)}{\hat{\epsilon}^\mathsf{T}\hat{\epsilon}}}.$$

Let T be t-distributed on $n-K-2$ degrees of freedom. The p-value of the two-sided t-test is

$$p_i = 2\Pr(T \geq |t_i|).$$

The updated set of regressors is then

$$I^* = I \setminus \left\{x^{i^*}\right\}, \qquad i^* = \operatorname*{argmax}_i p_i.$$

This method is an iterative search for the optimal regression model. It is an alternative to the exhaustive subset selection when r is large. The iterative search will not allways find the preferred model defined by (4.11), while the exhaustive search always do. When implementing CSE, a

combination of exhaustive selecetion and iterative selection is suggested.

IV Maximum correlation estimation (MCE)

Section II explains how we can use the sample cross-correlation function $r_{x_j y}(k)$ to decide which lags to include in the regression model. The sample CCF is computed pairwise between output and each input x_j, which results in a set of lags to include for each x_j. These sets Ω_j are defined in (4.5). However, we previously stated that $r_{x_j y}(k)$ is influenced by ρ_{x_j}. Consequently, Ω_j may be a suboptimal choice depending on ρ_{x_j}. In this section we will look more into this concept and a possible solution is provided.

Consider the AR(2) processes in figure 3.5. Let these be the basis for six examples of model identification. For each process $i \in \{a, b, c, d, e, f\}$, let x_i be a realization of one thousand elements. For each input x_i, we generate output as a sum of x_i on lags 3, 4, and 5. Independent standard normal noise ϵ_i is also added to the model. This results in

$$y_{it} = x_{i,t-3} + x_{i,t-4} + x_{i,t-5} + \epsilon_{it}, \qquad t = 1, 2, \ldots, 1000, \quad i \in \{a, b, c, d, e, f\}. \tag{4.12}$$

This is a regression model where the coefficients are all one. These six examples all consist of one input x_i and one output y_i. The first step of model identification is computing $r_{x_i y_i}(k)$. Based on the sample CCF, we choose the lags to include in the regression model, as the data generating process in (4.12) is unknown.

Figure 4.2 includes the true regression coefficients, the ACF of x_i and the sample CCF between x_i and y_i for each example. Each column correspond to an example. Consider example a, which is described by the first column. The true regression coefficients in (4.12) are one for lags 3, 4 and 5 and zero otherwise. These are plotted in the top-left panel. In the panel below, we can see the theoretical ACF of the process generating x_a. No lags except zero are significant. The lower panel shows a plot of $r_{x_a y_a}(k)$. The realtionship suggested by the sample CCF is accurate. This is due to a small autocorrelation in input. However, this is not the case for the other examples.

By comparison of the ACF and the sample CCF in the examples, we can see a strong connection.

The sample CCF inherit the behavior of the input ACF. Recall that the ACF is a damped sine wave. The sample CCF is approximately the same wave shifted by 4, which is the average lag between input and output. Recall that Ω_i is the set of all lags between the smallest and largest lag such that the sample CCF is significant. We can see that Ω_a includes excatly the relevant lags, while Ω_f includes many more. Having too many lags in the model is suboptimal, even though these sets are later subsetted. With a large pool of lags to chose from, we are not guaranteed to retrieve the optimal subset. This is the reason why we should consider an alternative approach, especially when the autocorrelation in input is dominant.

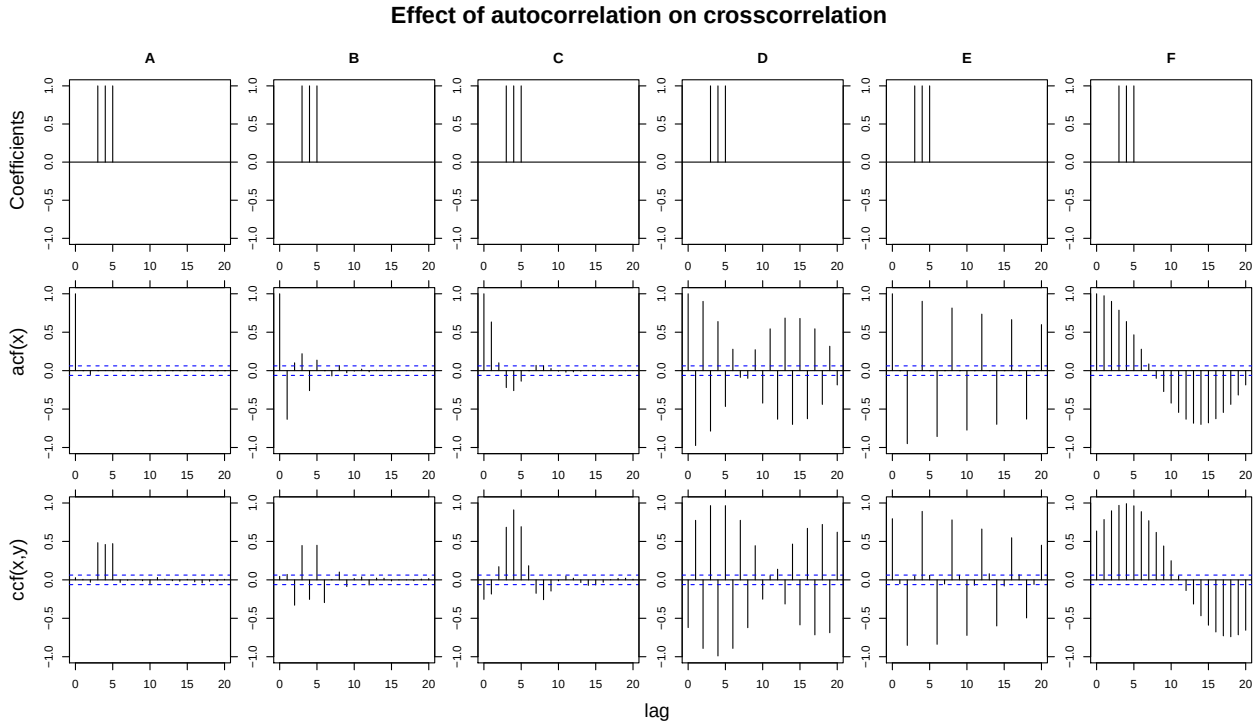

Figure 4.2: A summary of six regression models. Each column represent an example where the top panel is the regression weight, the center panel is the input ACF and the lower panel is the sample CCF between input and output.

Instead of performing CSE, we can build a solution by using the maximum of the sample CCF. Note that the lag where the sample CCF takes its maximum coincides well with the lag between input and output. Let the lag of the maximum sample CCF be $\mu_i = \text{argmax}_k \, r_{x_i y_i}(k)$. The idea is to include all lags in some neighborhood of μ. This neighborhood should be wide enough to

cover the important lags, but not wider. Consider the set of all lags closer than Δ from μ, i.e.

$$\Theta_i = \left\{k \in \mathbb{N} : \mu - \Delta \leq k \leq \mu + \Delta\right\}.$$

This set suggest which input lags to include when modeling output. Instead of building the model in (4.7) based on Ω_i, we may use Θ_i. Furthermore, the final model is the optimal subset defined in (4.11). This approach will be referred to as Maximum Correlation Estimation (MCE). Let us compare Ω_i and Θ_i for the AR(2) processes in figure 4.2. The result when $\Delta = 2$ is shown in table 4.1. Each row correspond to an example. The last column in the table shows the lags included for CSE. The regressor set Θ_a is perfect. However, as ρ_{x_i} decays more slowly, Θ_i eventually includes all lags in the specified range. In comparison, MCE will allways include $2\Delta+1 = 5$ lags. We can also see that Θ_i consistently includes the true set of lags. MCE clearly performs well for these examples. However, the result would be less optimal with a Δ specified differently.

Table 4.1: Comparison of the regressors sets Θ_i and Ω_i of MCE and CSE respectively. Six examples are included, with the name of each example in the first column. The second column includes the lags in the true model. The third and fourth column includes the lags in the MCE and the CSE.

i	True	Θ_i	Ω_i
a	$\{3,4,5\}$	$\{1,\ldots,5\}$	$\{3,\ldots,5\}$
b	$\{3,4,5\}$	$\{3,\ldots,7\}$	$\{1,\ldots,12\}$
c	$\{3,4,5\}$	$\{2,\ldots,6\}$	$\{0,\ldots,15\}$
d	$\{3,4,5\}$	$\{2,\ldots,6\}$	$\{0,\ldots,20\}$
e	$\{3,4,5\}$	$\{2,\ldots,6\}$	$\{0,\ldots,20\}$
f	$\{3,4,5\}$	$\{2,\ldots,6\}$	$\{0,\ldots,20\}$

We have seen that MCE outperforms CSE for complex AR(2) processes. This result can be generalized to processes where the input variables have a slowly decaying autocorrrelation function. However, there are some challenges with MCE, including specification of Δ. Regardless of chosen estimation tecnique, it is necessary to perform analysis on the reliability of the resulting model. The next section includes tests for accuracy of the model.

V Diagnostics

Performing diagnostics of a regression model is first of all an assessment of the statistical assumptions. Consider a regression model $y = \beta X + \epsilon$. The least squares estimator $\hat{\beta}$ is only valid under certain assumptions (Hayashi, 2000). These are

1. **Homoscedasticity** Same variance σ_ϵ^2 in the error term of every observation, i.e. $\mathrm{E}[\epsilon_t^2] = \sigma_\epsilon^2$.
2. **Strict exogeneity** The regression errors have mean zero, $\mathrm{E}[\epsilon_t] = 0$.
3. **No serial correlation** Errors are uncorrelated, which involves no repeating patterns. Formally, $\mathrm{E}[\epsilon_i \epsilon_j] = 0$, for all (i, j) with $i \neq j$.

There exist numerous methods for assessing these assumptions. A natural first step is plotting the error sequence. The plot will give us an idea of whether the error variance is constant in t. This is a necessary condition for homoscedasticity. Additionally, we can perform a formal test such as the Breush-Pagan test (Breusch and Pagan, 1979).

As for the third assumption, we can simply use the sample ACF and sample PACF defined in chapter 3. Recall that both these are derived from estimates of the autocovariance, $\gamma_{i-j} = \mathrm{E}[\epsilon_i \epsilon_j]$. Hence, the third assumption may not hold if either the ACF or PACF is significant for some lag. Autocorrelation in input may also violate strict exogeneity. Especially if not all appropriate lags for the input is included in the regression.

Let us continue with the example introduced in section II. The output y is explained by input variables x_1 and x_2. The sample cross-correlations between input and output are shown in figure 4.1. When modeling the data by (4.11) and estimating the coefficients by OLS, we obtain the fitted model

$$y_t = -0.04 + 2.59x_{1,t-3} + 2.69x_{1,t-4} + 2.13x_{1,t-5} + 2.29x_{1,t-7} + 1.79x_{2,t} + 2.80x_{2,t-1} + \hat{\epsilon}_{\hat{I},t}. \quad (4.13)$$

The coefficients on the regressors are in the same proportions to each other as for the sample CCFs in 4.1. Apparently, there are no decisive complications in the identification of the dynamic relationship between input and output. The residuals $\hat{\epsilon}_{\hat{I},t}$ are plotted in figure 4.3 along with

the sample ACF and sample PACF of the residuals. The plot in the top frame suggests that the variance in the residuals is not changing much with time. Thus, there is no clear violation of homoscedasticity. The middle panel shows that there are a few borderline significant estimates of the autocorrelation among non-zero lags. From the lower plot, we can see that it is the same for the sample PACF. There are no clear correlation patterns in the residuals. This is promising for the validity of assumptions two and three of the least squares estimates.

