Universiteit Utrecht

UMC Utrecht
Wilhelmina Kinderziekenhuis

MSc Artificial Intelligence

Final Thesis

Machine Learning for Ventilation Decision Support

Author:
Michiel Pieter Bron
(3836649)

Supervisors (UU):
dr. A.J. Feelders
prof. dr. A.P.J.M. Siebes

Supervisors (WKZ):
drs. E. Koomen
dr. J. Nijman
dr. T.H. Kappen

September 2019

Abstract

Introduction. For the last six decades, mechanical ventilation has become an established therapy in intensive care units for patients who have respiratory problems. However, this therapy faces some challenges; finding the optimal ventilation mode and settings is not always straightforward. Recent advances in the field of Artificial Intelligence, especially deep learning, enable learning from big datasets. Deep learning models that are trained on patient data may help to establish the appropriate mode and settings.

Objectives. In this project, we aimed to solve a part of this problem; creating a model that can predict values of several vital signs of the patient, given historical values. We analyzed the PICU dataset, which contains time series of vital signs and ventilator settings of 1547 patients of the Pediatric Intensive Care Unit of the Wilhelmina's Children's Hospital. The goal of this project was two-fold. First, we performed an extensive exploratory analysis on the dataset. We used time series analysis methods such as Vector Autoregression (VAR), Panel VAR, and Multilevel Graphical VAR models to describe relations that exist between variables and time steps. Secondly, we created and trained predictive models using several neural network architectures; the Long Short-Term Memory Network (LSTM), Convolutional Neural Networks (CNN), and an architecture that combines both of them. These models directly predict, given the (historical) time series of several vital signs, (a) future value(s) of the target vital signs.

Results. In our exploratory analysis, we found several correlations that were present between variables that can be explained by clinical theory. Furthermore, we observed that the vital signs exhibit a strong autocorrelation; the current measurement largely depends on recent measurements. The predictive models learned to mimic the naïve persistence model, which uses the last available measurement as its prediction. The trained models perform relatively well during stable periods. However, they fail to detect sudden changes that are the most interesting from a clinical standpoint. The trained CNN-based model slightly outperforms the persistence model, according to several error-metrics.

Discussion. The vital signs of the included patients remain mostly stable. Therefore, the naïve model performs relatively well, which may cause the mimicking. We suggest that clustering may enable selecting unstable periods in order to make the dataset more balanced. Furthermore, we give some directions for further research on the verification of predictive time series models.

Contents

1 Introduction

For the last six decades, *mechanical ventilation* has become a fundamental, established therapy in intensive care units. Ventilators provide respiratory support for patients who have trouble breathing on their own. The goal of this therapy is restoring the respiratory system of the patient so that ultimately, the patient can be extubated when the patient is ready to breathe on his own. The ventilators aim to improve the oxygenation of the patient (optimizing the oxygen intake) while relieving him in his efforts to breathe. Succeeding in these aims enables the patient to heal. This therapy, however, faces some challenges; while the goal is to heal the patient, ventilators can also damage the patient; the unwanted side-effects are referred to as *ventilator-induced injuries* [18]. Moreover, there are many methods of delivering ventilation to the patient, but finding the most suitable method for each patient is not trivial [7].

AI methods could help to establish the appropriate ventilation mode and settings. In recent years, several papers on the use of decision support systems, machine learning, and data mining aimed towards improving mechanical ventilation were published. Moreover, some ventilators already have intelligent ventilation modes that automatically adjust parameters based on the current condition of the patient [8]. Kilic and Kilic [23] describe a decision support system that decides whether a patient is ready to wean from ventilation based on fuzzy logic. Brigham et al. [7] describe a project whose goal it was to predict the response or efficacy of the current ventilation method on preterm newborns, by predicting future values of the minute volume metric, the volume that a patients expires during a minute, using deep learning. A first step in improving clinical decision making regarding ventilation could be to provide a prognosis of the vital signs

Figure 1.1: A patient monitor that shows a number of vital signs. Physicians and nurses use the vital signs to assess the condition of the patient. Furthermore, the monitor can alarm the caregivers if a vital sign moves outside a predefined range.

for the current ventilation method [7]. *Vital signs* are, in essence, sensor data measured by patient monitoring equipment. The vital signs can be used to assess the patient's condition. A *patient monitor* (see figure 1.1) provides a live overview of the vital signs. Some of the vital signs are plotted to give the caregivers an overview how the vital signs have changed over time. An example ventilator related vital sign shown in figure 1.1 is the *end tidal* CO_2 ($\text{ET}CO_2$, in white). Throughout this project, the aim is to predict future values of several vital parameters that indicate the current condition of the patient. A prediction model will be created based on the *PICU dataset*. The PICU data set is a multivariate time annotated data set of 1547 patients that were admitted to the *Pediatric Intensive Care Unit* (PICU) of the *Wilhelmina's Children's Hospital*, (*Wilhelmina Kinderziekenhuis* in Dutch and abbreviated as *WKZ*). The WKZ is a part of the University Medical Center Utrecht. The PICU dataset contains time series of these vital signs; we will refer to them further in this work as *Vital Signs Time Series* or VSTS. In chapter 2, we will describe these vital signs and ventilation practice in more detail. In addition to the Vital Signs Time Series, the dataset contains the settings of the ventilator and a limited set of static patient information, such as age and weight. During this project, this novel dataset is used for the first time. In order to gain more insight in the dataset and the data generating process, we performed an exploratory analysis, besides the creation of predictive models.

1.1 Clinical Context

The WKZ is currently working on new build plans for its Pediatric Intensive Care Unit. In order to provide a better healing environment for the patient and his/her family, patients get single rooms instead of being in a ward with other patients. While a single room provides privacy and rest for the patient, the caregivers lose situational awareness and oversight of the patient that they used to have in the open layout of the ward. The work process of the physicians and nurses relies on data, information, and alarms, but currently, these arrive at a tumbling speed. A *Clinical Decision Support* system is needed to help to make the clinical data more comprehensive for the healthcare professional, in order for him to provide the optimal care to the patient.

For the last two years, a group within WKZ has been exploring the possibilities of Clinical Decision Support in collaboration with the Department of Information and Computing Sciences of Utrecht University. Because of the expertise in the group on the ventilation practice and experience with the software that resides in modern ventilators, it was decided that solely improving the ventilator software was not the right way to start this process. Instead, they chose a more data-driven approach by leveraging the data stored in the *Patient Data Management System* (PDMS). The MetaVision PDMS provides a database with minute by minute data of more than 50 ventilation related variables, among other vital signs. This database can be queried using *SQL*. The PICU dataset that we employ in this project is derived from the PDMS of the WKZ's PICU. *Ventilation Decision Support* (VDS) could help the healthcare professionals at the PICU to optimize ventilation and weaning of ventilation by providing trend monitoring, and by giving a prognosis of the patient's vitals. This project was started in this context.

1.2 Time series analysis

As mentioned above, the PICU dataset contains *time series*. Time series are sequences of variables in the following form; $\{y_1, y_2, \ldots, y_t\}$. The subscripts denote the time period of the variable y and determine the ordering [47]. Because of the temporal component / temporal correlation, we cannot analyze a time series dataset as cross-sectional data. However, the PICU dataset is also a cross-sectional dataset, as the vitals are measured for each patient separately. For each of these patients, multiple vitals are measured simultaneously. Possibly, there are relations between the different vital signs; for example, a low heart rate might precede a decrease in the respiratory rate in a later time step. If this behavior is present in a majority of the patients, knowing the heart rate might help to predict future values for the respiratory rate. In chapter 4, we will discuss several analysis methods for time series and use these methods to investigate the presence of possible causal relations between the different variables that are present in the PICU dataset.

1.3 Deep Learning and AI

In the last decade, we have seen AI systems surface in many fields. Especially in recent years, big advances are made by systems that use *deep learning* methods like *Artificial Neural Networks*. Recent increases in the availability and use of distributed computing (mainly in the cloud), make the use of big datasets feasible. The size of datasets also increases, enabling learning algorithms to train on more examples. These circumstances combined enable us to model problems of increasing complexity [19]. Deep learning methods enable us to model complex problems; this could also be problems we do not fully know how to solve ourselves using traditional methods [26]. The PICU dataset is a vast dataset that contains many data points. When the amount of available training data increases, these methods become more useful [19]. In chapter 5, we will discuss several deep learning methods and show how these methods could be used to provide a prediction on the patient's vital signs, based on historical information. Furthermore, we will discuss how these models performed in our experiments.

1.4 Project goals

We already described some of the problems that the caregivers face and how AI methods may be useful to solve this problem. In this section, we describe the project's scope and objectives in more detail and discuss their motivation.

1.4.1 Exploratory Analysis

We will perform a descriptive analysis of the PICU dataset. In this analysis, we will describe the properties of the time series featured in the data set. We will investigate, for example, if time series are stationary or are autocorrelated. There may also be associations or causal relations between these time series.

These relations can be described using *Vector Autoregression* (VAR) models. A VAR model is a frequently used analysis technique for multivariate time series. Together with Granger causality testing, we can investigate possible causal relations between time series [5]. However, ordinary VAR models are patient-specific models and do not describe relations between vital signs time series that are present in the whole dataset, only those present within a specific patient. *Panel VAR* or *Multilevel Graphical VAR* models can be used to develop models of multivariate time series concerning multiple subjects. The latter technique can be used to visualize relations between variables in a graph.

As this was the first time this data set was analyzed, it is very worthwhile to make an extensive descriptive analysis. The exploratory analysis of the PICU dataset could give us valuable insights into the relations between the time series of the vital signs.

1.4.2 Predictive Models

We mentioned before that selecting the optimal ventilation mode and settings is not an easy task, as many different factors influence the efficacy of the chosen ventilation mode. This vast number of factors that influence the patient's condition makes it difficult to predict the patient's future state given the current ventilation mode. However, deep learning methods may be able to learn the relation between historical data points and future values. The aim is thus to create a model that, given a series of historical vital signs and settings, predicts future values of a subset of these vital signs. Deep learning methods like Recurrent or Convolutional Neural Networks may enable us to process this vast amount of data available in this dataset.

In consultation with the doctors of the WKZ, we chose the following target variables:

1. Oxygen Saturation (SpO_2);
2. End-tidal CO_2(EtCO_2);
3. Expiratory Tidal Volume ($V_{T_{\text{exp}}}$);
4. Peak Pressure (P^{PEAK})

These variables give some insight into the condition of the patient and the efficacy of the current ventilation method.

Furthermore, the model should be generic; we mean by this is that it should be useable for multiple patients. Therefore, the model should not be tailor-made for an individual patient. Ideally, given a history $\{t_{0-h}, \ldots, t_0\}$, the model should make reliable predictions for t_{0+s}, where s is in the range of a couple of hours. Brigham et al. [7] performed similar experiments, but in their work s was very small; s was at most 15 seconds or ten time steps. We investigated what was feasible. Besides solely predicting a single output t_s, we also attempted to predict the intermediate values between t_0 and t_s.

1.4.3 Research Objectives

We can summarize the two sections above, with the following two research objectives:

Research Objective 1. To describe the properties of the individual time series and possible (causal) relations between them. We will perform these tasks to gain more insight into the data generating process.

Research Objective 2. To create models that, after being trained on multiple time series of both measured variables and settings entered by the ventilator's operators, predict future values of the patient's oxygen saturation, end-tidal CO_2, expiratory tidal volume, and peak pressure.

1.5 Other research questions

In addition to the two objectives discussed above, some other questions need to be answered in order to accomplish these. We will discuss these questions in the following sections.

1.5.1 Concerns regarding the PICU dataset

As opposed to some data sets the PICU data set is not entirely "clean" since it was collected from an actual intensive care unit, where this data was passively gathered; the recordkeeping of the vitals is not primarily aimed towards research. At the PICU, manual interventions take place, but these are not always directly visible in the dataset. One of the reasons is that some actions have to be manually entered by the physicians and nurses, and if it is registered, the data entry method may not be consistent over time. An example of manual intervention is the administration of medication. Certain drugs can influence the respiratory system and, in turn, these drugs affect the recorded vital signs [18]. The administration of drugs is recorded in the PDMS, but not at the exact time of the administration, and therefore, not included in the PICU dataset. The absence of recorded interventions in the dataset may cause problems in determining causal relations between these variables. Fortunately, changes in ventilator settings are recorded in the dataset. This problem, however, persists. The fact that we do not know if a manual intervention took place is not something that can be (easily) fixed in the dataset afterward, and it may raise questions on the trustworthiness of a model that was trained on this dataset. A methodology on how to deal with this problem is not yet available, and solving this is probably not feasible in the scope of this project. However, during the project, we kept in mind that interventions took place and that these may influence the predictions and causal relations.

In addition to these more methodological problems, there are also some other problems present in the dataset. Throughout all patients and variables, some values are absent or unreliable which may be due to equipment failure, manual interventions or the patient's behavior. Moreover, many machine learning methods or analysis methods cannot deal with missing data points. In chapter 3, we will discuss the treatment of missing data points and how we prepared the dataset for the actual data analysis.

Research Question 1. How do we prepare the raw dataset for further analysis?

1.5.2 Patient subsets

The PICU dataset is very heterogeneous, in the sense that it contains pediatric patients from all age groups between newborns and adolescents and with a broad spectrum of diseases. We will describe some aspects of the patient population in section 3.3. Brigham et al. [7] had similar aims as we have. The number of patients that were included in their dataset was small (10 patients), but the patients were in a very similar

condition, and they all had the same age approximately. The homogeneity level of their dataset is not present in ours. We could, however, decide to create different models for each ventilation mode or for specific patient groups instead of a generic model to increase the performance. During this project, we will investigate if this will improve performance or influence the results of analysis methods:

Research Question 2. Will the creation of models that are trained on subsets of patients improve the model's accuracy?

1.5.3 Learning method

Multiple deep learning methods may be suitable for modeling this problem. Networks that belong to the class of *Recurrent Neural Networks* are used a lot for learning on data in which a specific ordering or temporal relation is present. For example, RNNs are used to model natural language and time series analysis/forecasting. Brigham et al. [7] used a *Long Short-Term Memory network*, (a RNN variant) to create a prediction model. In addition to Recurrent Neural Networks, *Convolutional Neural Networks* may also be useful for time series analysis. In this project, we will investigate which methods are suitable and how these models can be applied to the PICU dataset and time series in general.

Research Question 3. Which deep learning methods are suitable for making predictive models for Vital Signs Time Series?

1.6 Thesis outline

The structure of this thesis is as follows: In chapter 2, we first discuss the clinical concepts of ventilation further. In chapter 3, we will discuss the PICU dataset in detail and explain how the dataset was made usable for analysis. Next, in chapter 4 we will discuss a range of time series analysis methods and use these methods to learn the presence of causal relations between the different variables that are present in the PICU dataset. In chapter 5, we will discuss several deep learning methods and how these methods could be used to provide a prediction on the patient's vital signs, based on historical information. After that, we will discuss the results to research aim two. In chapter 6, we will provide a global discussion and conclusion on the results discussed in chapters 4 and 5.

Of all the experiences the physician must undergo, none can be more distressing than to watch respiratory paralysis in a child with poliomyelitis - to watch him become more and more dyspneic, using with increasing vigor every available accessory muscle of neck, shoulder and chin-silent, wasting no breath for speech, wide-eyed, and frightened, conscious almost to the last breath.

Dr. James L. Wilson, in [11]

.

2 Ventilation

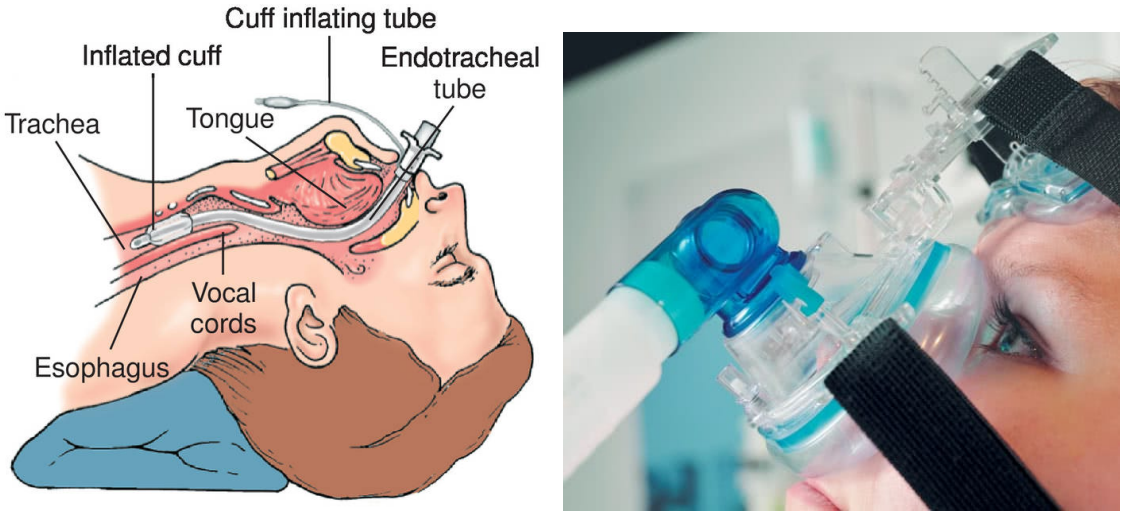
2.1 Introduction

The disease polio(myelitis) can be seen as the main reason for the development of the first machines that provide artificial respiration. Many patients during polio epidemics were experiencing respiratory problems [18]. The first device that provided mechanical ventilation was the tank respirator, perhaps more known as the *iron lung* [31]. Drinker and Shaw made one of the first prototypes in 1929, describing them in their paper called "An apparatus for the prolonged administration of artificial respiration: I. A design for adults and children" [11]. Patients were put inside the tanks, and the machine-generated a negative pressure, causing the size of the lungs to increase. In this way, the negative pressure simulates an inhalation. In the 1960s, more advanced more machines emerged into practice. These machines provided *positive pressure ventilation*; instead of forcing the lungs to increase, the machines pushed air into the lungs. These machines also allowed patients to trigger the ventilation themselves. In the 1980s, ventilators with screens were introduced; the curves of vital parameters could now be plotted on these screens, enabling operators to monitor the patient condition more precisely. In the following years, computing became more and more ubiquitous; eventually, computers were also embedded in ventilators, providing more advanced and patient-tailored ventilation. This evolution resulted in an enormous number of settings and parameters that can be set on the ventilators to provide this patient-tailored care. For now, we will disregard all these parameters; then, there are two main ventilation modes that the caregivers can choose from:

Invasive ventilation uses an endotracheal tube, which is directly placed through the mouth into the trachea (see figure 2.1a for an illustration). An inflatable cuff seals the normal airway, preventing leakage. The tube is then connected to the ventilator.

Non-invasive ventilation uses a face or nasal mask and is more suitable for patients that are in a better condition, in contrast to the invasive method. As non-invasive ventilation is much more comfortable than invasive, this method is preferred when possible. Invasive ventilation, however, gives more control, as there is less leakage because of the cuff. In figure 2.1b, a nasal mask is shown.

When the caregivers have decided between these, then they have to decide how to configure the ventilator. However, further in this work, we will only consider invasive ventilation. In the following sections, we will first describe and formalize some essential concepts of normal breathing and how they connect to mechanical ventilation. After that, we will describe the essential concepts and settings of mechanical ventilators. In these sections, we will mainly follow the information presented in Chatburn et al. [8] and Gommers and Rosmalen [18], as they provide a good overview of this domain.



(a) Invasive ventilation **(b)** Non-invasive ventilation

Figure 2.1: The two main ventilation methods: invasive[1] and non-invasive ventilation.

2.2 Breathing

Breathing, we do it almost without thinking, but defining or more formally describing what we do during a breath is not trivial. The goal of breathing is to provide the body

[1]Figure adapted from https://medical-dictionary.thefreedictionary.com/endotracheal%2Btube

with oxygen (O_2) and rid the body of carbon dioxide (CO_2). In the alveoli, *diffusion* takes place; differences in concentration enable the gas CO_2 to travel from the blood to the alveoli and oxygen to enter the blood from the alveoli. During inhalation, the chest size is increased, and as the size of the lungs also increases, negative pressure is created in the alveoli. Subsequently, air flows from the mouth to the alveoli to balance this negative pressure. When the pressure is balanced, diffusion, as described above, takes place. During expiration, the chest muscles are relaxed, and then the lungs return to their smaller form. The smaller form causes an increase in pressure within the alveoli, causing the air to flow outwards [18]. There are several variables in that characterize our breathing. In this section, we will discuss the most important variables:

2.2.1 Time related variables

A breath is defined as a single cycle of inspiration and expiration. During an inspiration there is a positive air *flow* from the mouth to the lungs and during expiration, the flow is negative, as the air flows back from the lungs to the mouth. *Flow* denotes the speed of the airflow, or how much air flows in or out the lung per minute. In figure 2.3, we can see that during inspiration, the airflow is constant, while during expiration, the (absolute) flow decreases towards the end of each breath. Flow is often expressed in the unit $\mathbf{l/min}$.

Figure 2.3 describes, besides flow, several other variables that are included in the dataset. The most obvious one is the *respiratory frequency*, which can be derived in the following way:

$$t_{\text{cycle}} = \frac{1}{f_{\text{resp}}} \tag{2.1}$$

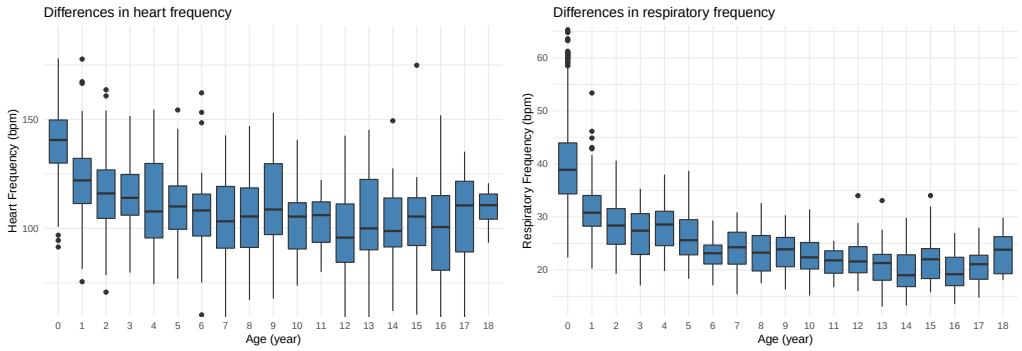
$$f_{\text{resp}} = \frac{1}{t_{\text{cycle}}} \tag{2.2}$$

where

t_{cycle} : time needed for a full breathing cycle
f_{resp} : respiratory frequency

Our *respiratory frequency* is amongst others related to our *heart rate*. An increased heart rate (caused by increased labor for example) means that more O_2 is consumed and as a consequence, the arterial CO_2 pressure ($PaCO_2$) rises as the body produces CO_2 when O_2 is consumed. An increased $PaCO_2$ causes an increase in the respiratory rate as an expiration usually lowers the $PaCO_2$. By increasing the respiratory frequency, the body can normalize the $PaCO_2$ levels. We denote the respiratory frequency in the number of breaths per minute (BPM). For the heart rate, we record the number of heart beats per minute in a similar way.

The ratio between the time needed for the inspiration and expiration is defined as



(a) Heart rate **(b)** Respiratory rate

Figure 2.2: The relation between age and the heart and respiratory rate. The boxplots are based on the PICU dataset.

the *inspiratory-expiratory ratio* (I:E-ratio), which is defined in the following way:

$$ie = \frac{t_{\text{insp}}}{t_{\text{exp}}} \tag{2.3}$$

On a ventilator, caregivers can set the optimal I:E-ratio for each patient. The ventilators also record the actual measured ratio, especially for patients who breath spontaneously and are only supported in their breathing efforts.

The heart rate and respiratory frequency are related to age. This is illustrated in figure 2.2. For babies, both frequencies are higher on average compared to other age groups.

2.2.2 Volume related variables

Besides time and frequency, we can also describe a breath in terms of volume. Formally, the *Tidal Volume* (V_T) is defined as the integral of the flow time curve, or less formally, the flow multiplied by time [8]. In the PICU dataset, this variable is expressed in milliliters. There are two types of Tidal Volume recorded; the *Expiratory Tidal Volume* ($V_{T_{\text{exp}}}$) and *Inspiratory Tidal Volume* ($V_{T_{\text{insp}}}$). During invasive ventilation, still some *leakage* can occur. Leakage is defined as the difference between the $V_{T_{\text{insp}}}$ and the $V_{T_{\text{exp}}}$. When the patient receives non-invasive ventilation, the amount of leakage is often even bigger, as there is less control over the air flow.

During the child's development, his lung capacity increases and in turn, his tidal volume. When we compare weight and Tidal Volume, we can see this relation. In figure 2.4, this relation is visualized, based on the data available in the PICU dataset. The tidal volume is also used as a setting. On the ventilator, caregivers can set the amount of air the patient should receive each breath or should receive during a minute. The latter is referred to as the *minute volume* or *flow rate*.

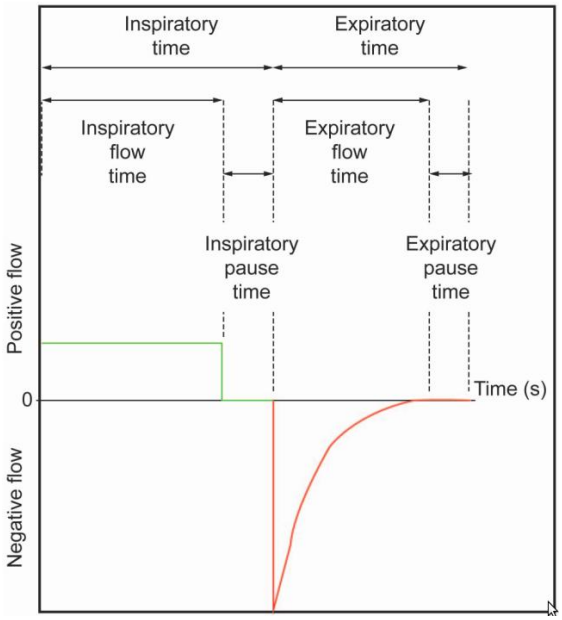


Figure 2.3: A breath is defined as a single cycle of inspiration and expiration. This figure can be used to describe several variables; for example, flow, respiratory frequency and the I:E ratio. This figure is adapted from [8].

2.2.3 Diffusion related variables

The ventilators and monitors also measure the success of the diffusion, which can give some indication of the lung's health. Below we will describe two variables and one setting related to the diffusion of O_2 and CO_2:

Oxygen saturation (SaO_2 and SpO_2). Oxygen saturation is defined as the percentage of red blood cells (hemoglobin) that are saturated with oxygen. Healthy saturation levels are in the range of 95 to 100 %. When we look at figure 2.5, we can see that in the PICU dataset, most saturation levels are in this range. Dangerous lower levels are recorded, but they do not occur a lot in the dataset. Low saturation levels are immediately counteracted by the physicians and nurses.

The saturation can be measured in two ways. We can determine the oxygen saturation by blood gas analysis and by pulse oximetry. The first method is more accurate; however, this cannot be measured continuously because this method is invasive. Saturation measured by blood gas analysis is referred to as the arterial saturation (SaO_2). A pulse oximeter is a non-invasive device, which is slightly less precise but can be measured continuously. The saturation measured by a pulse oximeter is referred to as periferous saturation (SpO_2). In the PICU dataset, only SpO_2 measurements are included.

End tidal CO_2 ($EtCO_2$). This variable describes the concentration of CO_2 in the expired air. The ventilator measures this variable. $EtCO_2$ can be expressed as a frac-

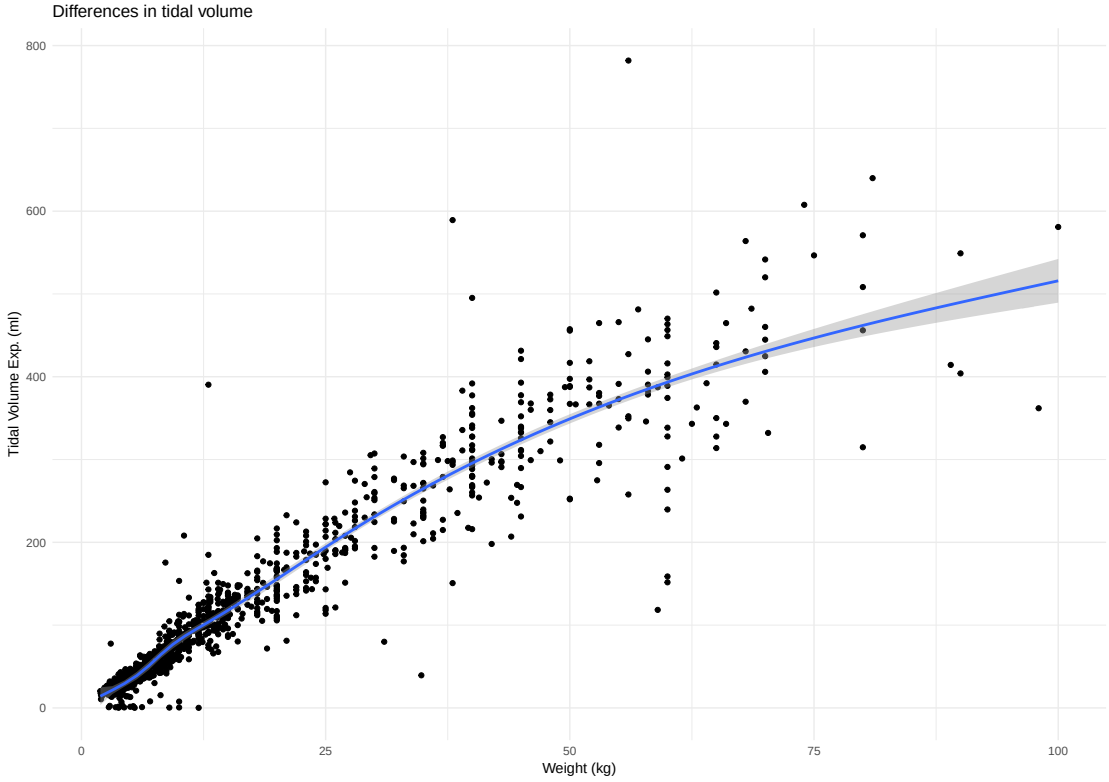


Figure 2.4: Weight vs. Tidal volume, based on the patients included in the PICU data set

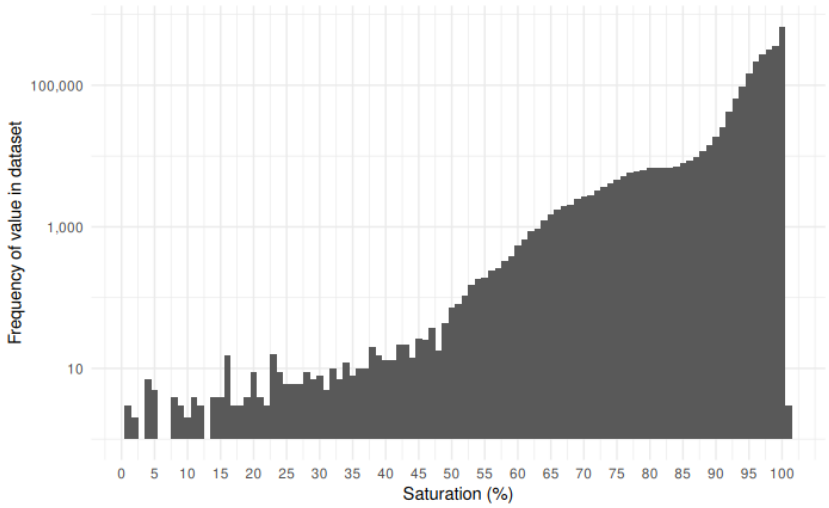


Figure 2.5: Frequency of saturation values in the PICU dataset.

tion/concentration or as a pressure. In the PICU dataset, the $EtCO_2$ is recorded as a pressure using the *mm Hg* unit. Typical values of the $EtCO_2$ variable lie between 35 and 45 mmHg, which corresponds to a 5 to 6 % concentration of CO_2 [18]. This is also visible when we look at figure 2.6, which shows the frequency of $EtCO_2$ values in the PICU dataset.

Fraction of inspired oxygen (FiO_2). The earth's atmosphere contains roughly 21 % oxygen. However, in some cases, it would be beneficial for a patient if the oxygen levels were higher (for example, when the SaO_2 is low). The O_2 fraction of the applied gas mixture can be set using the FiO_2 setting. The average FiO_2 level is 34 % in the PICU data set, so above atmospheric levels. The FiO_2 level does not depend on static variables like weight or age. The ventilator also measures the FiO_2 level. Because the FiO_2 level is higher for more ailing patients, this value could give be indicator of the patient's health.

2.2.4 Pressure related variables

We already described how pressure relates to air flow. The ventilators can also regulate and measure several pressure related variables:

Positive End Expiratory Pressure (PEEP). After expiration, there is some residual pressure still present in the lungs. The word positive refers that this pressure level is above atmospheric levels. Patients on ventilators are also given an applied PEEP. The PEEP is applied to prevent *alveolar collapse* by ensuring that the alveoli stay open. When alveoli collapse, no diffusion can take place, preventing oxygen from entering the bloodstream, while carbon dioxide cannot be expired either. The

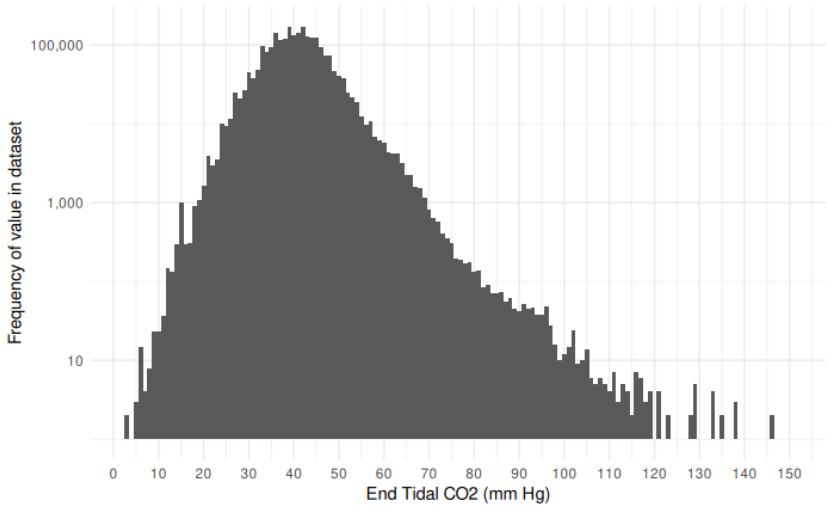


Figure 2.6: Frequency of $EtCO_2$ values in the PICU dataset.

amount of PEEP that needs to be applied depends on the lung's health. Normal PEEP levels are around 5 cm H_2O, and higher levels are associated with more ailing lungs.

Pressure above PEEP. During ventilation, a positive pressure level, above the PEEP level, is needed to ensure that air flows into the lungs. The amount of pressure that is applied can be set for several ventilation modes.

Peak Pressure (P^{PEAK}). For some ventilation modes, the pressure is not a fixed setting but varied based on the condition of the patient. The ventilators measure the applied pressure levels to ensure that the pressure levels do not reach dangerously high levels. The pressure levels rise and decline while a breath progresses. Therefore, the peak pressure level of each breath is recorded.

2.3 Ventilation modes

In the previous section, we described several parameters that can be set on a ventilator. Besides these settings, there are also several methods of applying ventilation, these are referred to as *ventilation modes*. A ventilation mode consists of three parts [8]:

1. Breath Control Variable
2. Breath Sequence
3. Targeting Scheme

In the sections below, we will introduce these parts and describe what influence they have on the settings and outcome.

2.3.1 Breath control variable

The Breath Control Variable decides Ventilation can be described using the following equation (assuming that there is no respiratory effort from the patient) [8]:

$$P(t) = E \cdot V(t) + R \cdot \dot{V}(t) \tag{2.4}$$

where

P : pressure
E : elastance or compliance; indication of the lung's condition
V : volume
R : resistance; this rises when there are obstructions in the trachea and/or tube
$\dot{V}$: flow
t : time, with $P(t)$, $V(t)$ and $\dot{V}(t)$ as continuous functions of time t

If we predetermine the pressure function, then we can derive the volume and flow. The same holds the other way around, so when the volume and flow functions are derived, the pressure can be derived. Therefore, we can control ventilation in two ways:

Volume Control (VC). In modes where this breath control variable is used, flow and volume are set on the machine by the operator. The pressure varies based on the lung condition; more precisely, the pressure is influenced by the compliance and resistance as per equation 2.4. In figure 2.7, it is shown how flow, pressure, and volume behave over time. Volume Control guarantees a certain minute volume by setting the frequency, as $MV = f_{\text{resp}} \cdot V_T$. Volume Control is a suitable mode for patients whose arterial CO_2 pressure needs to be constant. However, there is a downside; if the pressure rises due to decreasing compliance or perhaps a higher resistance due to sputum lung damage can be inflicted on the patient [18].

Pressure Control (PC). In this mode, the peak inspiratory pressure is fixed while the Tidal Volume and flow vary. Suppose the operators want to increase the patient's minute volume. We cannot achieve this by increasing the respiratory frequency and the tidal volume, as these variables are not the control variables in this modality. We can achieve this by increasing the peak pressure and the time for each inspiration. The latter can be controlled by changing the I:E ratio.

During both modes, PEEP is applied to ensure that the alveoli stay open. The pressure variables that vary during volume control are thus the Pressure above PEEP and the Peak Pressure.

[2]Adapted from https://commons.wikimedia.org/wiki/File:Pressure_regulated_volume_control_graphic.jpg

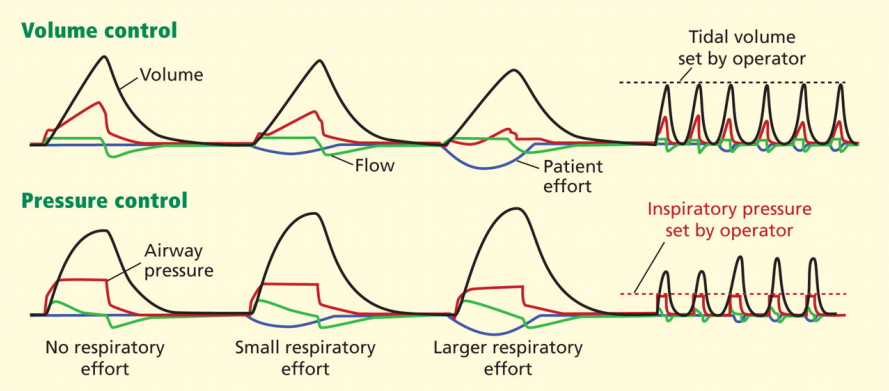


Figure 2.7: Volume control vs. Pressure control [2]

2.3.2 Breath Sequence

The former part described the regulation of the airflow. The operators also have to decide who is in charge of starting a breath; the patient or the ventilator. There are patients who cannot start the inspiration themselves. The operators choose one of the following three *breath sequences*, according to the condition of the patient.

Continuous Mandatory Ventilation (CMV). When this setting is used, the machine always initiates the inspiration. Inspiration is then time-cycled, meaning that inspiration and expiration occur on a fixed time defined by the frequency on the ventilator.

Continuous Spontaneous Ventilation (CSV) . This mode is the complete opposite of CMV, as patients ventilated by this mode initiate the inspiration themselves and the ventilators only support the breathing of the patient by applying some additional pressure. The patient triggers all the work of the ventilator. A disadvantage of this method is that there is no guarantee on the minute volume, as the patient is fully autonomous over his respiratory frequency.

Intermittent Mandatory Ventilation (IMV). This mode is somewhat in the middle of the two previous ventilation modes, by allowing the patient to breathe spontaneously, but still having some control on the respiratory frequency: The patient is given a window in which he can start an inspiration. Within this window the patient can trigger a supported inspiration by the machine. If the patient, however, does not trigger the machine in this window, the machine starts a mandatory inspiration directly after missing the window. With this method, some guarantees on minute volume can be made.

The Volume Control is designed to guarantee a minute volume; therefore, there does not exist a CSV mode for Volume Control.

2.3.3 Targeting Scheme

While our condition changes, so do our breaths change as well. This also is the case for patients who receive mechanical ventilation. With *Targeting Schemes*, the ventilators can adjust the Breath Control Variables according to the condition of the patient:

Setpoint. In this targeting scheme, all settings are fixed. A benefit of this system is that it is known beforehand what this system is going to do. However, as the patient's condition changes slightly over time, the setting should be adapted manually.

Servo. This mode increases the pressure when the flow decreases, which is caused by a higher resistance. By increasing the pressure, more air flows in the lungs, mitigating possible decreased oxygenation.

Adaptive. This scheme is among the more advanced targeting schemes. Among practitioners, this mode is probably more known as **Pressure Regulated Volume Control**. In this scheme, a goal tidal volume is set. By slightly adjusting the pressure based on the work of the patient, the ventilator ensures that this goal is reached. The pressure is measured over time and should remain between safe bounds, as with regular volume control. The tidal volume is reached by controlling the pressure; therefore, this mode does not belong in the volume control category, but to the Pressure Control category.

There are more targeting schemes than the aforementioned three. These can be found in Chatburn et al. [8] and Gommers and Rosmalen [18].

2.4 Predictors and Target variables

Frequently used ventilation modes on the WKZ PICU are the following:

- PC-CMV-SetPoint
- PC-CSV-SetPoint
- PC-IMV-Adaptive

Further in this work, we will only consider the PC-IMV-Adaptive mode. In section 3.5, we will further elaborate on this choice. Here, we will briefly list some of the measured variables and settings that are important for this ventilation mode and will be target variables or predictors in our predictive models.

Peak Pressure ($\mathbf{P^{PEAK}}$). This mode falls into the Pressure Control class. However, as in this mode the pressure is automatically adapted by the ventilator, the applied pressure is not set by the operators. Instead, the Tidal Volume and the patient's condition determine the applied pressure. The Peak Pressure is, however, recorded and could give some indication on the resistance in the air way.

Tidal Volume. As this is the most important setting for this ventilation mode, this value is included as a predictor. We will also include the measured Expiratory Tidal Volumeas a target variable. The difference between the setting and the two could be an indication of *leakage*.

SpO_2. Low saturation levels should be counteracted as soon as possible, therefore, a prognosis for this vital sign is helpful.

$EtCO_2$. Dojat et al. [10] and Kilic and Kilic [23] used the $EtCO_2$ variable as a predictor for determining if a patient is suitable for weaning[3]. The $EtCO_2$ variable is, therefore, also an indicator of the efficacy of the ventilation.

Respiratory Frequency. To prevent hyper- and hypoventilation, it is good to monitor the respiratory frequency.

Heart rate. This vital sign is not directly related to ventilation, but we described in section 2.2.1 how the heart rate and respiratory frequency are related.

In the predictive models, these above mentioned target variables will also be used as predictors. In addition to these, all the settings and corresponding measured versions of them will be included.

2.5 Summary

Above, we have discussed the most important ventilation principles. We described how we can formalize breathing and how we can characterize breathing using several types of variables. These variables describe the timing, volume, applied pressure, air composition and the quality of the diffusion. These variables can also be used to assess the lung's health. We also described the main ventilation modes and how they differ. We have also described why these are chosen as predictors and target variables in our predictive models. In chapter 4, we will first describe possible causal relations between these variables.

[3] Kilic and Kilic [23] actually used the $PaCO_2$; however, there is a similar relation between $EtCO_2$ and $PaCO_2$ as there exists between SpO_2 and SaO_2.

3
PICU Dataset

3.1 Introduction

In this work, a data set gathered at the Pediatric Intensive Care Unit of the Wilhelmina Children's Hospital will be analyzed. This dataset is referred to in this work as the *PICU dataset*. The dataset contains time-stamped patient information collected from the database used by the *MetaVision Patient Data Management System* (PDMS). A PDMS is an application that is used by physicians and nurses at the PICU to monitor the vitals of the patients. The data in this database is provided by a range of medical equipment; bedside monitors, ventilators, pumps, et cetera. These devices provide the measurements they perform. Devices that are not only monitoring but are also actively involved in treatment (e.g., ventilators), provide their settings also. All data streams provided by medical equipment are recorded at a frequency of once every minute.

The PICU dataset is a derivation from this PDMS. In the PICU dataset, only patients are included that were attached to mechanical ventilation for at least 24 hours. However, patients with congenital heart disease were excluded. The dataset contains 1547 patients, who were admitted between the years 2008 and 2018. The patient population is very diverse; it contains pediatric patients from all age groups, ranging from age 0 to 18. In this section, we will describe the main characteristics of this dataset. First, we will describe the organization of the dataset in section 3.2. After that, we will describe the patient population in section 3.3. In section 1.5.1, we already expressed some concerns regarding this dataset. One of the main concerns is missing data points; within several vital signs time series, observations are missing. Many data analysis methods cannot

handle missing data points. We will discuss in section 3.4 how we prepared the raw dataset for further analysis.

3.2 Dataset organization

Initially, a big dataset containing anonymised patient information was provided in a single file in the CSV format. This file had a size of approximately 5.0 GiB and it had the following structure:

```
patientid,time,eng_venttype,other_variable, ...
9,2119-11-10 10:35:00,,3.5, ...
...
```

For every patient, there are 88 variables recorded at each timestamp. The CSV then contains 90 columns in total; including the timestamp and the patient identifier as columns. The patient identifier and timestamps are anonymised; the date part of the timestamps are fictional, in order to preserve patient privacy. Because of the file's size, it cannot directly be used in analytical programming environments like R. The dataset was adapted as follows.

First, the original file was split in smaller chunks; we created a separate file for every patient. Then a parsing schema was created for the whole dataset. There is a reason we did this after splitting, because of the file size we could not determine the right parsing schema for the columns of the whole dataset. After considering a number of rows from a sample of the patient population, we were able to form a parsing schema. One of the reasons why auto-discovery of a parsing schema did not work is the sparsity of the dataset; not all variables are recorded for every patient, as there are separate columns for the variables and parameters of the individual ventilators (e.g., `eng_venttype` and `ser_venttype` for the Engstrom and Servo ventilators respectively). Sampling several rows from the whole data set enabled us to make a type schema of all the columns of the entire data set. This schema enabled parsing every patient file correctly and access all the records in a type-safe way.

Using a program written in F♯, a language in the functional programming paradigm that is very suitable for data discovery and analysis, we were able to aggregate some cross-sectional statistics for each patient. We created an overview containing some basic statistics for some of the variables of interest. This resulted in a table which included the variables given in table 3.1. The parameters in table 3.1 include the most indicative parameters. The aggregated table may be used to group patients on age, and perhaps condition. Indicators for patient health are the vital signs and the settings themselves. By finding extreme settings, for example, the PEEP and FiO_2 parameters, we may be able to find interesting patients. The resulting table was used to produce some of the graphs and plots in chapter 2.

Monitor	**Ventilator**	**Patient**
$ETCO_2$	Expiratory Tidal Volume	Identifier
Heart Rate	Total Frequency	Admission Date
SPO_2	PEEP	t_{start}
Resp. Frequency	FIO_2	t_{end}
	Ventilator Type	Length of Stay
	Ventilation Mode	Age
		Weight

Table 3.1: In this table, the variables are given that are included in the aggregated table. From the monitor and ventilator variables, the minimum, maximum, and average values are calculated. The ventilator types and ventilation modes are aggregated as lists.

3.3 Patient population

As said earlier, the PICU data set contains 1547 patients, who are all between the ages 0 and 18. Most patients, however, are very young; 821 of them are between 0 and one years old. In figure 3.1, the age distribution is shown. The number of older children that stay longer than 24 hours on the PICU is small compared to the group of younger infants. In turn, most of the patients that are younger than a year are also between 0 and one months old at the moment of admission.

The average time spent on the PICU is 296 hours, which translates to 12 days. The median length of stay is 170 hours, which is approximately seven days. From this observation, we may assume that the patients' time series are long enough to correctly analyze them. We did not find a relation between age and length of stay.

In addition to age, we can also divide the patients over the several ventilator types. As there are differences between ventilator types, it may be desirable to create ventilator-specific prediction models. In figure 3.2, the results are shown. There are four types of ventilators:

- Engstrom
- Carescape
- Servo-i (also referred to as Servo)
- Servo-u

A large group of patients ($n = 800$) was solely connected to the Servo-i ventilators. The second biggest group was solely connected to an Engstrom ventilator. By looking at figure 3.2, it would seem that a large group ($n = 169$) was not connected to any ventilator. We found out, however, that this was not the case; the patients who were not associated with a ventilator were connected to the Servo-300 ventilator, a predecessor

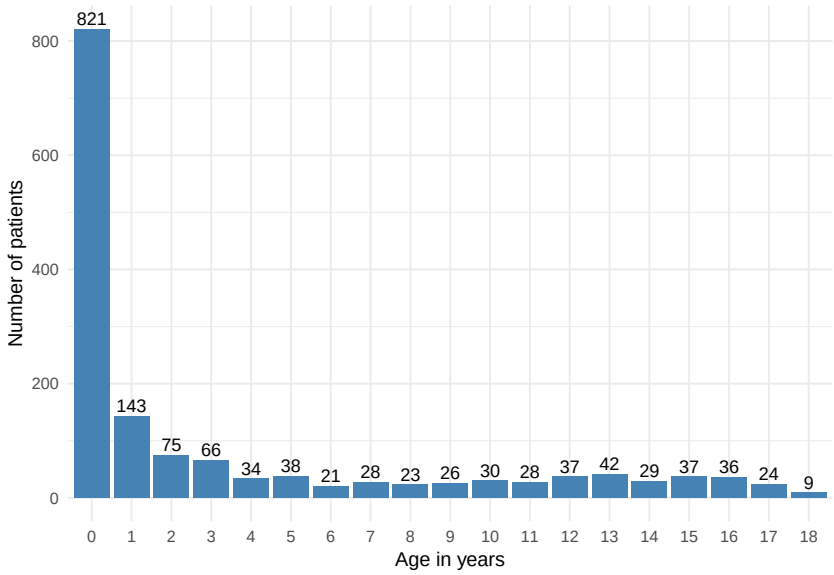


Figure 3.1: Age distribution in the PICU data set

of the Servo-i and Servo-u ventilators. The Servo-300 did not report its settings to the PDMS, and therefore, we could not identify it (as identification was based on the settings columns).

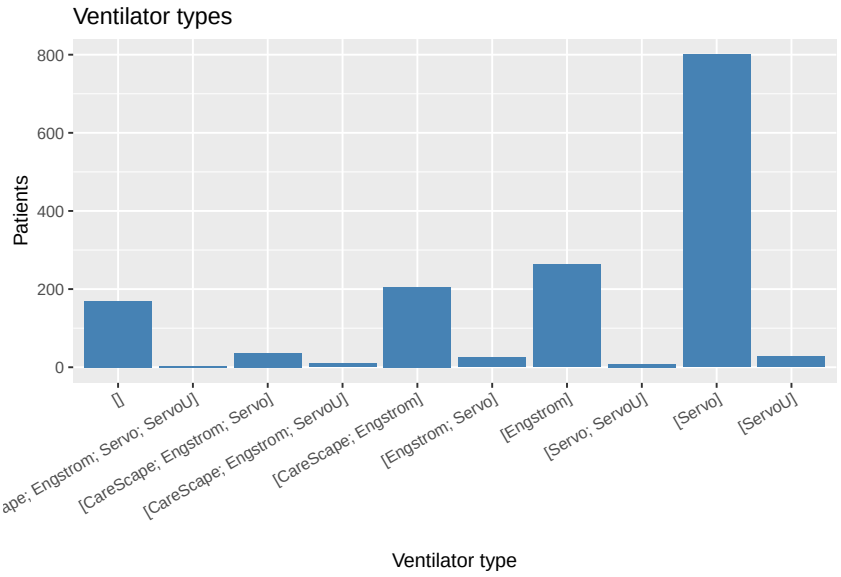


Figure 3.2: Number of patients per ventilator type

3.4 Dataset preparation

We already mentioned in section 1.5.1 that the raw PICU dataset cannot be used in analysis methods directly. Below, we will provide several reasons for this.

Differences between ventilators and ventilation modes. Each ventilator stores its settings in columns that are only used by that specific ventilator type. Most settings are present for all types, but some are only present for a specific type. In table 3.2, an overview is provided of the setting parameters.

Besides the difference between ventilator types, there are also variations between ventilator modes. For example, when patients are ventilated in a CMV mode, only the `vendor_pres_pc`[1] pressure settings are used, whereas in a IMV / CSV mode, the `vendor_pres_ps`[2] settings are used. Furthermore, when a VC or PC-IMV-Adaptive mode is used, the pressure settings are not used; the systems aim to achieve a goal tidal volume/minute volume and vary the pressure themselves. This difference is also reflected in the measurements; it depends on the ventilation mode, which variables are measured. The fact that several columns remain empty for certain modes means that we cannot create models that work for all ventilation modes.

Gaps in the data. There are periods in the dataset in which the patient is disconnected from the ventilator and monitoring equipment. These episodes are recorded as

[1] `pc` stands for *Pressure Control*, which is a CMV mode
[2] `ps` stands for *Pressure Support*, which is, depending on the configuration, a CSV or IMV mode

gaps in the data; there are no measurements stored for these timestamps. Patients are disconnected from regular equipment during surgery or transportation. After surgery, the patient is again connected to the equipment at the PICU. We cannot treat the gaps as part of a time series, as within these gaps, the state of the patient may have fundamentally changed.

Missing measurements. There are also small periods in the dataset where data from only one or a few variables is missing. This can happen, for example, due to equipment failure, patient behavior, or a manual intervention by the caregivers. These missing values are stored as **NA** values. With imputation methods, we can fill in possible values for the missing variables.

In the following sections, we will describe how we adapted and prepared the PICU dataset for further analysis. In section 3.4.1, we will discuss how we solved inter-ventilator differences and how we divided the dataset into uninterrupted episodes. In section 3.4.2, we will discuss how we imputed the remaining missing values.

3.4.1 Dataset redesign

Setting	Servo-i	Servo-u	Engstrom	Carescape
Mode	ser_ventmode	seru_ventmode	eng_ventmode	car_ventmode
Patient Type	ser_pttype	seru_pttype	eng_pttype	eng_pttype
PEEP	ser_peep	seru_peep	eng_PEEP	eng_PEEP
Pressure	ser_pres_pc, ser_pres_ps	seru_pres_pc, seru_pres_ps	eng_pres_pc, eng_pres_ps	eng_pres_pc, eng_pres_ps
Frequency	ser_freq	seru_freq_min	eng_freq	eng_freq
Minute Volume	ser_mv	-	-	-
Inspiration Time	ser_insptime	-	eng_time_insp, eng_time_supp	eng_time_insp, eng_time_supp
Inspiration Rise Time	ser_insprisetime	seru_insprisetime, seru_insprisetime_msec	eng_insprisetime_ps, eng_insprisetime	eng_insprisetime_ps, eng_insprisetime
Trigger Sensitivity Below PEEP	ser_trigsens	-	-	-
F_IO_2	ser_fio2	seru_fio2	eng_fio2	eng_fio2
I:E-Ratio	ser_ieratio	seru_ieratio	-	-
Tidal Volume	ser_tv	seru_tv	eng_tv	eng_tv

Table 3.2: In this table, all the settings that describe the same concept are placed on the same row. In the preparatory phase, the values of the original columns are stored in new ventilator generic settings columns. Some settings have multiple variables (e.g., pressure). The choice between them often depends on the chosen ventilation mode.

During the preparatory phase, we created a new schema for all the variables. We combined all the vendor-specific columns that describe the same concept into one singular column. We reduce the sparsity of the dataset by doing this; in turn, this made the dataset much more comprehensive. We also combine some of the measured variables (e.g., frequency of the Engstrom and Carescape ventilators). Some variables are based on other variables. An example derived variable is the compliance variable. Not every ventilator records this value, and therefore, we omitted this variable in the new schema. We keep only the measured variables that are measured by all ventilators. We combine these into a new record schema. The new schema for each recorded time step is

shown in table 3.3. The ventilator mode is stored according to the classification method provided by Chatburn et al. [8]. In the original dataset, the modes were encoded as integers, and each ventilator type had a different encoding.

As said earlier, as some variables and parameters are omitted for some ventilation modes, we cannot create a generic model for all modes. Therefore, for each patient, we will create episodes. In these episodes, the ventilation mode remains the same. These episodes are also without gaps. When a gap is found, a new episode starts. We already split the dataset based on the patient identifier. We did this by reading the original CSV-file line-by-line and appending the contents verbatim to patient-specific files. Here, we read these patient-specific files and parse the results according to a type-schema. We store the information in a record-type that corresponds with table 3.3. If this is the first measurement, a new episode is created. All subsequent records are parsed and added unless the ventilation mode differs from the previous record or the timestep has increased by more than one minute. The latter indicates a gap. When one of these situations occurs, the last read record is added to a new episode. When all the contents of the patient's file have been read, the new episodes are saved to a new patient-specific file. Each episode has a unique identifier. For each patient, we also stored the total number of episodes and global parameters as age, weight, length of stay, and admission date in a separate file. All the episodes are also stored in a separate file, so we can make selections based on ventilation mode.

Record Info	Measurements:	Settings:	Actions	Ventilator Info:	Comments:
ID	HR	Pressure PS	Suction	Model	SettingsChange
Time	SpO_2	Pressure PC	Nebula	Mode	
	RF	Inspiration Time	Plasters		
	$EtCO_2$	Inspiration Rise Time			
	$V_{T_{exp}}$	Tidal Volume			
	$V_{T_{insp}}$				
	Frequency	Frequency			
	P^{PEAK}				
	FiO_2	FiO_2			
	PEEP	PEEP			
	I:E-ratio	I:E-ratio			
		End Inspiratory Cycle			

Table 3.3: The new schema for each recorded time step. We only include variables that are measured by all ventilators and those that are not derived from others. We also include the vendor-generic settings, the recording of an action (intervention, boolean variables), the ventilator model, and if a change in the settings has occurred.

3.4.2 Handling Missing Data Points

The episodes that we have now are gapless; from t_{start} to t_{end}, there are measurements present. However, within these measurements, a few variables can still be missing or contain extreme outliers. First, we replace all extreme outliers with **NA** values. These extreme outliers are impossible sensor readings, i.e., negative values and impossible values outside the possible range, like $FiO_2 > 100$. In this way, we treat the extreme outliers as missing data points. Typically, there are two options for handling missing

data:

Complete case analysis or listwise removal. We remove the rows that have missing data. After removal, the resulting dataset only contains complete cases.

Imputation. We fill in probable values in for the missing data.

For time series, listwise removal is not always suitable. Removing entries / time-stamped rows with missing measurements may disrupt temporal patterns for the variables that are complete. However, this is not the case at the beginning and the end of the episode. These are spots where many missing data points are present. At some episode's beginnings, the patient is admitted to the PICU, intubated, or brought back from surgery. Up to the moment when the patient is fully connected, sensor readings can be unreliable or even missing. The same happens at the end of an episode; before a gap or a mode change, some variables may be missing due to the interventions of the caregivers. In our case, we removed entries at the beginning and the end of an episode up to the first or last complete entry respectively. We could impute these values, but we suspect that some of the other values may also be unreliable within these periods. This removal process is referred to as *trimming*. Trimming does not disrupt temporal patterns, however, removing remaining entries with missing data does disrupt temporal patterns as these entries are surrounded by complete ones, and removing them causes gaps. Therefore, we did not remove these entries and imputed the remaining missing values. A limited set of episodes remained empty after trimming; this may happen when one of the columns is empty for the entirety of the episode.

In table 3.4, the percentage of missing values for each variable is shown after trimming . These percentages ranged from almost zero to 3 %. The ETCO_2 variable had the highest percentage of missing values (3.32 %).

ETCO_2	$V_{T_{\text{exp}}}$	FIO_2	Freq	HR	I:E-ratio	PEEP	P^{PEAK}	RF	SPO_2
3.32 %	0.18 %	0.00 %	0.18 %	0.64 %	2.81 %	0.18 %	0.18 %	1.93 %	1.05 %

Table 3.4: The percentage of missing data for each of the measured variables. The EtCO_2 variable contains the most missing data points.

Types of Missing Data

The suitability of imputation depends on the type of missing data. We distinguish between three types of missing data [22, 42]:

Missing at random (MAR). The chance that a data point is missing is not related to the missing information itself.

Missing completely at random (MCAR). This is a stronger version of the former, but here the missing information has also not related to all the other recorded variables.

Missing not at random (MNAR). In this case, the missing of information is directly related to the value. In surveys, MNAR often takes place at privacy-sensitive variables like income. The chance that a variable is absent may also depend on another variable.

For the MNAR case, we cannot perform imputation without modeling the underlying process, which makes imputation much more difficult. We assume, however, that this is not the case for our missing data. All the data is gathered passively (so there is no incentive to withheld information) Furthermore, we assume that during normal operation the sensors work correctly, that is, they do not store **NA** values for specific values. Of course, sensor failure occurs within our dataset. A possible cause can be a discon-

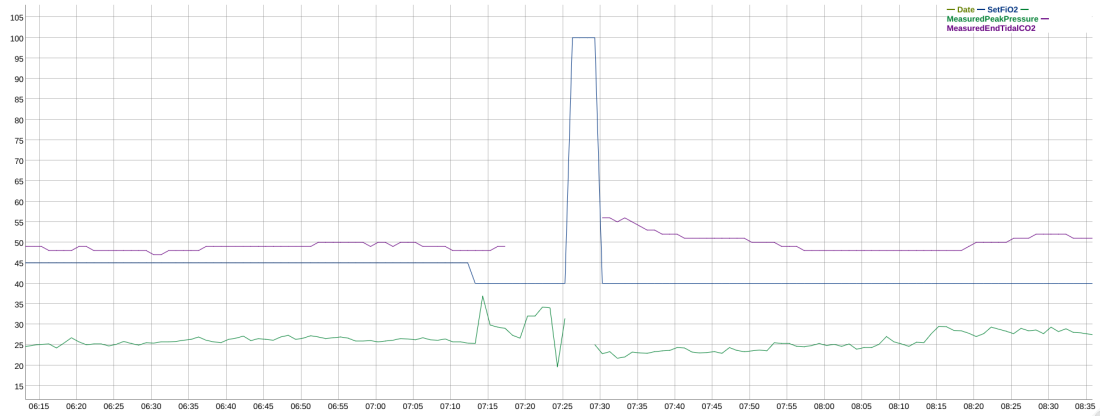


Figure 3.3: Typical behavior of missing values during the manual suction intervention. Some of the variables are omitted for clarity. The FiO_2 setting (blue) is set to 100 %, and during the intervention, missing values occur in the ventilator measured variables $EtCO_2$ (purple) and P^{peak} (green).

nection due to the movement of the patient or intervention of a caregiver. We assume that the failure does not depend on the actual value and happens at random.

Sensor failure also occurs when the caregivers perform the *manual suction* intervention; in this case, the tube is temporarily disconnected for cleaning out *sputum*. When this is done, the FiO_2 setting is set to 100 % (see figure 3.3). During the period that the tube is cleaned, the ventilator measured variables ($EtCO_2$, $V_{T_{exp}}$, Freq, I:E-ratio, PEEP, and P^{PEAK}) do not work and report **NA** values to the PDMS, which is correct behavior, as they cannot perform measurements. Note that in table 3.4, the $V_{T_{exp}}$, Freq, PEEP, and P^{PEAK} variables show similar percentages of missing values. The fact that these variables are missing could also be used as an indicator or feature. One might also argue that this is a case of MCAR. However, the VAR models we discuss in chapter 4 cannot deal with missing values, so keeping the missing values is not an option. Furthermore we cannot know what the actual values are during these interventions, so we treat the missing values caused by this intervention the same as the other missing data points.

Imputation

There are several methods suitable for time series imputation. The R-package `dplyr` [55] and `tsibble` [54] provide several basic imputation methods suitable for time series:

Last Observation Carried Forward. This imputation method provides the last available measurement as the imputed value.

Next Observation Carried Backward. This method works similarly as the previous, only the next available observation is used as the imputed value.

Linear interpolation. The last and next available values are interpolated. When these values differ, the imputed values follow the trend between these. This method was used by Bose et al. [5].

In addition to these more naïve methods, there are also some more sophisticated methods. The Kalman filter is a frequently used method for imputing missing data in a univariate time series (e.g., [24, 21]), which does take temporal characteristics into account. The dataset that we use, however, contains multiple time series. It may be possible to use the other time series to find the most suitable value for imputation.

The `MTSDI` package, which stands for *Multivariate Time Series Data Imputation* can be used to impute missing values in multivariate time series. The algorithm is based on the *Expectation-Maximization* (EM) algorithm [22]. According to Junger and De Leon [22], the method is suitable for modeling both spatial and temporal correlations. During a simulation study, the authors tested the performance of the algorithm. On a test set where the amount of missing data was at most 5 %, the algorithm yielded good results. When the amount of missing data was increased, the performance of the algorithm decreased likewise. For more information on the implementation of the MTSDI algorithm, we refer to [22]. Below we will describe how we used the MTSDI algorithm for imputing the missing data in the PICU dataset.

The MTSDI algorithm aims to keep the mean of the time series intact. The *convergence* parameter specifies how far the resulting mean can differ from the original mean. The convergence parameter has as default a very small value (0.001). This default value was not fruitful with respect to the PICU dataset; the algorithm produced a lot of extreme values that were (far) out of the ranges that were present in the original dataset. We observed that the precision of the variables in the dataset is often only one decimal; therefore, a small convergence parameter does not make sense. After taking a higher value for the convergence parameter (0.1), the results greatly improved; however, some imputed values were still out of range. During a second pass, these out of range variables are removed and again imputed using the Kalman filter method. We use the Kalman filter provided in the `imputeTS` package [35]. The combination of these two methods resulted in a successfully imputed dataset that kept the means of the dataset intact and does not contain out of range values. We performed imputation on each episode separately, as the MTSDI algorithm and the Kalman filter cannot take the episode identifier

into account. In figure 3.4, the results of this combined approach are shown on the missing values of figure 3.3.

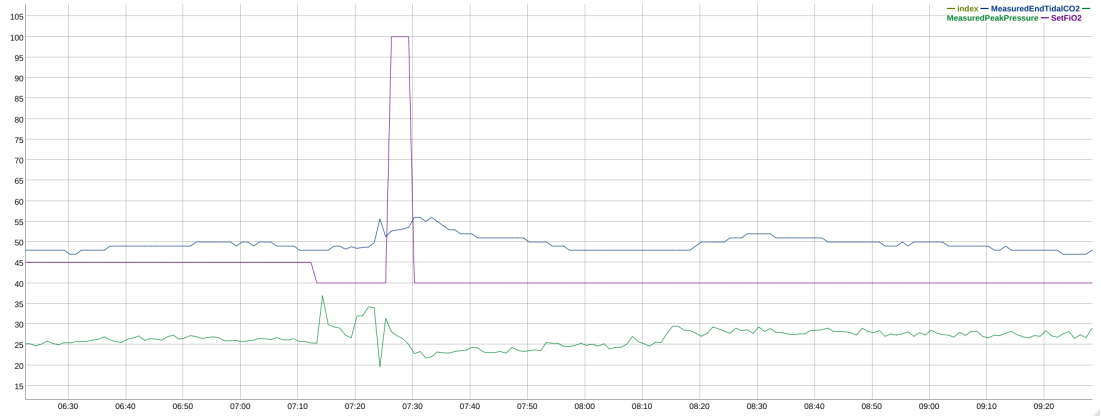


Figure 3.4: This figure shows the results of the combined MTSDI and Kalman filter imputation method. The original data are shown in figure 3.3.

3.5 Focus on a single ventilation mode

In addition to choosing to focus our research on one single ventilator type, we also decided to focus or research on one specific ventilator mode. We explained in section 3.4.1 how we divided the patients' time series into episodes for each mode because the set of used variables differs among them. As our time was limited, we decided to focus our research on one single ventilation mode. We chose the PC-IMV-Adaptive / PRVC mode, as this is the most used mode on the WKZ PICU for the Servo-i model (see table 3.5). For our further research, there are almost 2700 days (nearly 7.5 years) of Vital Signs Time Series available.

Ventilation Mode	**Days of data available**
PC-CMV-SetPoint	1460.17
PC-CSV-Adaptive, Adaptive	14.54
PC-CSV-Adaptive	65.00
PC-CSV-Servo	36.34
PC-CSV-SetPoint	2256.15
PC-IMV-Adaptive, Adaptive	49.68
PC-IMV-Adaptive	2693.78
PC-IMV-SetPoint, SetPoint	169.37
VC-IMV-Dual, Dual	2.05

Table 3.5: The used ventilation modes in the WKZ PICU

3.6 Summary

In this chapter, we have described the main characteristics of the PICU dataset and the patient population it describes. We have described the major flaws of the raw dataset; these were the mixing of different ventilators and modes, the presence of gaps, and missing values that are present in the measured variables. We have described how we adapted the dataset for further analysis. This chapter answers Research Question 1: *How do we prepare the raw dataset for further analysis?* We achieved this by first adapting the architecture by storing the time series in episodes of continuous measurements in the same ventilation mode. After that, we removed the beginning and endings of the episodes that contained missing data. Finally, we imputed the remaining missing values by using a combination of the MTSDI-algorithm and a Kalman Filter. This adapted dataset can now be used for further analysis. An extensive exploratory study of the PICU dataset will be given in the next chapter.

4

Exploratory time series analysis

4.1 Introduction

Before we make predictive models for vital signs time series, we will provide an exploratory analysis. We will introduce several analysis methods and time series modeling methods. We can use these methods to describe the Vital Signs Time Series that are present in the PICU Dataset. First, an introduction to time series analysis will be given by discussing some fundamental concepts and notation. These can then later be used to describe the time series models.

Several models can be used to describe a multivariate time series. The first model that we will introduce is *Vector Autoregression*, which models how the current measurement relies on the earlier time steps of these variables.

Secondly, the *Panel VAR* model will be discussed, which is a variation on the Vector Autoregression model. This model can be used to model data when there is also a cross-sectional characteristic present in the data set (in addition to the temporal relation). This cross-sectional characteristic can, for example, be different countries or subjects/patients. In our case, this will be the identifier of the episodes we discussed in chapter 3. One can see this identifier as a patient identifier; however, a patient can have multiple episodes.

The third model we will discuss is the *Multilevel Graphical VAR Models* (MLGVAR). This model is an exploratory analysis tool based on the *Gaussian Graphical model* (GGM). GGMs can be used to visualize relations in various types of datasets. The MLGVAR

model can analyze multivariate time series data of multiple subjects [15].

This chapter is structured as follows: First, some fundamental concepts of time series analysis will be introduced, followed by a description of these aforementioned analysis methods. Furthermore, we will describe how these methods will be applied to this dataset and how the analysis is implemented in the R language. We will conclude this chapter by discussing the results and their interpretation.

4.2 Time series analysis

One of the main goals of time series analysis is to develop models that describe the data in question. A time series is a sequence of random variables $\{x_1, x_2, \ldots, x_t\}$, for which the subscripts denote the time period of the variable x [47]. In literature, the sequences $\{x_t\}$ of random variables are referred to as stochastic processes. A realization of a stochastic process consists of its observed values [47]. The time series we discuss in this work are *discrete time parameter series*; the data is measured at a fixed discrete time interval. In practice, all time series are discrete [47].

Traditionally time series are assumed to be composed of several components (classical additive decomposition) [34]:

$$x_t = m_t + s_t + \epsilon_t \tag{4.1}$$

where

x_t : the observed series
m_t : trend
s_t : seasonal effect / effects based on the calendar
ϵ_t : residuals or error term

4.2.1 Univariate measures

Several (statistical) measures can describe a time series. These measures will be discussed in this section.