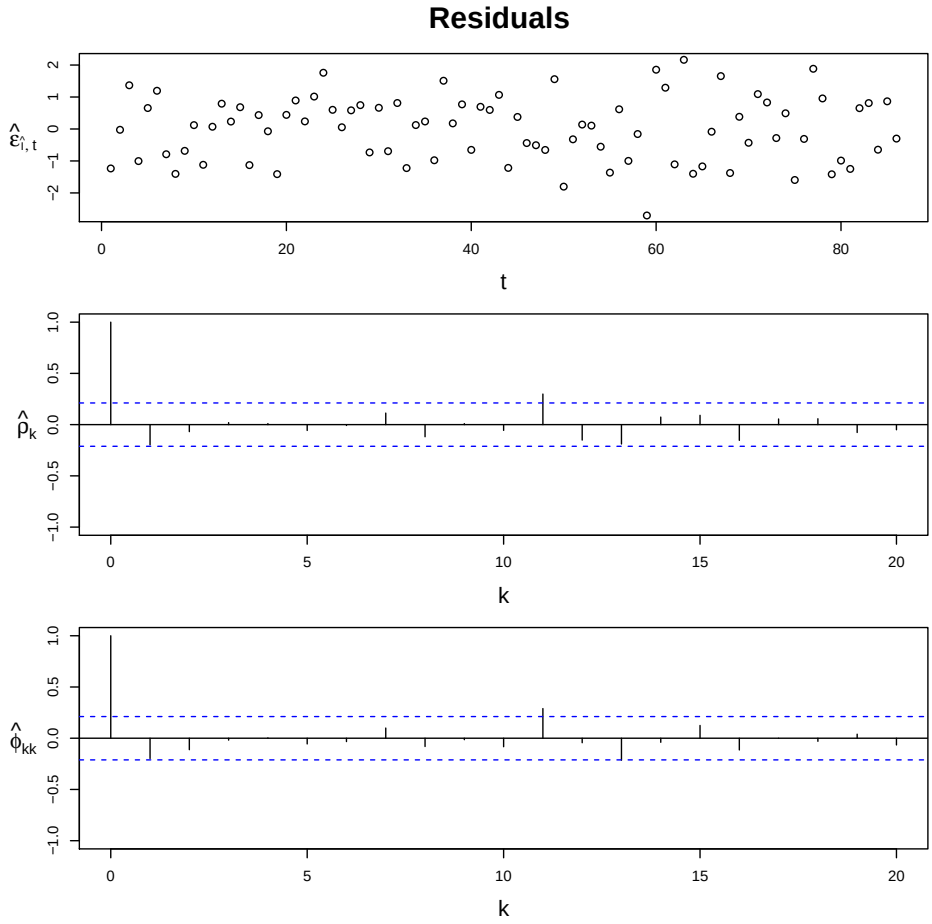

Figure 4.3: The top panel displays the residuals in a model fitted by ordinary least squares. The middle panel includes the sample ACF of these, and the bottom panel shows the sample PACF. The blue dotted lines are 95 percent significance limits for the estimates.

Imagine that the sample PACF and the sample ACF in figure 4.3 were significant for some lags. This would suggest that there are dynamics in the output not explained by our regression model. Let us continue with the same example as before. The prediction in (4.13) is based on the model in (4.11), where $\hat{I} = \{x_{1,t-3}, x_{1,t-4}, x_{1,t-5}, x_{1,t-7}, x_{2,t}, x_{2,t-1}\}$. Consider instead a subset where

some regressors are dropped, say $I^* = \{x_{1,t-7}, x_{2,t}, x_{2,t-1}\}$. The resulting model fitted by OLS is

$$y_t = 0.21 + 1.83x_{1,t-7} + 1.99x_{2,t} + 2.74x_{2,t-1} + \hat{\epsilon}_{I^*,t}. \tag{4.14}$$

The residuals $\hat{\epsilon}_{I^*,t}$ are shown in the top-left plot in figure 4.4. There is a clear oscillating trend in the residuals. From the sample ACF and sample PACF plotted below the residual sequence, we can see that there is a correlation pattern in the residuals. Specifically, $\hat{\rho}_k$ experience damped oscillations while $\hat{\phi}_{kk}$ cuts off after $k = 2$. Thus, the residuals follow an AR(2) process. It seems reasonable to model the errors by the same model, i.e.

$$\begin{aligned} & \epsilon_{I^*,t} - \phi_1\epsilon_{I^*,t-1} - \phi_2\epsilon_{I^*,t-2} = a_t \\ \text{or} \qquad & \phi(B)\epsilon_{I^*,t} = a_t, \end{aligned} \tag{4.15}$$

where a_t is white noise. The knowlegde about the structure of the residuals should be incorporated into the regression model. Let the coefficients in the model with y and I^* be denoted $\beta_{I^*} = (\beta_0^*, \beta_1^*, \beta_2^*, \beta_3^*)^\mathsf{T}$. Then, we can write the model as

$$\begin{aligned} & y = \beta_{I^*} X_{I^*} + \epsilon_{I^*} \\ \text{or} \qquad & y_t = \beta_0^* + \beta_1^* x_{1,t-7} + \beta_2^* x_{2,t} + \beta_3^* x_{2,t-1} + \epsilon_{I^*,t}. \end{aligned}$$

The problem with this model is the autocorrelation in $\epsilon_{I^*,t}$. We can use the result from (4.15) to fix this problem. If the operator $\phi(B)$ is applied to the errors, the result is independent white noise. Thus, applying this operator to each term in the original model results in

$$\begin{aligned} \phi(B)y_t &= \phi(B)\beta_0^* + \beta_1^*\phi(B)x_{1,t-7} + \beta_2^*\phi(B)x_{2,t} + \beta_3^*\phi(B)x_{2,t-1} + \phi(B)\epsilon_{I^*,t} \\ &= \phi(B)\beta_0^* + \beta_1^* x_{1,t-7} - \beta_1^*\phi_1 x_{1,t-8} - \beta_1^*\phi_2 x_{1,t-9} + \beta_2^* x_{2,t} - \beta_2^*\phi_1 x_{2,t-1} \\ &\quad - \beta_2^*\phi_2 x_{2,t-2} + \beta_3^* x_{2,t-1} - \beta_3^*\phi_1 x_{2,t-2} - \beta_3^*\phi_2 x_{2,t-3} + a_t \\ \implies \quad y_t &= \phi(B)\beta_0^* + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \beta_1^* x_{1,t-7} + (-\beta_1^*\phi_1)x_{1,t-8} + (-\beta_1^*\phi_2)x_{1,t-9} \\ &\quad + \beta_2^* x_{2,t} + (-\beta_2^*\phi_1 + \beta_3^*)x_{2,t-1} + (-\beta_2^*\phi_2 - \beta_3^*\phi_1)x_{2,t-2} + (-\beta_3^*\phi_2)x_{2,t-3} + a_t. \end{aligned}$$

Renaming the intercept and coefficients of each regressor results in the expanded regression model

$$y_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \beta_3 x_{1,t-7} + \beta_4 x_{1,t-8} + \beta_5 x_{1,t-9} + \beta_6 x_{2,t} + \beta_7 x_{2,t-1} + \beta_8 x_{2,t-2} + \beta_9 x_{2,t-3} + a_t. \tag{4.16}$$

We will call this the AR expanded model. Note that there are lagged versions of the output on the right hand side of the equation. The purpose of the expanded regression is fixing the problem of autocorrelation in the residuals of the original model. The idea of new model is to have uncorrelated errors a_t.

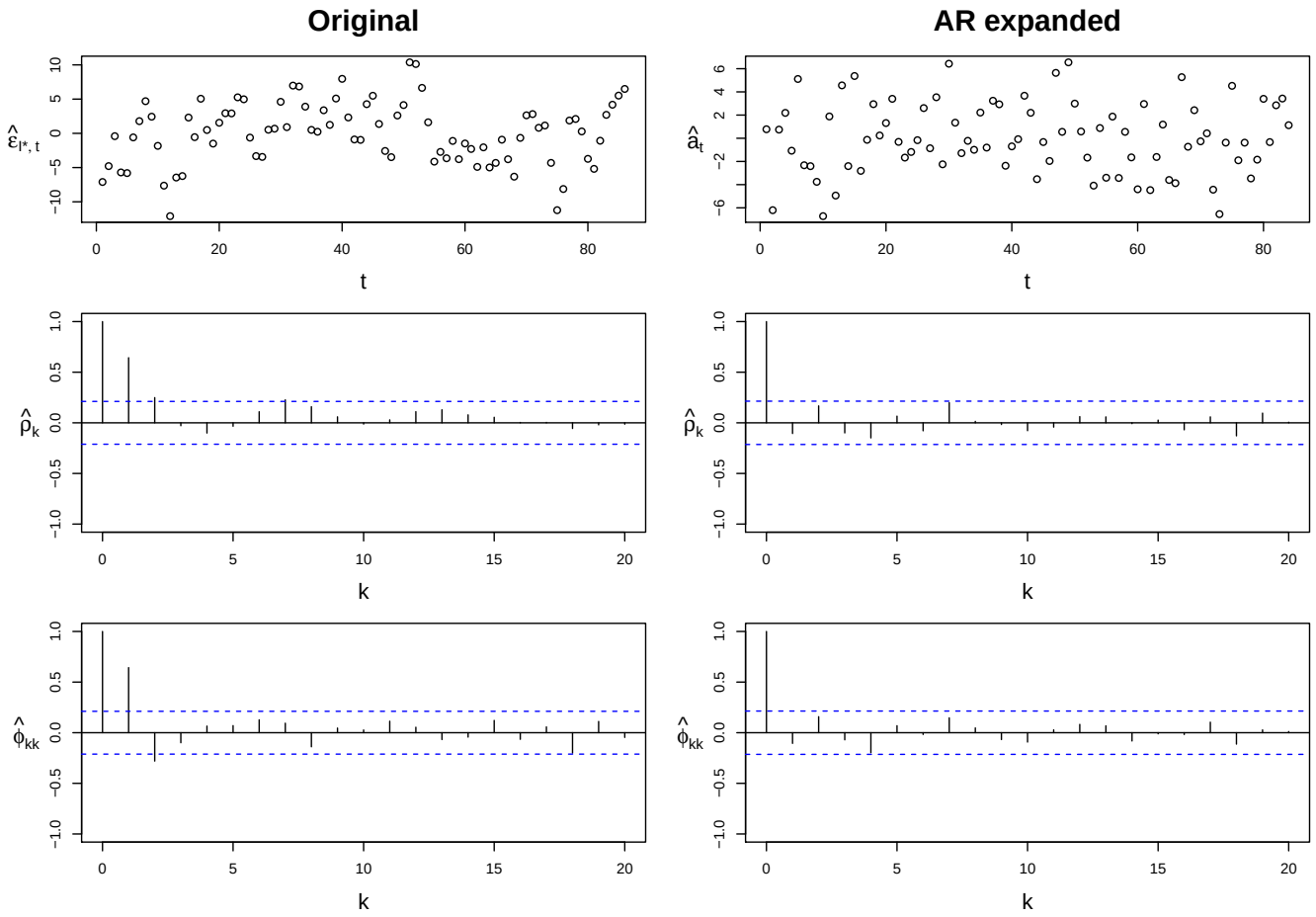

Figure 4.4: Comparison of residuals of two models fitted by least squares. The left column shows the residuals of a suboptimal model and the sample ACF and sample PACF of these residuals. The right column displays the resulting residuals after expanding the original model by an AR polynomial.

Fitting the model in (4.16) by ordinary least squares results in

$$y_t = 0.15 + 1.02y_{t-1} - 0.53y_{t-2} + 3.00x_{1,t-7} - 2.38x_{1,t-8} + 1.36x_{1,t-9} + 2.38x_{2,t} + 0.76x_{2,t-1} - 1.72x_{2,t-2} + 1.33x_{2,t-3} + \hat{a}_t. \quad (4.17)$$

The residuals $\hat{a}_t$ are plotted in the top-right panel of figure 4.4. The sample ACF and sample PACF of the residuals are also included in the right column of the figure. All three indicate that the coefficient estimates in (4.17) are valid. There is a vague pattern in the residual plot, but as long as the correlation plots are fine, this is of little concern. It is clear that the AR expanded model do better than the original.

The AR expansion worked well in the above example. However, a small modification of this method will be necessary in some cases. Consider the fitted model in (4.14). Since the residuals are autocorrelated, we might have biased coefficient estimates $\hat{\beta}_i^*$. These estimates are used for modelling of the errors as described in (4.15), and consequently the error model can be biased. This means that the coefficients $\hat{\beta}_i$ in the resulting AR expanded model might also be biased. If the original estimates $\hat{\beta}_i^*$ are unbiased, they will be similar to the updated estimates corresponding to the same regressors. Hence, we should repeat the AR expansion procedure until the coefficient estimates remains unchanged. If the coefficient estimates does not stabilize after a couple of iterations, more drastic changes are required.

Recall the OLS assumptions in the beginning of this section. In general, fitting the residuals by an AR(p) model is recommended if these assumptions are violated. Furthermore, the resulting AR polynomial operator is applied to both sides of the regression model as demonstrated in the example above. The number of regressors will increase by doing this. More terms will be added when the order of the AR polynomial is large. Since we want to be conservative in adding terms, p should not be too large.