A fundamental statistical measure used in all analysis is the mean. The mean of a time series is defined as follows [47]:

$$\mu_{x_t} = E(x_t) = \int_{-\infty}^{\infty} x \cdot f_t(x)\, dx \tag{4.2}$$

where

μ_{x_t} : the mean of the series
x_t : the observed series
$E(x_t)$: the expected value of a random value in x_t
$f_t(x)$: the probability density function

Another measure is the autocovariance function. This function measures the linear dependence between two points on a series (observations measured on different times s and t). Smooth series have larger autocovariance values, also when the distance between s and t is increased. In contrast, choppy series have values that tend to zero. The autocovariance function is defined as follows [47]:

$$\gamma_X(s,t) = \text{cov}(x_s, x_t) = E[(x_s - \mu_s)(x_t - \mu_t)] \tag{4.3}$$

In time series modeling, we often include earlier or lagged variables in equations. We can define the lag or *backshift operator*, which can access the previous value of a variable, as follows [47]:

$$B(x_t) = x_{t-1} \tag{4.4}$$

This backshift operator can be repeated; this can be done in the following way:

$$B^3(x_t) = B(B^2(x_t)) = B(B(B(x_t))) = x_{t-3} \tag{4.5}$$

In addition to these measures, standard statistical measures as the variance and standard deviation can also be used to describe the series.

4.2.2 Stationarity property

The time series need to be stationary to correctly analyze and model time series. This is necessary to prevent the construction models based on *spurious correlations*. A *stationary series* is a series without trends and periodic fluctuations (e.g., seasonality). Without stationarity, it is not possible to make correct statements about, for example, the means and correlations that characterize the time series [5]. Shumway and Stoffer [47] define a *strictly stationary* time series as follows:

Definition 1. *A strictly stationary time series is a series for which the probabilistic behavior of every collection of values*

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

is identical to that of the time shifted section:

$$\{x_{t_{1+h}}, x_{t_{2+h}}, \cdots, x_{t_{k+h}}\}$$

So that:

$$\Pr\{x_{t_1} \leq c_1, \cdots, x_{t_k} \leq c_k\} = \Pr\{x_{t_{1+h}} \leq c_1, \cdots, x_{t_{k+h}} \leq c_k\}$$

This definition is often too strict; it is not always realistic to assume that real-world data conforms to this definition, even after transformation. There is also the notion of weakly stationarity time series, which is a relaxed version of strictly stationary, only specifying constant behavior over time for two probabilistic properties [47]:

Definition 2. *A* weakly stationary *time series x_t, is a finite variance process such that,*

1. *the mean value function* (*equation 4.2*) *is constant and does not depend on t*
2. *the autocovariance function* (*equation 4.3*) *depends on s and t only through their difference* $|s - t|$.

Stationarity is often linked to the presence of *unit root* in the series. Suppose we have a series x_t that can be written as an autoregression with p lags:

$$x_t = a_1 x_{t-1} + a_2 x_{t-2} + \cdots + a_p x_{t-p} + \epsilon_t. \tag{4.6}$$

Using the coefficients a, we can formulate a characteristic polynomial equation:

$$m^p - m^{p-1} a_1 - m^{p-2} a_2 - \cdots - a_p = 0 \tag{4.7}$$

When for one of the polynomial roots of these equations $m = 1$ holds, then the series has a unit root. In a stationary time series, no unit roots are present, we can find out through unit root testing if a series is stationary or not.

Several tests can be used to find unit roots in a series. One commonly employed test is the *Augmented Dickey-Fuller* (ADF) test. The null hypothesis is that a unit root is present in the series, while the alternative states that the series is (weakly) stationary.

Another test that can be used is the *Kwiatkowski–Phillips–Schmidt–Shin* (KPSS) test. Contrary to the ADF test, here the presence of unit root is the alternative. Furthermore, the null hypothesis states that a function is trend stationary, which is weaker than weakly stationary. A trend stationary process can be written in the following form:

$$x_t = f(t) + \epsilon_t \tag{4.8}$$

where

$f(t)$: a function $\mathbb{R} \to \mathbb{R}$

A process can be non-stationary and not have a unit root present. A trend stationary process is mean-reverting, meaning that when a shock has taken place, it will converge to the current mean on the trend. During this project, we will perform unit root testing using ADF tests.

4.2.3 Transforming time series

When a time series is not stationary, it can be adjusted by taking a first-order difference of the series (subtracting the current value from its lag)[1]:

$$\Delta x_t = x_t - B(x_t) \tag{4.9}$$

[1]NB: In time series literature like Shumway and Stoffer [47], instead of the Δ, the ∇ symbol is also used. However, in our work, we will use the Δ, as we will use the ∇ to denote the gradient in chapter 5.

Often, these first-order difference time series are stationary, but when this is not the case, this operation can be repeated to create a difference of order d [47]:

$$\Delta^d = (1 - B)^d \tag{4.10}$$

Another operation that is sometimes performed in time series is demeaning or centering, removing the mean of the series so that the mean of the resulting time series is 0. This is done in [15] and [48]. In this analysis, we will use *within-subject centering*. With this method the mean that will be subtracted from the time series is the mean of the subject. Another possibility is that the population mean is subtracted, but this is not something that is done in the literature referenced in this work.

4.2.4 Granger Causality

Suppose that we want to investigate if changes in the $ETCO_2$ variable may cause changes in the SPO_2 variable. We can test this using the Granger Causality test. The *Granger Causality test* is one of the most commonly employed techniques to investigate causal relations between two time series. The test is named after Clive W.J. Granger, who introduced it in his paper called "Investigating causal relations by econometric models and cross-spectral methods" (1969) [20]. It has to be noted that Granger causality cannot be equated to general causality: it is not necessary that an event in time series a, preceding an event in time series b, actually causes the event in series b. The event in a could indeed have been the cause, but there may also be another event that causes both the events in a and b. Granger causality is defined as follows [40]:

Definition 3. *A variable Granger-causes another variable if knowing the first (significantly) helps to predict the latter.*

If we want to determine if there exists a Granger causality relationship between two time series, we can follow the following procedure: Suppose that we have two time series, x_t and y_t. Then, we pose the following two hypotheses:

H_0: y_t does not Granger-cause x_t

H_1: y_t does Granger-cause x_t

First we define a regression on x using only x's own lagged values.

$$x_t = c_x + a_{x_1} \cdot x_{t-1} + a_{x_2} \cdot x_{t-2} + \cdots + a_{x_p} \cdot x_{t-p} + \epsilon_t \tag{4.11}$$

where

a : coefficients
p : the number of included lags

Then the lagged values of y are added to the equation. Again regression is performed to find suitable values for the coefficients:

$$\begin{aligned} x_t = c_x + a_{x_1} \cdot x_{t-1} + a_{x_2} \cdot x_{t-2} + \cdots + a_{x_p} \cdot x_{t-p} \\ + a_{y_1} \cdot y_{t-1} + a_{y_2} \cdot y_{t-2} + \cdots + a_{y_p} \cdot y_{t-p} + \epsilon_t \end{aligned} \tag{4.12}$$

In the regression above, the coefficients that are not significant (according to t-statistics) are removed as they do not add explanatory power to the regression measured by an F-test [56]. H_0 is rejected if at least one lagged term of y is kept in the regression equation. The null hypothesis is, therefore, only accepted if all lagged values of y are removed from the regression equation.

4.3 Analysis methods

4.3.1 Vector Autoregression

Before we discussed individual time series, however, the data set that is used in this work is multivariate. *Vector Autoregression* (VAR) models enable us to model multivariate stationary time series. In a VAR model, we perform regression on the outcomes of a vector $\boldsymbol{y}_t$, which is a $(K \times 1)$ random vector, with K denoting the number of variables that are modeled.

Using lagged values of a variable $\boldsymbol{y}_i$ in $\boldsymbol{y}$ and the other remaining variables contained in $\boldsymbol{y}$, a linear regression is formed. A VAR model has an order p, specifying the lag-length; e.g. a VAR(3) model includes the observations of $t-3$, $t-2$ and $t-1$ of all the variables in $\boldsymbol{y}$. We can describe a general VAR(p) model using the following equation:

$$\boldsymbol{y}_t = \boldsymbol{c} + \boldsymbol{A}_1 \boldsymbol{y}_{t-1} + \boldsymbol{A}_2 \boldsymbol{y}_{t-2} + \cdots + \boldsymbol{A}_p \boldsymbol{y}_{t-p} + \boldsymbol{\epsilon}_t \tag{4.13}$$

Suppose that we want to model the ETCO_2 and SPO_2 variables in terms of each other. We can then create a bivariate VAR(p) model. Suppose that y_1 describes the ETCO_2 variable, and y_2 the SPO_2 variable and that we want to include the two previous observations $(p = 2)$. We can then create a bivariate VAR(2) model. We can rewrite equation 4.13 by explicitly notating all the elements of $\boldsymbol{y}$ and $\boldsymbol{A}$:

$$\begin{bmatrix} y_{1,t} \\ y_{2,t} \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \begin{bmatrix} A_{1_{1,1}} & A_{1_{2,1}} \\ A_{1_{1,2}} & A_{1_{2,2}} \end{bmatrix} \begin{bmatrix} y_{1,t-1} \\ y_{2,t-1} \end{bmatrix} + \begin{bmatrix} A_{2_{1,1}} & A_{2_{2,1}} \\ A_{2_{1,2}} & A_{2_{2,2}} \end{bmatrix} \begin{bmatrix} y_{1,t-2} \\ y_{2,t-2} \end{bmatrix} + \begin{bmatrix} \epsilon_{1,t} \\ \epsilon_{2,t} \end{bmatrix} \tag{4.14}$$

We can also rewrite the above equation in scalar notation:

$$y_{1,t} = c_1 + A_{1_{1,1}} \cdot y_{1,t-1} + A_{1_{2,1}} \cdot y_{2,t-1} + A_{2_{1,1}} \cdot y_{1,t-2} + A_{2_{2,1}} \cdot y_{2,t-2} + \epsilon_{1,t} \tag{4.15}$$

$$y_{2,t} = c_2 + A_{1_{1,2}} \cdot y_{1,t-1} + A_{1_{2,2}} \cdot y_{2,t-1} + A_{2_{1,2}} \cdot y_{1,t-2} + A_{2_{2,2}} \cdot y_{2,t-2} + \epsilon_{2,t} \tag{4.16}$$

The c terms in these equations are intercept values (cf. regular linear regression). All the coefficients for the different lagged values are contained in the matrices $\boldsymbol{A}$. The matrices $\boldsymbol{A}_p$ have a size of $(K \times K)$ [28]. If we decompose equation 4.14 into scalar notations, we see that no coefficients are shared between the two equations. This means that the coefficients of each equation can be estimated separately [5].

Lag selection

For a VAR model, we have to find out which lag order is the best. This lag order is important as too few lags can cause autocorrelation errors and a lag size that is too big can cause overfitting [28, 5]. *Information Criteria* can be used to assess the complexity of models. For VAR models, the following four Information Criteria can be used [28]:

- Akaike Information Criterion (AIC)
- Final Prediction Error (FPE)
- Hannan-Quinn Criterion (HQ)
- Schwarz Criterion (SC), which is equivalent to the Bayesian Information Criterion (BIC)

The AIC and FPE are roughly equivalent. Lütkepohl [28] shows that AIC and FPE never underestimate the lag order, but they can overestimate the order with a positive probability. Paulsen and Tjøstheim [38] show that the probability for overestimating the lag order decreases when the dimension K increases, so when more variables are present in the multivariate time series. When $K \geq 5$, the probability for overestimating the true lag order becomes negligible.

In small samples sizes, AIC and FPE are mostly able to find the correct order, while HQ and SC will find the correct order more often than the former when the sample size is larger. However, AIC and FPE can still provide better forecasts than HQ and SC when working on larger sample sizes, as thOne way to do thisese methods are created with the intent to minimize predictive errors, sometimes sacrificing finding the correct order. HQ and SC will more often find the correct order [28].

Residual autocorrelation

After fitting a VAR model, it is desired to determine the quality of the fitted model; answering the question if the model gives a good representation of the data. is to examine the residuals [5], the ϵ term in equation 4.13. If there is information still present that could have been modeled, then this will influence the performance of the model. When there is information still present, this is referred to as *residual autocorrelation*. With the *Lagrange Multiplier Test* this can be assessed. If there is information still present, this can

be modeled with the following equation:

$$\epsilon_t = D_1\epsilon_{t-1} + \cdots + D_h\epsilon_{t-h} + v_t \tag{4.17}$$

where

D_i : some constant
v_t : white noise or the true error term
h : the number of lags

In this test the following two hypotheses are tested [28]:

H_0: $D_1 = \cdots = D_h = 0$ or in other words $\epsilon_t = v_t$

H_1: $D_i \neq 0$ for at least one $i \in \{1, \ldots, h\}$

The test that we will use to test these hypotheses is the *Breusch-Godfrey Lagrange Multiplier test* [6].

VAR Model stability

After a VAR model is fitted, we have to assess if the model is *stable*. A stable VAR model generates stationary time series. If a model is unstable, the model can have a trend or even exhibit seasonal fluctuations [28]. Assuming that the model is fitted on stationary time series, the model should, of course, also generate stationary time series.

The stability of a VAR(p)-process can be determined by calculating the *eigenvalues* of the coefficient matrix. The eigenvalues are complex numbers, so the eigenvalues have a real and imaginary part. If all eigenvalues lie within the unit circle, which has as axes the real and imaginary part of $\mathbb{C}$, then the system is considered stable (see figure 4.1). Bose et al. [5] performed visual inspection of a unit circle plot to assess stability. However, visual inspection is not needed. If we take the modulus[2] of all eigenvalues z and determine if they are smaller than 1 (so $z < 1$), the VAR-process is considered stable [39].

4.3.2 Panel VAR

Panel datasets are a class of datasets which have a cross-sectional dimension in addition to the time dimension. The PICU dataset is a panel dataset, as it contains the same set of observations for each patient, which are measured at a fixed frequency over multiple time periods. A Vector Autoregression model cannot take the cross-sectional dimension into account, whereas a *Panel VAR model* does take this into account. This model combines a *Dynamic Panel Model* (DPM) with the VAR model that we discussed before [48].

[2]The modulus r of a complex number $z = x + yi$ is $r = |z| = \sqrt{x^2 + y^2}$

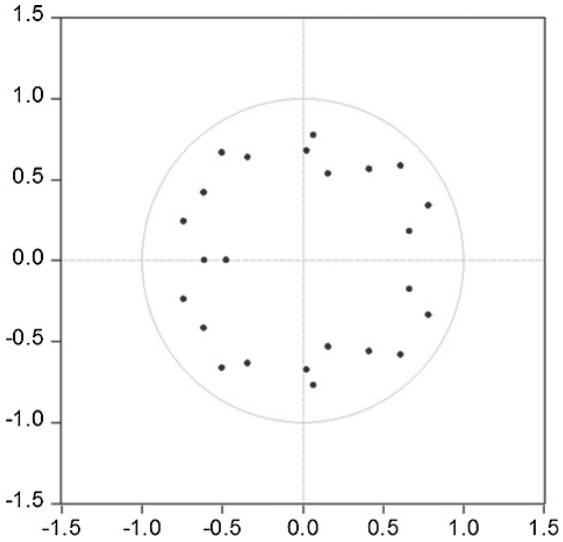


Figure 4.1: When all the roots of the VAR system lie within the unit circle, the VAR system is considered stable. The x-axis describes the **real** part of the root, the y-axis the **imaginary** part. Adapted from [5].

Panel models have the following form [57]:

$$y_{i,t} = \alpha + \boldsymbol{\beta}^T \boldsymbol{X}_{i,t} + u_{i,t} \tag{4.18}$$

$$u_{i,t} = \mu_i + \epsilon_{i,t} \tag{4.19}$$

where

i : the index of the cross-sectional dimension
t : the index of the time dimension
α : intercept
$\boldsymbol{\beta}^T$: coefficient vector
$\boldsymbol{X}_{i,t}$: the independent variables
μ_i : effects that vary between individuals i, but do not vary over time
$\epsilon_{i,t}$: effects that vary over individuals and time

In a Dynamic Panel Model, lagged variables are also added to the regression of equation 4.18 [57]:

$$y_{i,t} = \alpha + \boldsymbol{\beta}^T \boldsymbol{X}_{i,t} + \gamma y_{i,t-1} + u_{i,t} \tag{4.20}$$

where

γ : coefficient for the lagged variable

If we translate equation 4.20 to our case, we observe that there is a patient-specific "mean" μ_i. The coefficients $\boldsymbol{\beta}$ and γ are assumed to be the same for all patients. The use of multiple lags and multiple dependent variables is still missing in equation 4.20, as

this equation has only one dependent variable and only includes one lag. By combining equations 4.13 and 4.20, we can form the regression equation for a Panel VAR model [48]:

$$\boldsymbol{y}_{i,t} = \boldsymbol{\mu}_i + \sum_{l=1}^{p}(\boldsymbol{A}_l \boldsymbol{y}_{i,t-l}) + \boldsymbol{B}\boldsymbol{x}_{i,t} + \boldsymbol{C}\boldsymbol{s}_{i,t} + \epsilon_{i,t} \tag{4.21}$$

where

- $\boldsymbol{y}_{i,t}$: an $(m \times 1)$ vector of m endogenous variables for the ith cross-sectional unit at time t
- $\boldsymbol{y}_{i,t-l}$: an $(m \times 1)$ vector of m endogenous variables at time step $t - l$
- $\boldsymbol{x}_{i,t}$: a $(k \times 1)$ vector of k predetermined variables
- $\boldsymbol{s}_{i,t}$: an $(n \times 1)$ vector of n exogenous variables
- $\boldsymbol{A}_l$: an $(m \times m)$ matrix containing the coefficients for lag-l
- $\boldsymbol{B}$: an $(m \times k)$ matrix containing the coefficients for the k predetermined variables
- $\boldsymbol{C}$: an $(m \times n)$ matrix containing the coefficients for the n exogenous variables
- $\boldsymbol{\mu}_i$: the subject-specific mean

This model allows the use of *predetermined* and *exogenous* variables. Exogenous variables are variables that are not influenced by other variables. Examples of exogenous variables are age and weight, as the other variables do not influence them; these variables even remain constant over time in our dataset. Predetermined variables have a weaker form of exogeneity; their value is not correlated to previous values, but a sudden change may affect future values [58]. However, for the PICU dataset, there are no exogenous variables for which this relaxation is needed. All the other variables are *endogenous* variables. For the estimation of the coefficients of the Panel VAR model, we refer to Sigmund and Ferstl [48].

4.3.3 The Multilevel Graphical VAR Model

In the article "The Gaussian Graphical Model in Cross-Sectional and Time-Series Data", by Epskamp et al. [15], a novel graphical exploratory data analysis method is introduced; the *Multilevel Graphical VAR* model[3]. This model is based on the *Gaussian Graphical Model* (GGM). A GGM is an undirected graph or network with nodes and edges. The nodes in the network represent the different variables, and the edges describe the correlation coefficients between them. In short, a GGM shows which variables can possibly be used to predict one-another. An example GGM is shown in figure 4.2. GGM's are designed to be used in three kinds of datasets [15]:

1. Cross-sectional datasets, where no temporal ordering is present, and every subject is measured only once. These can be modeled using a basic GGM.

[3]Epskamp et al. [15] refer to this model as the *multilevel VAR model* (mlVAR). However, in this work, we will add the word *Graphical* to emphasize that this model can be used to visualize correlations.

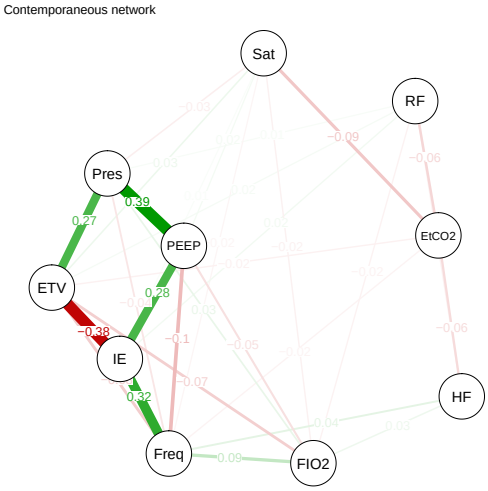


Figure 4.2: A GGM based on the PICU dataset (The Contemporaneous Network)

2. Temporally ordered datasets of one subject, ($n = 1$ time series). This is modeled through a Graphical VAR model.
3. Temporally ordered datasets of multiple subjects($n \geq 2$ time series). This is modeled through a Multilevel Graphical VAR model.

We will describe the process of deriving a GGM. First, the basic GGM for cross-sectional data is introduced. This model will later be extended for modeling time series data. In this section, we will largely follow the contents as presented by Epskamp et al. [15].

Deriving a GGM

The edges in a GGM describe the *partial correlation coefficients* (PCC's). A PCC denotes the correlation between two variables after conditioning on all the other variables in the dataset [15]. Suppose we have a random vector $\boldsymbol{y}_C^\top = [\boldsymbol{Y}_{C_1}, \boldsymbol{Y}_{C_2}, \ldots, \boldsymbol{Y}_{C_m}]$ with $\boldsymbol{y}_c$ as its realization. We assume that $\boldsymbol{y}_c$ is centered / demeaned (see 4.2.3). C denotes a case, in the case of a cross-sectional dataset this C can denote the subject identifier, or C could denote a temporal ordering or a combination of these. Next, we define a matrix $\boldsymbol{\Sigma}$, which will denote the variance-covariance matrix. We also define a matrix $\boldsymbol{K}$, which is the inverse of $\boldsymbol{\Sigma}$, that is $\boldsymbol{K} = \boldsymbol{\Sigma}^{-1}$. The elements of $\boldsymbol{K}$, (κ_{ij}), are related the partial correlation coefficients of the Gaussian Graphical Network as follows:

$$\text{Cor}(\boldsymbol{Y}_i, \boldsymbol{Y}_j \mid \boldsymbol{y}_{-(i,j)}) = -\frac{\kappa_{ij}}{\sqrt{\kappa_{ii}}\sqrt{\kappa_{jj}}} \tag{4.22}$$

where

$\text{Cor}(\dots)$: The partial correlation between $\boldsymbol{Y}_i$, and $\boldsymbol{Y}_j$
$\boldsymbol{y}_{-(i,j)}$: all the values of $\boldsymbol{y}$ without the elements i and j
κ_{ij} : the elements of matrix $\boldsymbol{K}$

We can represent the PCCs in a graph. We visualize each variable $\boldsymbol{Y}_i$ as a node. For each significant PCC, we draw an edge between the nodes. The edges all have a weight, which is equal to the PCC. In practice, none of the partial correlations will be exactly zero. Some PCCs are significant and others are not. We can use thresholding and regularization to force some values in the matrix to zero. The resulting graph is more sparse and only shows relevant correlations.

GGM on Cross-sectional data (independent cases)

Because in cross-sectional datasets only one observation per subject is available, we cannot estimate subject-specific means or GGM networks that are based on a single subject. All the rows in the dataset are assumed to share the same multivariate distribution:

$$\boldsymbol{y}_p \sim N(0, \boldsymbol{\Sigma}) \tag{4.23}$$

We can make similar statements when we have an $n = 1$ time series dataset when there is no temporal relation between measurements (for example, when time interval between measurements is very large).

$$\boldsymbol{y}_t \sim N(0, \boldsymbol{\Sigma}) \tag{4.24}$$

The Temporal Network

We have seen in section 4.3.1 that we can use regular VAR models to model the effect that consecutive earlier measurements have on the current measurement. Let us adapt the VAR equation of 4.14 into the current notation for a lag order of $p = 1$.

$$\begin{aligned} \boldsymbol{y}_t &= \boldsymbol{B}\boldsymbol{y}_{t-1} + \boldsymbol{\epsilon}_t \\ \boldsymbol{\epsilon}_t &\sim N(0, \boldsymbol{\Theta}) \end{aligned} \tag{4.25}$$

The matrix $\boldsymbol{B}$ fulfills the same role as matrix $\boldsymbol{A}$ from equation 4.14. From matrix $\boldsymbol{B}$ we can derive a temporal network. This network is not a GGM, as the temporal direction does not go backwards (the future cannot influence the past). Therefore, this network is directed and not a GGM (as GGMs are by definition undirected graphs).

A *temporal network* is thus a directed graph and created by drawing weights as arcs between the corresponding variables. The weights are derived from the regression coefficients β_{ij} in $\boldsymbol{B}$. The weights determine the size of the arcs. An arc in the temporal network indicates that the origin variable can be used to predict the target variable node.

We visualize the current measurement and lagged measurement as a single node. This fact results in the presence of *self-loops* in the graph.

Graphs are commonly encoded as matrices in which the row indicates the node of origin and the column the node of destination. The matrix $\boldsymbol{B}$ is transposed ($\boldsymbol{B}^\top$) to achieve this. Because of the temporal ordering, arcs between different variables are always from a lagged variable to the current measurement and not the other way around.

Equation 4.25 describes a $p = 1$ model. Suppose that we want to include more lagged predictors in the model. Let $L \subseteq \mathbb{N}$ be a set of lags that are included in the model. As each lag-l has its own matrix $\boldsymbol{B}_l$, each lag has its own temporal network.

In section 4.2.4, we described the Granger Causality test. When we want to determine a causal relation between two variables using Granger-causality testing, we can include these two variables in the comparison. The temporal network, however, gives us more information. When there is a correlation between two variables, this is conditioned on all the other variables in the network. An arc $e : A \to B$ (for $A, B \in \boldsymbol{y}_t$) when A helps predicting B after controlling on all other variables in $\boldsymbol{y}_t$. This information could indicate potential causal relations between these variables, assuming that no other (unobserved) variables than those in $\boldsymbol{y}_t$ cause those variables.

The Contemporaneous Network

In addition to the temporal network, we can make a GGM based on the *contemporaneous effects*. The residuals $\boldsymbol{\epsilon}_t$ of equation 4.25 can be used to derive the contemporaneous network. The variance-covariance matrix of $\boldsymbol{\epsilon}_t$, which is denoted as $\boldsymbol{\Theta}$ (see equation 4.25), can be inverted to obtain a matrix $\boldsymbol{K}^{(\boldsymbol{\Theta})}$ (that is, $\boldsymbol{K}^{(\boldsymbol{\Theta})} = \boldsymbol{\Theta}^{-1}$). By using equation 4.22, we can use the elements $\kappa_{ij}^{(\boldsymbol{\Theta})}$ to calculate the partial correlations between variables i and j after controlling for the temporal effects. This networks shows how i and j are correlated, or can predict each other within the same time step; if we know that the ETCO_2 and SPO_2 have a significant coefficient in this network, we can use the ETCO_2 value at time step t to "predict" the SPO_2 value at the same time step t, or vice versa.

Graphical VAR (GVAR)

A *graphical VAR model* (GVAR) is a model that is composed of both the temporal network and the contemporaneous network. When there are no temporal effects (when all the measurements are independent of time), this model is equivalent to the GGM model for cross-sectional data. This is the case when $\boldsymbol{B} = \boldsymbol{O}$, where $\boldsymbol{O}$ is a matrix containing only zeros.

A fully saturated GVAR model, in which all nodes are connected to all nodes, can be estimated as follows: first a regular VAR model is fitted (just as in 4.3.1). The variance-covariance matrix $\boldsymbol{\Theta}$ of the residuals is then inverted to acquire the contemporaneous model. There is also a step-wise model selection that can be used to estimate sparse GVAR models. *LASSO estimation* (least absolute shrinkage and selection operator) can

be used to estimate both the temporal and contemporaneous network structure. LASSO estimation reduces the model's fit on the training set by only using a subset of the set of the predictors. This may seem counterintuitive, but this can reduce the risk of overfitting. A good set of predictors can be found using step-wise selection, according to the Extended Bayesian Information Criterion (EBIC).

The Multilevel Graphical VAR Model (MLGVAR)

Before, we described the Graphical VAR model that can model an $n = 1$ time series. However, the PICU dataset contains data of multiple patients. The GVAR model can be adapted to be able to deal with multiple subjects/persons. This model allows that the number of measurements can be different for each person p. The adapted regression equation is defined as follows [15]:

$$\begin{aligned} \boldsymbol{y}_{[t,p]} &= \boldsymbol{\mu}_p + \boldsymbol{B}_p(\boldsymbol{y}_{[t-1,p]} - \boldsymbol{\mu}_p) + \boldsymbol{\epsilon}_{[t,p]} \\ \boldsymbol{\epsilon}_{[T,p]} &\sim N(0, \boldsymbol{\Theta}_p) \\ \boldsymbol{\Theta}_p^{-1} &= \boldsymbol{K}_p^{(\boldsymbol{\Theta})} \end{aligned} \tag{4.26}$$

where

$\boldsymbol{\mu}_p$: the stationary mean vector of subject or person p
$\boldsymbol{B}_p$: the subject-specific temporal network

We need to combine the information found in the individual networks to acquire insight into the structure of the global network of the patient population. By multilevel modeling (investigating models at a second level), we can obtain matrices $\boldsymbol{B}_*$ and $\boldsymbol{K}_*^{(\boldsymbol{\Theta})}$ that represent the temporal and contemporaneous model of a random subject. These are related to the individual networks as follows:

$$\begin{aligned} E(\boldsymbol{\mu}_P) &= 0 \\ E(\boldsymbol{B}_P) &= \boldsymbol{B}_* \\ E(\boldsymbol{K}_P^{(\boldsymbol{\Theta})}) &= \boldsymbol{K}_*^{(\boldsymbol{\Theta})} \end{aligned} \tag{4.27}$$

The matrix $\boldsymbol{B}_*$ encodes the average partial correlation coefficients of the population, creating a *fixed effects* model, which is similar to the Panel VAR model discussed in section 4.3.2. The differences $\boldsymbol{B}_P - \boldsymbol{B}_*$ are referred to as the *random effects*. Similarly, the $\boldsymbol{K}_*^{(\boldsymbol{\Theta})}$ encodes the population's contemporaneous network.

In addition to the two networks that were also present in the Graphical VAR model, a third network can be estimated based on the means $\boldsymbol{\mu}_P$. This network is referred to as the *between-subjects network*. We can invert the variance-covariance matrix $\boldsymbol{\Omega}$ of stationary means $\boldsymbol{\mu}_P$ in the following way:

$$\begin{aligned} \boldsymbol{\mu}_P &\sim N(0, \boldsymbol{\Omega}) \\ \boldsymbol{K}^{(\boldsymbol{\Omega})} &= \boldsymbol{\Omega}^{-1} \end{aligned} \tag{4.28}$$

This network shows how the variables predict each other on average [15].

Estimation of a multilevel VAR for $n \geq 2$ time series

A multilevel VAR model can be estimated in two steps [15]:

1. First, univariate regression models are estimated. The regressions contain all lagged variables and the subject sample means of the variables as between-subjects predictors. The lagged variables are within-subject centered (see equation 4.26). The subject sample means can be used to estimate a between-subjects network.
2. The contemporaneous model can be estimated using the results of the previous step by applying univariate regression on all the residuals $\boldsymbol{\epsilon}$ of step 1.

Regularization takes place in the form of thresholding, in other words, removing all effects that have a very small coefficient. In the contemporaneous networks and between subjects network, there are two ways to perform regularization, as there are two p-values that denote the significance of an edge; from $A \rightarrow B$ and vice versa ($B \rightarrow A$). An edge can be retained in the network if one of these values is significant (this is called the "or-rule", or only if both of them are significant ("and-rule").

Interpretation

Epskamp et al. [15] describe how each of these networks can be interpreted. They note, however, that these networks may only be used to create hypotheses for further research. No hard (causal) relations can be established, but the results of these models can give some indication for a starting point of further research. Below, we will discuss the interpretation of the model's networks according to Epskamp et al. [15]:

The temporal networks can be interpreted as a network that directly describes causal influences as they happen over time, visualizing the correlations between a past time step $t - h$ and the current time step t. When the model is based on multiple lags, this can be hard to interpret, as there is a separate models for each lag. The sign of the corelation coefficient can even change between time steps.

The between-subjects network describes how variables predict each other on average, describing global relations between variables. These can be causal and reflect the temporal structure, but this may not always be the case. Take this example as provided by Epskamp et al. [15]: If people are forced to exercise, they will have a higher heart rate after starting the exercise. This will probably result in an arc with a positive coefficient in the temporal network between the variables *exercise* and *heart rate*. In the between-subjects model, however, a negative correlation exists between these variables, as patients who exercise more often have a better condition, and a lower heart rate on average than patients who exercise less.

The contemporaneous network shows the correlations between variables that are still present in the residuals. The contemporaneous model could give some indication of what interactions take place between variables between the last lag and the current measurement. Strong correlations in this network could be an indication that the measurement frequency should be increased, as there are possible temporal relations present between the lags' measurements that could have been modeled through a temporal network if the frequency was higher.

4.4 Research methodology

We have described three different methods that can be used to analyze the PICU dataset:

- Traditional *subject-specific* Vector Autoregression (VAR) model
- Panel Vector Autoregression model
- Multilevel VAR model

In this section, we will describe how we can use these methods to analyze the dataset. First, we will analyze the dataset using traditional VAR models. After that, we will analyze the results of the Panel VAR model and the Multilevel Graphical VAR model. For all of these models and tests, we will illustrate how these methods can be used in the R programming language.

4.4.1 VAR Methodology

In this section, we will describe how we analyzed the dataset using Vector Autoregression models. We will do this according to the methodology presented by Bose et al. [5]. In their work, the authors examined the Vital Sign Time Series using a VAR model and assessed causal relationships with Granger Causality testing. This paper (based on the work in Lütkepohl [28]) presents a clear guide on how to perform such an analysis. Their methodology consists of the following steps:

1. Determine the stationarity of the Vital Sign Time Series
2. Determine the lag-length p using selection based on an information criterion
3. Build a VAR model with p lags
4. Assess Residual Auto correlation (with the Lagrange Multiplier test)
5. Assess the stability of the VAR Model
6. Perform Granger Causality testing do determine causal relations between time series

Bose et al. [5] determined causal relations between time series prior to a Cardio-Respiratory Instability event (violations of the alarm thresholds of either the heart rate, SpO_2 or respiratory frequency vital signs). As VAR models are subject-specific, they determined the causal relations for each patient separately. The estimation procedure is not computationally complex; this enables to estimate the VAR models for a large collection of series. In the following sections, we will follow the steps described above and show how these can be performed in R.

Stationarity testing

Stationarity testing is done using the Augmented Dickey-Fuller Test, which is provided in the `tseries` [53] as the `adf.test( ... )` function. The test is performed on every column containing a measured variable and its first-order difference. As we expect that the presence of a unit root within an individual univariate series does not mean that all other instances of that variable also share the presence of unit root, this test will be performed on all the time series present in the dataset.

Lag order

Subsequently, we will determine the ideal lag-length. The function `VARselect( ... )` accepts a `xts` [46] multivariate time series object [40]. Given an `l.max` it will determine the ideal lag order, according to the AIC, HQ, SC and FPE criteria (see section 4.3.1):

```
1 varselect ← VARselect(prvc.subset.xts, lag.max = l.max, type = "const")
```

The `type = "const"` is chosen, because that is the default for VAR models where no seasonality (or other non-stationary properties) and/or exogenous variables are present. As we assume that only stationary time series are present, this setting is suitable. The object `prvc.subset.xts` includes all the measured variables of a single patient's uninterrupted time series. We take for `l.max ← 30`, as including more lags will make the computation only longer and will limit the usefulness of some smaller series, as there are series present in the dataset that have a length of two hours. We will assess all subsequent test based on the results of both the AIC and SC criteria.

VAR Model estimation

According to the results of the lag-order selection test, an individual VAR(p) model for each of the patient's uninterrupted series will be estimated. The `VAR( ... )` function's signature is very similar to `VARselect( ... )` function's signature, but instead of estimating the lag order, this one can be used to estimate the VAR model of the current series.

```
selectp   ← varselect$selection[[1]] # 1 → AIC, 2 → HQ, 3 → SC, 4 → FPE
varm      ← VAR(prvc.subset.xts, p = selectp, type = "const")
```

The function creates a `varest` object, which contains all the coefficients of the estimated equations that describe the process. This object can be used for further testing and Granger Causality testing, which we will describe below. It has to be noted that no variables can be introduced that remain constant over time, such as the settings. These variables have a variance of zero, and this causes problems as this makes one of the matrices in the model to be singular (non-invertible).

Residual autocorellation testing

Residual testing can be performed to assess whether a model describes the data well enough, or that we should increase the order p. We can perform the Breusch - Godfrey test, which is implemented in the `serial.test( ... )` function, which is also included in the `vars` package [40]. Recall the autocorrelation hypotheses described in section 4.2.1 and equation 4.17. Let `l.max` $\leftarrow h$ from that equation, which denoted the furthest autoregressive term in history.

```
> serial.test(var.est, type="BG", lags.bg = l.max)

    Breusch-Godfrey LM test

data:  Residuals of VAR object var.est
Chi-squared = 1253.5, df = 1000, p-value = 7.065e-08
```

In this example, H_0 is rejected, as $p < 0.05$. This result means that in the residuals of this VAR model, autocorrelation is still present, which might indicate that the lag order of this model should be increased.

Stability testing

We can assess the stability of the VAR model by looking at the eigenvalues of the coefficient matrix using the `roots( ... )` function.

```
# Getting the roots of describing polynomial
rs.complex  ← roots(var.est, modulus = FALSE)
rs.modulus  ← roots(var.est, modulus = TRUE)

```

```
visualInspect(rs.complex) # See Appendix
# Given a VAR model, var.stable calculates if it is  stable
var.stable ← function(varmodel){
    rs ← roots(varmodel, modulus = TRUE)
    return(all(rs < 1))
}
```

The function `var.stable( ... )` returns **TRUE** when the model is stable. The function can easily be applied to a list of VAR models.

Granger Causality testing

In the former steps, all Vital Signs Time Series were included. However, for Granger Causality testing, we will only compare two VSTS. We use the lag order that was determined earlier for creating a bivariate VAR model. Let $c_1, c_2 \in K$ where K is a set containing all the VSTS.

```
prvc.colselect ← prvc.subset.xts[,c(c1,c2)]

varm      ← VAR(prvc.colselect, p=selectp, type="const")
test.a    ← causality(varm, cause=c1.name)
test.b    ← causality(varm, cause=c2.name)
```

Using the `causality( ... )` function we can test if c_1 causes c_2 and vice versa. The function gives an output like the following:

```
$Granger

    Granger causality H0: dHF do not Granger-cause dRF

data:  VAR object prvc.colselect
F-Test = 142.32, df1 = 1, df2 = 954, p-value < 2.2e-16


$Instant

    H0: No instantaneous causality between: dHF and dRF

data:  VAR object prvc.colselect
Chi-squared = 44.232, df = 1, p-value = 2.917e-11
```

In this example, H_0 (no Granger Causality) is rejected, as $p < 0.05$. After establishing the presence of a Granger-causal relationship between all pairs of variables for all patients, the results can be aggregated to get an overview of the global causal relations between the variables.

4.4.2 Panel VAR modeling

The R package `panelvar` [48], that implements the PanelVAR model we discussed in section 4.3.2, provides several methods for the estimation of a Panel VAR model:

`pvarfeols( ... )`: A fixed-effects Panel VAR estimator using an Ordinary Least Squares estimator

`pvargmm( ... )`: A fixed-effects Panel VAR estimator using a Generalized Method of Moments (GMM) estimator

The `pvargmm( ... )` function can estimate the coefficients for equation 4.21. Unfortunately, the required computation time and (more importantly) the amount of required memory make the use of this function impossible on (subsets of) this dataset. However, we can use the `pvarfeols( ... )`:

The data structure that was used before has to be adapted in order to be usable in this model. The DateTime column is adapted for each patient to range from $1, 2, \ldots, T$, where T is the index of the last recorded entry of the patient. The `PanelID` is the identifying index for each episode of the patient. Each variable is demeaned (like in the Multilevel VAR model). We provide a list of dependent variables and exogenous variables. This model cannot estimate the coefficients for predetermined variables, but we concluded earlier that there are no variables that fall into this category, so this does not pose any problem. However, during experiments, we learned that we cannot include static variables like age and weight. Therefore, we will only include the measured vital sign variables, just as in the traditional VAR models. This function can then be used as follows:

```
prvc.pvar ←
  pvarfeols(
    dependent_vars = dep_vars, # e.g., c("HR", "ETCO2", ... )
    exog_vars = exo, # e.g., c("AGE", "WEIGHT"). Not used in our work
    lags = 30,
    transformation = c("demean"),
    data = panel.subset,
    panel_identifier = c("PanelID", "Time")
  )
```

4.4.3 Graphical and Multilevel Graphical VAR Models

Epskamp et al. [15] list two R packages that can be used to create the graphical models of time series data.

`graphicalVAR`, for $n = 1$ time series modeling and pooled estimation of a generic model for $n \geq 2$ [13]

`mlVAR`, for $n \geq 2$ time series modeling [14]

Creating graphical VAR models using the `graphicalVAR` package turned out to be more difficult than initially expected, and ultimately not even feasible. While creating a Lag-1 model is possible, it was not possible to take more lags into account. According to the package's documentation, it should be possible to supply a vector of lags to the estimation function [13]. However, after supposedly successfully estimating the model, something goes wrong in saving the results in the data structure. As the function ungracefully terminates with an error, the intermediate models' results are lost. After reverting to a Lag-1 model, this error disappears. Resolving this problem was not deemed feasible during the scope of this project, and it was decided not to investigate the dataset using this package any further. As with the VAR models, we cannot include the settings in the model, as these models still require a non-zero variance.

We can, however, estimate a Multilevel Graphical VAR Model. This model can be estimated with the `mlVAR` package as follows:

```
mlvar.res ← mlVAR(
  prvc.subset,        # Data set
  mvars,              # A chr vector containing included variables
  "ID",               # Specifying the Panel ID, the ID of the episode / patient
  lags = c(1:30),     # A vector containing the lags that have to be included
  temporal = "fixed", # For fixed effects modelling
  contemporaneous
    = "orthogonal",   # To obtain the contemporaneous network
  nCores = 1          # The number of cores.
)
```

The completion of the estimation procedure takes much time; however, this procedure can be sped up by parallelization, but this has a cost: When a (big) dataset is analyzed, the dataset and intermediate results are not shared between the processes, but instead, each process receives its own copy. For estimating the fixed effects model (Temporal and Between-Subjects) model, this is still feasible in some cases. However, when the contemporaneous network needs to be determined, this becomes problematic. The intermediate results have already consumed a tremendous amount of memory. These

results are, for this phase, again distributed over the cores, exceeding the memory limits of our equipment. It is, therefore, better to sacrifice computation speed to prevent memory shortage. For instance, the model that is estimated using the code above saved as an RDS-file (a compressed serialized R object), consumes 3.4 GiB of space. However, when the model is loaded into R, it will be expanded, causing that session to consume approximately 12 GiB of RAM. On the used computing equipment, this was, however, feasible in controlled conditions. We cannot consider the whole dataset with this package; therefore, we will draw a random sample of 100 episodes from the dataset.

A fitted `mlVAR` model contains the three different networks discussed earlier. These can be retrieved by using the `plot( ... )` function on the object, by using `type = "temporal"`, `"contemporaneous"`or `"between"` respectively. When the temporal network is plotted, the Lag-1 model is plotted by default[4]. The other lags can be accessed through the function call `plot( ... , type = "temporal", lag = 5)`. Note that 5 does not indicate lag-5, but the 5th lag in the original function call (e.g. lag-25 in `c(1,5,10,20,25,30)`). It is therefore possible to skip lags in the model.

By setting the `temporal` parameter of the estimation function to `"orthogonal"` a random effects model can be determined. This is, however, not feasible on the PICU dataset, as the dataset's dimensions cause the model to consume too much memory. In line with the Panel VAR model, we will limit our experiments to fixed effects modelling.

It is also possible to create unique models for each subject by using the following function call:

```
model ← mlVAR( ... , temporal = "unique", contemporaneous = "unique")
```

However, in this case no between-subjects model is created, as the means of the variables are not included as predictors. These individual models can be accessed by supplying an additional parameter in the plotting function call:

```
plot( ... , type = "contemporaneous", subject = 12)
```

4.4.4 Patient subset

For some of the models, we will perform the analysis on a subset of the dataset. This subset is based on age; we will only include patients that were younger than 30 days at the time of admission. Clinically, it might be better to select patients on symptoms or a common disease, but we cannot easily do this as this information is not recorded in the dataset. For the VAR model and Panel VAR model creation, we will only perform this on the subset. The Multilevel Graphical VAR models will be based on randomly drawn samples of both the whole dataset and the under 30 days subset.

[4] or the first model that was supplied in the `mlVAR( ... , lags = c( ... ), ... )` function call.

4.5 Results

In this section, we describe the exploratory analysis of the PC-IMV-Adaptive part of the dataset. From this part, we only retain the episodes which are longer than 200 minutes, and these were imputed according to the procedure in 3.4.2. The remaining dataset in question contains 1753 episodes. The average duration of an episode is 22 hours and 48 minutes. There is data from 526 unique patients, so approximately three episodes per patient. In table 4.1, we listed the variables that we intended to include in our analysis.

Measured Variables	**Settings**
End-tidal CO_2 ($ETCO_2$)	
Expiratory Tidal Volume ($V_{T_{exp}}$, ETV)	Tidal Volume
Fraction of inspired Oxygen (FIO_2)	FIO_2
Respiratory Frequency (Ventilator) (Freq)	Respiratory Frequency
Respiratory Frequency (Monitor) (RF)	
Heart Rate (HF)	
I:E – Ratio (IE)	I:E – Ratio
PEEP	PEEP
Peak Pressure (P^{PEAK}, PRES)	
Saturation (SPO_2, SAT)	
	Inspiration Time
	Inspiration Rise Time

Table 4.1: The variables that are considered in the exploratory analysis. The settings and measured variables that directly correspond to each other are placed on the same line. We only briefly consider the settings, as they cannot be included in the VAR models.

However, during initial experimentation, we found that the settings cannot be included in the VAR models. The settings remain mostly constant over time. When we included the settings, somewhere in the estimation procedure a non-invertible matrix was created. Therefore, the VAR models could not be estimated.

In table 4.2, the changes in measurements and settings are compared. Also the average standard deviation is shown for all the measured variables. The settings that are changed the most are the FIO_2 and the Tidal Volume. The fact that these variables remain constant causes these non-invertible matrices. In further experiments, we will therefore, not include the settings. However, some of these settings are also measured, so not all information is lost.

When we look at standard-deviation in table 4.2c, we see that the many variables do not see big changes over time. The I:E-ratio is very stable, whereas the Heart Rate and Expiratory Tidal Volume deviate much more from the mean.

	FiO_2	TV	PEEP	IE	FREQ	INSP	INSPR
n unique values	3.53	2.12	1.67	1.14	1.75	1.14	1.10

(a) Settings

	$EtCO_2$	ETV	FiO_2	FREQ	HF	IE	PEEP	PRES	RF	SAT
n	65.96	71.12	44.35	114.63	62.62	68.57	36.09	142.06	52.07	86.24

(b) Measurements

	$EtCO_2$	ETV	FiO_2	FREQ	HF	IE	PEEP	PRES	RF	SAT
SD	3.31	17.34	7.98	3.69	11.33	0.17	0.77	3.26	3.78	1.93

(c) Measurements - Standard Deviation

Table 4.2: These values indicate how many changes occur for a variable within an episode. There is a large difference between the measurements and the settings. The standard deviation in (c) is also the average standard deviation within an episode.

4.5.1 Determining stationarity

Before a VAR model can be created, we first need to determine if the VSTS are stationary. Stationarity tests are performed using the Augmented Dickey-Fuller test provided by the `tseries` package in R (`adf.test`). This function was applied using a standard lag order criterion for each time series y, which grows as y becomes longer [53]:

$$p(y) = \left\lfloor (\text{length}(y) - 1)^{\frac{1}{3}} \right\rfloor \tag{4.29}$$

We performed stationarity tests for each measured time series and its first-order difference. The results of this test are shown in table 4.3. As can be concluded from these results, most time series become stationary after taking their first order difference. The heart rate (HF) series are more often non-stationary than stationary; a sample graph is shown in figure 4.3. However, for the FiO_2 series, the results differ slightly; 20 out the 1753 series were still non-stationary. A manual intervention, the suction of the tube, may cause this non-stationarity. For the duration of the procedure, FiO_2 is raised to 100 %. Afterward, the levels are restored to the previous level. These spikes in the FiO_2 levels may persist after taking the first-order difference.

4.5.2 Determining the lag order

The lag order is determined using the `VARSelect( ... )` function. After running this function on all individual series, the results were aggregated in table 4.4. There is a remarkable difference between the AIC / FPE and the SC and HQ criteria. The latter two prefer a smaller model; $p = 2$ is the most preferred for the Schwarz Criterion, and

Variable	Stationary	Non Stationary	Variable	Stationary	Non Stationary
Δ^3FIO2	1742	9	EtCO2	1130	623
Δ^2FIO2	1738	13	ETV	1668	85
ΔEtCO2	1749	4	FIO2	1534	217
ΔETV	1753	0	Freq	1361	388
ΔFIO2	1731	20	HF	849	904
ΔHF	1750	3	IE	1563	188
ΔIE	1748	3	PEEP	1526	227
ΔPEEP	1751	2	Pres	1504	249
ΔPRES	1750	3	RF	1499	253
ΔRF	1750	2	Sat	1281	471
ΔSat	1749	3			

Table 4.3: Stationarity testing on the PC-IMV-Adaptive subset, note that some of the columns do not add up to 1753, due to the fact that some of the stationarity tests failed.

for the Hannan Quinn criterion $p = 4$ is the most preferred. The choice for $p = 6$ is based on the observation that a majority of the patients is covered by this lag order (i.e., no underfitting takes place). On the other hand, for the Akaike Information Criterion, the preferred p is 30. After increasing the maximum lag order to 35, the preferred order also goes to 35. After another increase to 40, the preferred p again becomes 40. A higher lag order seems to increase the predictive performance, but as said earlier, this may not always be practical. Decreasing the number of included VSTS reduces the number of lags. If we perform lag order selection during bivariate Granger Causality testing, the resulting order is much lower according to the AIC, usually around 9.

4.5.3 VAR model creation

The VAR models were estimated using the `VAR( ... )` function for a sample of 100 patients, with $p = 30$ (AIC) and $p = 6$ (SC, HQ). All of the time series are first-order differenced to ensure stationarity of the series. A sample output for the Oxygen Saturation for a single patient is shown in appendix B. Four out of six of dSat lags are significant. Other variables with a significant coefficient were dFIO2 and dPEEP. Considering all the equations and variables, the number of coefficients is enormous; it is impossible to make some general remarks about them.

We performed the Breusch-Godfrey LM test on all the estimated VAR models to test for residual autocorrelation. For both $p = 6$ and $p = 30$ there is residual autocorrelation present in all the models. When there is residual autocorrelation, p is usually increased. However, increasing the lag order from 6 to 30 did not result in a decrease of residual autocorrelation.

In addition to testing for residual autocorrelation, we also determined if the VAR models were stable. The VAR(6) models were mostly stable. Out of the 100 VAR(6) models, four were unstable, so there were 96 stable models. However, this was not the

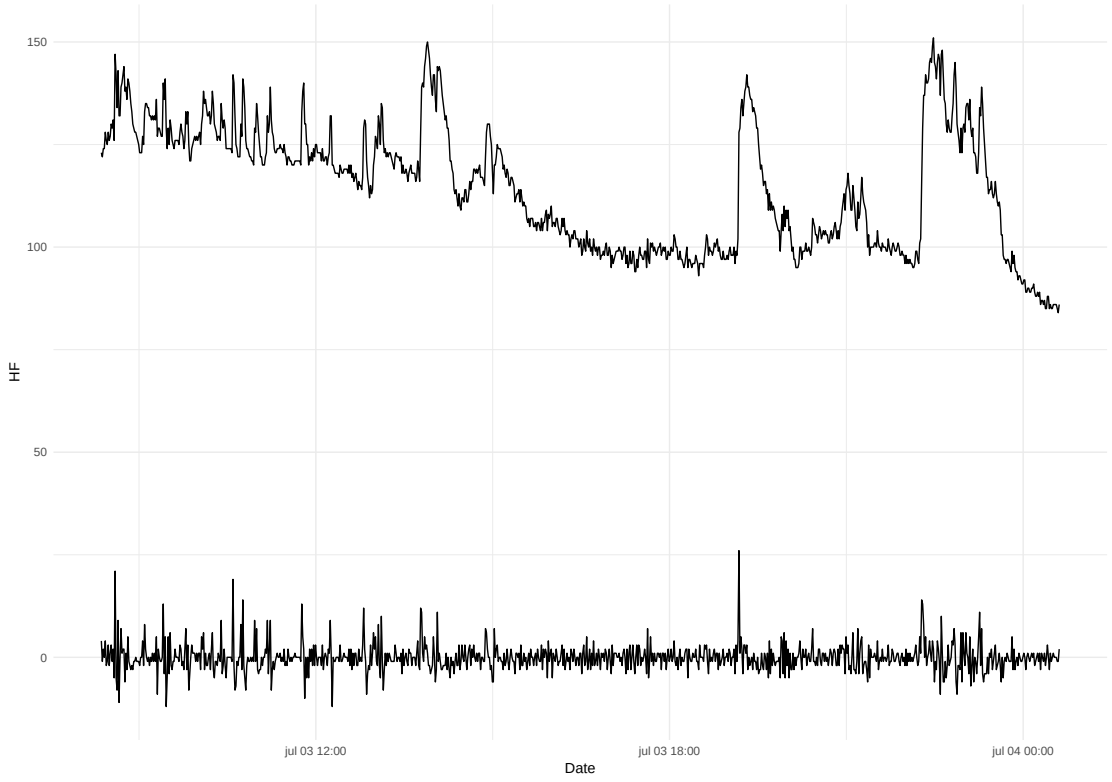


Figure 4.3: A non-stationary heart frequency series (above) and a stationary ΔHF (below)

case for the VAR(30) models; only 56 models were stable and 22 became unstable because the episode became too small due to the increased lag order. The smaller episode size caused **NA**s in the coefficients of the characteristic polynomial of the model. Furthermore, 22 models were otherwise unstable.

4.5.4 Granger causality

We performed Granger causality testing on the under 30 days subset. Using this test, we determined the presence of many Granger-causal relations. This is especially visible if we aggregate the results on all the episodes that we analyzed. These results are shown in table 4.6. The most frequently found relation exists between the ventilation measured frequency (ΔFreq) and the ΔE_TCO_2, which exists in 329 out of 361 episodes. The frequency Granger-causes most of the other variables. We aggregated the results of table 4.6 on the causes. The resulting table is shown in table 4.5. We can learn from this table that the Frequency Granger-causes most of the other VSTS overall.

p	HQ	AIC	SC	FPE
1	3	1	194	2
2	53	2	**438**	4
3	171	0	412	8
4	**326**	7	265	21
5	234	19	126	50
6	188	41	82	69
7	122	68	23	94
8	114	84	11	102
9	61	98	2	113
10	53	93	2	104

p	HQ	AIC	SC	FPE
11	27	62	0	76
12	18	74	2	83
13	15	84	2	92
14	9	53	1	66
15	3	56	1	63
16	5	54	0	54
17	9	53	2	53
18	23	55	12	61
19	30	60	20	59
20	21	52	15	55

p	HQ	AIC	SC	FPE
21	26	52	17	57
22	24	35	12	36
23	28	43	16	39
24	26	45	17	44
25	19	32	12	32
26	17	29	11	35
27	22	37	13	35
28	17	30	13	32
29	20	38	10	41
30	61	**388**	14	**165**

Table 4.4: These tables show the results of the determination of the lag order p. The numbers denote the number of episodes that had the ideal lag length according to the column's Information Criterion.

	Cause	n
1	ΔFreq	2558
2	ΔPRES	2399
3	ΔETV	2362
4	ΔFIO2	2329
5	ΔPEEP	2318

	Cause	n
6	ΔIE	2180
7	ΔEtCO2	1784
8	ΔSat	1780
9	ΔHF	1720
10	ΔRF	1637

Table 4.5: Results after aggregating only on the causes. n indicates how many times the variable is the cause in a Granger causal relation. Number of included patients is 361.

	Cause	Effect	False	True
1	ΔFreq	ΔEtCO2	32	329
2	ΔFreq	ΔHF	37	324
3	ΔFreq	ΔRF	48	314
4	ΔFIO2	ΔSat	53	309
5	ΔPRES	ΔEtCO2	56	306
6	ΔFIO2	ΔEtCO2	61	301
7	ΔETV	ΔFIO2	68	294
8	ΔFreq	ΔIE	72	290
9	ΔETV	ΔEtCO2	73	289
10	ΔPRES	ΔFIO2	73	289
11	ΔFreq	ΔSat	74	287
12	ΔPRES	ΔSat	76	286
13	ΔPEEP	ΔFIO2	81	281
14	ΔPEEP	ΔEtCO2	83	279
15	ΔIE	ΔEtCO2	85	277
16	ΔFIO2	ΔRF	85	276
17	ΔETV	ΔRF	87	274
18	ΔIE	ΔRF	89	273
19	ΔPRES	ΔHF	91	271
20	ΔPRES	ΔRF	91	270
21	ΔFreq	ΔPEEP	92	269
22	ΔPEEP	ΔSat	93	269
23	ΔEtCO2	ΔSat	95	267
24	ΔETV	ΔPRES	95	267
25	ΔETV	ΔSat	95	267
26	ΔPRES	ΔFreq	96	265
27	ΔIE	ΔHF	98	264
28	ΔPEEP	ΔRF	98	263
29	ΔPEEP	ΔFreq	99	262
30	ΔFIO2	ΔFreq	104	257
31	ΔETV	ΔHF	106	256
32	ΔFreq	ΔFIO2	108	253
33	ΔHF	ΔSat	109	253
34	ΔFreq	ΔETV	109	252
35	ΔPEEP	ΔHF	110	252
36	ΔPEEP	ΔETV	111	251
37	ΔETV	ΔPEEP	113	249
38	ΔPEEP	ΔPRES	113	249
39	ΔPRES	ΔETV	115	247
40	ΔFIO2	ΔHF	117	245
41	ΔFIO2	ΔIE	117	245
42	ΔETV	ΔFreq	117	244
43	ΔHF	ΔEtCO2	119	243
44	ΔIE	ΔSat	121	241
45	ΔFreq	ΔPRES	121	240
46	ΔSat	ΔEtCO2	123	239
47	ΔIE	ΔFreq	125	237
48	ΔPRES	ΔPEEP	125	237
49	ΔFIO2	ΔETV	127	235
50	ΔFIO2	ΔPRES	128	234
51	ΔIE	ΔPEEP	131	231
52	ΔSat	ΔFIO2	131	231
53	ΔIE	ΔETV	133	229
54	ΔPRES	ΔIE	134	228
55	ΔFIO2	ΔPEEP	135	227
56	ΔIE	ΔFIO2	137	225
57	ΔRF	ΔEtCO2	137	224
58	ΔEtCO2	ΔHF	140	222
59	ΔETV	ΔIE	140	222
60	ΔRF	ΔSat	147	214
61	ΔHF	ΔRF	148	213
62	ΔPEEP	ΔIE	150	212
63	ΔEtCO2	ΔRF	151	210
64	ΔIE	ΔPRES	159	203
65	ΔEtCO2	ΔFIO2	162	200
66	ΔSat	ΔPEEP	165	197
67	ΔSat	ΔPRES	168	194
68	ΔSat	ΔRF	168	193
69	ΔSat	ΔHF	174	188
70	ΔEtCO2	ΔPEEP	176	186
71	ΔSat	ΔFreq	175	186
72	ΔHF	ΔIE	179	183
73	ΔRF	ΔHF	178	183
74	ΔSat	ΔETV	179	183
75	ΔHF	ΔFIO2	184	178
76	ΔEtCO2	ΔETV	185	177
77	ΔEtCO2	ΔPRES	186	176
78	ΔRF	ΔPEEP	186	175
79	ΔEtCO2	ΔFreq	187	174
80	ΔRF	ΔIE	188	174
81	ΔRF	ΔFIO2	188	173
82	ΔEtCO2	ΔIE	190	172
83	ΔHF	ΔFreq	191	170
84	ΔSat	ΔIE	193	169
85	ΔRF	ΔETV	193	168
86	ΔRF	ΔFreq	194	167
87	ΔHF	ΔPEEP	196	166
88	ΔHF	ΔPRES	202	160
89	ΔRF	ΔPRES	202	159
90	ΔHF	ΔETV	208	154

Table 4.6: Granger causality between variables

4.5.5 Panel VAR models

We created a Panel VAR model using the `pvarfeols( ... )` function on a random sample of 100 episodes drawn from the under 30 days subset. Creating a PanelVAR model using the GMM estimator was not feasible due to the size of the dataset (see section 4.4.2). After estimating, we determined if the PanelVAR model was stable using a similar test provided in the package. The PanelVAR model was considered stable, as all the roots of the characteristic polynomial lie within the unit circle. The most important predictors are from the first two lags. The average contribution of a lagged variable to one of the Panel VAR model's equations is shown in figure 4.4. We see that the contribution lowers, as the distance t_0 grows.

We can also show the contribution of each variable. From figure 4.5, we can conclude that the frequency variable is the most important as its lags are the most represented in all the models' regression equations. Note that we made a similar observation in the VAR model from the Granger causality test in table 4.5.

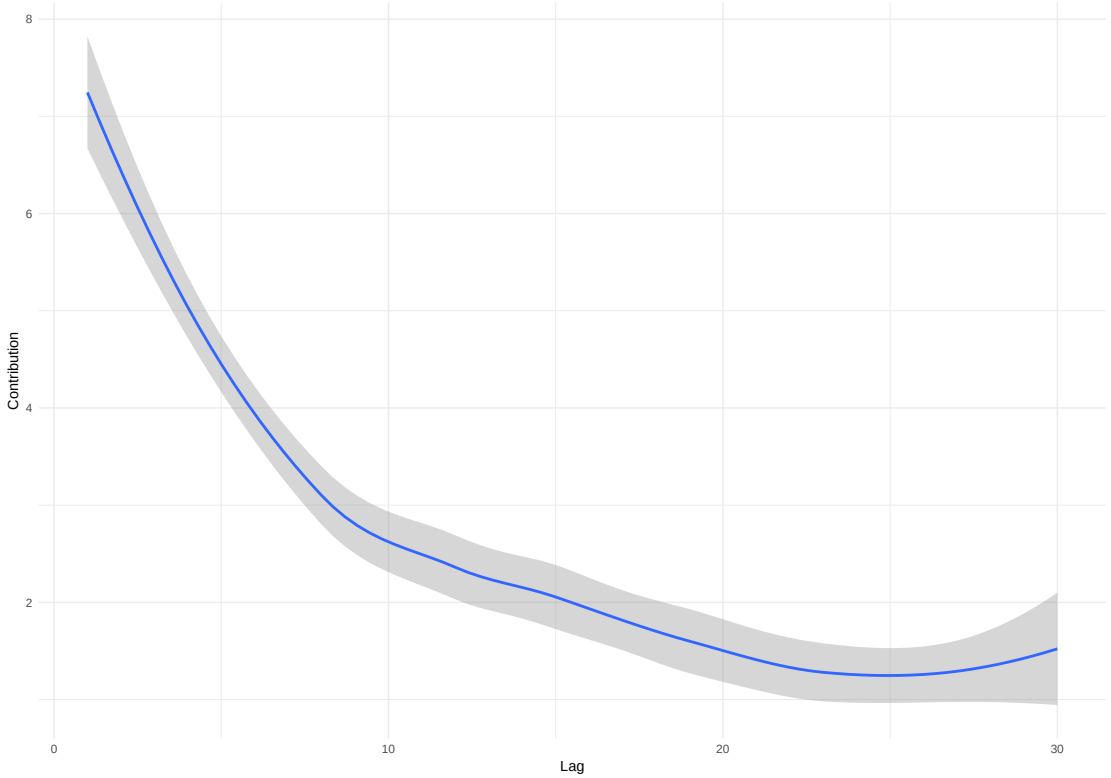


Figure 4.4: Number of times a variable of lag-t is included in a regression equation.

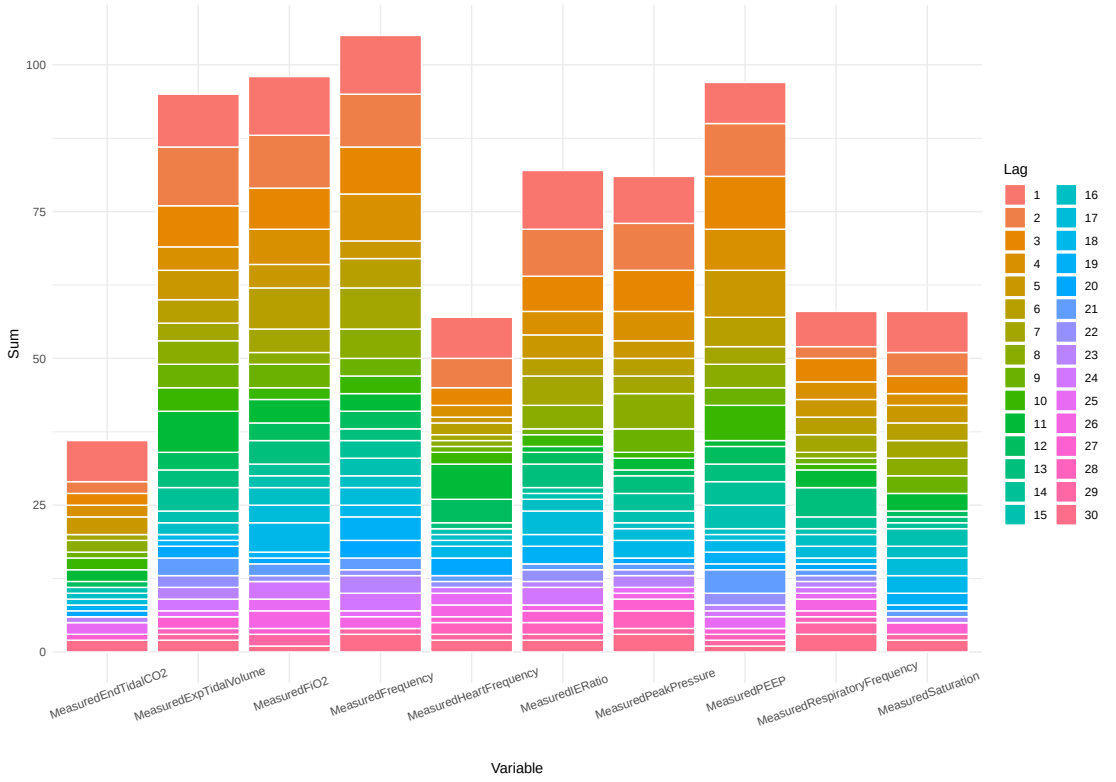


Figure 4.5: The contribution (predictive power) of each variable to the Panel VAR model

4.5.6 Graphical VAR models

We used the original series without differencing as input for the Multilevel Graphical VAR models. However, due to the computational complexity of the estimation algorithms and limitations of the used equipment, it was not possible to fit the model to the full dataset. Instead, we created samples of smaller size. The first was a random sample of 100 patients drawn from the whole dataset. The second was a random sample of 100 patients that were 30 days old or younger at the day of admission to the PICU. For these two subsets, we created the following networks:

- 30 Temporal networks for each lag in $L = \{1, 2, \ldots, 30\}$
- A contemporaneous network
- A between-subjects network

The temporal networks of lag-1 and lag-2 are shown in figure 4.6 and 4.7 respectively. The first thing that comes to mind when looking at the graph is the self-loops that are present for every node. The numbers on the arcs are the correlation coefficients between the variables. Among these correlations, the autocorrelations are the strongest. Note that the strongest correlation in the graph determines the thickness of the graph's other arcs. Because of this, we cannot rely on the visual aspects of the graph solely, as the width of the arcs of the lag-1 temporal network has a different meaning as the thickness of the arcs in the lag-2 graph. Using the underlying `qplot` package which was used in the `plot( ... )` function, the arc labels can be included in the plot. The maximum absolute correlation decreases a lot in the temporal networks beyond lag-2: In the lag-2 network, the maximum absolute correlation coefficient is already 0.14, coming from 0.73. In lag-3 it is 0.08, and at lag-10 it has gone down to 0.03. The sign of the correlation of the correlations also changes between lags; for example, the sign of the autocorrelation of the FiO_2 variable changes between lag-1 and lag-2. If we consider the intervariable relations, we see a relatively strong arcs coming out from the Frequency in both the lag-1 and lag-2 models. This is a result which was to be expected, in view of the results of the Granger-causality tests and the PanelVAR model.

The between-subjects networks (figures 4.8 and 4.9) show a significant correlation between the Respiratory Frequency (RF) and Freq, the Respiratory Frequency measured by the ventilator. This association was expected as they describe the same concept. Besides this relation, there is a strong association between Peak Pressure and PEEP. There is also a difference between the network of the below-30-days subset and the whole dataset; the number of correlations is smaller in the network for the entire dataset (6 vs. 15). Possible clinical explanations for the correlations that are present in the between-subjects-network will be discussed in section 4.6.2.

The contemporaneous networks (in figure 4.10 and 4.11) do not contradict the between-subjects network as the signs of edges do not change. However, the between-subjects network is more sparse. The contemporaneous relation between PEEP and PRES (P^{PEAK})

is very strong. This is to be expected in view of the relation between them that was described in section 2.2.4. In section 4.6.3, we will further discuss clinical explanations of the correlations of this network.