I suggest the folllowing approach. For each value of p from one to some upper bound P, fit the residuals by an AR(p) model. This is done by standard computer approaches discussed in chapter 3. Then we compare the model fit for the various orders p. The Bayesian informasion criterion (BIC) (Schwarz et al., 1978) is suitable for this purpose. BIC is a criterion for model

selection that heavily penalizes the number of parameters in the model. It is defined as

$$\text{BIC} = (p+1)\log n + 2\log \hat{L},$$

where $\hat{L}$ is the maximized value of the likelihood of the model. The AR expansion will be of the order $\hat{p}$ that resulted in the lowest BIC. Consider a model with an input variable y and output variables $x_1, x_2, \ldots, x_p$. Let λ_j be the lags included for each variable. Then, the lags included for variable x_j in the AR expanded model are

$$\Lambda_j = \left\{k \in \mathbb{N} : \ l \leq k \leq l + \hat{p}, \ \ \forall \ l \in \lambda_j\right\}.$$

The resulting AR expanded regression model is

$$y_t = \beta_0 + \sum_{k=1}^{\hat{p}} \beta_k y_{t-k} + \sum_{j=1}^{p} \sum_{k \in \Lambda_j} \beta_{j,k} x_{j,t-k}, \qquad t = K + \hat{p} + 1, \ldots, n.$$

Recall that the expansion should be repeated if the coefficient estimates were changed a lot by the current expansion.

Chapter 5

Results

Chapter 4 express the relationship between variables by a regression model. The model is fitted by observations and can be used for prediction of future output. Say we want to apply this method to a a full scale industry process like a furnace, where there is an output variable that we wish to minimize or maximize. This may be done by adjusting the process input variables according to the prediction model. However, the precision in the model must be very high before we can rely on it to make changes in production.

In this chapter we will discuss approaches for assessing the accuracy of the model. Furthermore, we will consider cases where the methods will fail. The methods will fail when applied to systems where the assumptions do not hold. First of all, recall that the suggested regression model only includes first-order terms. When the system exhibits higher-order relationships or more complex functions, the model fit is not satisfactory. However, it will be clear when important regressors are left out. The diagnostics of the final model should always be considered.

I Data simulation

One way of assessing the method is to simulate a process with a known relationship between input and output and see whether the fitted model is similar. In order for the simluation to be relevant, it must mimic the actual process. In the ElMet project we are interested in how well

the model describes a furnace. Hence, the simulated data should be similar to furnace vaiables. First we will need to decide the distribution of the input variables. Let the input sequence x_j be a realization of a multivariate random variable $X_j = (X_{j,1}, \ldots, X_{j,n})^\mathsf{T}$ governed by either of three mechanisms, the normal distribution, the uniform distribution or an ARMA process. These data generating processes are chosen because they frequently arise in nature. The idea is to test the performance of the method for standard distributions with a wide range of shape and scale variations. The parameters in each mechanism are chosen at random for each X_j, but constant in time. The input sequence X_j is generated as either V, W or Z with equal probability, where

$$\begin{aligned} V_t &\sim \mathcal{N}(\mu, \sigma_v^2), \\ W_t &\sim \mathcal{U}(-1,1), \\ Z_t &= p\phi_1 x_{t-1} + q\theta_1 \epsilon_{t-1} + \epsilon_t, \\ \epsilon_t &\sim \mathcal{N}(0,1) \\ \mu, \sigma_v^2, \phi_1, \theta_1 &\sim \mathcal{U}(-1,1). \end{aligned}$$

The value of (p,q) is either $(1,0)$, $(0,1)$ or $(1,1)$ with equal probability. This corresponds to AR(1), MA(1) and ARMA(1,1) processes. The simluation approach above is used to draw input sequences $x_1, x_2, \ldots, x_p$. The input sequences are then standardized.

Next, we must decide the lags between input and output. We assume that y depend on x_j on consequtive lags $m_j, m_j + 1, \ldots, M_j$. The minimum and maximum lags are integers drawn randomly from discrete uniform distributions. Specifically,

$$\begin{aligned} m_j &\sim \mathcal{U}\{0,7\}, \\ M_j &\sim \mathcal{U}\{m_j, m_j + 3\}. \end{aligned}$$

The distribution of the lag interval is chosen by consideration of a real furnace process. The maximum lag is $K = \max_j M_j$. Furthermore, the coefficients $\beta_{j,k}$ corresponding to the regressors are drawn uniformly from the interval $[-3,-1] \cup [1,3]$. Note that coefficients close to zero are excluded. The smallest coefficients are not of of interest with standardized variables. Furthermore, standardization means that only the *relative* sizes of the coefficients are of interest.

Finally, we express the output by the linear combination of the regressors and added i.i.d. normal noise $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$, i.e.

$$y_t = \sum_{j=1}^{p} \sum_{k=m_j}^{M_j} \beta_{j,k} x_{j,t-k} + \epsilon_t, \qquad t = K+1, \ldots, n. \tag{5.1}$$

The output variable is standardized the same way as the input. The generated data now consists of input variables $x_1, x_2, \ldots, x_p$ and the output variable y. Next, we will discuss how well the methods described in chapter 4 identifies the relationship between the simulated variables.

II Performance

In this section we will describe the performance of CSE as defined in section III of chapter 4. We will fit the simulated data by the model in 4.11, where the coefficients are estimated by least squares. Recall that the algorithm first choose a large set of regressors which is then subsetted. The number of regressors in the large set is denoted r. Recall that we discussed two approaches for finding the optimal subset $\hat{I}$. An exhaustive search is used for $r \leq 25$ and an iterative search otherwise. The final estimates for the regression coefficients are the OLS estimates of the optimal subset.

If we want to know the accuracy of CSE, we must first define a measure. It is reasonable to compare the true coefficients in (5.1) with the least squares estimates of (4.11). Let $\hat{\beta}_{j,k}$ denote the estimates. The two models does not necessarily include the same terms. However, we can simply consider regressors not included to have a coefficient equal to zero. Let b be the number of pairs (j,k) such that etiher $\hat{\beta}_{j,k}$ or $\beta_{j,k}$ is non-zero. The distance between the true coefficients and the estimated ones may be defined as the mean square difference, i.e.

$$\delta = \frac{1}{b} \sum_{j=1}^{p} \sum_{k \geq 0} \left(\beta_{j,k} - \hat{\beta}_{j,k} \right)^2. \tag{5.2}$$

When a simulation has a small δ, the estimation was successful. Consider the simulation procedure in section I. There are a few parameters to be decided for each simulation. We must choose

the number p of input variables, the length n of the sequences and the variance σ^2 in the output defined in (5.1). The values of these parameters may affect the error δ. This will be clear if we run multiple simulations with various parameter values. We will discuss an example where we have a few values to choose from for each parameter. Then, for every combination of parameter values we compute δ. However, as the simulations are stochastic, one thousand replications of each experiment is performed, and the average $\bar{\delta}$ is computed. The levels of n is 500, 1000 and 5000. For p the levels are 2, 5, 10 and 30. The levels of the noise variance σ^2 are 0.1, 0.5, 1, 1.5 and 2. We can number the sixty different combinations by $s = 1,2,\ldots,60$. The total number of experiments is 60000.

Furthermore, we can do regressions with $\bar{\delta}$ as output to see which parameters are important for the performance of CSE. Intially, the included regressors are p, n and σ^2, pairwise interactions, squares and square roots. Then, backward elimination is performed with Akaike information criterion (AIC) as the model fit criterion (Venables and Ripley, 2002). For each model, let $\hat{L}$ be the maximized value of the likelihood function. Then the model fit criterion is defined as AIC $= 2(p - \hat{L})$. The backward elimination remove regressors that are not important for explaining $\bar{\delta}$. The resulting model for a specific parameter combination s is

$$\bar{\delta}_s = \beta_0 + \beta_1 n_s + \beta_2 p_s + \beta_3 \sigma_s^2 + \beta_4 n_s p_s + \beta_5 n_s^2 + \beta_6 \sqrt{p}_s + e_s,$$

where β_i are regression coefficients and e_s is a zero-mean normally distributed regression error. The OLS estimates of the regression coefficients are given in table 5.1. The right column states the p-values of a two-sided t-test with a null hypotheses that the coefficient is equal to zero. All regressors are significant on a 95 percent significance level.

Table 5.1: The effect of a parameter on the performance of CSE. The table is a summary of a regression with the estimation error $\bar{\delta}$ as the output. The left column names the regressors. The center column displays the OLS coefficient estimates, and the right column states the p-values of a two-sided t-test for the coefficient being zero.

Param.	$\hat{\beta}$	p -value
1	6.1×10^{-1}	$<2.0 \times 10^{-16}$
n	-6.0×10^{-5}	7.0×10^{-5}
p	1.1×10^{-2}	1.0×10^{-11}
σ^2	6.6×10^{-3}	3.7×10^{-2}
np	-8.5×10^{-7}	4.6×10^{-12}
n^2	1.1×10^{-8}	1.8×10^{-5}
$\sqrt{p}$	-5.6×10^{-2}	3.7×10^{-8}

Table 5.1 provide useful information about when CSE is reliable. The results concur with expectations. The error $\bar{\delta}$ is low for high n as we have more information. When the number of variables increases, the error increases because the model is more complex. Noise also reduce the accuracy of the estimation. The error is at the highest when p is close to n. This is the general case for any regression model. The significance of n^2 and $\sqrt{p}$ suggest that the error is not linear in n and p.

It is clear that the performance of CSE is heavily incluenced by the parameters in the observed data set. We should keep this in mind in applications. Next, we will consider CSE and MCE applied to an example.

III Example

Two estimation methods were introduced in chapter 4, Cross-correlation Selection Estimation (CSE) and Maximum Correlation Estimation (MCE). In this section we will see these methods applied to observed data. The two resulting models will be compared and we will discuss the diagnostics of both. One output variable y will be explained by three input variables x_1, x_2 and x_3. There are $n = 852$ observations of each variable. The variables describe a furnace process. Missing values are imputed before doing estimation. Specifically, there are four missing values in x_2 that is imputed by the method described in section V of chapter 3. The resulting time

series are plotted in figure 5.1. The imputed values $x_{2,549}$, $x_{2,597}$, $x_{2,598}$ and $x_{2,599}$ are joined by red dotted lines. As these values appear reasonable, we continue with estimation.

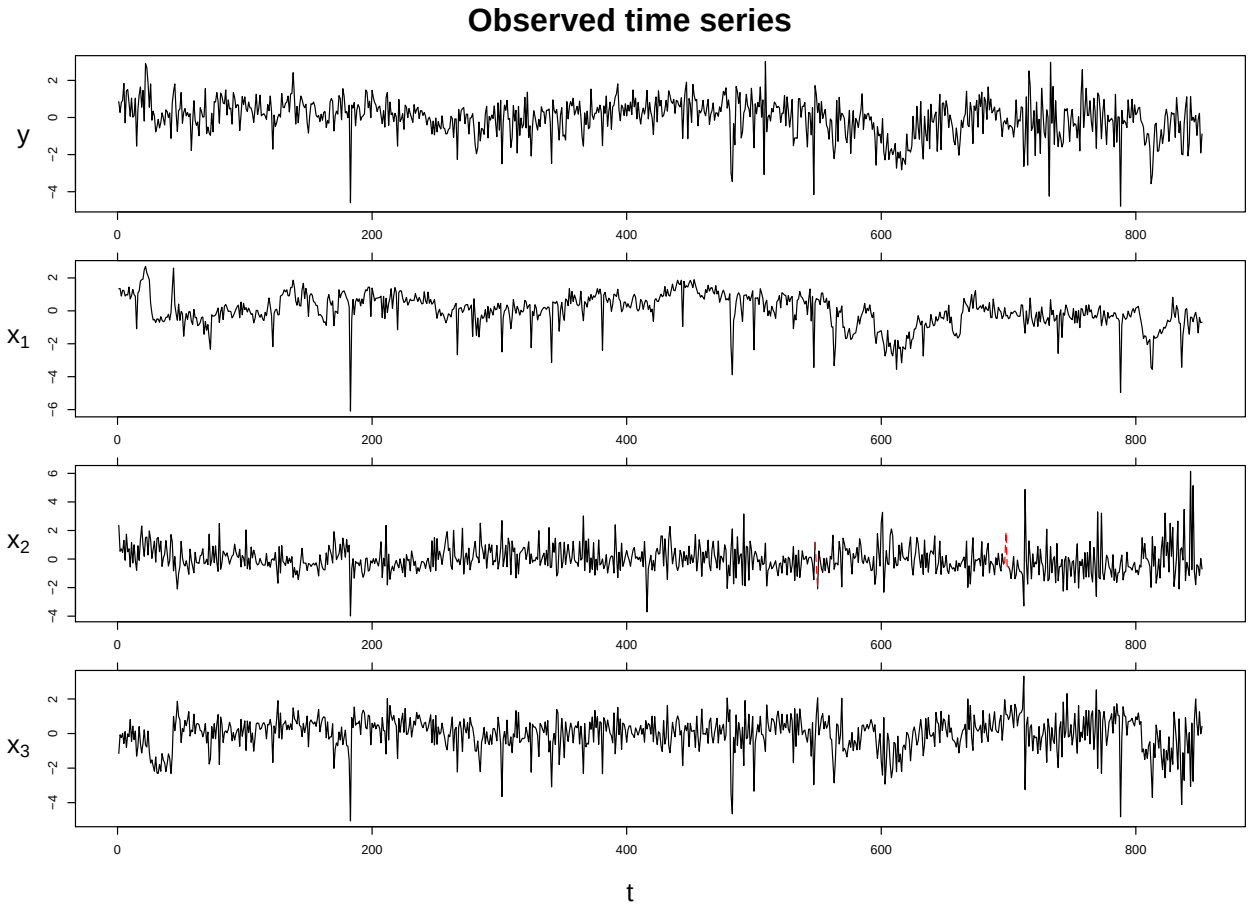

Figure 5.1: Plots of an observed output variable y and observed input variables x_1, x_2 and x_3. Some entries of x_2 were missing and has been replaced by inputed values. These are connected by red dotted lines.