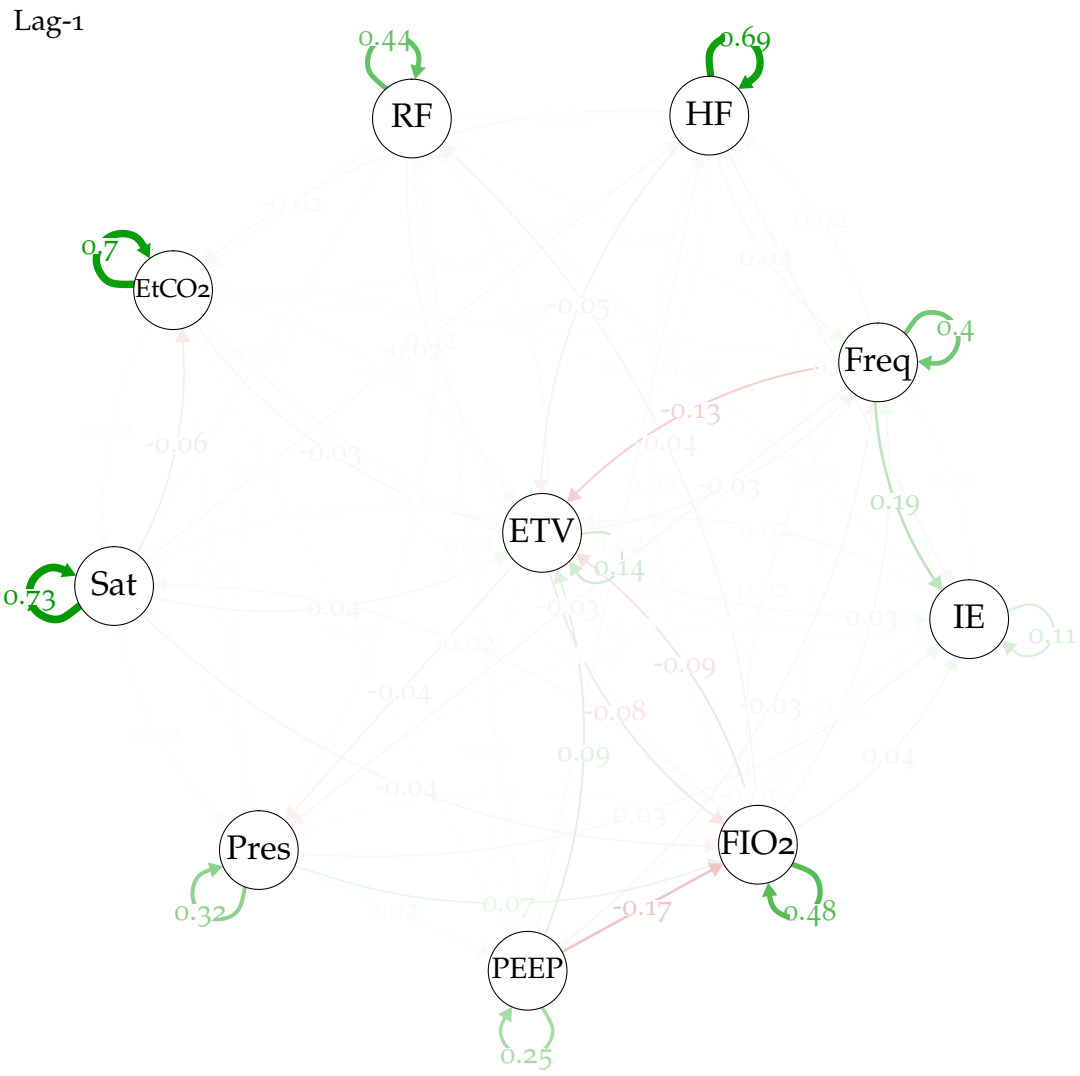


Figure 4.6: Fixed effects temporal model of Lag-1, for the below 30-days subset

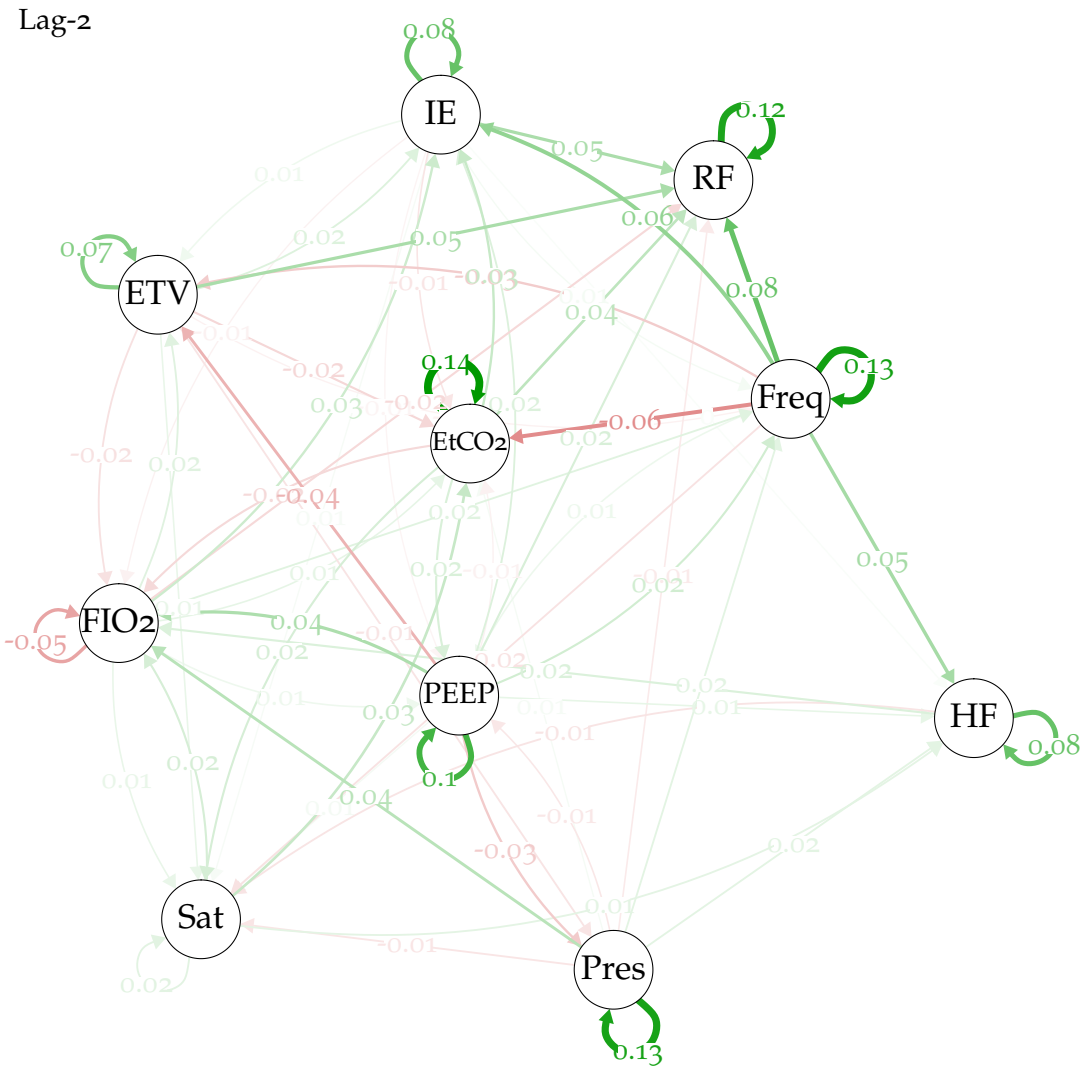


Figure 4.7: Fixed effects temporal model of Lag 2, for the below 30-days subset

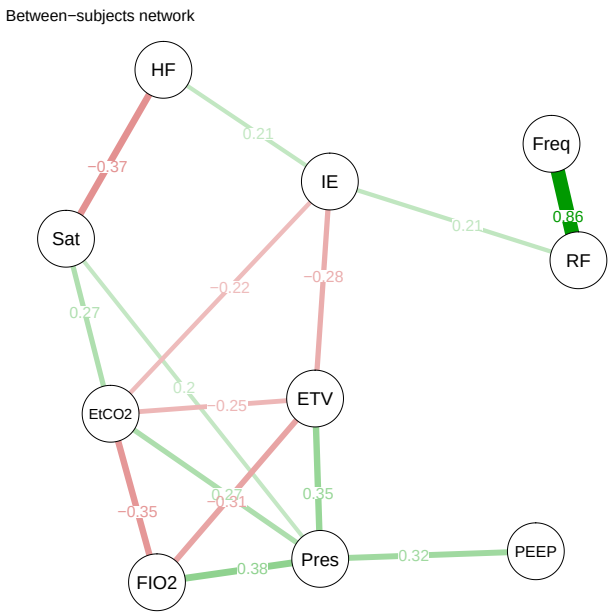


Figure 4.8: Between subjects model of the under 30 days subset

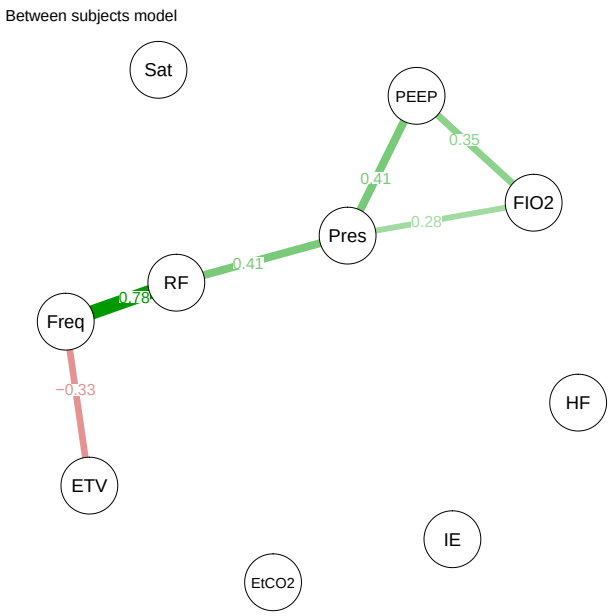


Figure 4.9: Between subjects model of the whole dataset

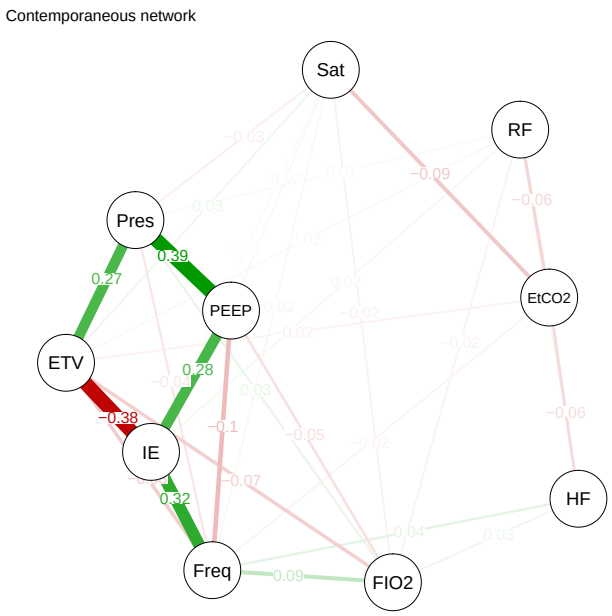


Figure 4.10: Contemporaneous model of the under 30 days subset

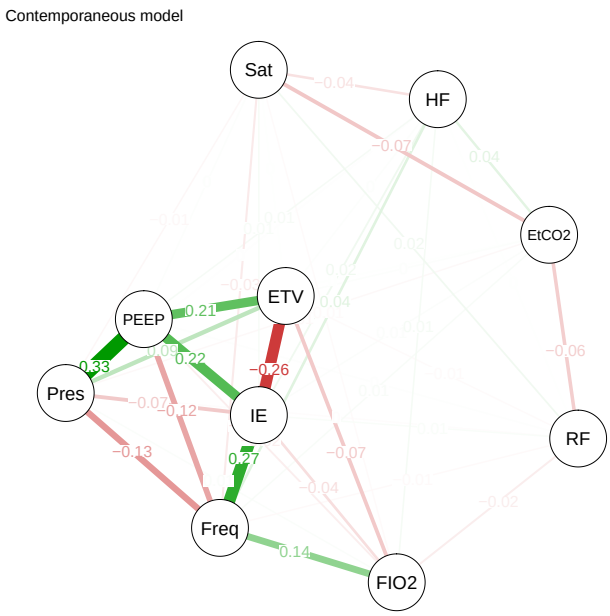


Figure 4.11: Contemporaneous model of the whole dataset

4.6 Discussion

In this section, we will discuss the results of the previous section. We discuss here the results of the three models; the classical VAR model, the Panel VAR model, and the Multilevel Graphical VAR model.

4.6.1 Temporal models

When considering the temporal effects, we can make several observations. The first is that there are no apparent contradictions between models. For example, for all the models, it holds recent lags (e.g., lag-1 and lag-2) are most influential in determining the value of the current measurement. This observation is more evident in the Panel-VAR and the Multilevel Graphical VAR model, as they model the whole dataset and not individual patients. Also, we have seen that the Frequency variable is included in many regression equations and is listed as a cause for many Granger-causal relations. For the MLGVAR model, this is also the case; arcs from the Frequency node have high (absolute) correlation coefficients.

The high number of Granger-causal relations between variables (and indicators of Granger-causal relationships in the PanelVAR / MLGVAR models) could be an indication that when we make predictive models, that including other variables might indeed help in predicting future values for the Vital Signs Time Series. This could mean that these variables are indeed related, and do not solely rely on their own history. However, the coefficients of the self loops in the temporal networks are among the highest. The high coefficients in the temporal network for lag-1 (figure 4.6) show a significant correlation between the previous measurement and the current. This might be an indication that extending the prediction horizon beyond $t + 1$ might be difficult, as the contribution of the lags decreases linearly, an observation which is evident from figure 4.4.

The high number of Granger-causal relations and a high number of temporal networks make it difficult to assess these clinically; especially, when the number of variables is also high. Creating VAR models for each episode and then aggregating their results is feasible when the number of episodes is not too high; we can even sometimes discuss individual patients. In our case, we found many relations between variables. This observation, combined with the fact that our dataset contains many patients and episodes, makes the traditional VAR methodology unpractical.

4.6.2 Between-subjects model

In this section, we will attempt to interpret the results of the between-subjects network of both the under 30 days subset and the whole dataset (in figure 4.8 and 4.9). These networks show some interesting correlations between variables. Below, we will provide some possible (clinical) relations that might explain these correlations:

Respiratory Frequency and Frequency. We expected to find this relation beforehand, as these variables describe the same concept; they only differ in the method of measurement. The Frequency is recorded by the ventilator, whereas the Respiratory Frequency is based on the leads that are used for electrocardiography (ECG). This difference causes slight variations in the recorded values. The fact that these variables record the same concept probably causes them to have the highest correlation (0.86 and 0.78) coefficient in this network for both datasets.

(Respiratory) Frequency and Expiratory Tidal Volume. There is a negative correlation between the frequency and the Expiratory Tidal Volume(-0.33, whole dataset); in words this correlation says the following: patients who breathe faster, breathe less air out per breath. By judging this network alone, one might assume that this relation could have a pathological origin; however, there is a cause for this relationship that is more reasonable. This relation is probably related to age and weight, two variables that are not included in this model. The fact that this relation is present in the network for the whole dataset, but this (direct) relation is absent in the network of the 30 days subset is is also an indication that this relation is not of pathological origin. The youngest patients, especially babies, breathe faster, and we have seen in figure 2.4 that the $V_{T_{\text{exp}}}$ variable is related to weight. Weight is, in turn, related to age. The fact that older patients have a higher $V_{T_{\text{exp}}}$ and a lower RF and the fact that the opposite applies to younger patients, explain this relation in the network of the whole dataset. Figure 4.12 visualizes the relation between age, weight, $V_{T_{\text{exp}}}$ and RF.

Peak Pressure, PEEP, and FiO_2. When the PEEP is set higher, the lung has more oxygenation problems; in other words, the lung is ailing. Recall that the PEEP is used to prevent alveolar collapse (see section 2.2.4). When the PEEP is higher, the pressure needs to be set appropriately above the PEEP level to ensure airflow. The measured peak pressure levels will, in turn, also be higher. This is reflected in both networks, which have a positive correlation of 0.41 for the whole dataset and 0.32 in the under 30 days dataset.

In addition to raising the pressure levels, an ailing lung is treated by changing the fraction of inspired oxygen (FiO_2). When the inspired air contains more oxygen than usual, the body can easier maintain proper blood oxygen levels (SaO_2), despite the diminished lung capacity. When the patient's lung is ailing, the caregivers will increase both pressure levels and oxygen levels. These relations that are an indication of the lung's health level and treatment regimen are visible in both networks (0.35 for the whole dataset and 0.38 for the subset).

Peak Pressure and (Respiratory) Frequency. The network of the whole dataset shows a positive correlation of 0.41. When a patient is ailing, more pressure is needed to ensure that the same amount of air reaches the lungs. To ensure that the EtCO_2 re-

mains within safe boundaries, the set respiratory frequency is increased. Also, when the frequency is increased (and in turn, the breaths are shorter), more pressure is needed to move the same volume of air.

I:E ratio, Expiratory Tidal Volume, and RF. In practice, when the patient is connected to the ventilator in the PRVC / PC-IMV-Adaptive mode, the inspiration time is set and the ratio between the inspiration and expiration time (I:E ratio) is set to be $1 : 1.5$. When the patient attempts to breathe on his own, the expiration will be shorter. When this happens, the value will be higher as the time for the expiration is in the denominator (recall equation 2.3). In turn, when the time for the expiration is shorter, the Expiratory Tidal Volume could be smaller. In turn, the respiratory frequency will increase. These correlations are, however, only present in the under 30 days subject (0.21 with RF, and -0.28 with $V_{T_{\text{exp}}}$).

Heart Frequency and Saturation. In the network for the under 30 days subset, a negative correlation (-0.37) between the heart frequency and saturation is shown. This relation could have the following explanation: patients who are severely ill can have a higher heart rate and lower saturation level. During recovery, the heart rate lowers, and the saturation rises to normal levels of 95 % - 100 %.

$EtCO_2$, FiO_2, and P^{peak}. In practice, when the lung is more ailing, the $EtCO_2$ levels will be higher. The $V_{T_{\text{exp}}}$ levels will be lower as well. These clinical observations might explain the negative correlation between $V_{T_{\text{exp}}}$ and $EtCO_2$ (-0.25) in the network of the under 30 days subset. The pressure and FiO_2 levels might be increased as treatment (correlation -0.31 with $V_{T_{\text{exp}}}$). The negative association between $EtCO_2$ and FiO_2(-0.35) may seem strange; as an ailing lung may have higher $EtCO_2$ levels. However, when a patient is healing, he will start to breathe more by himself. In turn, this may lead to higher $EtCO_2$ levels (but safe). Moreover, healthier patients do not need a high FiO_2.

$EtCO_2$ and SpO_2. In the same line as above, we could explain the positive correlation of 0.27 in the under 30 days subset, could be explained by the fact that when a patient gets healthier and starts more and more breaths himself, which leads to higher $EtCO_2$ levels (ventilated patients have an artificial lower $EtCO_2$ level). Healthier patients also have higher SpO_2 levels.

In addition to these possible explanations of the correlations, we can also make some general remarks on the between-subjects model. When we compare both networks, we can clearly see that more relations are found on the under 30 days subset (6 vs. 15). This fact might confirm our conjecture that creating predictive models for specific subsets of patients might be beneficial for the model's predictive accuracy, as there may be relations between variables that are only valid for specific groups of patients.

The fact that the model relates the Respiratory Frequency to the Expiratory Tidal Volume, points out that we should heed the advice by Epskamp et al. [15] seriously;

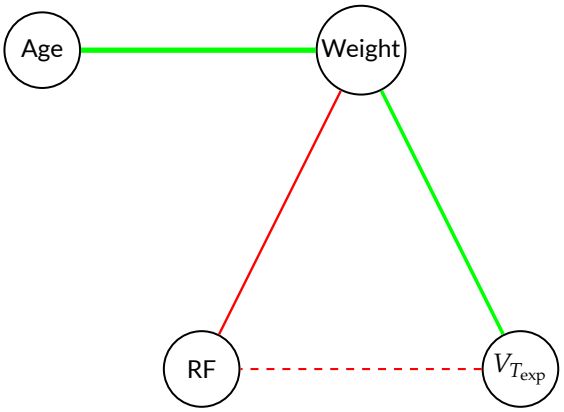


Figure 4.12: The relation between Age, Weight, RF and $V_{T_{exp}}$. The dashed line expresses the (spurious) correlation found in figure 4.9, which was caused by the absence of age and weight in the MLGVAR model.

that we should not derive causal conclusions from these models, only hypotheses for further research.

4.6.3 Contemporaneous model

In figures 4.10 (under 30 days subset) and 4.11 (whole dataset), several relations are shown. We will describe some of them below.

PEEP and P^{PEAK} (Pres). This relation is the most prominent among them, and found in the networks of both datasets (0.33 whole, 0.39 subset). This relation is also present in the between-subjects model. It says that within the same measurement we can use the PEEP to predict the P^{PEAK} and vice versa. In section 2.2.4, we described that a higher PEEP results in a higher applied peak pressure.

Expiratory Tidal Volume, I:E ratio and Frequency. The negative correlation between $V_{T_{exp}}$ and I:E is among the most strong in both graphs (-0.26 whole, -0.38 subset). The relation between the Frequency and I:E-ratio is also strong (0.27 whole, 0.32 subset) in both models. The relations between can be explained along the same lines as for these variables in the between-subjects network.

Expiratory Tidal Volume and P^{PEAK}. We have seen in section 2.2.4 and 2.3.3 how the pressure can be regulated to achieve a higher $V_{T_{exp}}$. The contemporaneous correlation between $V_{T_{exp}}$ and P^{PEAK} (0.21 whole, 0.27 subset) might capture this relation.

RF and $ETCO_2$. When our respiratory frequency is lower (*hypoventilation*), our $ETCO_2$ levels increase. This is well visible in a *capnogram*, which is the plot of the $ETCO_2$ levels [18]. Vice versa, the same holds. The networks of both datasets show a negative correlation of -0.06 between these variables.

Epskamp et al. [15] describe how these models might be interpreted. This network models relations between variables that are present within a measurement, so that if we know the current value of variable A, we can use that to predict the current value of variable B. However, these relations may also be caused by temporal effects that cannot be modeled through the lagged variables; for example, when they happen within the one minute window. By increasing the measurement rate, some of the contemporaneous correlations might disappear and instead be captured as temporal ones.

4.6.4 Limitations

The models that we discussed are all linear models. Between the time series, or even within a time series, non-linear relations can be present. Brigham et al. [7] suggest that the Vital Signs Time Series are non-linear. The VAR models can not capture these non-linearities, as they can only model linear relations. The relations that we discussed above may only be valid if the linearity assumption is correct.

4.7 Summary

In this chapter, we detailed how time series analysis methods can be used to describe time series datasets like the PICU dataset. Whereas the classical VAR models can be used to create patient-specific models, the PanelVAR and Multilevel Graphical VAR models can be used to create models that describe the whole population. We could not include variables that remain (most of the time) constant (e.g., settings); therefore, we only included the measured variables in the models. We used VAR models for Granger-causality testing. We found a considerable number of Granger-causal relations between variables. However, these relations we patient-specific. The number of patients is also large. These two observations combined led us to the conclusion that applying the methodology of Bose et al. [5] is not very practical. However, the PanelVAR and, especially the MLGVAR model, enable us to make statements that may hold for the whole population.

We saw that the current measurement $\boldsymbol{x}_t$ greatly depends on recent lags $\boldsymbol{x}_{t-1}$ and $\boldsymbol{x}_{t-2}$. When p in $\boldsymbol{x}_{t-p}$ becomes higher, the correlation becomes smaller and smaller. This is visible in figure 4.4. Figures 4.6 and 4.7 show that for each variable its own lags are the most important, as the correlations of the self-loops are the strongest. However, there are some significant temporal associations between variables, for example, between the Respiratory Frequency (FREQ) and the Expiratory Tidal Volume.

In addition to the temporal effects, we also discussed the between-subjects effects and contemporaneous effects. The between-subjects network shows how variables predict each other on average, and the contemporaneous network shows how variables are related within the same measurement, after accounting for the temporal effects. We have seen relations that could be explained by the lungs health and the treatment regi-

men (e.g., between P^{PEAK}, PEEP, and FiO_2). However, we have also seen that the model might make wrong assumptions. For example, the relation between RF and $V_{T_{exp}}$ (see figure 4.12), because the static variables age and weight are missing.

We also observed that when we only consider a specific patient-cohort (based on age), the models differ. The between-subjects network for the under 30 days subset shows more correlations than the network for the whole dataset. This leads us to the hypothesis that creating models for specific groups of patients might be beneficial.

In the next chapter, we will discuss predictive models that also capture non-linear functions (as well as linear functions). Moreover, these models can use the variables that we did not include in our exploratory analysis; static variables, like age and weight, and the ventilator's settings.

5 Predictive models

5.1 Introduction

In the last decade, we have seen intelligent systems surface in almost any field or market. We have seen how IBM's Watson defeated the best human players of the TV-show Jeopardy, the defeat of the best human Go player by Alpha-Go, the advent of self-driving cars. In newspapers, popular scientific magazines and, of course, research papers, applications of AI are ubiquitous. Many of these recent advances use Artificial Neural Networks. Significant increases in the availability and use of distributed computing (mainly in the cloud), make the use of big datasets feasible. The increase in computing power also contributed to the growth of the networks' capacity, enabling the ability to model more complex problems. Besides the increase in computing power, the introduction of new neural network architectures, for example, the Long Short-Term Memory networks and Convolutional Neural Networks, made these advances possible.

Artificial Neural Networks can also be applied to time series forecasting. The ability of Neural Networks to model very complex (non-linear) functions could be beneficial to predict future values of Vital Signs Time Series. Making a prediction model using ANNs is not a trivial task. We have to make very conscious choices on what model to use, as no machine learning algorithm is suitable for all datasets (no free lunch theorem) [19]. By carefully designing and configuring a Neural Network, we might be able to create a successful model.

In the following sections, neural networks and the field of deep learning will be discussed. First, we will discuss the primary building blocks of Neural Networks, the per-

ceptron, and some other principles that are also used in more advanced architectures. Then we discuss how we can model sequences like time series using two neural network architectures we mentioned earlier, the Long Short-Term Memory Networks, and Convolutional Neural Networks. After that, our experimental setup will be discussed, followed by their results and interpretation.

5.2 Artificial Neural Networks and Deep Learning principles

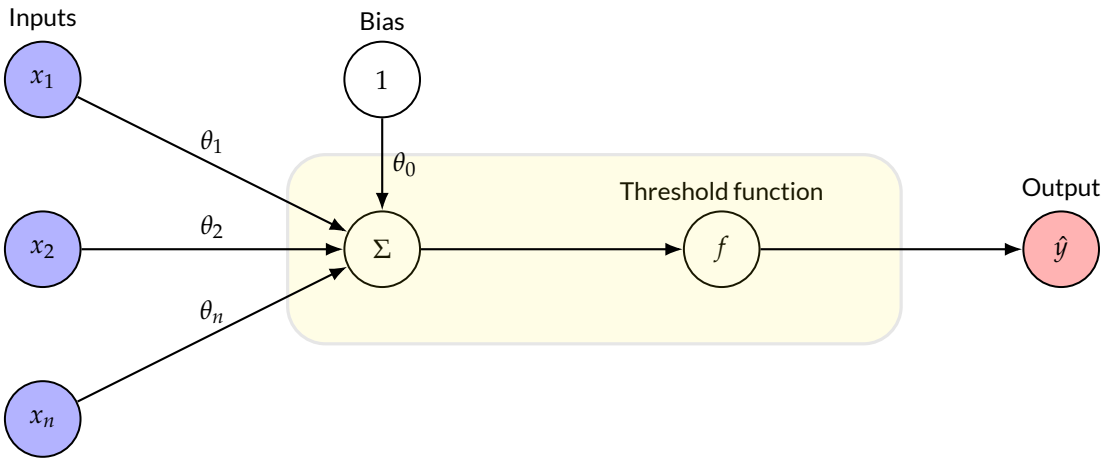


Figure 5.1: The perceptron by Rosenblatt. Several inputs x_i and corresponding weights θ_i, combined with a threshold function, produce an estimate $\hat{y}$. This figure is based on a figure in [2], but adapted to the notation used in this work.

In 1943, the first paper on neural networks was published by Warren McCulloch and Walter Pitts [33, 17]. The paper called "A logical calculus of the ideas immanent in nervous activity" described a computational model of a biological neuron. The human nervous system is composed of millions of neurons, which enables them to sense, act, and think about the world. The theoretic model that they proposed was able to perform Boolean operations, and therefore theoretically, a neural network should be able to calculate the result of every computation. In 1958, Frank Rosenblatt implemented one of the first artificial neurons, the *perceptron* (see figure 5.1). By configuring the weights, this network could be used as a classification method. A linear threshold function could be used to distinguish two classes. However, it was also flawed in the sense that this implementation was perceived to be unable to learn the XOR-operation, which is a boolean operation that was thought to be trivial to learn. This inability caused a deception amongst researchers and funding agencies, and therefore funding into neural network research plummeted.

In the 1980s there was a revival of neural networks after publications in the "Parallel Distributed Processing" volumes by Rumelhart et al. and McClelland et al. [44, 32]. Neural networks, traditionally being a modeling tool for cognitive science, have also

since then be used for many machine learning tasks. ANNs excel in pattern recognition and modeling complex functions [17]. According to Kurd et al. [26], the main benefits of Artificial Neural Networks over other machine learning methods are as follows:

The ability to learn. This ability is also present when there is no or little understanding of the relation between the input and the desired output of the learning problem. In other words, no specific knowledge about the problem is needed; this enables bottom-up learning as no specific information on the problems needs to be made explicit.

Operational Performance. ANNs are often able to make generalizations on the training data, meaning that ANNs can also make correct predictions on unseen data. Their generalization ability often outperforms other machine learning techniques.

Computational efficiency. Neural networks can be more efficient in both memory consumption and processing speed than other methods.

The first advantage refers to a difference that exists between traditional machine learning methods and deep learning. In more traditional machine learning algorithms, such as logistic regression, important *features*, pieces of relevant information, are handpicked [19]. When a traditional method needs to make a recommendation for a self-driving vehicle, for example, whether to brake or not, that model is given features like the presence of an object, the distance to the next object, the velocity of the vehicle. These features give meaning to and simplify the raw data; an example feature could be (the output of) an algorithm that detects the presence of objects in an image. If a traditional method were instead given the raw output of a camera, the model would not be able to deal with this, as these individual pixels can have a negligible relation with a reason for needing to brake. Representing the environment with designed features enables these methods to make decisions. However, in some cases, it is not always known or clear which features can be extracted from the raw data [19]. Finding these features is often hard, for example, it is challenging to represent abstract concepts like objects in terms of raw pixels; or in this project's case, a disease in terms of Vital Signs Time Series. Deep learning methods can learn the representation of these features in the raw data themselves [19].

5.2.1 The Multilayer Perceptron

The *Multilayer Perceptron* (MLP) or *deep feed forward* network is often considered as the primary deep learning method. The name references to the original perceptron by Rosenblatt, by effectively being an extension to his perceptron. The MLP has the following alterations:

- The usage of multiple layers of nodes instead of a single layer
- The usage of a non-linear activation function instead of a linear threshold function.

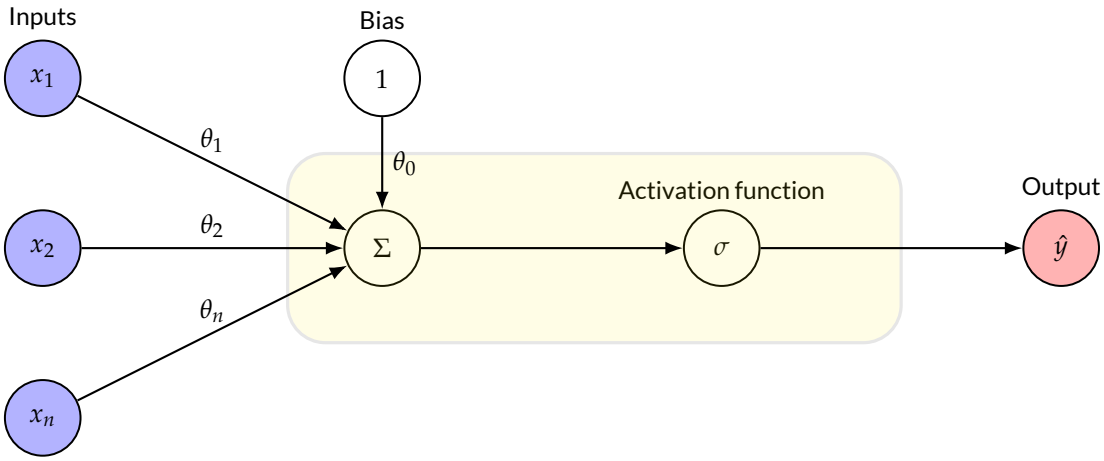


Figure 5.2: A single perceptron unit of a Multilayer Perceptron Network (in yellow). The architecture is almost identical to the original perceptron (see figure 5.1), with the only change being the activation function. This figure is based on a figure in [2], but adapted to the notation used in this work.

A single perceptron unit is displayed in figure 5.2. A unit has multiple inputs x_i, and a bias node, which are connected through weights θ_i. This bias has a similar role as the intercept in linear regression. The bias has its own weight θ_0, which actually defines the bias (because of the multiplication by one). The output $\hat{y}$ is calculated using the following function [2]:

$$\hat{y} = g(\theta_0 + \sum_{i=1}^{n}(x_i \cdot \theta_i)) \tag{5.1}$$

$$= g(\theta_0 + \boldsymbol{x}^T\boldsymbol{\theta}) \tag{5.2}$$

where

$g(z)$: a non-linear activation function (e.g., the sigmoid function σ)
$\boldsymbol{\theta}$: $\begin{bmatrix}\theta_1 & \theta_2 & \cdots & \theta_n\end{bmatrix}^T$
$\boldsymbol{x}$: $\begin{bmatrix}x_1 & x_2 & \cdots & x_n\end{bmatrix}^T$

We can view the perceptron as a function f which maps an input with type $\mathbb{X}$ to an output with type $\mathbb{Y}$. This is expressed in the following equation:

$$\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta}) \tag{5.3}$$

where

f : a function of $\boldsymbol{x}$, parametrized by $\boldsymbol{\theta}$ with type $f : \mathbb{X} \to \mathbb{Y}$
$\boldsymbol{\theta}$: a vector describing the weights of the neural network
$\boldsymbol{x}$: a vector containing the inputs $\begin{bmatrix}x_1 & x_2 & \cdots & x_n\end{bmatrix}^T$

We can make a Multilayered Perceptron by tying the output of a node to the input of another node. The following equation is visualized in figure 5.3.

$$\hat{y} = f(\boldsymbol{x};\boldsymbol{\theta}) = f_3(f_2(\begin{bmatrix} f_{1,1}(\boldsymbol{x};\boldsymbol{\theta}_{1,1}) \\ f_{1,2}(\boldsymbol{x};\boldsymbol{\theta}_{1,2}) \end{bmatrix};\boldsymbol{\theta}_2);\boldsymbol{\theta}_3) \tag{5.4}$$

The MLP is designed as follows: Suppose having an input $\boldsymbol{x}$, for which the network

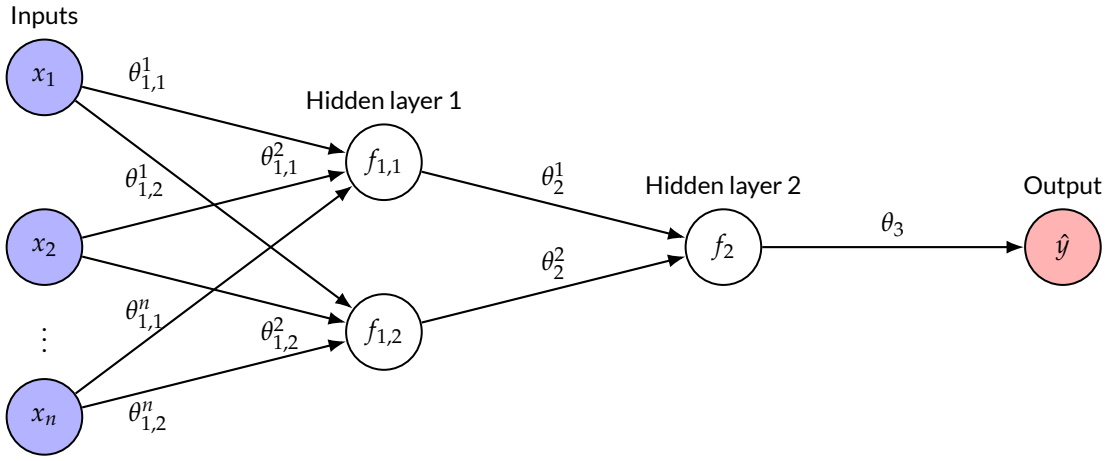


Figure 5.3: By combining multiple perceptron units, a multilayer perceptron is formed. A white node depicts a full perceptron unit (recall the yellow box in figure 5.2). This figure visualizes equation 5.4. The bias nodes are omitted in this figure.

needs to estimate for $\boldsymbol{y}$, outputting $\hat{\boldsymbol{y}}$. The information in $\boldsymbol{x}$ is given to a series of input nodes (in blue). These are connected to a series of layers with multiple nodes (in white). The last layer is called the output layer (in red), which encodes the value of $\hat{\boldsymbol{y}}$. Output nodes are also perceptron units.

When the network is given an input $\boldsymbol{x}$, the information flows through from the input nodes, to each of the hidden layers and its units, and finally, it determines an output for $\hat{\boldsymbol{y}}$. This process is called forward propagation [19]. Each unit is effectively in itself function, accepting the output of the nodes of the previous layer. A perceptron produces thus an intermediate output or final output. The layers after the input layer are often referred to as hidden layers, as their weights are not known beforehand.

We could use a Multilayered Perceptron for time series forecasting. This can be done by training an MLP that approximates a function that maps a vector containing t time steps to a vector containing s predictions. Suppose we want to predict the next three $ETCO_2$ measurements, given a history of t measurements. We give a MLP a vector $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_t \end{bmatrix}$, which contain the last t $ETCO_2$ measurements. The input flows through a network with k hidden layers, with each its own number of nodes d_k, and finally produces an output. This is a vector $\hat{\boldsymbol{y}} = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}$, which contains a prediction for the next three $ETCO_2$ measurements. This network is visualized in figure 5.4. As

said before, the weights of the network are not known. The network can find suitable weights through the application of the Gradient Descent algorithm.

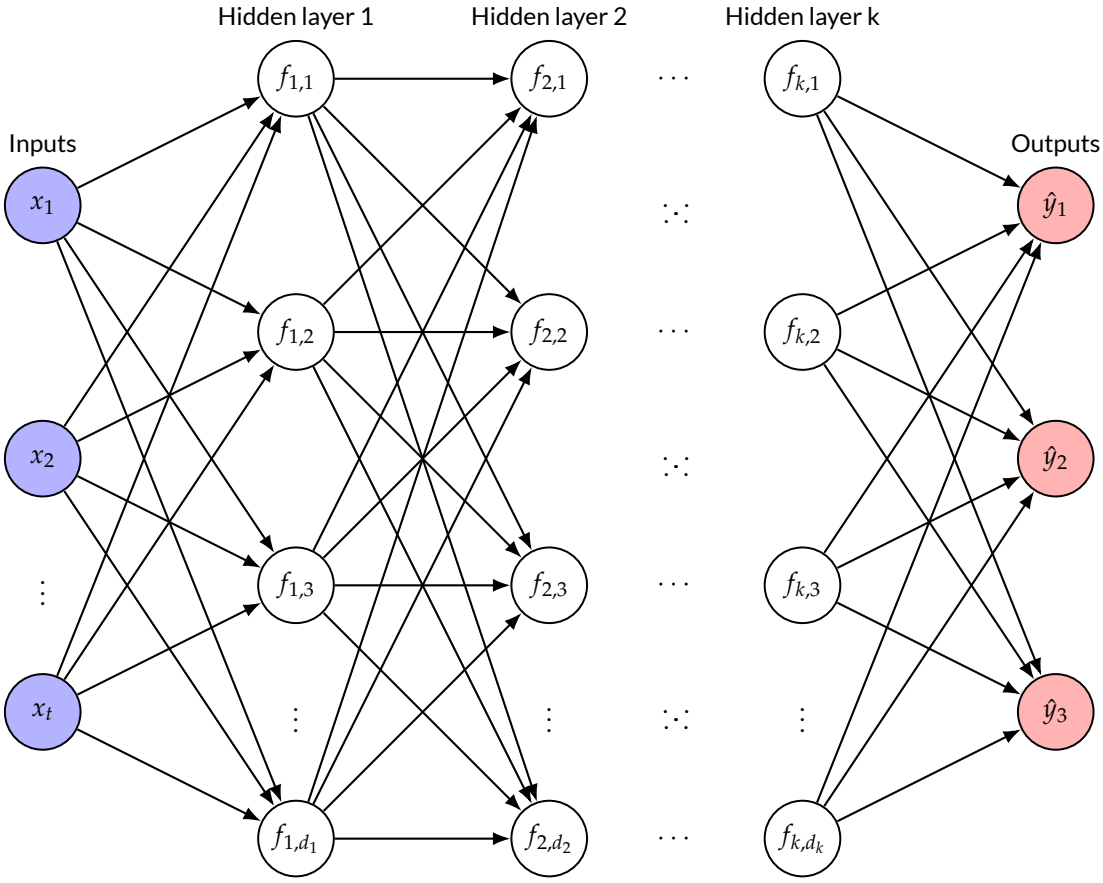


Figure 5.4: The multi layered perceptron used for time series forecasting, with k hidden layers. Each white node is a perceptron cell. The bias node for each cell is omitted in this figure.

5.2.2 Gradient Descent and Back Propagation

Initially, the weights $\boldsymbol{\theta}$ of a neural network f are set randomly. After estimating $\hat{\boldsymbol{y}}$, we can calculate the cost function $J(\boldsymbol{\theta})$. The exact definition of the cost function depends on the problem, but generally it will describe how well the network predicts $\hat{\boldsymbol{y}}$ given $\boldsymbol{\theta}$. The training of a neural network can be described as an optimization problem. Optimization is defined as the task to minimize (or maximize) a function $f(x)$ by altering x, in other words, find x for which $f(x)$ is at its lowest or highest. In this particular case, we want to minimize the cost function $J(\boldsymbol{\theta})$. The minimum of a function $f(x)$ can be determined by using its derivative $f'(x)$.

$$f(x - \epsilon \operatorname{sign}(f'(x))) < f(x) \tag{5.5}$$

If ϵ is small enough, we can use the above equation 5.5 to move towards a minimum of $f(x)$. This method is known as *gradient descent* [19]. Note that using this method, a minimum of $f(x)$ can be found; however, this minimum is not necessarily the global minimum. In the case of deep learning, functions are optimized that possibly have multiple local minima, which makes finding this global minimum difficult. Therefore, approximate minimization is applied. Parameter settings $\boldsymbol{\theta}$ that perform well enough, corresponding with low values of $J(\boldsymbol{\theta})$ are therefore accepted.

As $\boldsymbol{\theta}$ is multidimensional, *partial derivatives* are necessary. A partial derivative $\frac{\partial}{\partial x_i} f(\boldsymbol{x})$ describes how f changes when only x_i is changed. Then the notion of a *gradient* can be introduced, which is the ensemble of all the partial derivatives for f with respect to a vector $\boldsymbol{x}$. The gradient for $f(\boldsymbol{x})$ is denoted as $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$. Using gradient descent, we will move towards a new point $\boldsymbol{x}'$ using the following equation [19]:

$$\boldsymbol{x}' = \boldsymbol{x} - \epsilon \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{5.6}$$

where

ϵ : denotes *the learning rate*, which is a small positive scalar, determining the step size

Finding the right learning rate is essential for finding the correct weights for the network. A learning rate that is too large might not find the minimum, as a minimum can be missed because of the step size (overshoot). However, a learning rate that is too small may take forever to converge, or it can get stuck in a local minimum that is far from optimal. Gradient descent converges when all the individual values of the gradient are zero or very close to zero. In pseudo code, gradient descent can be described as follows:

```
def gradient_descent(learning_rate, cost_fun, network):
    network.weights = random.normal(network.dimensions)
    converged = False
    while not(converged):
        gradient = compute_gradient(cost_fun, network)
        # θ' = θ − ε · ∇_θ J(θ)
        network.weights = network.weights - learning_rate * gradient
        converged = check_converged(gradient)
    return network.weights
```

The function `compute_gradient( ... )` is usually an implementation of the *Back propagation algorithm*, first presented by Rumelhart et al. [45]. We described earlier in this section how information flows forward when calculating the result of the neural network when after supplying an input. The back propagation algorithm allows the errors (the result of the cost function) to flow backwards through the network. How this algorithm works can best be illustrated using the example network shown in figure 5.5: By

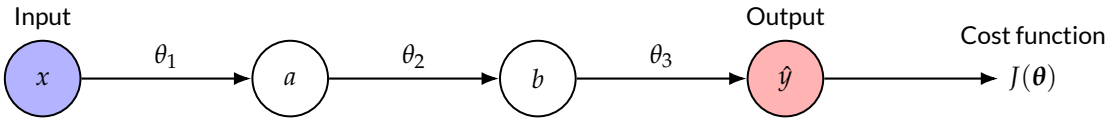


Figure 5.5: Sample Computational Graph

repeatedly applying the *chain rule*, we can calculate for each weight θ_i the partial derivative, answering the question on how a minor change in weight θ_i affects the final cost of the network [19]. By using the gradients from later layers (for example, the gradients from b in a) we can calculate the partial derivative for each weight θ_i [2]. An example calculation for the gradient of the weights $\boldsymbol{\theta}$ of figure 5.5 is shown below. For the weight θ_3 we only need to apply the chain rule once:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_3} = \frac{\partial J(\boldsymbol{\theta})}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_3} \tag{5.7}$$

However, for θ_2 we have to apply the rule twice:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_2} = \frac{\partial J(\boldsymbol{\theta})}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_2} = \frac{\partial J(\boldsymbol{\theta})}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial b} \cdot \frac{\partial b}{\partial \theta_2} \tag{5.8}$$

And for θ_1 thrice:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_1} = \frac{\partial J(\boldsymbol{\theta})}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_1} = \frac{\partial J(\boldsymbol{\theta})}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial b} \cdot \frac{\partial b}{\partial \theta_1} = \frac{\partial J(\boldsymbol{\theta})}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial b} \cdot \frac{\partial b}{\partial a} \cdot \frac{\partial a}{\partial \theta_1} \tag{5.9}$$

Finally, we can combine these all into the gradient, which can can then be used in the gradient descent algorithm:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_1} & \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_2} & \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_3} \end{bmatrix}^T \tag{5.10}$$

5.2.3 Activation functions

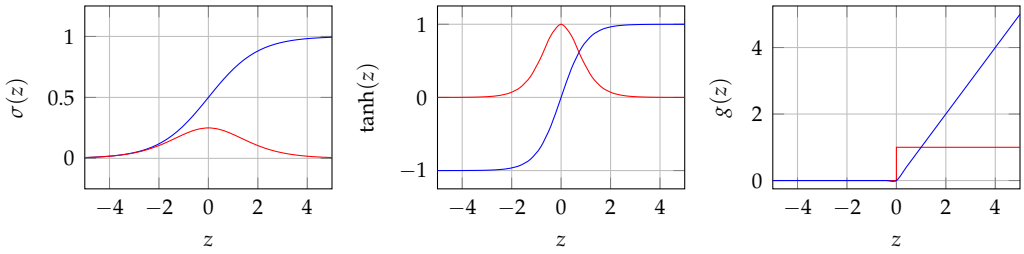
Neural networks are often used to solve problems that cannot be solved by using linear methods. For example, consider a complex classification problem with two classes, where the border between the two classes cannot be described using a linear function. An example is the XOR-operation, which could not be learned by the original perceptron, as there is no linear threshold function that can model the XOR-operation. However, by using a non-linear activation function instead of a linear function, Multilayer Perceptrons are able to learn or model this operation. The non-linearities enable us to make an approximation of functions that can be arbitrarily complex [2]. There are three frequently used activation functions [2]:

Sigmoid function. The logistic sigmoid function is defined as $\sigma(z) = \frac{1}{1+e^{-z}}$ and maps z between 0 and 1. This function is used especially in output units in binary classification problems [19]. This activation function is not used a lot within the units

of the hidden layers. When we look at the graph of the function (figure 5.6a), we see that towards the "ends" of the S-curve, the amount of change decreases. This is also visible if we look at the derivative of the sigmoid (in red), which converges to zero when z goes towards -1 and 1. This is not very helpful in the learning process, as no learning takes place when the derivative is (close to) zero (see section 5.2.2). However, this function is used in LSTM units, discussed in section 5.3.2

Hyperbolic tangent. This function is defined as $\tanh(z) = \frac{\sinh(z)}{\cosh(z)} = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{e^{2z}-1}{e^{2z}+1}$. It is very similar to the sigmoid function, and has a similar S-curve, as can been seen in 5.6b. However, it maps z between -1 and 1 and is steeper than the sigmoid. The same learning problem is present here, as can been seen in the graph. This activation function is used in RNN and LSTM units.

Rectified Linear Unit (ReLU). This function is defined as $g(z) = \max(0, z)$. This function is linear for positive inputs, and zero for negative inputs (see figure 5.6c). This linearity makes them suitable for learning, as the derivative is always 1 when z is positive. Therefore, the ReLU is not prone to the problems of the sigmoid and the hyperbolic tangent, and therefore, they are nowadays the recommend choice as activation function [19].



(a) Sigmoid **(b)** Hyperbolic tangent **(c)** Rectified Linear Unit (ReLU)

Figure 5.6: The plots show the three most commonly used activation functions, the sigmoid $\sigma(z)$, the hyperbolic tangent $\tanh(x)$ and the Rectified Linear Unit. The activation functions have been plotted in blue and their derivatives in red.

5.3 Sequence modeling

In situations where time and order is present, the information of earlier moments can be of importance to say something about later time steps. If we consider a language model where we want to predict the next word, the choice does not only depend on the previous word or even the current sentence, but it may also depend on words uttered in previous sentences or paragraphs. Sequences can be modeled using MLPs, but this architecture has some practical issues: MLPs have a fixed input size; the size of the input

layer needs to be the same for every prediction. For long sentences or even paragraphs as input, the input layer needs to be very big. Often sequences have a variable length, and due to the fixed size of the input layer, this cannot easily be represented in the network.

Furthermore, in sequences, several patterns may not always be present on the same place in a temporal ordering. Take the language model as an example; there are sometimes several options for word order that are equivalent (i.e., the meaning of the sentence remains the same). Alternatively, in vital signs time series prediction, some curve characteristics can have a variable length. Even though their exact manifestation or time of appearance may differ, they still can cause some similar effects in the curve that we aim to predict. These are all things that are difficult to model with MLPs, as each time step has its fixed place in the network, and therefore, the automatically derived features that are represented in the network also. So if these patterns can happen at multiple time steps, then they have to be learned for each time step, which makes learning more difficult.

Sequence modeling can, in theory, be done using MLPs, for example, by increasing the number of hidden units and corresponding weights and padding the inputs, but this is a very cumbersome process. Ultimately, a machine learning method is desired that can do the following [50]:

1. Handle variable length sequences
2. Keep track of temporal ordering
3. Handle long term dependencies
4. Share parameters across the sequence

5.3.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a class of neural networks that are especially suitable for processing sequential data [19]. They were first described by Rumelhart et al. [45]. The main RNN structure is shown in figure 5.7. The loop, and when unfolded the links between the time steps, allow information to flow from previous time steps to later ones. By unfolding the network, a process visualized in figure 5.7, we can see how the network can handle variable length sequences. Suppose we have an input vector $\boldsymbol{x}_t$ for t time steps. Instead of representing the network as a recurrence (left-hand side), we can unfold this recurrence (right-hand side), by drawing a component for each time step t. Each instance of $\boldsymbol{h}_t$ represents the current state of the network, which can be modified by each input $\boldsymbol{x}_t$. We can also see that the network can keep track of temporal ordering, as information flows from earlier time steps to following steps, and not the other way around. The network can be defined using the following equation [19]:

$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t; \boldsymbol{\theta}) \tag{5.11}$$

When the network is trained, the network uses a fixed size vector $\boldsymbol{h}$ to encode information from previous time steps. Because of this fixed size, not all information can be

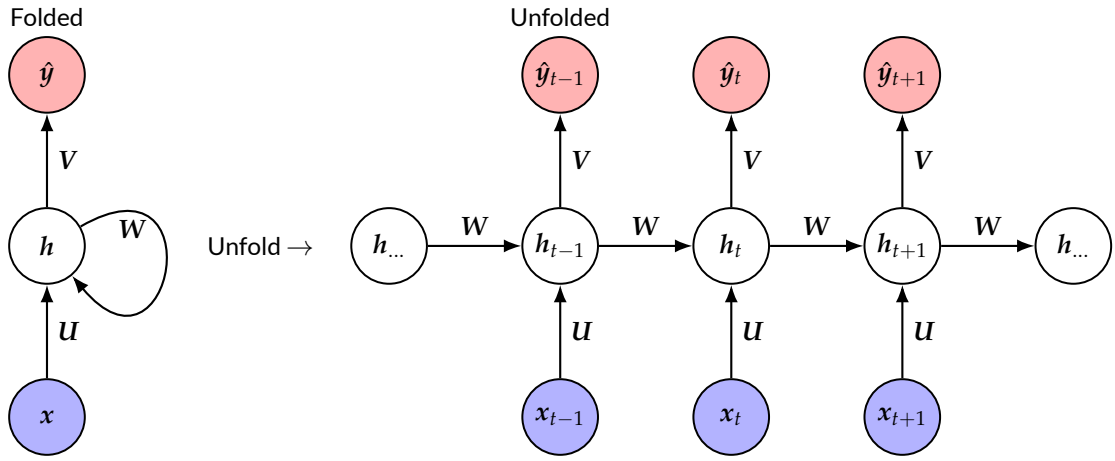


Figure 5.7: A recurrent neural network visualized as a computational graph. The weight parameter $\boldsymbol{\theta}$ is composed of three weight matrices; a matrix $\boldsymbol{U}$ for parametrizing the weights between the input and the hidden RNN layer, a matrix $\boldsymbol{W}$ for the weights between the recurrent units, and finally a matrix $\boldsymbol{V}$ that parametrizes the weights between the recurrent units and the output units. This figure is a simplified version of figure 10.3 in [19].

saved, as the number of previous time steps is arbitrary. By adjusting the weights $\boldsymbol{W}$ between the time steps' components, it is determined how information from previous time steps flows further to successors.

Back and forward propagation

In section 5.2.1, we described how information from the inputs flows through the network to produce an output; this process was called forward propagation. We can apply the same principle here, by chronologically traversing $\boldsymbol{x}_t$. Provided we have an initial state $\boldsymbol{h}_0$, the following equations will be performed for every time step t [19]:

$$\boldsymbol{a}_t = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t \tag{5.12}$$

$$\boldsymbol{h}_t = \tanh(\boldsymbol{a}_t) \tag{5.13}$$

$$\hat{\boldsymbol{y}}_t = g(\boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}_t) \tag{5.14}$$

where

- $\boldsymbol{b}$: bias
- $\boldsymbol{c}$: bias
- $\boldsymbol{U}$: weights between the input and recurrent units
- $\boldsymbol{W}$: weights between the recurrent units
- $\boldsymbol{V}$: weights between the recurrent units and the outputs
- $g(z)$: activation function like ($\sigma(z)$, $\tanh(z)$ or ReLU)

The weight matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ are shared across all time steps. The *sharing of parameters* by the RNN ensures a reduction in the number of parameters in total that would

be needed compared to an MLP equivalent. This reduction makes learning easier. The weights of the recurrent neural network can be learned using a slightly modified version of the back-propagation algorithm, *Back Propagation Through Time* (for a detailed description of the algorithm, see section 10.2.2 in Goodfellow et al. [19]). A figure that illustrates how the errors propagate through the network is shown in figure 5.8.

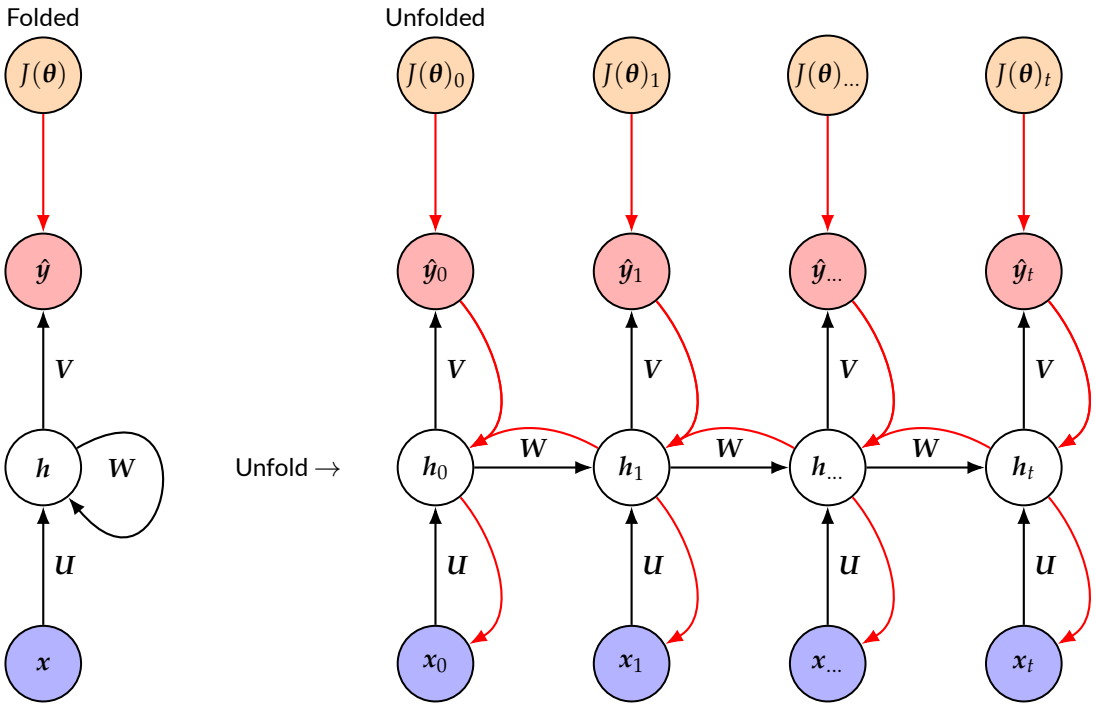


Figure 5.8: Back Propagration Through Time. Forward propagation is in black, back propagation is in red. This figure is based on a figure in [50], but adapted to the notation used in this work.

Long term dependencies and the problem of vanishing gradients

In some cases, early moments in history can have a significant influence on later moments. We have learned from the results in section 4.5.2 and 4.5.6 that the in VAR models, lag-30 still had some influence on the current measurement. Theoretically, RNNs should be able to deal with long term dependencies. In practice, however, this is not the case: when the gap between cause and effect increases, so does the chance decrease that a learning algorithm like Back Propagation will find the correct weight for a network that models this dependency. However, at the same time, by manually adjusting the weights, a human operator could model this dependency successfully in an RNN [19]. How is this possible?

Suppose we want to calculate gradient with respect to $\boldsymbol{h}_0$. We see in figure 5.8 that the gradient depends on many factors $\boldsymbol{W}$. When these factors in $\boldsymbol{W}$ are bigger than

1, then the gradient could become very large. However, when many values are smaller than 1, then the gradient may "vanish". These effects are both caused by the many times these factors are multiplied. We have seen that the number of times the chain rule has to be applied depends on how many nodes precede a node in the backward pass (causing these multiplications, see equations 5.7 - 5.9). As the gap between $\boldsymbol{h}_0$ and $\boldsymbol{h}_t$ increases, the number of chain rule applications increases as well, causing the gradient to become smaller and smaller (or bigger and bigger).

Because of this *vanishing gradient problem*, RNNs are unable to learn long term dependencies. Vanishing gradients make it difficult to find an improvement in the weights, because it is difficult to determine for small gradients which way the learning should go [19]. Bengio et al. [4] presented theoretical and experimental evidence that it is not possible to learn long term dependencies by performing gradient descent. This result would mean, as we do not want to rule out long term dependencies, that Recurrent Neural Networks are unsuitable for our problem. However, we can adapt the underlying architecture of an RNN to solve this problem. A popular RNN architecture that can deal with long term dependencies, called *Long Short-Term Memory networks* (LSTMs), will be discussed in the next section.

5.3.2 Long Short-Term Memory Networks

Long Short-Term Memory networks are an extension to the classical RNN architecture that we described in the previous section. There are two main differences between traditional RNNs and LSTMs:

Cell state. Besides the links $\boldsymbol{h}_t$, the hidden state, that RNNs also have, there is another layer called the cell state $\boldsymbol{c_t}$. The main difference between these two is that the cell state is not influenced by weights $\boldsymbol{W}$ between the units. Therefore, information that was provided in early time steps does not suffer from the vanishing gradient problem. The inputs and the hidden state can, however, influence the information stored in the cell state through the use of gates.

Gates. The original RNN architecture had one neural network unit per recurrent unit with a $\tanh(z)$ activation function (see equation 5.13). The LSTM has four neural network units that serve as gates. These gates may add and subtract information from the cell state, and finally, determine the output for the current cell.

Figure 5.9 shows a single LSTM cell. Just as the traditional RNNs, the LSTMs are chained. Because of the two states, there are two links between the recurrent units. We can describe an LSTM network using the following recurrence relation.

$$\boldsymbol{h}_t, \boldsymbol{c}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{c}_{t-1}, \boldsymbol{h}_t; \boldsymbol{\theta}) \tag{5.15}$$

There are four gates per LSTM cell. Here will follow a description of all four of these:

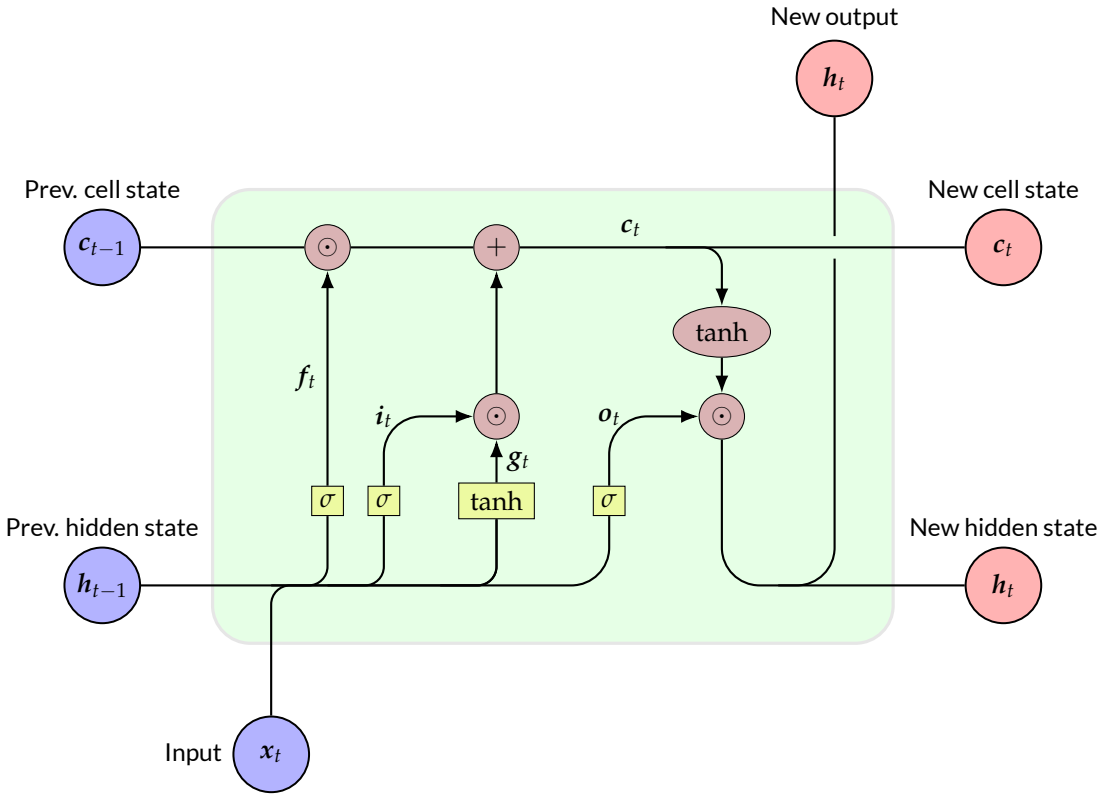


Figure 5.9: A single Long Short-Term Memory cell (in green). The yellow rectangles denote a neural network layer. The purple circles and ellipse denote point-wise operations. Joining arrows describe the concatenation of two tensors, while diverging arrows describe the copying of information. This figure is based on the figures in [37], but adapted to the notation in this work.

Forget gate. The forgetting of information may seem counterintuitive to learning. However, this is not the case. Recall from the original RNN architecture that the tensor or vector holding the hidden state $\boldsymbol{h}_t$ has a fixed size. The same holds for the cell state and, therefore, it is not possible to keep all information in the cell state. By using this gate, the network can decide to forget some information. Depending on the context, information can become irrelevant. For example, a spike in the heart rate could be so critical that the information in the cell state describing a previous steady curve becomes irrelevant. When this spike occurs, the network can decide to forget this steady curve. Formally, the forget gate can be described using the following equation:

$$\boldsymbol{f}_t = \sigma(\boldsymbol{U}_f \boldsymbol{x}_t + \boldsymbol{W}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \tag{5.16}$$

We know from section 5.2.3 that the range of the sigmoid is $[0, 1]$. The vector $\boldsymbol{f}_t$ has the same dimensions as the the cell state, and it will contain for every element in $\boldsymbol{c}_{t-1}$ a value between 0 and 1, encoding how much information should be kept [37].

Input gate. We can also store new information in the cell state. Depending on the context and the input, the input gate decides which information is relevant, so in our case, this would be the information describing this spike in the heart rate. The input gate decides which pieces of information of the hidden state and the current input combined will be supplied to the cell state. The input gate is defined using the following equation:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{U}_i \boldsymbol{x}_t + \boldsymbol{W}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \tag{5.17}$$

Because of the sigmoid, the vector $\boldsymbol{i}_t$ contains for every element of the concatenation of $\boldsymbol{h}_{t-1}$ and $\boldsymbol{x}_t$ a value between zero and one, describing which information will be transferred to the cell state.

Update Gate. The information in the cell state can be represented in a different way than the way information is stored in the inputs and hidden states. The neural network layer with the the $\tanh(z)$ activation function creates candidate values for the cell state [37]; the information contained in both $\boldsymbol{h}_{t-1}$ and $\boldsymbol{x}_t$ is put between the values -1 and 1.

$$\boldsymbol{g}_t = \tanh(\boldsymbol{U}_g \boldsymbol{x}_t + \boldsymbol{W}_g \boldsymbol{h}_{t-1} + \boldsymbol{b}_g) \tag{5.18}$$

After calculating $\boldsymbol{g}_t$, the cell state can be updated using the following equation:

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t \tag{5.19}$$

First, we forget the information according to the vector $\boldsymbol{f}_t$, which was supplied by the forget gate, by point-wise multiplying ($\odot$) the two vectors. Then, the new candidate values are added to the cell state. Which information gets updated was decided by the input gate ($\boldsymbol{i}_t$). If we look again at our example, the cell state now contains updated information about the spike.

Output Gate. We have now updated the cell state, but we still have not given an output. The output gate decides what information from the cell state flows back to the new hidden state and output of that LSTM unit, given the current input and previous hidden state:

$$\boldsymbol{o}_t = \sigma(\boldsymbol{U}_o \boldsymbol{x}_t + \boldsymbol{W}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \tag{5.20}$$

The vector $\boldsymbol{o}_t$ is then a vector that decides for each element in the cell state if it will be transferred to the hidden state and output. Before transferring, each value in the cell state is transformed by applying the tanh function point-wise. Then the new hidden state is determined using the following equation:

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \tag{5.21}$$

Note that the matrices $\boldsymbol{U}$ and $\boldsymbol{W}$ that contain the weights are different for each gate (hence the subscripts in the equations). Therefore, the behavior of each gate can be unique.

LSTMs meet all the four requirements discussed earlier. Chung et al. [9] show that gated networks like LSTMs outperform traditional RNNs (an alternative is the *Gated Recurrent Unit* (GRU), which performs on par with LSTMs). All these arguments combined make this architecture a good candidate architecture for the prediction of future vital sign time series.

5.3.3 Convolutional Neural Networks

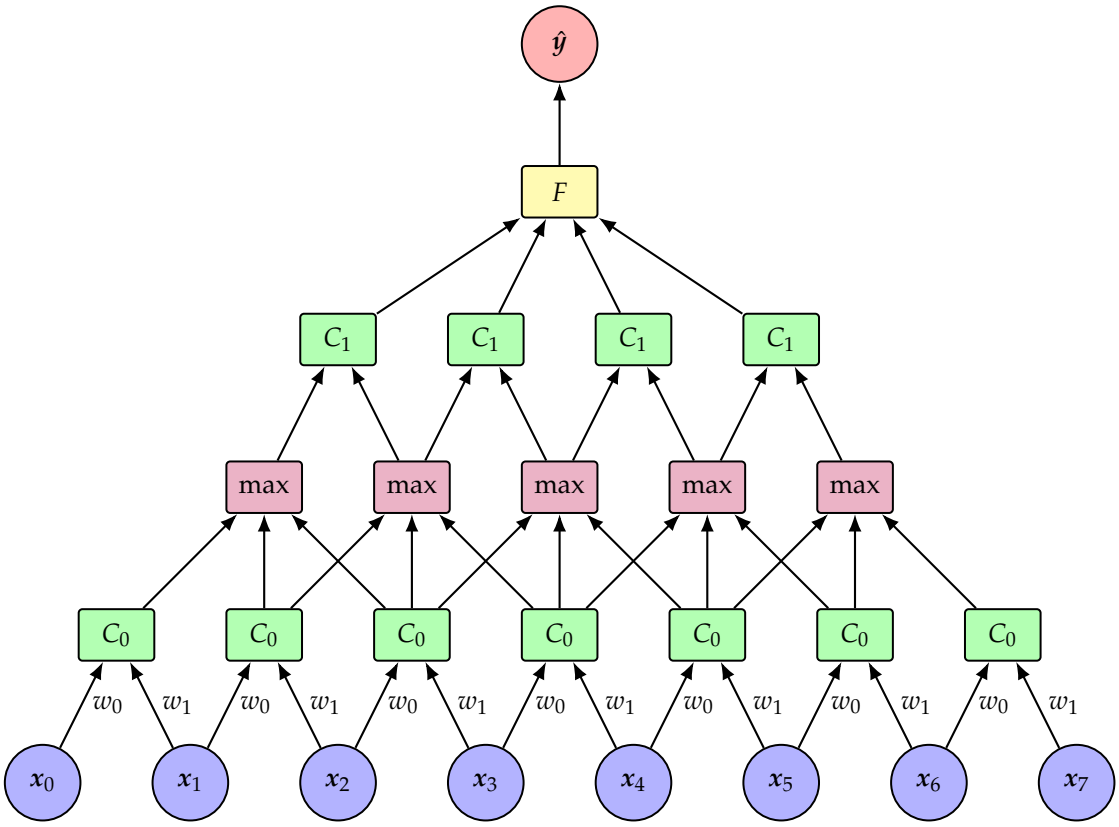


Figure 5.10: A sample one-dimensional Convolutional Neural Network. There are two convolution layers (green), interspersed by one max pool layer. The last convolution layer is connected to a fully connected layer F, which will determine an output $\hat{y}$. Based on a figure in [36], but adapted to our notation.

Besides Recurrent Neural Networks like LSTMs, there are other choices for modeling sequences. *Convolutional Neural Networks* (CNNs) have also been used as a way to model sequences. Traditionally, CNNs have been used a lot for processing images. Convolutional Neural Networks share many properties with some techniques found in traditional image processing (e.g., kernels). Mostly, CNNs process 2D images, yet time series are typically one-dimensional. It is, however, also possible apply CNNs to one-dimensional data. In figure 5.10, a simple CNN is shown that works on time series data. CNNs generally consist of three different components or layers [36]:

Convolution layers consist of cells that look for features in an input patch. Note the difference between the fully connected layers of a Multilayer Perceptron (see figure 5.4) and the CNN; the input nodes are fully connected to the hidden nodes in the first, while this is not the case in the latter. The convolutions work on patches of the data, working on multiple time steps at once. In the figure, there are only two time steps that are combined; however, in reality, the number of connected time steps will be larger.

The convolution cells themselves are in fact, an array of nodes that are trained to detect different features [36]. In our case, these could be the detection of plateaux, spikes and dips in a time series. The presence of these features will be provided to the next layer. This could be either another convolution layer, a max pool layer or a fully connected layer.

Max pool layers. A max pool cell summarizes the information found by the convolutional layer below. For some features, we do not necessarily need to know when they happened, but only *if* they happened. The max pool cell picks the maximum value for each feature of the nodes below and provides it to the next layer. In this way, networks can work on larger series and detect more high-level features [36]. If we assume that $\boldsymbol{x}$ encodes a heart rate time series, the max pool layer in figure 5.10 will report the presence of a spike and dip in the four time steps that are connected to it.

Fully connected layer. This layer is, in essence, a hidden layer like those found in an MLP. The units will combine the features found in the convolution layers below. These layers can be stacked if desired. The (high-level) curve characteristics of the heart rate series can now be used to make a prediction. Given the series of high-level features, we can now produce an estimate for a future value of this series.