The sets of regerssors to be included in the model is determined by the sample cross-correlation between input and output. The sample CCFs are plotted in figure 5.2. Recall that CSE create sets Ω_j of lags based on the significance of the sample CCF. On the other hand, MCE creates sets Θ_j based on the most extreme lag of the sample CCF. For this example we estimate all correlations with a lag of a week or less. The radius of the MCE interval is set to $\Delta = 2$. The resulting sets are displayed in table 5.2. Recall that both methods find the subset of the highest adjusted R-squared. These sets are included in the rightmost columns of the table.

The pools of regressors Ω_j and Θ_i are overlapping. Note that the sample CCF takes it most

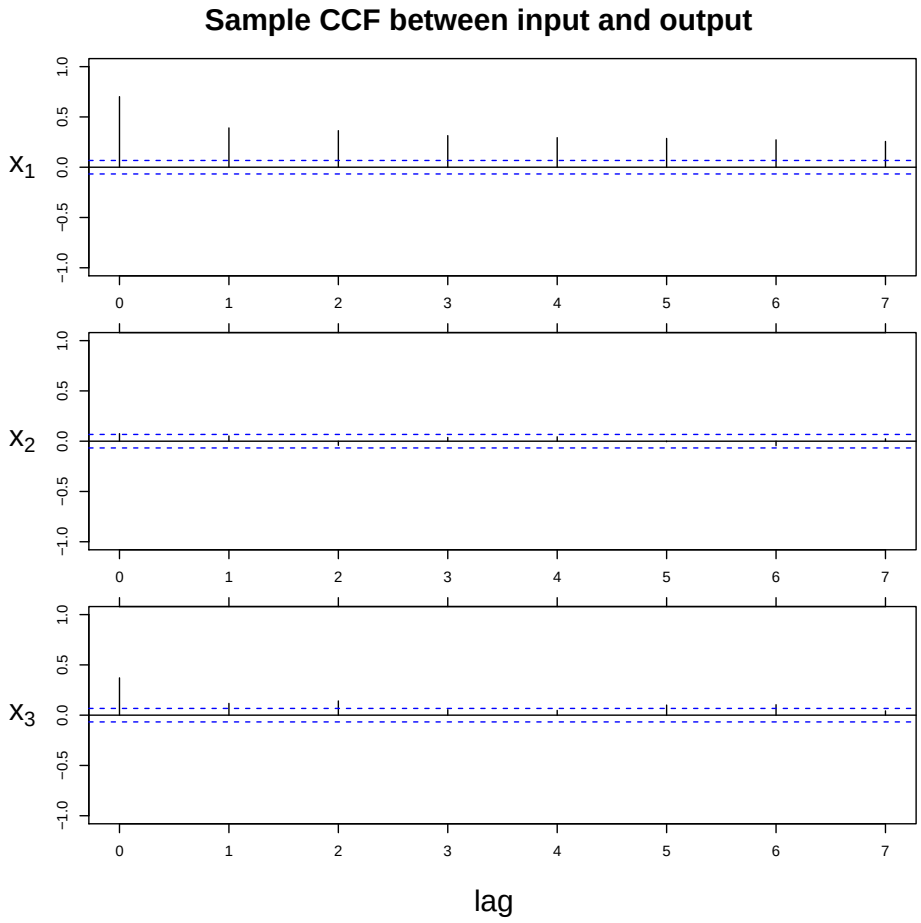

Figure 5.2: The sample CCF computed pairwise between an output variable and three input variables.

extreme value at lag zero for each input variable. The number of regressors in the final model of CSE is nine, while it is five for MCE. The coefficients on the regressors are estimated by ordinary least squares. We can write the final model from CSE as

$$y_t = -5.4\times10^{-5} + 0.60x_{1t} + 0.22x_{2t} + 0.23x_{3t} - 0.056x_{3,t-1} + 0.043x_{3,t-2} \\ - 0.078x_{3,t-3} - 0.031x_{3,t-4} + 0.037x_{3,t-5} + 0.033x_{3,t-6} + \hat{a}_t, \quad t = 7,8,\ldots,852,$$

where $\hat{a}_t$ are the regression residuals. The R-squared of this model is 0.52. Similarily, we may write the MCE model as

$$y_t = 1.1\times10^{-3} + 0.60x_{1t} + 0.22x_{2t} - 0.046x_{2,t-2} + 0.22x_{3t} - 0.058x_{3,t-1} + \hat{b}_t, \quad t = 3,4,\ldots,852,$$

Table 5.2: Comparison of the regressors sets of CSE and MCE respectively. The index j represent the input x_j. The second and third column shows the pools of regressors for the two methods. The last two columns displays the best subset of lags for each method.

j	Ω_j	Θ_i	$\hat{I}_{\text{CSE}}$	$\hat{I}_{\text{MCE}}$
1	$\{0,\ldots,7\}$	$\{0,1,2\}$	$\{0\}$	$\{0\}$
2	$\{0\}$	$\{0,1,2\}$	$\{0\}$	$\{0,2\}$
3	$\{0,\ldots,6\}$	$\{0,1,2\}$	$\{0,\ldots,6\}$	$\{0,1\}$

where $\hat{b}_t$ are the residuals. The R-squared of this model is 0.51. Note that the intercept in both models is small. Some rergessors are common for the two models. The estimated coefficients on these regressors are similar. The residuals of the two models are compared in figure 5.3. The left column displays the results from CSE and the right column shows the same for MCE. The residual plots exhibits no clear trends which would violate the regression assumptions. The sample ACF and sample PACF of the residuals for both models have several borderline significant entries, but no large ones. If the correlations were larger we should have considered AR expansion as defined in section V of chapter 4. Overall, both sets of residuals are promising. For this example there are no big difference between CSE and MCE. Still, the model from MCE is probably prefered as it is simpler.

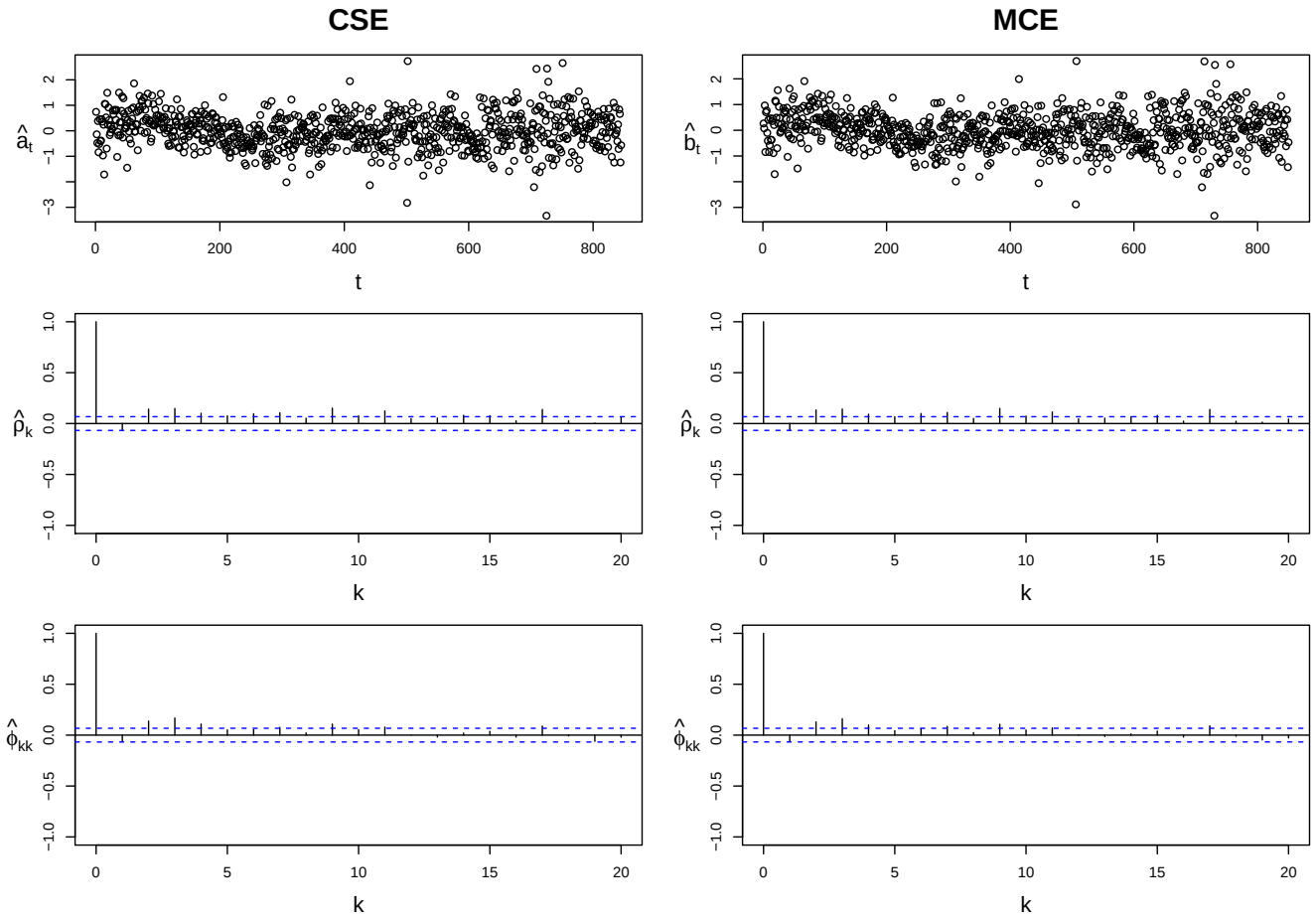

Figure 5.3: Residuals from a model fitted to observed data. The residuals of CSE is compared to those of MCE. The top row displays the residuals by day. The second row shows the sample ACF and the third row includes plots of the sample PACF of the residuals.

Chapter 6

Discussion and Conclusion

The methods presented in chapter 4 is a step towards an accurate model for the process of producing ferromanganese in electrical furnaces. However, the utility of these methods is not restricted to metall processes. In fact, we may apply them to many situations where we have observed input variables and a lagged output variable. Certain assumptions are made. The variables are equally spaced discrete time sequences. The output is a linear combination of simultaneous or lagged input. Furthermore, we allow for noise in the observations. The furnace data is measured daily, but this is not a general requirement. The interval between observations might as well be seconds or years.

Two main models are presented, Cross-correlation Selection Estimation (CSE) and Maximum Correlation Estimation (MCE). They are identcal with exception of the initial variable selection. Which model that is preferred depends partially on the dynamics of the input variables. We have seen examples of how the autocorrelation in input affects the cross-correlation between input and ouput. Specifically, when the input is heavily correlated with itself, the output will also be correlated with the input on multiple lags. Thus, the cross-correlation does not reveal the direct causal relationships. Still, MCE appears to do well even with a large autocorrelation in input on multiple lags. Autocorrelation is a bigger problem for CSE. However, in some other cases, CSE is the preferred model. For example when the output depend on the input on every other lag in some interval.

To perform diagnostics of the methods is essential. Tests include assessing the regression as-

sumtions. If these are violated, the estimates are of little worth. Autocorrelation in the residuals is a possible violation. However, section V in chapter 4 presents a solution to this problem, through a method called AR expansion. The idea is that the structure in the residuals of the current model will suggest a new model. AR expansion is a recommended extension to CSE and MSE.

The main result of chapter 5 is presented in table 5.1. The performance of CSE is explained as a function of three parameters, the dimensions n and p of the data set, and the variance σ^2 of the noise in the output. As we would expect, the performance is poor when n is small, p is large or σ^2 is large. The performance of CSE is especially poor when p is close to n. When applying CSE to an industry process, n and p is known, but not σ^2. Still, we can make a guess of σ^2 based on the complexity of the process. In a specific example, the values of these three parameters can be used to estimate the uncertainty of the model from CSE.

A viable extension of this work is a more compound algorithm, possibly based on machine learning. Multiple methods have been presented in this thesis and they have the potential to work well together. Say we have n observations of one output variable and p input variables. Assume the noise variance σ^2 is known. A potential algorithm can be outlined as following. First, we compute the sample ACF and sample PACF of each input sequence. Then we choose the optimal regressor selection method based on the parameters n, p and σ^2 as well as the sample correlation functions. The regressor selection method can be CSE, MSE or other already existing methods. Next, we estimate the coeffients by OLS. Based on the residuals we can do AR expansion once or repeatedly. The optimal choices in this algorithm can be trained by machine learning. There are many challenges in building this algorithm, but it is definitely achievable.

In the end of the previous chapter CSE and MSE were applied to a ferromanganese furnace. First, missing values were imputed by the forecasting approach presented in chapter 3. The fitted models for CSE and MSE are similar, but the MSE model is simpler. The R-squared values are just above one half, which is decent for an industry process. The regression residuals are close to being i.i.d normal as we have assumed. In conclusion, both CSE and MSE can be feasible in ferromanganese production and similar processes.

Bibliography

Barker, I. (2011). Some considerations on future developments in ferroalloy furnaces. *Journal of the Southern African Institute of Mining and Metallurgy*, 111:691 – 696.

Bisgaard, S. and Kulahci, M. (2011). *Time series analysis and forecasting by example*. John Wiley & Sons.

Box, G. and Jenkins, G. (1976). *Time Series Analysis: Forecasting and Control*. Holden-Day, San Francisco, 2nd edition.

Breusch, T. S. and Pagan, A. R. (1979). A simple test for heteroscedasticity and random coefficient variation. *Econometrica: Journal of the Econometric Society*, pages 1287–1294.

Brockwell, P. J. and Davis, R. A. (2013). *Time series: theory and methods*. Springer Science & Business Media.

Devlin, Susan J., R. G. and Kettenring, J. R. (1975). Robust Estimation and Outlier Detection with Correlation Coefficients. *Biometrika*, 62:531 – 545.

Diebold, F. X. (1998). *Elements of forecasting*. Citeseer.

Durbin, J. (1960). The fitting of time-series models. *Revue de l'Institut International de Statistique*, pages 233–244.

Eksteen, J., Frank, S., and Reuter, M. (2004). Towards predictive control of ferroalloy furnaces: combining thermochemistry, inventory modelling and systems engineering. In *Proceedings International Ferroalloy Congress*, pages 1–4.

Hayashi, F. (2000). *Econometrics*. Princeton University Press.

Hyndman, R. and Khandakar, Y. (2007). Automatic time series forecasting: The forecast package for r 7. 2008. *URL: https://www. jstatsoft. org/article/view/v027i03 [accessed 2016-02-24][WebCite Cache]*.

Kendall, M. G., Stuart, A., and Ord, J. (1968). *The advanced theory of statistics*, volume 3. London.

Rottmann, K. (2011). *Matematisk Formelsamling*. Spektrum Forlag, 11th edition.

Rubin, D. B. (1974). Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology*, 66(5):688.

Schwarz, G. et al. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.

Theil, H. (1961). Economic forecasts and policy.

Venables, W. N. and Ripley, B. D. (2002). Random and mixed effects. In *Modern applied statistics with S*, pages 271–300. Springer.

Wei, W. W.-S. (1994). *Time series analysis*. Addison-Wesley publ Reading.

Wold, H. (1939). A study in the analysis of stationary time series.