Convolutional Neural Networks also use parameter sharing; the convolution units in each layer share the same weights. They will perform the same operation on each patch of data, a behavior that is similar to a kernel in image processing. This ability makes learning easier. Besides parameter sharing, this architecture does also fulfill an other requirement that was discussed in section 5.3: CNNs are able to handle long-term relations. Relations between features are only made in the fully connected layer. As the weight of a feature that is based on $\boldsymbol{x}_0$ does not depend on a feature of $\boldsymbol{x}_t$,there is no temporal bias like we found in the Recurrent Neural Network architecture. However, CNNs are not able to handle variable length sequences. In our case, this does not necessarily pose a problem: given a fixed length history, we may be able to predict one or more future values for the vital signs.

5.4 Methodology

By using exploratory analysis methods, we aimed to discover relations between vital signs time series. In this part, we aim to create models that given a set of historical vital signs time series predict future values of these time series. In the previous section, we described two different Neural Network architectures that are suitable for time series prediction; Long Short-Term Memory Networks and Convolutional Neural Networks. However, there are still some choices left to be made in the configuration of these networks. In the following sections, we will describe which configurations are chosen and how they are assessed.

5.4.1 Inputs, outputs and network architecture

In the previous sections, we have seen two different neural network architectures that are suitable for time series prediction, namely Long Short-Term Memory Networks and Convolutional Neural Networks. However, we still have to decide how to precisely design the computational graph. There are several options concerning how information enters the network and how predictions are made. Considering the Recurrent Neural Network / LSTM architecture, there are several options for input and output [50]:

One to Many. Given a single time step of vital signs, we can produce multiple outputs. This approach is not suitable for our case, as we have seen that there are correlations between early time steps and newer ones (see section 4.5.6).

Many to One. Given multiple time steps of vital signs, produce a single output. This architecture is mainly used for classification of time series and is less suitable for prediction, as information provided in earlier predictions is not reused.

Many to Many. For each time step we provide to the network, an output is produced. This approach seems similar to one to one modeling, but this is not the case. We have seen in figure 5.7 and 5.9 that there are links between the recurrent nodes that transfer information from earlier predictions to later ones. The information that is provided in earlier steps can be used to predict later ones.

Sequence to Sequence. We take as input a (multivariate) time series. First, we produce an internal representation of that series. The part of the network that produces this representation is called the *Encoder*. Subsequently, this internal representation is provided to the *Decoder*, which produces the prediction of multiple time steps. In our case, this architecture transforms historic sequences into future sequences.

This architecture seems suitable for time series prediction; however, there are some limitations to this approach. One of the limitations is that the feature dimension of the input has to be the same as the output's dimension. In this work, the number of Vital Signs Time Series that we provide to the network will be larger than the

number of VSTS we want to predict. Moreover, we will provide static variables like age and some ventilator settings to the network. While we can also include these variables in the prediction, this will, probably, not make learning easier.

From these approaches, the *Many to Many* approach is the most suitable for modeling our problem, as information about earlier predictions can be reused, and we are not limited by the dimensions of the input and output. When we consider CNNs we do not have to make this choice; there is a *One to One* relation between an input (a history of vital signs) and the prediction.

Input structure

As described above, the many to many method provides an output for every input $\boldsymbol{x}_t$ that is given. Given the dataset, there are two choices on how to structure the input of each prediction:

Single Time Step Input (STSI). The vector $\boldsymbol{x}_t$ has k dimensions, containing the current values of k time series. Information about earlier time steps flows through the hidden state $\boldsymbol{h}$ and the cell state $\boldsymbol{c}$. A down side to this approach is that when there is not a lot of information recorded in the state, the outputs will be unreliable. During a *warm-up period*, the predictions will, therefore, be unreliable. After this period, the predictions should be reliable.

$$\boldsymbol{x}_t = \begin{bmatrix} \text{HR}_t & \text{ETCO}_{2t} & \cdots & P_t^{\text{peak}} \end{bmatrix} \tag{5.22}$$

Multiple Time Step Input (MTSI). Instead of a vector, we supply each input node with a matrix $\boldsymbol{X}_t$ with dimensions $h \times k$. Instead of only providing a single step, now information about a history of h time steps is already available through each input node. However, when LSTMs are used, information from previous predictions still flows through recurrent links to the current prediction.

$$\boldsymbol{X}_t = \begin{bmatrix} \text{HR}_{t-h} & \text{ETCO}_{2t-h} & \cdots & P_{t-h}^{\text{peak}} \\ \text{HR}_{t-(h-1)} & \text{ETCO}_{2t-(h-1)} & \cdots & P_{t-(h-1)}^{\text{peak}} \\ \vdots & \vdots & \ddots & \vdots \\ \text{HR}_t & \text{ETCO}_{2t} & \cdots & P_t^{\text{peak}} \end{bmatrix} \tag{5.23}$$

In section 5.3.3 we described how a Convolutional Neural Network can make predictions on a time series with a fixed size h. Therefore, the MTSI approach is also suitable to be used by a CNN. We can also combine a CNN with a LSTM by giving the output of the CNN to a LSTM. Therefore, the LSTM will work on high level features that describe the input data. These abstractions will enable the network to work on larger amounts of data.

Output structure

There are also several options for the output of the network. In time series forecasting literature, several forecasting methods are mentioned that may be suitable for our problem [29]:

Iterated forecast (IF). We predict the next time step for all our k input variables. The forecast is then used to predict the next step. Suppose we use MTSI and use the input of equation 5.23 as input. The desired output will be as follows:

$$\widehat{\boldsymbol{y}}_t = \begin{bmatrix} \widehat{\mathrm{HR}}_{t+1} & \widehat{\mathrm{ETCO}_2}_{t+1} & \cdots & \hat{P}^{\mathrm{peak}}_{t+1} \end{bmatrix} \tag{5.24}$$

The next input will include the prediction and lose the earliest measurement:

$$\boldsymbol{X}_{t+1} = \begin{bmatrix} \mathrm{HR}_{t-(h-1)} & \mathrm{ETCO}_{2\,t-(h-1)} & \cdots & P^{\mathrm{peak}}_{t-(h-1)} \\ \mathrm{HR}_{t-(h-2)} & \mathrm{ETCO}_{2\,t-(h-2)} & \cdots & P^{\mathrm{peak}}_{t-(h-2)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{HR}_{t} & \mathrm{ETCO}_{2\,t} & \cdots & P^{\mathrm{peak}}_{t} \\ \widehat{\mathrm{HR}}_{t+1} & \widehat{\mathrm{ETCO}_2}_{t+1} & \cdots & \hat{P}^{\mathrm{peak}}_{t+1} \end{bmatrix} \tag{5.25}$$

This cycle is repeated for the following predictions, until the desired prediction window is reached. When LSTMs are used, this method is also suitable with the STSI method. Of course, then only the prediction is supplied as input.

The prediction quality of further time steps mainly relies on the accuracy of the prediction of the one-period ahead model. If this is not the case, the rest of the predicted will be unreliable [29]. We suspect that the one-period ahead model will not be fully reliable. Therefore, we will not investigate Iterated forecasts.

Direct forecast with single step output (DFSO). We directly predict the value of time step $\boldsymbol{x}_{t+s}$, without predicting intermediate time steps. We do not necessarily need to predict the same number of parameters as the input, as the output will not be used in further predictions.

$$\widehat{\boldsymbol{y}}_t = \begin{bmatrix} \widehat{\mathrm{HR}}_{t+s} & \widehat{\mathrm{ETCO}_2}_{t+s} & \hat{P}^{\mathrm{peak}}_{t+s} \end{bmatrix} \tag{5.26}$$

Direct forecast with multi step output (DFMO). We directly predict the value of all the time steps from $t+1$ till s, returning a matrix of size $k_{\mathrm{output}} \times s$:

$$\hat{\boldsymbol{Y}}_t = \begin{bmatrix} \widehat{\mathrm{HR}}_{t+1} & \widehat{\mathrm{ETCO}_2}_{t+1} & \hat{P}^{\mathrm{peak}}_{t+1} \\ \widehat{\mathrm{HR}}_{t+2} & \widehat{\mathrm{ETCO}_2}_{t+2} & \hat{P}^{\mathrm{peak}}_{t+2} \\ \vdots & \vdots & \vdots \\ \widehat{\mathrm{HR}}_{t+s} & \widehat{\mathrm{ETCO}_2}_{t+s} & \hat{P}^{\mathrm{peak}}_{t+s} \end{bmatrix} \tag{5.27}$$

From a clinical perspective, the direct forecast with multi step output would be ideal, as more intermediate predictions provide more information to caregivers. We will, however, create both single and multi step output models, so we can compare their performance.

Summary

We will assess the performance of the following network architectures:

1. LSTM:
   (a) Single Time Step Input (STSI) - Direct forecast Single Output (DFSO)
   (b) Multi Time Step Input (MTSI):
      - Direct forecast Single Step Output (DFSO)
      - Direct forecast Multi Step Output (DFMO)
2. CNN + Multi Time Step Input (MTSI):
   - Direct forecast Single Step Output (DFSO)
   - Direct forecast Multi Step Output (DFMO)
3. LSTM + CNN + Multi Time Step Input (MTSI) - Direct forecast Multi Step Output (DFMO)

As CNNs cannot work on single time step inputs, we will not assess this combination.

5.4.2 Dataset

For our predictive models, we will again use the PC-IMV-Adaptive subset of the dataset, as described in chapter 3. Whereas in the exploratory analysis, not all the variables could be used, due to limitations of the VAR models, we can now use a broader range of inputs (see table 5.1). The static parameters *age* and *weight* are treated as time series (just like the other variables), even though they remain constant over time.

Downsampling

Esteban et al. [16] used a similar dataset, namely the eICU dataset [41], to predict alarm threshold violations and for the creation of synthetic vital signs time series. That dataset also records vital signs time series; however, it does not contain the series of vital signs that were measured by ventilators. Besides the different set of variables, the time series were also recorded in a different frequency; one measurement every 5 minutes. The classifiers that were trained on that dataset predicted future alarm rule violations quite successfully [16]. We can mimic this dataset by downsampling the data by only storing the median of a window of five minutes. We will compare the predictive quality between the networks that are trained on the original dataset and the adapted dataset.

Measured Variables	**Settings**	**Static Parameters**
End-tidal CO_2 *(*$ETCO_2$*)*		Age
Expiratory Tidal Volume ($V_{T_{\text{exp}}}$)	Tidal Volume	Weight
Fraction of inspired Oxygen (FIO_2)	FIO_2	
Respiratory Frequency (*Ventilator*)	Respiratory Frequency	
Respiratory Frequency (Monitor)		
Heart Rate (*HR*)		
IE – Ratio	IE – Ratio	
PEEP	PEEP	
Peak Pressure (P^{PEAK})		
Saturation (SPO_2)		
	Inspiration Time	
	Inspiration Rise Time	

Table 5.1: The input and output variables that are used in the prediction models. The settings and measured variables that directly correspond to each other are placed on the same line. The target variables (the ones that we aim to predict) are marked in *italics*.

Patient specific subsets

Instead of training on the whole dataset, we can see if we can improve predictive quality by building a model that is suitable for only a subset of the patients. We should know beforehand to which subset the patients belong; therefore, we can only select on static parameters. We will see if training on a subset of patients does improve predictive quality. As a subset, we will take the patients that are 40 days or younger at the moment of admission, which is a more homogenous subset of patients, which is still sufficiently large to produce a reasonable amount of training data.

Scaling

Initial experimentation on the unaltered dataset did not provide good models, nay, often training failed when the amount of training data was increased. The different, large and small, ranges in which the individual vital signs are represented caused increased weights w in $\boldsymbol{W}$ ($w \geq 1$), which, in turn, resulted in the *exploding gradient problem* (see section 5.3.1). Exploding gradients make learning unstable [19]; in our experiments, symptoms of this instability were (impossible) negative values in the loss/cost function during training. We scaled all the values of the whole dataset between zero and one, according to the global minimum and maximum of that variable to mitigate this problem. The following equation expresses the scaling method:

$$x'_{i,p,t} = \frac{x_{i,p,t} - \min(\boldsymbol{x}_i)}{\max(\boldsymbol{x}_i) - \min(\boldsymbol{x}_i)} \tag{5.28}$$

where

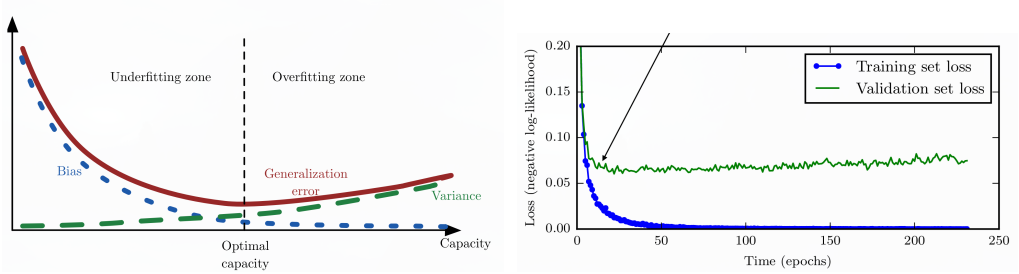
i : column (variable) index
p : patient identifier
t : time index
$\boldsymbol{x}_i$: column vector

We do not let the scaling depend on the patient, like in the Graphical VAR models, as the system should be able to deal with new patients for which we do not know the maximum and minimum values beforehand. The scaler's parameters are saved so that new patients can be scaled using the same method.

5.4.3 Achieving generalization

In supervised learning, the dataset is often divided into two disjoint parts; a training set and a test set [19]. The training set is provided to the neural network during the training phase. In order to evaluate the model's performance, a part of the dataset is held out. The test set is then used to assess the model's performance on unseen data, to see if *generalization* is achieved. The *generalization error* refers to the error on the test set [19].

Usually, in time series forecasting, the division between test and training is made across the temporal dimension, where the first part is used for training and the second part for the testing of the model's performance [30]. The PICU dataset is a panel dataset, as it consists of a panel of patients. In our case, we will make the division on the panel axis, so that we can verify that the model also works when predictions are made for patients whose time series were not (partially) included in the training phase. By making the division on the patient axis, we aim to create a generic prediction model, that will work for a broad range of patients.



(a) Capacity **(b)** Early Stopping

Figure 5.11: The plots show how overfitting can occur during training. When the capacity of the network is too large or training continues too long, overfitting takes place. The arrows point to the optimal capacity and the ideal epoch to stop training. These figures are adapted from figures 5.6 and figures 7.3 in [19].

A part of the training data will be held out for validation. During training, we aim to minimize the loss/cost function. As we minimize the loss, we could reach a point

where we are overfitting to the training data, and the predictive accuracy on the test data will again decrease. After each epoch of training, we will calculate the loss function on the predictions on the validation set and store the model if it is an improvement on the validation loss of the previous epoch. We will do the same for the training loss, so we can compare these two configurations when the training phase has been completed. We can also stop training after x epochs of no improvement in the validation loss, a method that is referred to as *Early Stopping* [19]. We effectively achieve the same result by continuously saving the best performing model on the validation set; only we continue training to minimize the loss on the training set, in order to ensure that we do not stop to early in a phase were the model is still underfitting. We stop training when the training loss function has not been improved for over 100 epochs of training.

Early Stopping is a *regularization method*. Regularization is defined as any alteration of a machine learning method that aims to reduce the generalization error, while still maintaining the same learning error [19].

We will divide the patients as follows: the training set will contain 80 % of the patients and the test set will contain 20 %. From the training set 70 % will be used during training and 30 % for validation.

During training, we will use time series of at least five hours of uninterrupted measurements. From a time series we will select a window of four hours, by picking a random time step t_{start} that will still allow a window size of four hours. During each epoch of training, a batch of n randomly selected windows will be fed as training data.

5.4.4 Hyperparameters and Regularization

Besides the weights, there are several other parameters of a neural network. These parameters are not learned through gradient descent but are specified beforehand. These *hyperparameters* control the behavior of the learning method and the capacity of the network or apply some regularization [19]. In our case, there are several hyperparameters that we have to specify:

The LSTM dimensionality. We can specify the size of the hidden state $\boldsymbol{h}_t$. When we increase this parameter, more information can be stored. This hyperparameter says thus something about the network's capacity. This parameter is specified for each LSTM layer individually. This parameter is often referred to as the number of LSTM units, a misnomer, as it does not say anything about the number of LSTM cells in each layer.

Number of filters in the Convolution Layers. In section 5.3.3, we described how each CNN cell determines the presence of a couple of features on the patch of data on which it is working. This hyperparameter determines the number of features that each cell detects. This parameter is specified for each convolution layer individually. Again, this hyperparameters says something about the capacity of the network.

Kernel size and Pool size. The kernel size determines the size of the input patch of each convolution cell. The pool size determines the same for each pooling unit. A smaller kernel size results in more fine-grained features, while an increased kernel size will result in more high-level features.

Learning rate. In section 5.2.2, we discussed how the learning rate affects the learning process; if this parameter is not set correctly, we might miss the optimal weights of the network. Usually, the optimal learning rate depends on the phase of learning. We can adapt the learning rate through an *optimizer*. In the beginning, we learn faster, and once we reach a certain phase (the increase in loss function slows done), the learning rate becomes smaller. Besides this, the Adam optimizer [25] adapts the learning rate for each weight. Ruder [43] lists several optimizers and gives an overview of their performance. Adam outperforms in most of the cases other optimizers. Because of this, we will use the Adam optimizer to control the learning rate.

Dropout is a regularization method, introduced by Srivastava et al. [51]. When dropout is applied to a layer, a number of its nodes are removed. A scalar value p (between zero and one) determines for each epoch of training the chance that a node is kept. This method counteracts overfitting of the different nodes in the network, as the final prediction will not rely on a few nodes [19]. During training, each epoch the network had a different layout. These networks can be seen as an *ensemble* of networks, of which we average the prediction. This approach is, however, expensive. Alternatively, we will multiply all the layer's outgoing weights with p. This method is an approximation of averaging, as we will get the expected output of the node [51]. Besides the application of dropout on layers, it is also used between recurrent units. This is referred to as recurrent dropout.

5.4.5 Evaluation

After a model is fitted, we have to assess its predictive quality. During training, this is assessed through the use of a loss function. We use the *Mean Squared Error* metric as loss function [19]:

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 \tag{5.29}$$

where

m : the number of predictions

The MSE measures the error between the predicted $\hat{y}$ and the actual data points y. This error is squared for two reasons:

1. The error should always be positive; otherwise estimations that are too low cancel the estimations out that are too high (because of the summation).

2. Wrong estimations that are close to the actual value should not be punished as much as the estimations that are far from the actual value. By squaring, this is achieved as the slope increases linearly.

By taking the root of the MSE, we obtain the *Root Mean Squared Error* (RMSE), which describes the error in the original unit of measurement. For the STSI case, we slightly change the MSE loss function: we will calculate the MSE for each series, but we do not take the ten first predictions into account. During the warm-up period, predictions are unreliable, as no history is available through the hidden and cell state. By conceding to this fact, we aim to find models that will make accurate forecasts after the warm-up period has ended, and shift the focus during training to after this period.

In addition to the MSE, there are two alternative metrics that are frequently used in time series forecasting literature to assess the predictive quality of the models:

Mean Absolute Error (MAE). This method is very similar to the MSE metric, but instead of squaring the error we take its absolute value. This results in the fact that large errors are not punished more severely than small errors.

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}|\hat{y}_i - y_i| \tag{5.30}$$

Mean Absolute Percentage Error (MAPE). This metric enables one to assess the predictive accuracy without knowing the scales of the different parameters. It is so to say, scale invariant. It will describe the average error percentage. However, there are is one main issue: when the actual value y_i that we aim to predict is zero, we cannot calculate this metric, as we cannot divide by zero (a value that is absolutely plausible for some parameters). Therefore, we cannot use this method on all the predictions. We can, however, assume that the number of actual values that are zero is negligible, and therefore, their contribution to the average error too. After ignoring these predictions, we can calculate this metric using the following equation:

$$\text{MAPE} = \frac{100}{m}\sum_{i=1}^{m}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{5.31}$$

Persistence Model

Besides comparing trained models on the prediction error, we will also compare the models to a naïve model; the persistence model. This model is straightforward. We have seen in section 4.5.6 that there is a strong correlation between the previous and current measurement. The persistence model lets the prediction rely entirely on the last available measurement [34]:

$$y_{t+s} = y_t + \epsilon_t \tag{5.32}$$

where

s : the time step in the future

This model provides a baseline for predicting future values. Complex models like Artificial Neural Networks are only useful if they (at least) outperform the naïve model.

Plots

We will also visualize the predictions of the models. In this way, we can visually compare them to both the actual data and the forecasts of the persistence model. Furthermore, we can provide consecutive visualizations of multistep output models that show how these models transform the input to output, by plotting both the history followed by the prediction. In this plot, we will also visualize the actual data so we can assess the predictive quality. Consecutive plots allow us to see how the model's predictions change over time. Moreover, this approach reflects how a predictive model could be applied in practice.

5.4.6 Implementation

All these networks will be implemented through the *Keras* library (with TensorFlow backend) in the *Python* programming language. By using a *Tensor Processing Unit*, a GPU designed for Machine Learning, we can achieve faster training times and process more data than using the CPU. The training will take place on a Virtual Machine that is hosted on the *Google Cloud Platform*. The machine that was used had 4 CPU's, 24 GB of RAM and an NVIDIA Tesla K80 Accelerator, which provides 12 GB of memory.

5.4.7 Comparison to related work

In previous sections, we already cited and mentioned the article "Predicting responses to mechanical ventilation for preterm infants with acute respiratory illness using artificial neural networks", by Brigham et al. [7]. This article describes a study conducted at Durham University and the neonatal intensive care unit of "University Hospital of North Tees" in Stockton-on-Tees, UK. In this section, we will briefly summarize this paper and describe how it relates to our work and methodology.

Background

Preterm infants (neonates) are susceptible to respiratory illness, partly because the lungs are often underdeveloped at birth. Mechanical ventilation is, therefore, usually applied to neonates after birth [49]. Various ventilator types and methods are currently being tried on patients, in order to determine which method is most suitable; in other words, finding the method which achieves the most optimal results for this patient. The overall goal of the study was to reduce the need for *in vivo* testing by using deep learning models that ideally predict the most suitable method. As a first step, they investigated whether

deep learning methods can predict the response of the current ventilation method, by forecasting an indicative vital sign. Motivated by the assumption that the Vital Signs Time Series are non-linear, Brigham et al. opted to use Long Short-Term Memorys to model time series.

Data collection

Brigham et al. used data from four different ventilation methods (terminology cf. [8]):

1. PC-CSV-Adaptive
2. PC-CMV-SetPoint
3. PC-IMV-Adaptive
4. PC-IMV-SetPoint

The data were collected as follows: one of the four modes was applied for 30 minutes the data generated by the ventilators and monitors were recorded. This measurement period was followed by a 15-minute washout period, after which a new measurement period was used for another mode. From each period of measured data, a window of 10, artifact-free, minutes was selected. The following parameters were measured every 1.5 seconds:

- Inspiratory Tidal Volume
- Spontaneous Minute Volume (the amount of air that a patient expires during patient triggered breaths).
- Respiratory Frequency
- FiO_2
- Pressure
- Compliance

The settings in the patients were fixed at an Expiratory Tidal Volume of 4 - 6 ml/kg and the inspiratory time between 0.25 and 0.35 seconds. The patient cohort consisted of 10 preterm babies, who were born after less than 30 weeks of pregnancy.

Model

Brigham et al. used LSTMs to create a prediction model. Based on five time steps of historical data $\{t_{-4}, t_{-3}, \ldots, t_0\}$, they predicted t_1 (1.5 seconds ahead), where t_0 is the current time step. They also created models that predicted t_{10} (15 seconds ahead). For this prediction, the history there goes back to t_{-9}. Only the Minute Volume vital sign was predicted (this variable is related to the the Expiratory Tidal Volume in our dataset,

see section 2.2.2). A network containing one hidden LSTM layer was used to predict t_1. For t_{10}, two architectures were tested; the first was aforementioned one layer architecture, and the second was a network with an additional hidden layer.

Results

The predictions of the next time-step ahead model seemed to be successful at first glance; the model has a higher accuracy as the error-metric RMSE is low (in terms of the original unit of the Minute Volume). However, if we look at the plots that compared the prediction to the actual data, it seems that the predictions seem to be lagging. For the t_{10} models, the results are worse in terms of the RMSE-metric. After adding a second LSTM layer, the results of the model improved; the resulting RMSE was lower, but not as low as the t_1 model. However, in our opinion, we see the same lagging behavior in some of the plots.

Relation to our project

The context of both projects is similar, and therefore, the projects share have many similarities. In our dataset, more patients are included, whereas, in their work, the patient cohort was limited to a more homogenous group: the patients were of similar age and had similar complications. Furthermore, the data was collected and selected systematically, which is not the case in our dataset. The most significant difference is the frequency of measurements, which is much higher in their dataset.

We have implemented their architecture to see how this architecture will perform on our dataset; we refer to their architecture as the LSTM-MTSI-DFSO model. In the reflection of the results, the authors note that the LSTM networks have trouble keeping up with frequent changes. This is one of the reasons that we will also try if the CNN architecture may be a better fit. In their models, no static parameters were included like age and weight. The authors expect that by including this the accuracy of the model improve will improve. In our models, these variables will be included, so if their hypothesis is correct, we might see some improvement compared to their models. The use of Dropout and other (deeper) architectures is also suggested, and we will experiment with both of them.

Their paper lacked an overview of the model's performance on the whole dataset/test set; only the model's performance on some patients was provided. By following the evaluation procedure provided in section 5.4.5, we aim to provide a report of the results that is more indicative of how the models perform in general.

5.5 Results

In this section, the results of our experiments, as discussed in the previous section, will be discussed. As the number of configurations is large, only a subset of the results is

reported here. Only the best-performing, category representative, or otherwise special models are reported for each category. We will report for each model the following:

- The configuration; architecture, input and output specifications, and hyperparameter values
- The performance per vital sign according to the following four metrics:
  - Mean Squared Error (MSE)
  - Mean Absolute Error (MAE)
  - Root Mean Squared Error (RMSE)
  - Mean Average Precision Error (MAPE)
- Visualizations of the predictions compared to the true signal

The dataset contains data of multiple patients, who are partitioned over the training, test, and validation sets. We call calculate these metrics for all the vital signs separately, as there are scale differences between them. Because of the presence of a patient axis, there are two ways to evaluate these metrics:

1. We calculate these metrics per patient and aggregate the results by averaging. In this way, we calculate the mean of means.
2. We ignore the cross-sectional axis and calculate the metric on the whole dataset.

We choose to go for option **two**, as we believe this is a more accurate way to describe these metrics. Moreover, during training, this is also the method in which the loss function is calculated. Also, the square root relation between the RMSE and MSE metrics is kept intact, whereas when the first method is being used, this is not the case.

We will first discuss the results of the LSTM models in section 5.5.1, followed by the CNN-based models' results in section 5.5.2. Furthermore, we will assess the performance of the combined LSTM-CNN architecture in section 5.5.3. Finally, in section 5.5.4, we will report on the results of some experiments where we adjusted the dataset by down-sampling and by only including a subset of the patients in the training set.

5.5.1 Long Short-Term Memory

The LSTM model is the only architecture with the option of a single time step input, as this option is nonsensical when used in combination with a CNN architecture. First, we will address the Single Time Step Input (STSI) approach, followed by the Multiple Time Step Input approach.

Single Time Step Input (STSI)

In this section we will describe the results of a LSTM network wit the following specifications:

Input:	1 time step for 19 variables
LSTM Layer 1:	No. of Units = 500, Dropout = 0.1
LSTM Layer 2:	No. of Units = 500, Dropout = 0.1
Fully Connected Layer:	6 units, one for each vital sign
Output:	1 time step, $t + 10$

The results for a forecast horizon of 10 minutes are shown in table 5.2. In terms of (Root) Mean Squared Error, there is an overall improvement over the naïve persistence model. However, in terms of the other error metrics, MAE and MAPE, this improvement is negligible or even absent. When we plot the predictions of both the models, we can see that the LSTM model has learned to mimic the naïve persistence model, the RMSE with respect to the naïve model is 2.37, while this is 6.14 with respect to the true signal. This behavior is also visible in the visualizations for the other vital signs (not shown). Furthermore, this behavior is also seen when we increase or reduce the forecast horizon.

Multi Time Step Input - Direct Forecast Single Step Output

In addition to the STSI models, we created MSTI models based on the LSTM-architecture. When we increased the number of input time steps, we also had to reduce the number of LSTM units, as the learning speed became untenably slow. In this section, we will show the results for an LSTM network with the following configuration:

Input:	60 time steps for 19 variables
LSTM Layer 1:	No. of Units = 100
LSTM Layer 2:	No. of Units = 100
Fully Connected Layer:	6 units, one for each vital sign
Output:	1 time step, $t + 10$

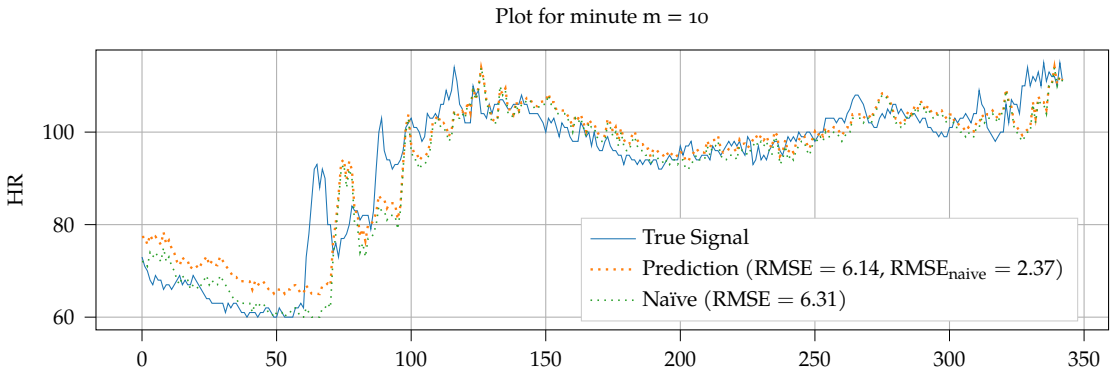
This model's results are shown in table 5.3. This model gives a slightly improved accuracy compared to the STSI model for all of the variables except the Expiratory Tidal Volume. The plots show that this model has also learned to mimic the persistence model. In the case of the P^{PEAK} variable (visualized in 5.3c), the variable which shows the biggest improvement in terms of the MAPE-metric, we see that the model can discern outliers/noise from the main signal, and the predictions will follow the latter more closely. However, the model still follows the persistence model, as the error between the persis-

	MSE	MAE	RMSE	MAPE
$ETCO_2$	6.81	1.49	2.61	3.66
$V_{T_{exp}}$	489.16	8.19	22.12	47.17
RF	12.68	1.97	3.56	5.37
HR	62.26	4.96	7.89	3.99
P^{PEAK}	8.50	1.71	2.91	12.26
SPO_2	4.07	1.02	2.02	1.13

(a) LSTM - STSI - DFSO, horizon = 10 minutes

	MSE	MAE	RMSE	MAPE
$ETCO_2$	8.07	1.49	2.84	3.64
$V_{T_{exp}}$	904.19	9.42	30.07	45.68
RF	18.82	1.86	4.34	4.82
HR	70.19	4.94	8.38	3.96
P^{PEAK}	13.58	2.01	3.68	13.56
SPO_2	5.57	0.94	2.36	1.05

(b) Naïve persistence model



(c) HR plot

Table 5.2: The LSTM-STSI-DFSO performance on the test set. The model has forecast horizon of 10 minutes.

tence signal and the model's prediction is smaller than the predictive error on the true signal. Just as in the STSI case, this behavior is also present when the predictive horizon is increased or decreased.

Multi Time Step Input - Direct Forecast Multiple Step Output

We adapted the above LSTM-MTSI-DFSO-architecture by increasing the number of output nodes. Each time step has its own output node in the network:

Input:	60 time step for 19 variables
LSTM Layer 1:	No. of Units = 100
LSTM Layer 2:	No. of Units = 100
Fully Connected Layer:	120 units (6×20)
Output:	20 consecutive time steps $(t_1, \ldots, t_{20})$, for 6 variables

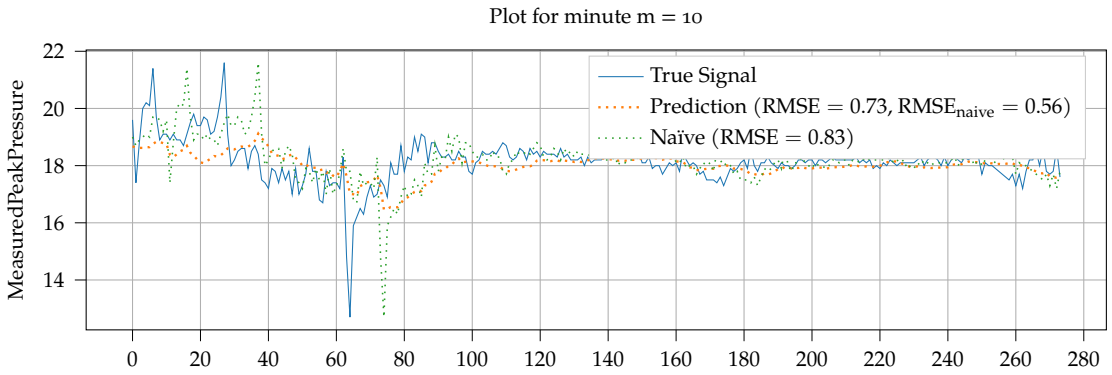
The results show (see table 5.4 that the model has trouble predicting four out of six variables (all except the heart rate and saturation); for example, there is a significant increase in the Mean Squared Error metric of the $ETCO_2$variable compared to the per-

	MSE	MAE	RMSE	MAPE
$ETCO_2$	6.31	1.46	2.51	3.53
$V_{T_{exp}}$	611.14	8.43	24.72	56.73
RF	12.12	1.82	3.48	4.91
HR	60.74	4.88	7.79	3.94
P^{PEAK}	8.03	1.66	2.83	11.09
SPO_2	3.64	0.96	1.91	1.06

(a) LSTM-MTSI-DFSO error metrics

	MSE	MAE	RMSE	MAPE
$ETCO_2$	8.07	1.49	2.84	3.64
$V_{T_{exp}}$	904.19	9.42	30.07	45.68
RF	18.82	1.86	4.34	4.82
HR	70.19	4.94	8.38	3.96
P^{PEAK}	13.58	2.01	3.68	13.56
SPO_2	5.57	0.94	2.36	1.05

(b) Naïve persistence model



(c) P^{peak} plot

Table 5.3: The results on the test set of the LSTM-MTSI-DFSO model.

sistence model (80.90 vs. 7.45). These increases in the error metric are caused by the fact that the model fails to learn some of the time steps correctly. For most of the time steps, all goes well. However, the model fails to learn the correct weights for 5 out of the 120 time steps and will predict a bogus constant value for all inputs (see figure 5.12). The model is still underfitting, and further training will not resolve this problem, as it persists after almost 100 epochs without improvement in the training loss.

We tested multiple configurations and alterations in the hyperparameter values for this architecture. However, the results were similar: the gaps in the prediction may happen at random places, but the presence of gaps could not be prevented. Moreover, despite all alterations, the model would still mimic the persistence model.

5.5.2 Convolutional Neural Networks

As said before, we only created models with the Multiple Time Step Input method for the Convolutional Neural Networks architecture. Considering the outputs, we created both models that predict a single time step of the output variables, and models that produce a multiple time step output.