Chapter 7

R code

```
########### Script information ###########
# Author: Andreas Strand
# R version 3.4.0 (2017-04-21)
# Platform: x86_64, linux-gnu
#
# Latest edit: June 19, 2017 by Andreas Strand
#####

########### Minor functions ###########
## Expand index vector by neighbouring elements
makeSeg= function(miss,tmax){
  vec = unique(c(miss-1,miss,miss+1))
  vec = sort(vec[vec<=tmax])
  len = length(vec)
  tf = sapply(X = 1:(len-1), function(X) vec[X]+1 == vec[X+1])
  breaks = which(!tf)
  dbreaks = sort(c(breaks, breaks+1))
  return(matrix(c(vec[1],vec[dbreaks] , vec[len]), ncol = 2, byrow=T))
}

## Computes aicc or bic
ic = function(model, type = "aicc"){
  npar = length(model$coef) + 1
```

```
24    nstar = length(model$residuals) - model$arma[6] - model$arma[7] * model$arma[5]
25    if(type == "aicc"){out = model$aic + 2 * npar * (nstar/(nstar - npar - 1) - 1)
26    } else if(type == "bic"){out = model$aic + npar * (log(nstar) - 2)
27    } else{out = NA}
28    return(out)
29  }
30
31  ## GG colors
32  ggCol <- function(n) {
33    hues = seq(15, 375, length = n + 1)
34    hcl(h = hues, l = 65, c = 100)[1:n]
35  }
36
37  ## Add interval containing significant entries
38  interval = function(vec,limit){
39    signif = which(vec>limit)
40    if(length(signif)){out = c(min(signif),max(signif))
41    } else{out = c(NA,NA)}
42    return(out)
43  }
44
45  ## Make list with lead zero from vector/list
46  addZero = function(var){
47    if(!is.list(var)) var = list(var)
48    return(append(var,0,0))
49  }
50
51  ## Make sequence from start-and-stop-vector
52  Seq = function(vec){
53    return(vec[1]:vec[2])
54  }
55
56  ## Append list
57  lappend = function(lst, index, value = rep(NA,length(index))){
58    if(!is.list(lst)) lst = list(lst)
59    for(i in 1:length(index)){lst = append(lst, value[i], index[i]-1)}
```

```
60   return(lst)
61 }
62
63 ## Distances to next entry of a sorted vector
64 maxOrder = function(lst, order){
65   unlist(sapply(lst, function(x){
66     if(length(x)==1){
67       out = order
68     } else{
69       t = 2:length(x)
70       out = c(pmin(x[t]-x[t-1]-1,order), order)
71     }
72     return(out)
73   }))
74 }
75
76 ########### Missing values ###########
77 ## Change extreme values to NA
78 clean = function(data, bounds){
79   xtreme = apply(as.matrix(1:ncol(data)),1, function(x){
80     data[x] < bounds[x,1] | data[x] > bounds[x,2]
81   })
82   data[xtreme] = NA
83   return(data)
84 }
85
86 ## Partial autocorrelation using D-L algorithm
87 durbin = function(gamma, forecast = FALSE, error = FALSE){
88   pp = gamma[2]/gamma[1]
89   p = pp
90   v = gamma[1]
91   for(i in c(2:(length(gamma)-1))){
92     new = (gamma[i+1]-p%*%gamma[i:2])/(gamma[1]-p%*%gamma[2:i])
93     p = c(p-new*rev(p),new)
94     pp = c(pp, new)
95     v = v*(1-new^2)
```

```
96    }
97    if(forecast){
98      return(if(error) c(v,p) else p)
99    } else{
100     return(pp)
101   }
102 }
103
104 ## Estimating variance of random deviations of a time series
105 sigma2 = function(ts){
106   A = which(!is.na(ts))
107   A = A[which(A>2)]
108   tsF = fillTs(ts, se =F, normalize = F)
109   se = rep(0,length(A))
110   for(i in 1:length(A)){
111     g = acf(tsF, lag.max = A[i]-1, type = "covariance", na.action = na.pass, plot = F)$
          acf
112     phi = durbin(gamma = g,forecast = T)
113     se[i] = (ts[A[i]]-sum(phi*tsF[(A[i]-1):1]))^2
114   }
115   return(mean(se))
116 }
117
118 ## Impute missing values in a time series by forecasting
119 fillTs = function(ts, se = T, plot = F, line = 5, plot.col = "red", name = "",
120                   normalize = F, hyp = NULL, j = NULL, compact = T){
121   ## Note: Will give error if the first value is NA or out of bounds
122   if(se) sa2 = sigma2(ts)
123   ts.old = ts
124   tmax = length(ts)
125   M = which(is.na(ts))
126   maxNotNA = max(which(table(c(M,1:tmax))==1))
127   for(m in M){
128     mlag = ifelse(m > maxNotNA, maxNotNA-1, m-1)
129     g = acf(ts, lag.max = mlag, type = "covariance", na.action = na.pass, plot = F)$acf
130     phi = durbin(gamma = g,forecast = T)
```

```
131     ts[m] = sum(phi*ts[mlag:1])+ ifelse(se, rnorm(n = 1, mean = 0, sd = sqrt(sa2)), 0)
132   }
133   if(normalize){
134    ts.old = scale(ts.old)
135    ts = scale(ts)
136   }
137   if(plot){
138     if(length(hyp)){
139       par(mar = c(0,4.5,0,1)+.1)
140       plot(ts.old, type = "l", lty=1, main = "", xlab = "t",
141            ylab = bquote(paste(.(hyp)[.(j)])), cex.lab = 1.5)
142     }else if(compact){
143       par(mar = c(3,5,0,1)+.1)
144       plot(ts.old, type = "l", lty=1, main = "", xlab = "", ylab = "")
145       if(nchar(name)==2) name = bquote(.(substr(name,1,1))[.(substr(name,2,2))])
146       mtext(bquote(.(name)), side = 2, line = 3, las = 2, cex= 1.3)
147       if(name != "y") axis(3, at = seq(0,800,200), labels = F)
148     } else{
149       plot(ts.old, type = "l", lty=1, main = "", xlab = "", ylab = "")
150     }
151     if(length(M)){
152       apply(X=makeSeg(M,tmax), 1, function(X){
153         lines(X[1]:X[2], ts[X[1]:X[2]], type = "l", lty=line, col=plot.col)})
154     }
155   }
156   return(ts)
157 }
158
159 ## Impute missing values in a data frame by forecasting
160 fillDf = function(df, se = T, plot = F, line = 5, plot.col = "red", normalize = F,
161                   hyp = NULL, format = c(min(ncol(df),4),1), compact = T){
162   if(plot){
163     if(compact) par(oma = c(3,0,3.5,0)+.1)
164     par(mfrow = format)
165     return(sapply(X=colnames(df), function(X){
166       fillTs(df[,X], se = se, plot = T, line= line, plot.col = plot.col, name =X,
```

```
167                 normalize = normalize, hyp = hyp, j = which(colnames(df)==X)-1)}))
168   } else{
169     return(sapply(X=colnames(df), function(X){
170       fillTs(df[,X], se = se, plot = F, normalize = normalize)}))
171   }
172 }
173
174 ## Impute missing values in certain columns of a data frame
175 fillCol = function(df, cols = 1:ncol(df), se = T, clean = F, bounds = NULL, plot = F,
176                    format = c(min(ncol(df),4),1), line = 5, compact = T,
177                    plot.col = "red", normalize = F, hyp =NULL, save = F){
178   if(clean) df[,cols] = clean(df[,cols], bounds)
179   center = rep(colMeans(df[,cols], na.rm = T),rep.int(nrow(df),length(cols)))
180   df[,cols] = df[,cols]-center
181   if(save) pdf("~/Elmet/Master/PDF/fig/ex.pdf", width = 11, height = 8)
182   df[,cols] = fillDf(df[,cols], se = se, plot = plot, line = line,
183                      plot.col = plot.col, normalize = normalize, hyp = hyp,
184                      format = format)
185   if(compact){
186     title(main = "Observed time series", outer = T, line = 1.5, cex.main = 2.5)
187     title(xlab = "t", outer = T, line = 1, cex.lab = 2)
188   }
189   if(save) invisible(dev.off())
190   return(df)
191 }
192
193
194 ########### Regression ###########
195 ## Lagged regression
196 lagReg = function(df, type = "cse", lags = NULL, w = NULL, drop = "none",
197                   ar = FALSE, max.ar = 4, best.sub = 1, conv.names = FALSE,
198                   neg.lags = FALSE, plot.sub = FALSE, delta = 1){
199   ## Variables
200   #          df: Data frame with output in first column
201   #        type: simple = Zero lag from input to output,
202   #                max = Lag decided by largest abs ccf,
```

```
203 #                 mce = Lag decided by a neighborhood delta of largest abs ccf,
204 #                 cse = Lags are all those between first and last significant ccf,
205 #                 big = Lags are all significant ccf,
206 #       lags: Overwrites "type" and may contain a list of custom lags for input
207 #          w: Vector of optional weighting of each column
208 #       drop: Vector of optional columns to drop
209 #         ar: Boolean of whether to expand each variable by autoregressive terms
210 #     max.ar: Max order of automatic ar-fitting
211 #   best.sub: Number of best regression subsets to display for each subset size
212 # conv.names: Use mathematical notation for variables in plot of best subsets
213 #   neg.lags: Allow for negative lags
214 #   plot.sub: Plot best subsets
215 #      delta: See type->mce
216
217 ## List of lags
218 if(!length(lags)){
219   if(type == "simple"){
220     lags = as.list(rep(0, ncol(df)))
221   } else{lags = ccLags(df, type, neg.lags = neg.lags, delta = delta)}
222 }
223
224 ## Weights
225 if(!length(w)) w = lapply(X = lags, function(X) rep(1,length(X)))
226
227 ## Drop columns
228 if(drop == "none") drop = which(is.na(lags))
229 if(length(drop)&length(drop) < (ncol(df)-1)){
230   df = df[,-drop]
231   lags = lags[-drop]
232 }
233
234 ## Make data frame
235 P = lagExpand(df, lags)
236 if(!conv.names) colnames(P) = sub("\\.0","", colnames(P))
237
238 ## Regression
```

```
239  form = as.formula(paste0(colnames(P)[1], "~",
240                           paste(colnames(P)[-1], collapse = "+")))
241  reg = lm(form, data=P)
242  out = list("ORIGINAL" = summary(reg))
243
244  ## AR expansion
245  if(ar){
246    # AR coefficients
247    bic = sapply(1:max.ar, function(X){
248      ic(arima(reg$residuals, order = c(X,0,0), include.mean = F),type = "bic")})
249    ar.order = which.min(bic)
250    ar.coeff = c(1,arima(reg$residuals, order = c(ar.order,0,0), include.mean = F)$coef)
251
252    ## Data frame
253    Q.idx = t(matrix(rep((1+ar.order):nrow(P),1+ar.order),
254                     nrow = 1+ar.order, byrow = T)-0:ar.order)
255    Q = as.data.frame(matrix(apply(P, 2, function(x){sapply(1:(ar.order+1),
256        function(y) x[Q.idx[,y]])}),nrow = nrow(Q.idx)))
257    subscript = c("",sapply(1:ar.order, function(x) paste0(".AR",x)))
258    colnames(Q) = c(sapply(colnames(P), function(x) paste0(x,subscript)))
259    P = rmDependence(Q,lags,ar.order)
260
261    ## Regression
262    form = as.formula(paste0(colnames(P)[1], "~",
263                             paste(colnames(P)[-1], collapse = "+")))
264    ar.reg = lm(form, data=P)
265    regComp(reg, ar.reg, names = c("Original", "AR expansion"))
266    out = c(out, list("AR EXPANSION" = summary(ar.reg)))
267  }
268
269  ## Best subset
270  if(best.sub){
271    if(ncol(P)<3){
272      if(ar){best.reg = summary(ar.reg)} else{best.reg = summary(reg)}
273    }
274    else if(ncol(P)<=25){
```

```
275       sub.reg = regsubsets(form, data = P, nbest = best.sub, nvmax = 20)
276       summ = summary(sub.reg)
277
278       ## Plot the optimal subsets
279       par(mfrow = c(1,1), cex.main = 1.5)
280       if(conv.names){lab = tfColnames(colnames(P))} else{lab = sub.reg$xnames}
281       if(plot.sub){
282         subPlot(sub.reg, scale= "adjr2", labels = lab,
283                 main = expression(bar(R)^2 ~ "of regression subsets"))
284         len = sapply(lags, length)
285         if(ar){copies = c(ar.order,maxOrder(lags, ar.order))+1
286         } else{copies = rep(1, sum(len))}
287         abline(v= cumsum(copies)[cumsum(len)[-length(len)]]+.5, col = ggCol(1), lwd = 3)
288       }
289       ## Best regression
290       vars = summ$which[which.max(summ$adjr2),]
291       form = as.formula(paste0(colnames(P)[1], "~", ifelse(vars[1], "", "0 +"),
292                                paste(colnames(P)[c(FALSE, vars[-1])], collapse = "+")))
293       best.reg = summary(lm(form, data = P))
294     } else{
295       best.reg = bigSub(P)
296     }
297     out = c(out, list("BEST SUBSET" = best.reg))
298   }
299
300   return(out)
301 }
302
303 ## For each column return a list of lags with high correlation to first column
304 ccLags = function(df, type = "max", ci = 0.95, max.lag = 7, neg.lags =F,
305                   plot = F, delta = 1){
306   par(mfrow=c(ncol(df)-1,1), mar = c(5.1,4.1,4.1,2.1))
307   limit = qnorm((1 + ci)/2)/sqrt(nrow(df))
308   cross = sapply(X = 2:ncol(df), function(X){
309     ccf(df[,1], df[,X], lag.max = max.lag, plot = plot,
310         main = paste0("ccf(y, x", X-1,")"), ylab = "")$acf})
```

```
311   if(type == "max"){
312     lags = apply(abs(cross), 2, which.max)-max.lag-1
313   } else if(type == "mce"){
314     mu = apply(abs(cross)[-c(1:max.lag),], 2, which.max)-1
315     lags = lapply(mu, function(X) max(X-delta, 0):(X+delta))
316   } else if(type == "cse"){
317     big.cross = apply(X = abs(cross), 2, interval, limit) - max.lag - 1
318     drop.col = which(colSums(is.na(big.cross))>0)
319     if(!length(drop.col)){
320       lags = alply(big.cross, 2, Seq)
321     } else if(length(drop.col) < ncol(big.cross)){
322       lags = lappend(apply(as.matrix(big.cross[,-drop.col]), 2, Seq),drop.col)
323     }
324   } else if(type == "big"){
325     lags = apply(X = abs(cross), 2, function(X) which(X>limit)- max.lag - 1)
326     lags[sapply(lags, function(x) length(x))==0]=NA
327   } else{
328     stop("No type called", type)
329   }
330   lags = addZero(lags)
331   if(!neg.lags) lags = lapply(lags, function(X) if(sum(X>=0, na.rm=T)){X[X>=0]
332     } else{NA})
333   return(lags)
334 }
335
336 ## Compare two regressions by their residuals
337 regComp = function(reg1, reg2, names = c("A","B")){
338   par(mar = c(4.1,4,4,1))
339   par(mfrow = c(2,2))
340   acf(reg1$residuals, main = paste("Regression:", names[1]))
341   acf(reg2$residuals, main = paste("Regression:", names[2]))
342   par(mar = c(4.1,4,1,1))
343   pacf(reg1$residuals, main = "")
344   pacf(reg2$residuals, main = "")
345 }
346
```

```
347 ## Remove linearly dependent columns from df
348 rmDependence = function(df, lst, order){
349   mo = maxOrder(lst,order)
350   vec = c(rbind(mo+1,order-mo))
351   tf = unlist(sapply(1:length(vec), function(x){
352     if(x%%2){
353       out = rep(1, vec[x])
354     } else{
355       out = rep(0, vec[x])
356     }
357     return(out)
358   }))
359   return(df[,tf==1])
360 }
361
362 ## Modified leaps::plot.regsubsets
363 subPlot = function(x, labels = obj$xnames, col = gray(seq(0, 0.9, length = 10)),
364                    main = NULL, scale = c("bic", "Cp", "adjr2", "r2")){
365   obj <- x
366   lsum <- summary(obj)
367   par(mar = c(7, 5, 6, 3) + 0.1)
368   nmodels <- length(lsum$rsq)
369   np <- obj$np
370   propscale <- FALSE
371   sscale <- pmatch(scale[1], c("bic", "Cp", "adjr2", "r2"),
372                    nomatch = 0)
373   if (sscale == 0)
374     stop(paste("Unrecognised scale=", scale))
375   if (propscale)
376     stop(paste("Proportional scaling only for probabilities"))
377   yscale <- switch(sscale, lsum$bic, lsum$cp, lsum$adjr2,
378                    lsum$rsq)
379   up <- switch(sscale, -1, -1, 1, 1)
380   index <- order(yscale * up)
381   colorscale <- switch(sscale, yscale, yscale, -log(pmax(yscale,
382                        1e-04)), -log(pmax(yscale, 1e-04)))
```

```
383   image(z = t(ifelse(lsum$which[index, ], colorscale[index],
384                      NA + max(colorscale) * 1.5)), xaxt = "n", yaxt = "n",
385         x = (1:np), y = 1:nmodels, xlab = "", ylab = "",
386         col = col)
387   laspar <- par("las")
388   on.exit(par(las = laspar))
389   par(las = 2)
390   axis(1, at = 1:np, labels = labels)
391   axis(2, at = 1:nmodels, labels = signif(yscale[index], 2))
392   if (!is.null(main))
393     title(main = main)
394   box()
395   invisible(NULL)
396 }
397
398 ## Choosing a good subset when there are many variables
399 bigSub = function(df){
400   prev = -1
401   current = 0
402   vars = colnames(df)[-1]
403   form = as.formula(paste0(colnames(df)[1], "~", paste(vars,collapse = "+")))
404   reg = summary(lm(form, data=df))
405   if(reg$df[2]==0) vars = reduce(vars, reg$df[3]-nrow(df)+1)
406
407   tol = 0.001
408   while(current-prev > -tol |length(vars)>=0.75*(ncol(df)-1)&length(vars)>0){
409     prev = current
410     prev.reg = reg
411     form = as.formula(paste(colnames(df)[1], "~", paste(vars,collapse = "+")))
412     reg = summary(lm(form, data=df))
413
414     current = reg$adj.r.squared
415     low.p = which.max(reg$coefficients[-1,4])
416     vars = vars[-low.p]
417   }
418   return(prev.reg)
```

```
419 }
420
421 ## Remove the most lagged variables using names
422 reduce = function(vars, amount =1){
423   cat("Removing", amount, "variables.\n")
424   str = sub("m", "",regmatches(vars, gregexpr("m.+$", vars)))
425   str[nchar(str)>2] = "0"
426  m = as.numeric(str)
427   return(vars[-order(m,decreasing = T)[1:amount]])
428 }
429
430 ## Transform colnames from code to formula
431 tfColnames = function(colnam){
432   ## Break apart string
433   n = length(colnam)
434   core = regmatches(colnam, regexpr("^(y)|([x]\\d+)", colnam))
435   shift = as.numeric(chartr(".pm", "  -",
436           regmatches(colnam, regexpr("\\..?\\d+", colnam))))
437   ap = regexpr("AR\\d+", colnam)[1:n]+2
438   an = rep(0, n)
439   an[which(ap!=1)] = as.numeric(sapply(which(ap!=1), function(x){
440     substr(colnam[x],ap[x],ap[x])}))
441   ss = shift-an
442   ss = sub("^0", "", ss)
443   pos = grep("^\\d", ss)
444   ss[pos] = paste0("+",ss[pos])
445   ss = paste0("t", ss)
446
447   ## Put string together in new format
448   out = sapply(1:n, function(x){
449     s = ifelse(grepl("\\d", core[x]), ",", "")
450     name = bquote(.(substring(core[x],1,1))[
451       .(paste(substring(core[x],2),ss[x], sep = s))])
452     if(an[x]){
453       name = bquote(.(name)%.%phi1[.(an[x])])
454     }
```

```
455     return(as.expression(name))
456   })
457   out[1] = "(Intercept)"
458   return(out)
459 }
460
461
462 ########## Simulation ##########
463 ## Make data frame with lagged variables
464 lagExpand = function(df, lags){
465   ## Index matrix
466   y.start = 1 + max(sapply(lags,max),0)
467   y.end = nrow(df) + min(sapply(lags,min),0)
468   if(sum(is.na(c(y.start,y.end)))) print(lags)
469
470   P.idx = lapply(X = lags, function(X){
471     t(matrix(rep(y.start:y.end,length(X)), nrow = length(X), byrow = T)-X)})
472
473   # ERROR TEST
474   if(length(P.idx) != ncol(df)){
475     print(colnames(df))
476     print(lags)
477     lapply(P.idx, function(X) print(head(X)))
478   }
479
480   ## Data frame from indices
481   P = sapply(X = 1:ncol(df), function(X){
482     sapply(1:ncol(P.idx[[X]]), function(y) df[P.idx[[X]][,y],X])})
483   P = as.data.frame(matrix(unlist(P), ncol = length(unlist(lags))))
484   add = sapply(lags,function(x){
485     ifelse(x==0, "", paste0(ifelse(x<0, ".p",".m"), abs(x)))}, simplify = FALSE)
486   colnames(P) = unlist(sapply(1:length(lags), function(x){
487     paste0(colnames(df)[x], add[[x]])}))
488   return(P)
489 }
490
```

```
## Construct a random df
makeX = function(n, k=length(type), type = sample(1:3,k,replace = T), print = TRUE){
  ## Produce data
  out = lapply(1:k, function(X){
    scale = 1 #runif(1, 1, 1000)
    if(type[X] == 1){
      SD = runif(1, .1, 1.5)
      MU = runif(1, -1, 1)
      v = rnorm(n, sd = SD, mean = MU)*scale
      v.str = sprintf("%-9s --- mean = %*.2f, sd = %*.2f",
                      "Normal", 7, mean(v), 7, sd(v))
    } else if(type[X] == 2){
      v = (runif(n, -1, 1))*scale
      v.str = sprintf("%-9s --- mean = %*.2f, sd = %*.2f",
                      "Uniform", 7, mean(v), 7, sd(v))
    } else{
      AR = runif(sample(0:1,1),-1,1)
      MA = runif(ifelse(length(AR),sample(0:1,1),1),-1,1)
      v = as.numeric(arima.sim(list(ar = AR, ma =MA), sd = scale, n = n))
      v.str = paste0("ARMA(", ifelse(length(AR), 1, 0), ",",
                     ifelse(length(MA), 1, 0), ") --- ",
                     ifelse(length(AR), sprintf("  AR = %*.2f, ", 7, AR), "   "),
                     ifelse(length(MA), sprintf("MA = %*.2f, ", 7, MA), ""),
                     sprintf("sd = %*.2f", 7, scale))
    }
    names(v) = v.str
    return(v)
  })
  ## Format output
  nam = paste0("x",1:k)
  if(print) cat(paste(nam, sapply(out, function(X) names(X)[1]),sep=" ~ "),sep="\n")
  out = as.data.frame(out)
  colnames(out) = nam
  return(out)
}

```

```
## Simulate df with response
makeDF = function(n = 100, k = 1, max.lag = 7, df = scale(makeX(n, k, print = FALSE)),
                  lags = NULL, beta = NULL, max.expand = 3, sd.noise = 1){
  ## Produce input
  if(!length(lags)){
    lags = replicate(ncol(df), seq(from = sample(0:max.lag,1),
               length.out=sample(1:max.expand,1)), simplify = FALSE)
  }
  if(!is.list(lags)) lags = list(lags)
  if(!length(colnames(df))) colnames(df) = paste0("x", 1:ncol(df))
  P = lagExpand(df,lags)

  ## Compute output
  if(!length(beta)) beta = runif(ncol(P), 1,3)*sample(c(-1,1),1)
  names(beta) = colnames(P)
  y = scale(as.matrix(P)%*%beta + rnorm(nrow(P), 0, sd.noise))
  sdy = sd(y)
  y = scale(y)
  y.start = 1 + max(sapply(lags,max),0)
  y.end = nrow(df) + min(sapply(lags,min),0)
  out = as.data.frame(cbind(y,df[y.start:y.end,]))
  colnames(out) = c("y",paste0("x", 1:ncol(df)))
  return(list(data = out, coeff = beta/sdy))
}

## Test performance of lagged regression
test = function(n, k, sd.noise = 1, print = FALSE){
  a = makeDF(n, k, sd.noise = sd.noise)
  b = lagReg(a$data, type = "cse", best.sub = 1)$'BEST SUBSET'

  real = a$coeff
  est = b$coeff[-1,"Estimate"]

  nam = sort(unique(c(names(est),names(real))))
  S = cbind(real[nam], est[nam])
  S[is.na(S)] = 0
```

```
563   rownames(S) = nam
564   S = cbind(S,(S[,1]-S[,2]))
565   colnames(S) = c("True", "Estimate", "Rel. error")
566
567   out = mean(S[,3]^2)
568   if(out>1){
569     print("Bad performance:")
570     cat("n =", n, "\nk =", k, "\nst.dev noise =", sd.noise, "\n\n")
571   }
572
573   if(print){
574   print(S, digits = 4, na.print = "", print.gap = 3)
575   cat(paste("\nAdj. R squared:",b$adj.r.squared, "\n"))
576   }
577   return(out)
578 }
579
580 ## Multiple tests
581 performance = function(reps = 100){
582   ## Variables
583   #   reps: Number of repetitions for each setting
584
585   n = c(500, 1000, 5000)
586   k = c(2, 5, 10, 30)
587   s = c(.1, .5, 1, 1.5, 2)
588   lvl = expand.grid(n,k,s)
589   y = apply(X = lvl, 1, function(X){
590     print(unname(X))
591     return(mean(replicate(reps, test(X[1],X[2],sqrt(X[3])))))})
592   R = cbind(y,lvl)
593   colnames(R) = c("y", "n", "k", "s")
594   fit = lm(y ~ (n + k + s)^2  + I(n^(.5)) + I(k^2) + I(s^2), data = R)
595   #+ I(n^(.5))
596   return(list(R, summary(fit)))
597 }
598
```

```
599
600 ########## AR ##########
601 ## Sampling stationary coeffs of an AR(2)
602 complexRoots = function(n = 1){
603   out = replicate(n, expr = {
604     repeat{
605       x = runif(1, -2, 2)
606       y = runif(1,-1,0)
607       if(y < -x^2/4){
608         break
609       }
610     }
611     return(c(x,y))
612   })
613   return(out)
614 }
615
616 ## Add AR(2)-triangle
617 addAR = function(line = 2.5, cex = 1.3){
618   title("")
619   mtext(expression(phi[1]), side = 1, line = line+.3, las = 1, cex = cex)
620   mtext(expression(phi[2]), side = 2, line = line, las = 2, cex = cex)
621   segments(x0 = -2, y0 = -1, x1 = 0, y1 = 1)
622   segments(x0 = 0, y0 = 1, x1 = 2, y1 = -1)
623   segments(x0 = -2, y0 = -1, x1 = 2)
624   curve(-x^2/4, -2, 2, add = T)
625 }
626
627 ## Plotting coeffs of an AR(2)
628 plotCoeff = function(coeff, pch = 1, bg = F, cex = 1.2){
629   plot(NA, ylab = "", xlab = "", ylim = c(-1,1), xlim = c(-2,2))
630   addAR()
631   x = coeff[1,]
632   y = coeff[2,]
633   if(bg){
634     ex = 0.2
```

```
635     lsw = strwidth(pch); w <- lsw/2*(1+ex)*cex
636     lsh = strheight(pch); h <- lsh/2*(1+ex)*cex
637     rect(x-w, y-h, x+w, y+h, col='white', border=NA)
638     text(x, y, pch, cex = cex)
639   } else{
640     points(x = x, y = y , pch = pch, cex = cex)
641   }
642 }
643
644 ## Compare ACFs of AR(2) processes
645 compARe = function(size = 5, bg = T, coeff = NULL, cpdim = c(2,2), cex = 1.5,
646                    save.size = NULL, filename = "~/Elmet/Master/PDF/fig/acf.pdf"){
647   ## Draw random coefficients if not supplied
648   if(length(coeff)){
649     size = ncol(coeff)
650   } else{
651     coeff = complexRoots(size)
652   }
653
654   ## Plot layout
655   if(length(save.size)) pdf(filename, width = save.size[1], height = save.size[2])
656   if(size > 25) stop("Size cannot exceed 25.")
657   if(size > 5){
658     dim =  nbym(size+prod(cpdim), square = F)
659     cdiff = dim[2]-cpdim[2]
660     rdiff = dim[1]-cpdim[1]
661     L = matrix(1,dim[1], dim[2])
662     if(cdiff>0) L[1:cpdim[1], -(1:cpdim[2])] = seq(2, length.out = cpdim[1]*cdiff)
663     if(rdiff>0){L[(cpdim[1]+1):dim[1],] =
664       matrix(seq(cpdim[1]*cdiff+2, length.out = rdiff*dim[2]), byrow=T, nrow=rdiff)
665     }
666     layout(L)
667   } else{par(mfrow = nbym(size+1, square = F))}
668
669   nam = LETTERS[1:size]
670   colnames(coeff) = nam
```

```
671   par(mar = c(5,4,2,1),oma=c(0,0,3,0))
672
673   ## Plot
674   plotCoeff(coeff, pch = nam, bg = bg, cex = cex)
675   par(mar = c(3.5,4,2,1))
676   l_ply(colnames(coeff), function(X){
677     plot(0:20, ARMAacf(ar = coeff[,X], lag.max = 20), type = "h", ylim = c(-1,1),
678          ylab = "", xlab = "")
679     mtext(sprintf("T = %.2f\nr = %.2f",period(coeff[,X]), sqrt(-coeff[2,X])), 3,
680           adj=0.99, line=-1, cex = 0.7, bg = "white")
681     title(ylab = "ACF", line = 2.2, adj = 0.5)
682     title(xlab = "lag", line = 2.2, adj = 0.5)
683     title(X, line = 0.5, adj = 0.5, cex.main = 1.5)
684     abline(h=0)
685   })
686   title("Autocorrelation of AR(2) models", cex.main = 2, outer=TRUE)
687   if(length(save.size)) invisible(dev.off())
688 }
689
690 ## Returning optimal grid dimensions for number of entries
691 nbym = function(size, square = TRUE){
692   dim = rep(ceiling(sqrt(size)),2)
693   if(!square && dim[1]*(dim[1]-1)>=size) dim = c(dim[1], dim[1]-1)
694   return(dim)
695 }
696
697 ## Period of the acf of an AR(2)
698 period = function(phi){
699   ifelse(phi[2] >= -phi[1]^2/4 | phi[2] <= -1,
700          -1, 2*pi/acos(phi[1]/(2*sqrt(-phi[2]))))
701 }
702
703 ## Damping factor of the afc of an AR(2)
704 amp = function(phi){
705   ifelse(phi[2] >= -phi[1]^2/4 | phi[2] <= -1,
706          -1, sqrt(-phi[2]))
```

```
}

## Plotting the period and damping factor of an AR(2)
plotAR = function(resol = 1000, ncolor = 64, maxT = 30, maxR = 1.05,
                  intensity = 0.7, save.size = NULL,
                  filename = "~/Elmet/Master/PDF/fig/tr.pdf"){
  resol = resol + 1
  xvals = seq(-2,2, length.out = resol)
  yvals = seq(-1,0, length.out = resol)
  per = matrix(apply(expand.grid(xvals, yvals), 1, period), ncol = resol)
  amp = matrix(apply(expand.grid(xvals, yvals), 1, amp), ncol = resol)

  if(length(save.size)) pdf(filename, width = save.size[1], height = save.size[2])
  par(mfrow = c(2,1), mar = c(4.1, 4.1, 3.1, 4.1), oma = c(0,0,0,0), cex.main = 1.3)
  image.plot(x = xvals, y = yvals, z=per, zlim = c(0,maxT),
             col = heat.colors(ncolor, alpha = intensity), ylab = "", xlab = "",
             ylim = c(-1,1), xlim = c(-2,2), main = "Period (T) of the ACF")
  addAR()

  image.plot(x = xvals, y = yvals, z=amp, zlim = c(0,maxR),
             col = heat.colors(ncolor, alpha = intensity),ylab = "", xlab = "",
             ylim = c(-1,1), xlim = c(-2,2), main = "Decay constant (r) of the ACF")
  addAR()
  if(length(save.size)) invisible(dev.off())
}

## Plot the ccf of a complex AR(2) and a lagged response
ccfAR = function(phi, subset = NULL, lags = c(3,4,5), beta = c(1,1,1), n = 1000,
                 sd = 1, lag.max = 20, xlim = c(0,lag.max), ylim = c(-1,1),
                 main = "ccf(x,y)", ylab = "", xlab = "lag", cex.lab = 1.5,
                 print = F){
  if(length(subset)) phi = phi[,subset]
  df = makeDF(df = matrix(arima.sim(list(ar = phi), n = n, sd = sd)), lags = lags,
              beta =  beta)$data
  cc = ccf(df[,1],df[,2], lag.max = lag.max, ylab = ylab, xlab = xlab, xlim = xlim,
           cex.lab = cex.lab, main = ifelse(length(subset),subset, main),
```

```
743             ylim = ylim)$acf[-(1:lag.max)]
744   if(print){
745     print("Significant lags:")
746     print(interval(abs(cc), qnorm((1 + 0.95)/2)/sqrt(n))-1)
747     cat("Max abs lag: ",which.max(abs(cc))-1, "\n")
748   }
749 }
750 
751 ## Plot the effect of autocorrelation on crosscorrelation
752 ccfChange = function(phi, subset = NULL, lags = c(3,4,5), beta = c(1,1,1),
753                      n = 1000, xlim = c(0,20), ylim = c(-1,1), idx = 1,
754                      start = min(1,idx), end = max(0,idx), cex.lab = 1.5,
755                      main = colnames(phi)[subset], print = F){
756   par(mar = c(3.1, 3.1, 3.1, 0))
757   plot(lags, beta, type = "h", ylab = "", xlab = "",
758        xlim = xlim, ylim = ylim, main = main, cex.lab = cex.lab)
759   abline(h = 0)
760   if(subset != end) axis(4, at = seq(-1,1,.5), labels = F)
761   if(length(subset)) phi = phi[,subset]
762   par(mar = c(3.1, 3.1, 0, 0))
763   plot(Seq(xlim), ARMAacf(ar = phi, lag.max = 20), type = "h", ylim = c(-1,1),
764        ylab = "", xlab = "", main = "", cex.lab = cex.lab)
765   lim = qnorm((1 + 0.95)/2)/sqrt(n)
766   abline(h = c(-lim, 0, lim), col = c("blue", "black", "blue"), lty = c(2,1,2))
767   axis(3, at = seq(0,20,5), labels = F)
768   if(subset != end) axis(4, at = seq(-1,1,.5), labels = F)
769   par(mar = c(3.1, 3.1, 0, 0))
770   ccfAR(phi = phi, main = "", ylab = "", xlab = "", print = print)
771   axis(3, at = seq(0,20,5), labels = F)
772   if(subset != end) axis(4, at = seq(-1,1,.5), labels = F)
773 }
774 
775 ## Multiple plots of the effect of autocorrelation on crosscorrelation
776 cchange = function(phi, filename = "~/Elmet/Master/PDF/fig/cchange.pdf",
777                    save.size = NULL, idx = 1:ncol(phi), print = F){
778   if(length(save.size)) pdf(filename, width = save.size[1], height = save.size[2])
```

```
779   mat = matrix(1:(ncol(phi)*3), nrow = 3, ncol = ncol(phi))
780   layout(mat, c(rep(1,6)), c(1.2,1,1))
781   par(oma = c(2,2,3,1))
782   cc = sapply(idx, function(x) ccfChange(phi, subset = x, idx = idx, print = print))
783   par(cex.main = 2, cex.lab = 1.7)
784   title(main = "Effect of autocorrelation on crosscorrelation", outer = T, line = 1)
785   label = c("ccf(x,y)","acf(x)","Coefficients")
786   adjv = c(.15, .545, .92)*(par("omd")[4]-par("omd")[3]) + par("omd")[3]
787   adjh = .51*(par("omd")[2]-par("omd")[1]) + par("omd")[1]
788   sapply(1:length(adjv), function(x) title(ylab = label[x], adj = adjv[x], outer = T,
          line = 0))
789   title(xlab = "lag", adj = adjh, outer = T, line = 0)
790   if(length(save.size)) invisible(dev.off())
791 }
792
793 ## Plot a series along with ACF and PACF
794 tsPlot = function(z, main = expression(z[t]), kmax = 20, res = 0, cols = 1,
795                   add = F, sub = substitute(hat(I))){
796   if(!add) par(mfcol = c(3,cols))
797   par(cex.lab = 1.5, cex.main = 2, mar = c(4,4.5,4,1)+.1)
798   plot(z, ylab = "", xlab = "t", main = main)
799   par(mar = c(4,4.5,1,1)+.1)
800   if(res==1){mtext(bquote(hat(epsilon)[scriptscriptstyle(list(.(sub), t))]),
801         side = 2, line = 2.4, las = 2)
802   }else if(res == 2){mtext(bquote(hat(epsilon)[list(.(sub), t)]),
803         side = 2, line = 2.4, las = 2)
804   }else if(res == 3){mtext(expression(hat(a)[t]), side = 2, line = 2.4, las = 2)
805   }else if(res == 4){mtext(expression(hat(b)[t]), side = 2, line = 2.4, las = 2)
806   }else{mtext(expression(z[t]), side = 2, line = 2.4, las = 2)}
807   acf(z, ylim = c(-1,1), ylab = "",  xlab = "k", main = "", lag.max = kmax)
808   mtext(expression(hat(rho)[k]), side = 2, line = 2.4, las = 2)
809   y = pacf(z, ylim = c(-1,1), ylab = "", main = "", plot = F, lag.max = kmax)$acf
810   plot(0:kmax, c(1,y), type = "h", ylim = c(-1,1),  ylab = "", xlab = "k")
811   lim = qnorm((1 + 0.95)/2)/sqrt(length(z))
812   abline(h = c(-lim, 0, lim), col = c("blue", "black", "blue"), lty = c(2,1,2))
813   mtext(expression(hat(phi)[kk]), side = 2, line = 2.4, las = 2)
```

```
814 }
815
816 ## Plot the sample CCF between input and output
817 cc = function(df){
818   par(mfrow = c(3,1), mar = c(3,5,0,1)+.1, oma = c(3,0,3.5,0)+.1)
819   sapply(1:3, function(X){
820     ccf(df[,1], df[,X+1], lag.max = 7, xlab = "",
821         ylab = "",main = bquote(paste(r[paste(x[.(X)],y, sep="")],(k))),
822         xlim = c(0,7), ylim = c(-1,1), cex.lab = 1.5)
823     if(X != 1) axis(3, at = seq(0,7), labels = F)
824     mtext(bquote(x[.(X)]), side = 2, line = 3, las = 2, cex= 1.3)
825   })
826   title(main = "Sample CCF between input and output",
827         outer = T, line = 1.5, cex.main = 2, adj = 0.53)
828   title(xlab = "lag", outer = T, line = 1, cex.lab = 2, adj = 0.53)
829 }
830
831 ########### Workspace ###########
832 library(readxl)    # Loading data
833 library(forecast) # ARIMA models
834 library(leaps)     # Subset regression
835 library(car)       # Subset regression
836 library(plyr)      # More apply-functions
837 library(fields)    # Plot bivariate functions
838
839 rm(list = ls())
840 setwd("~/Elmet/Master/Data")
841 load("~/Elmet/Master/Data/master28.RData")
842
843 ########### Model ###########
844 # M is a data frame with output in the first column
845
846 ## Lagged regression
847 lagReg(M)
848 lagReg(M, type = "big", ar = T, best.sub = F)
849 lagReg(M, type = "mce", ar = T, best.sub = F)
```

```
lagReg(M, type = "cse", ar = T, best.sub = F)


########## Performance ##########
set.seed(100)
test(100, 5, sd.noise = 1, print = T)

a = performance(reps = 1000) # 18h
fm = lm( y ~ (n + k + s)^2 + I(n^2) + I(k^2) + I(s^2)+
            I(n^0.5) + I(k^0.5) + I(s^0.5), data= a[[1]])
summary(step(fm, direction = "backward", trace=FALSE ))

########## ARMA ##########
## Plotting example time series
set.seed(100)
z1 = arima.sim(model = list(ar = c(.8, -.8)), n = 200)
z2 = arima.sim(model = list(ma = .8), n = 200)
var(z1)
var(z2)

pdf("~/Elmet/Master/PDF/fig/arma.pdf", width = 9, height = 7)
par(mfrow = c(2,1), cex.lab = 1.5, mar = c(5,3,3,1)+.1)
plot(z1, ylab = "", xlab = "t", main = "AR(2)")
plot(z2, ylab = "", xlab = "t", main = "MA(1)")
invisible(dev.off())

## Plotting a summary of AR(1)
set.seed(200)
z3 = arima.sim(model = list(ar = c(.8)), n = 200)
z4 = arima.sim(model = list(ar = c(-.8)), n = 200)

pdf("~/Elmet/Master/PDF/fig/ar1.pdf", width = 7, height = 7)
par(mfcol = c(3,2))
tsPlot(z3, main = expression(z[t] == 0.8*z[t-1] + epsilon[t]))
tsPlot(z4,  main = expression(z[t] == -0.8*z[t-1] + epsilon[t]))
invisible(dev.off())
```

```
886
887 ########## AR(1) ##########
888 ## Effect of autocorrelation on crosscorrelation
889 phi1 = cbind(-.95, -.5, .5, .95)
890 colnames(phi1) = LETTERS[1:ncol(phi1)]
891 cchange(phi1, save.size = c(ncol(phi1)*2,7),
892         filename = "~/Elmet/Master/PDF/fig/cchange1.pdf")
893
894
895 ########## AR(2) ##########
896 ## Plot of the period and damping factor of the acf as a function of phi
897 plotAR(save.size = c(7,7))
898
899 ## Example acfs
900 phi2 = cbind(c(0,-.05), c(-.95,-.5), c(.95,-.5),
901              c(-1.90, -.95), c(0,-.95), c(1.90,-.95))
902 colnames(phi2) = LETTERS[1:ncol(phi2)]
903 compARe(coeff = phi2, cpdim = c(2,3), save.size = c(7,7))
904
905 ## Effect of autocorrelation on crosscorrelation
906 set.seed(100)
907 cchange(phi2, save.size = c(ncol(phi2)*2,7), print = T,
908         filename = "~/Elmet/Master/PDF/fig/cchange2.pdf")
909
910 ## For presentation
911 phi3 = phi2[, c(1,2,6)]
912 colnames(phi3) = letters[1:ncol(phi3)]
913 cchange(phi3, save.size = c(ncol(phi3)*2,7),
914         filename = "~/Elmet/Master/PDF/fig/cchange3.pdf")
915 cchange(phi3)
916
917 ## AR triangle
918 pdf("~/Elmet/Master/PDF/fig/triangle.pdf", width = 10, height = 7)
919 plot(NULL, xlim = c(-2.3,2.3), ylim = c(-1.3,1.3), xlab = "", ylab = "",
920      main = "Staionary regions of the AR(2) process", cex.main = 1.5)
921 addAR()
```

```
922 text(x = c(0,0,1.2), y = c(-.5,.4,.7), cex = 1.5,
923      labels = c("Complex", "Real", "Non-stationary"))
924 invisible(dev.off())
925
926
927 ########## Performance ##########
928 pdf("~/Elmet/Master/PDF/fig/sampleccf.pdf", width = 10, height = 7)
929 set.seed(117)
930 par(mfrow = c(1,2), oma = c(0,0,3,0), mar = c(4,3,3,1)+.1, cex.main = 1.5)
931 N = makeDF(k = 2, lags = list(c(3,4,5,7), c(0,1)), sd.noise = 1)$data
932 invisible(sapply(1:2, function(X){ccf(N[,1], N[,X+1], lag.max = 20, xlab = "k",
933   ylab = "",main = bquote(paste(r[paste(x[.(X)],y, sep="")],(k))),
934   xlim = c(0,20), ylim = c(-1,1), cex.lab = 1.5)}))
935 title(main = "Sample CCF", outer = T, cex.main = 1.7)
936 invisible(dev.off())
937
938 r = lagReg(N, type = "cse", best.sub = 1)
939 r2 = lagReg(N, lags = list(0,7,c(0,1)), ar = T)
940
941 pdf("~/Elmet/Master/PDF/fig/arex.pdf", width = 10, height = 7)
942 tsPlot(z = r2$ORIGINAL$residuals, main = "Original", res = 2, cols = 2,
943        sub= substitute(I*"*"))
944 tsPlot(z = r2$`AR EXPANSION`$residuals, main = "AR expanded", res = 3, add = T,
945        sub= substitute(I*"*"))
946 invisible(dev.off())
947
948 pdf("~/Elmet/Master/PDF/fig/residuals.pdf", width = 7, height = 7)
949 tsPlot(z = r$`BEST SUBSET`$residuals, main = "Residuals ", res = 1)
950 invisible(dev.off())
```