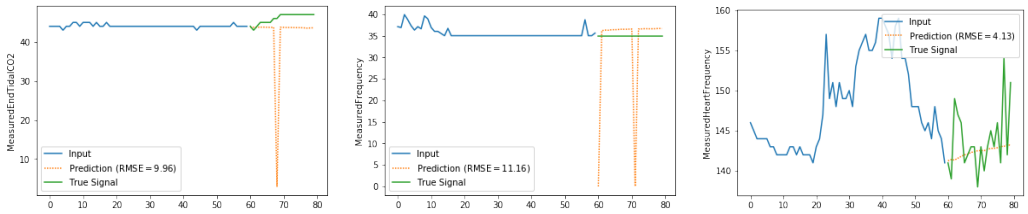
	MSE	MAE	RMSE	MAPE
$EtCO_2$	80.90	3.26	9.00	7.94
$V_{T_{exp}}$	613.75	9.33	24.77	55.90
RF	139.64	4.96	11.82	13.93
HR	61.83	4.82	7.86	3.84
P^{PEAK}	57.21	3.60	7.56	20.03
SpO_2	3.62	1.05	1.90	1.14

(a) Error metrics on the LSTM-MTSI-DFMO model

	MSE	MAE	RMSE	MAPE
$EtCO_2$	7.45	1.42	2.73	3.42
$V_{T_{exp}}$	1139.77	10.29	33.76	54.09
RF	17.83	1.78	4.22	4.69
HR	70.77	4.80	8.41	3.86
P^{PEAK}	12.77	1.95	3.57	12.70
SpO_2	4.96	0.91	2.23	1.00

(b) Error metrics on the Naïve Persistence model

Table 5.4: The results of the LSTM-MTSI-DFMO model.



(a) $EtCO_2$ **(b)** RF **(c)** HR

Figure 5.12: The results of the still underfitting LSTM-MTSI-DFMO model. The fails to learn the $EtCO_2$ time step t_9. For the RF variable, time steps t_1 and t_{12} were not well determined. However, for HR, weights were established for all time steps.

Direct Forecast Single Step Output (DFSO)

For the single time step case, we use a CNN model with the following configuration:

Input:	30 time steps for 19 variables
Convolution Layer:	Kernel size: 3, No. of filters = 256
Convolution Layer:	Kernel size: 5, No. of filters = 64
Fully Connected Layer:	200 units
Fully Connected Layer:	6 units, one for every vital sign
Output:	1 time step, t_{10}

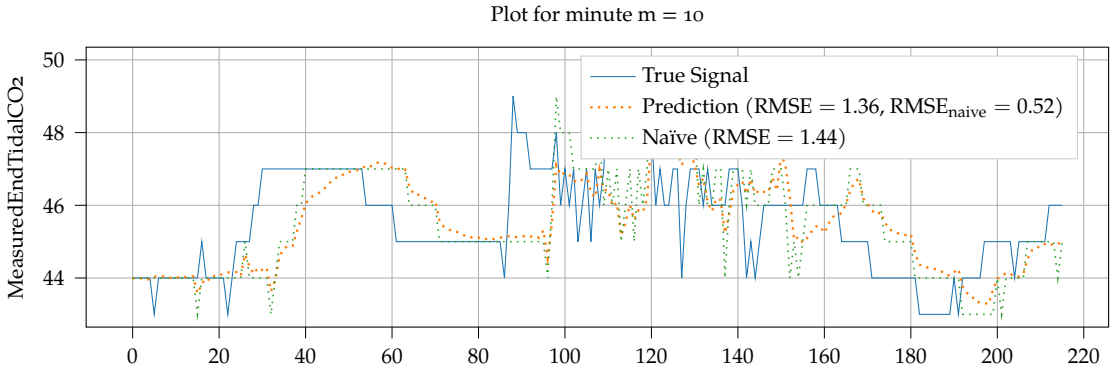
In table 5.5, the results are shown for the model with a prediction horizon of 10 minutes, so we can compare the results with the LSTM-STSI results in table 5.2. We have seen in section 5.5.1 that the LSTM-MTSI-DFSO model offers a minor improvement over the LSTM-STSI-DFSO model. The results of the CNN-DFSO model only provide a small decrease in error in terms of the MSE-metric over the LSTM-STSI-DFSO model, for some of the variables ($EtCO_2$, HR, P^{PEAK}). However, the CNN-DFMO only outperforms the LSTM-MTSI-DFSO model for the RF variable. Overall, there is no significant difference between the results of the three models (except for the Expiratory Tidal Volume). If we

	MSE	MAE	RMSE	MAPE
$EtCO_2$	6.73	1.50	2.59	3.80
$V_{T_{exp}}$	502.11	8.26	22.41	58.01
RF	12.89	1.73	3.59	4.58
HR	60.36	4.96	7.77	4.08
P^{PEAK}	8.41	1.69	2.90	11.79
SpO_2	4.11	0.96	2.03	1.08

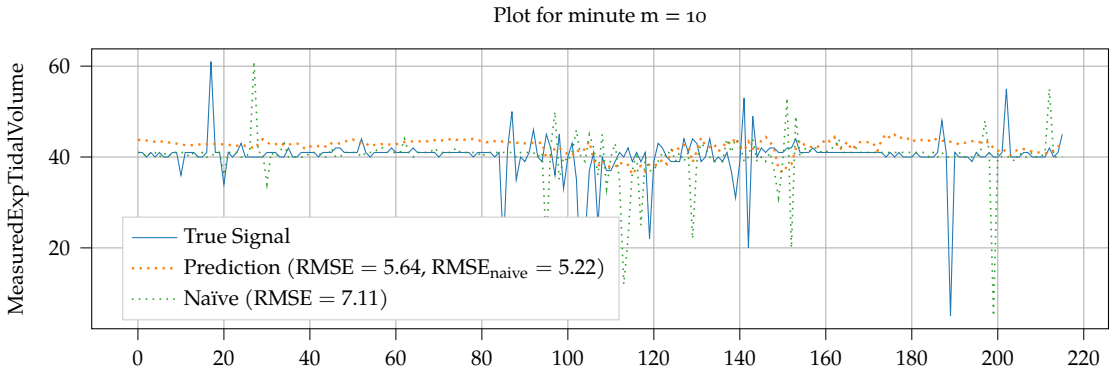
(a) The CNN model

	MSE	MAE	RMSE	MAPE
$EtCO_2$	7.88	1.48	2.81	3.63
$V_{T_{exp}}$	919.68	9.76	30.33	52.01
RF	19.44	1.87	4.41	4.91
HR	71.02	4.91	8.43	3.96
P^{PEAK}	13.39	2.01	3.66	13.37
SpO_2	5.70	0.95	2.39	1.05

(b) The naïve persistence model



(c) $EtCO_2$



(d) $V_{T_{exp}}$

Table 5.5: Forecasting with a horizon of 10 minutes using a CNN

look at the visualizations (in 5.5c), we can see that the model learned to use the last time step as a prediction, just as the LSTM models did. In terms of MAPE, the CNN-DFSO model outperforms the others for the RF variable. For the Expiratory Tidal Volume, we can see in 5.5d that it provides some generalization on the prediction, at least more than the other variables, as it does not religiously follow the curve with noisy spikes and dips as the predictions of the other variables do. Just as the LSTM-STSI-model, the CNN model offers a significant reduction of the MSE-metric (417) for the Expiratory Tidal Volume; however, this reduction is not reflected in the MAPE score. We will further discuss this phenomenon in section 5.6.1.

Direct Forecast Multi Step Output (DFMO)

We use for the multi step output almost the same configuration as in the single time step output case case:

Input:	30 time steps for 19 variables
Convolution Layer:	Kernel size: 3, No. of filters = 256
Convolution Layer:	Kernel size: 5, No. of filters = 64
Fully Connected Layer:	200 units
Fully Connected Layer:	120 units (6×20)
Output:	20 consecutive time steps $(t_1, \ldots, t_{20})$, for 6 variables

In table 5.6, the results of the model are shown for the CNN-DFMO model. The model is an overall improvement over the naïve model and the LSTM-DFMO model; however, when compared to the naïve model, the latter model is for some variables better in terms of MAPE. We can visualize the model's predictions in a similar way as in 5.5c and 5.5d, but this results in 120 plots. This type of plot cannot be used to visualize the behavior of the multi-step output prediction. Instead, we visualize the predictions as they would be used in a clinical setting; that is, in a plot which shows the history (input) of the vital signs followed by the predicted future values. We also plot the true signal so we can assess the quality of the prediction. Typical predictive behavior of the CNN-DFMO model is shown in figure 5.13. We have seen that the DFSO models seem to mimic the persistence model. This behavior is also present in the individual time steps of the variables that are predicted by the CNN-DFMO model (not shown). However, when we consider the ensemble $t_1, \ldots, t_{20}$ of DFMO predictions for a variable, this does not always seem to be the case: The persistence model's predictions can be visualized as a horizontal line $y = x_{t_0}$, where x_{t_0} is the last performed measurement. Predictions that follow this line are not present in the forecasts in figure 5.13. It seems that the start of the spike in the $\text{ET}CO_2$ curve is not foreseen until $t = 2$. After that, the model picks it up, but it seems to revert to some "mean" of approximately 45 mm Hg. The predictions are too low, but an approximation of the curve's true shape is visible in the predictions.

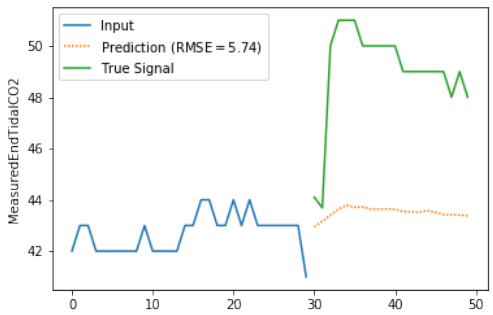
	MSE	MAE	RMSE	MAPE
$EtCO_2$	6.40	1.43	2.53	3.59
$V_{T_{exp}}$	539.11	8.32	23.22	49.33
RF	11.46	1.69	3.38	4.54
HR	54.50	4.61	7.38	3.75
P^{PEAK}	8.12	1.62	2.85	11.35
SpO_2	3.17	0.92	1.78	1.00

(a) CNN DFMO

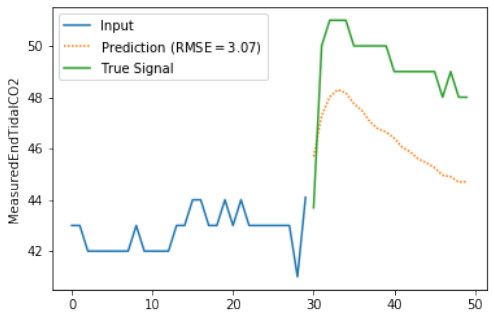
	MSE	MAE	RMSE	MAPE
$EtCO_2$	7.93	1.43	2.82	3.51
$V_{T_{exp}}$	1008.90	9.54	31.76	48.70
RF	17.20	1.77	4.15	4.68
HR	65.12	4.71	8.07	3.79
P^{PEAK}	13.13	1.96	3.62	13.03
SpO_2	4.65	0.88	2.16	0.97

(b) Naïve persistence model

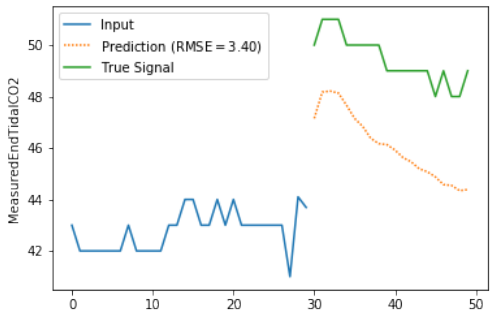
Table 5.6: The CNN-DFMO model. This model provides for every history of 30 minutes a prediction for the next 20 minutes.



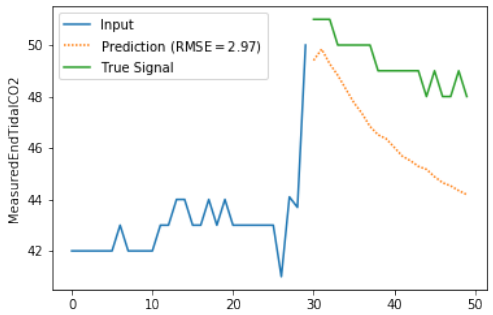
(a) t = 1



(b) t = 2



(c) t = 3



(d) t = 4

Figure 5.13: Four consecutive predictions of the CNN MTSI DFMO model are shown. It can be seen that the model does not just naively copy the last measured value, and it seems to be able to foresee a spike in the $EtCO_2$ curve. The estimates are, however, too low, and the predicted curve seems to revert back to some (pseudo) mean value.

5.5.3 LSTM + CNN

We only consider the Multi Time Step Input-method for networks that make use of the Convolutional Neural Network-architecture. For this combined architecture, we will only consider multiple time steps as the output method. This mixed architecture aims the use the benefits of both architectures. In this section, the results of the following configuration are discussed:

Input:	30 time steps for 19 variables
TD: Convolution Layer:	Kernel size: 3, No. of filters = 256
TD: Convolution Layer:	Kernel size: 5, No. of filters = 64
TD: Flatten	
LSTM Layer:	100 units, Dropout = 0.3
LSTM Layer:	200 units, Dropout = 0.5
LSTM Layer:	100 units, Dropout = 0.3
TD: Fully Connected Layer	120 units (6×20)
Output:	20 consecutive time steps $(t_1, \ldots, t_{20})$, for 6 variables

The convolution units are *Time Distributed* (TD); this means that a convolution unit provides the input for each LSTM cell.

The results in table 5.7 show that in terms of the MSE-metric, only the Expiratory Tidal Volume has a marginal improvement, while for the other variables the error increases. This increase is also visible in the other metrics. When we look at the visualizations in figure 5.14, the model seems to mimic the persistence model. The model predicts a straight line. The straight line appears to be based on some mean of the input data. We can see that as time progresses, the mean of the input values decreases, causing the prediction to go lower as well.

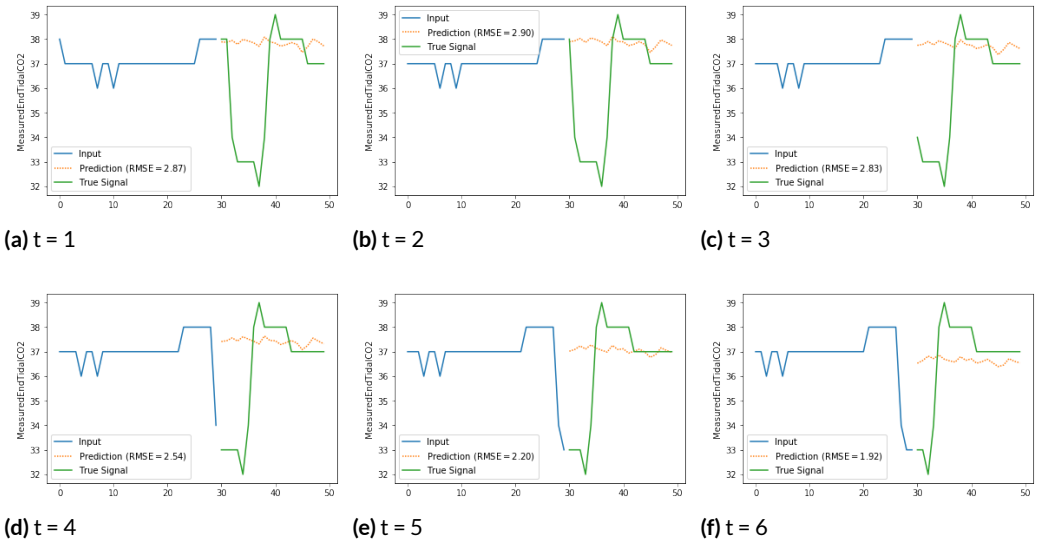
	MSE	MAE	RMSE	MAPE
$ETCO_2$	7.73	1.80	2.78	4.68
$V_{T_{exp}}$	525.86	9.68	22.93	51.11
RF	12.69	2.02	3.56	5.80
HR	64.86	5.30	8.05	4.35
P^{PEAK}	8.55	1.77	2.92	12.60
SPO_2	4.31	1.25	2.07	1.35

(a) LSTM - CNN

	MSE	MAE	RMSE	MAPE
$ETCO_2$	7.77	1.46	2.79	3.61
$V_{T_{exp}}$	916.01	9.41	30.27	46.68
RF	17.36	1.79	4.17	4.71
HR	65.95	4.68	8.12	3.77
P^{PEAK}	13.02	1.97	3.61	13.55
SPO_2	4.68	0.89	2.16	0.97

(b) Naïve

Table 5.7: The LSTM - CNN DFMO, forecasting the next 20 minutes



(a) t = 1 **(b)** t = 2 **(c)** t = 3

(d) t = 4 **(e)** t = 5 **(f)** t = 6

Figure 5.14: Six consecutive predictions of the LSTM - CNN MTSI DFMO model.

5.5.4 Dataset alterations

With the LSTM-CNN model, we performed some additional experiments by making some adjustments to the dataset. First, we trained the models only on a subset of the data. This adjustment did not cause any significant improvements in predictive accuracy. However, when we downsampled this subset and trained it with an LSTM-CNN model, we could create a model that overfitted to the dataset. With all the other methods, we could not even achieve overfitting, as we only stopped training after 100 epochs without improvement on the training set; after this, the best performing models were still underfitting. The LSTM-CNN model had the following architecture:

Input:	30 time steps with a 5 minute interval 150 minutes for 19 variables
TD: Convolution Layer:	Kernel size: 3, No. of filters = 5
TD: Max Pool Layer:	Pool size = 2
TD: Flatten	
LSTM Layer:	500 units, Dropout = 0.3
LSTM Layer:	400 units, Dropout = 0.3
TD: Fully Con. Layer	120 units (6×20)
Output:	20 time steps $(t_5, t_{10}, \ldots, t_{100})$, for 6 variables

The full results are shown in table 5.8. When we consider the results on the training

set, this model provides a very significant improvement in terms of predictive accuracy. The scores here show a lower bound on the error metrics; a lower score provided by a model that does not overfit is probably impossible. When we visualize the predictions on patients that are included in the training set (see figure 5.15), we can see that the model approximates the curves well. These good results disappear almost completely when we use this model on the test set (see figure 5.12). It performs much worse than the persistence model. If we look at the predictions for the $EtCO_2$ variable, we clearly see that the predictions do not make sense, so at least for the $EtCO_2$ variable, the training of this model did not help to create a usable model. For the P^{PEAK} variable, these results are better. The MAPE score is comparable to the CNN-DFMO and LSTM-CNN networks, and there are moments that this model outperforms the persistence model. One of these moments is visualized in figure 5.17.

During training, we also kept the model that still performed best on the validation set. This model, however, does not outperform the LSTM-CNN model's results given in 5.7.

	MSE	MAE	RMSE	MAPE
$EtCO_2$	4.50	1.36	2.12	3.48
$V_{T_{exp}}$	4.16	1.17	2.04	6.61
RF	4.59	1.30	2.14	3.22
HR	39.87	4.19	6.31	2.88
P^{PEAK}	2.60	1.07	1.61	5.60
SpO_2	2.58	0.98	1.61	1.08

(a) Results on the training set

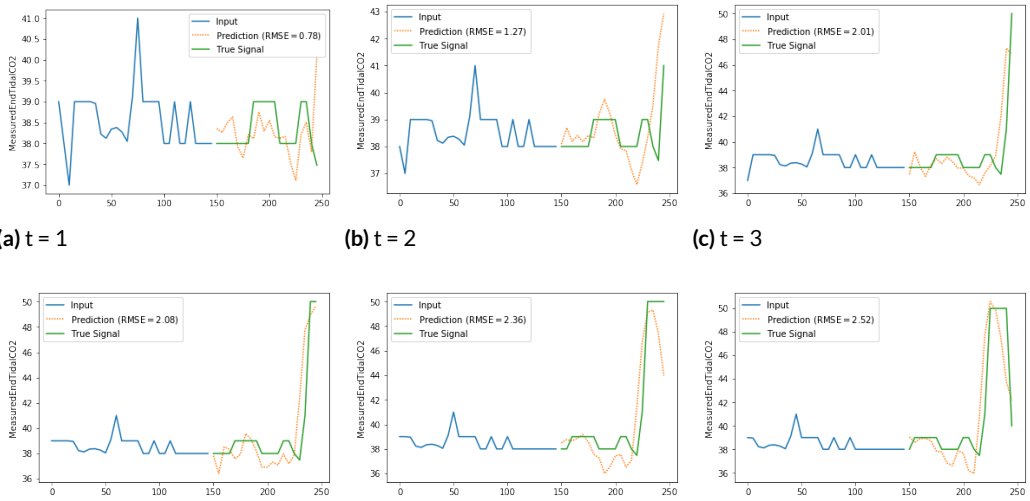
	MSE	MAE	RMSE	MAPE
$EtCO_2$	39.64	4.82	6.30	13.53
$V_{T_{exp}}$	17.56	2.71	4.20	15.49
RF	25.19	3.44	5.02	8.80
HR	447.32	16.05	21.15	11.03
P^{PEAK}	11.22	2.53	3.35	12.68
SpO_2	21.50	2.97	4.64	3.15

(b) Results on the test set

	MSE	MAE	RMSE	MAPE
$EtCO_2$	15.09	2.40	3.88	5.91
$V_{T_{exp}}$	16.38	2.04	4.05	13.53
RF	17.89	2.28	4.23	5.59
HR	139.10	7.60	11.79	5.48
P^{PEAK}	8.56	1.96	2.93	10.47
SpO_2	7.89	1.57	2.81	1.74

(c) Results of the persistence model on the test set

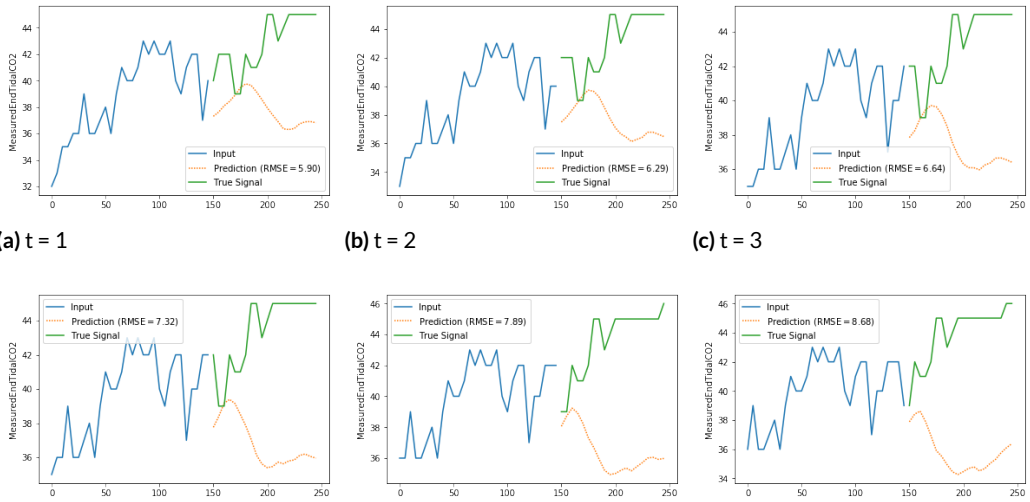
Table 5.8: A LSTM-CNN-DFMO was able to overfit on the training set



(a) t = 1 **(b)** t = 2 **(c)** t = 3

(d) t = 4 **(e)** t = 5 **(f)** t = 6

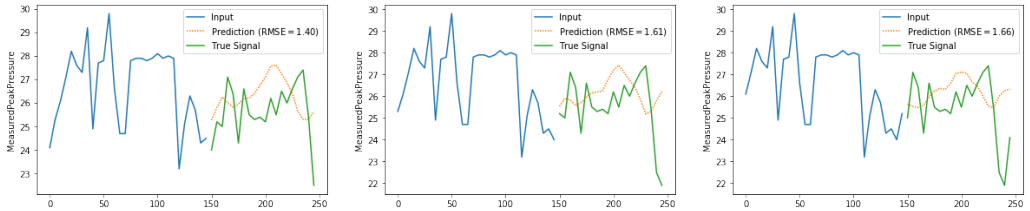
Figure 5.15: Six consecutive predictions of the LSTM - CNN MTSI DFMO model with an adapted training set. These are predictions made on a patient that was included in the training set



(a) t = 1 **(b)** t = 2 **(c)** t = 3

(d) t = 4 **(e)** t = 5 **(f)** t = 6

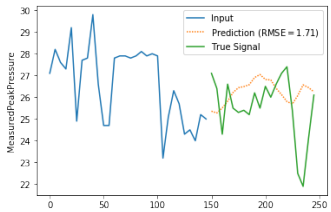
Figure 5.16: Six consecutive predictions of the LSTM - CNN MTSI DFMO model with an adapted training set. These are predictions for the $EtCO_2$ variable made on a patient that was included in the test set



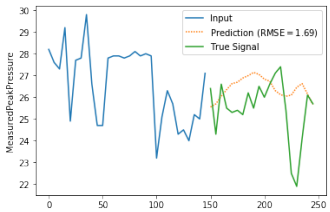
(a) t = 1

(b) t = 2

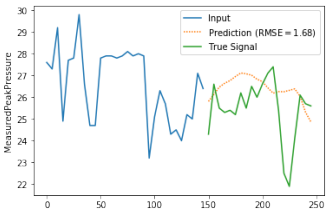
(c) t = 3



(d) t = 4



(e) t = 5



(f) t = 6

Figure 5.17: Six consecutive predictions of the LSTM - CNN MTSI DFMO model with an adapted training set. These are predictions for the P^{peak} variable made on a patient that was included in the test set

5.6 Discussion

In this section, we will discuss the results of the previous section. We have seen three different architectures that were used to create predictive models; the LSTM-architecture, the CNN-network, and the combined LSTM-CNN-architecture. First, we will discuss how we can explain and interpret the results of these models in section 5.6.1. Then we will compare the results with those presented in literature in section 5.6.2. After that, we will answer the research question we posed in the introduction in 5.6.3. Finally, we will propose some directions for further research in section 5.6.4.

5.6.1 Discussion of the results

We will first discuss the DFSO models that directly forecasted the variables with a forecast horizon of 10 minutes, without predicting the values of the time steps in-between. These results are shown in table 5.9. The DFSO model that performs best is the LSTM-MTSI model. Of all the trained models, the LSTM-MTSI performs the best on four out of the six variables. However, the difference between the four models is negligible. We have seen how all the models mimic the naïve persistence model, yet show a marginal improvement over the persistence model. We can also compare the results of one of the DFMO models. By selecting only the t_{10} part of the DFMO forecast, we could compare the results with the predictions of the DFSO models. This model slightly outperforms the DFSO models for three out of six variables. If we only compare the results with the CNN-DFSO model only, the CNN-DFMO-10 model outperforms this model for all variables. This shows that predicting multiple steps does not degrade the model's performance, it even improves the results slightly. For the DFMO models, we see a clear

	LSTM-STSI	LSTM-MTSI	CNN-DFSO	Naïve	CNN-DFMO-10
$ETCO_2$	3.66	**3.53**	3.80	3.63	3.76
$V_{T_{exp}}$	**47.17**	56.73	58.01	52.01	51.12
RF	5.37	4.91	**4.58**	4.91	*4.33*
HR	3.99	**3.94**	4.08	3.96	*3.81*
P^{PEAK}	12.26	**11.09**	11.79	13.37	11.73
SPO_2	1.13	**1.06**	1.08	**1.05**	*1.01*

Table 5.9: This table shows the results of the DFSO models, according to the MAPE-metric.

difference between the models (see table 5.10). The CNN outperforms all the other trained models; however, the difference between the persistence model and the CNN model in terms of MAPE is marginal. There is a reason why the naïve results differ between the adapted dataset and the original dataset. This downsampling process removes a lot of noise, especially in the Expiratory Tidal Volume signal, which is noisier than the other signals, which makes this signal difficult to predict [1] The models trained

[1]The noisiness may be caused by the absence of the cuff (see section 2.1) for the youngest patients (which are also included in the unaltered training set). The absence of the cuff causes unstable $V_{T_{exp}}$ values due to

MAPE	LSTM	CNN	LSTM-CNN	Naïve	Naïve (5)	Overfit (5)
$EtCO_2$	7.94	**3.59**	4.68	**3.51**	5.91	13.53
$V_{T_{exp}}$	55.89	**49.33**	51.11	**48.70**	13.53	15.50
RF	13.93	**4.54**	5.80	4.68	5.59	8.80
HR	3.84	**3.75**	4.35	3.79	5.49	11.03
P^{PEAK}	20.03	**11.35**	12.60	13.03	10.47	12.68
SpO_2	1.14	**1.00**	1.35	**0.97**	1.74	3.15

Table 5.10: This table shows the results of the DFMO models, according to the MAPE-metric.

on the original datasets perform denoising (especially on the $V_{T_{exp}}$ signal) themselves, as they provide a generalization over the naïve prediction, exposing the true signal hidden behind the noise. The error-metrics do not reflect this quality, as we make a comparison between the predictions and the noisy signal; the denoised moments are regarded as errors. This flaw is visible, especially in the MAPE-metric. However, the MSE-metric, the loss function during training, rewards this quality. The denoised prediction contains fewer outliers, which are punished more severely by this metric. This quality is the main reason why the trained models provide such significant improvement with respect to the predictions of the Expiratory Tidal Volume variable, when compared to the naïve model and according to the MSE-metric. If we perform the denoising beforehand, the naïve score on denoised patient data will automatically be better.

An observation that can be made that concerns all the trained models is that they all mimic the naïve persistence model. This is not clearly visible in the scores, but it is clearly visible if we compare the true signal, naïve predictions, and the model's prediction visually. When this happens, a question that needs to be answered is the following: Is the data generating process a random walk? When this is the case, the optimal prediction is indeed the naïve prediction, and there is not much that we can do to improve this.

There is also another reason why this behavior can happen: the dataset contains a lot of patients who show vital signs that, globally, remain constant over time. This is also visible when we compare the error metrics of the overfitted model to error metrics of the naïve model (see table 5.11): The overfitted model can be seen as an upper bound in

	Training set	Naïve	Test set
$EtCO_2$	3.48	5.91	13.53
$V_{T_{exp}}$	6.61	13.53	15.43
RF	3.22	5.59	8.80
HR	2.88	5.48	11.03
P^{PEAK}	5.60	10.47	12.68
SpO_2	1.08	1.74	3.15

Table 5.11: This table shows the results of the overfitted model, according to the MAPE-metric.

the fact that there is more leakage.

terms of accuracy. Suppose that we were indeed able to create a model that makes good predictions on the test set, then the results of that hypothetical model will probably not be better than those of the overfitted model on the training set. If we compare the naïve model with this upper bound, we can see that for the $ETCO_2$, RF, HR, and SPO_2 the difference between the MAPE-metrics is only 2 percentage points. Note that these are scores of a model that predicts time steps to 100 minutes ahead (with 5-minute intervals). If the naïve model makes relatively good predictions 100 minutes ahead, this can only be caused by this stability; otherwise, the difference between the overfitted model and the naïve forecast would be more significant. The difference between the MAPE of the CNN-DFMO model and the overfitted model on the training set concerning the $ETCO_2$ variable is only 0.20 percentage points. When we compare the results of the RMSE scores, there is also not a very significant difference between the CNN-DFMO model and the overfitted model, apart from the $V_{T_{\text{exp}}}$ variable. Finding a (global) optimum through gradient descent that improves on this score, is probably hard, as this relatively good local optimum is always near in terms of the loss function and relatively easy to find.

5.6.2 Results in literature

In section 5.4.7, we briefly discussed the paper "Predicting responses to mechanical ventilation for preterm infants with acute respiratory illness using artificial neural networks". In this paper, Brigham et al. performed similar experiments in which they created what we call DFSO models; models that given a history make a direct forecast for a single future value. The authors do not provide global error statistics, but only provide visualizations of the predictions compared to the actual data. They provided RMSE-metrics corresponding to the visualizations. Because the RMSE-metric is not scale invariant, and they do not provide global statistics, we cannot compare the results based on error-metrics. However, we can compare their visualizations to ours. In figure 5.18, some of their results are presented. While the last predictions are not explicitly plotted, we can clearly see the same lagging behavior that we found in our DFSO and DFMO models (for comparison, see 5.5c and 5.2c). A difference between their dataset and ours is that their data has a higher frequency. The fact that they faced the same problem could be an indication that the data at a higher frequency could also be a random walk. One could say that the fact that the one-minute resolution of the PDMS may cause some of the randomness, however the fact that Brigham et al. [7] face similar problems shows that increasing the rate of the measurements might not be the solution to solve the random walk problem. However, this should be verified by further research, as although the methodologies are similar, they are not entirely the same.

5.6.3 Answering the research questions

In this chapter, we aimed to accomplish the following research objective:

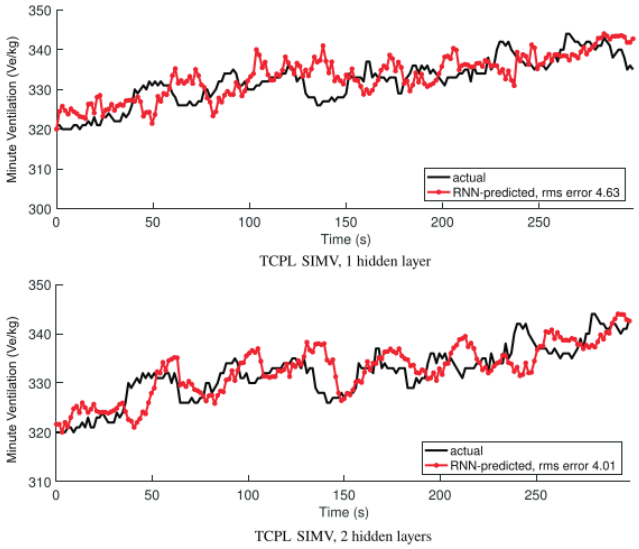


Figure 5.18: Predictions on the Minute Expiratory Tidal Volume, as presented in [7].

Research Objective 2. To create models that, after being trained on multiple time series of both measured variables and settings entered by the ventilator's operators, predict future values of the patient's oxygen saturation, end-tidal CO_2, expiratory tidal volume, and peak pressure.

While we succeeded in the creation of these models, we cannot say that these models are clinically usable. They do offer some improvement over the naïve persistence model, but they still mimic this predictor. As the model mostly relies on the last measurement, it will only make correct predictions when the patient is in a stable condition, which is mostly the case. These changes from stable to unstable are the most clinically interesting, and yet, these are not predicted accurately. While we have seen that in some cases (figure 5.13) the model is able to pick up some of these changes, this is not always the case and difficult to verify.

We also posed a subquestion on finding a suitable learning method. In this chapter, we have experimented with three deep learning architectures. All three of them are suitable for time series forecasting; however, we experienced some problems with the DFMO-method and the LSTM-architecture. The CNN-DFMO model only slightly outperformed the other methods.

Creating patient subgroup specific models did not yet prove to be beneficial for the model's performance. This observation is partly due to the fact that all the models mimic the persistence model; when the model only relies on the last available measurement, the patient's class does not matter and dividing the dataset based on patient classes will not help.

5.6.4 Directions for further research

We have seen that the patients who are ventilated using the PC-IMV-Adaptive mode have vital signs which remain stable for the majority of the time. This may be different for other ventilation modes, but probably this problem is present for these other patients. This observation raises the following question. *Can we improve the model by providing more examples of unstable patients?* Before we can answer this question, we first have to find patients who had unstable episodes. As the PICU dataset is not verified or even annotated by physicians, identifying these episodes is not an easy task. A method that could help finding these episodes is *clustering*. If we could cluster fragments of time series in different groups, we could identify patients that belong to a certain class. Among these groups, a group may be present that physicians would classify as unstable or dangerous. We could also predict the change between the different clusters, instead of predicting the actual scalar value. As a single scalar value of a vital sign is meaningless without its context, classifying time series fragments may even make the information more comprehensive to the caregivers.

The fact that these models seem to model the "normal" behavior of the vital signs given a set of historical data points could mean that these predictions can be used as an outlier/anomaly detection method. If the actual values are diverging of this signal, then this could be a reason for producing an alarm. In further research, the applicability and trustworthiness of this method can be assessed.

5.7 Summary

In this chapter, we described how Artificial Neural Networks could be used to create predictive models for multivariate time series data. We described several main principles of deep learning, like the perceptron, backpropagation, and activation functions. Next, the following three architectures were introduced: the *Long Short-Term Memory network* (LSTM), the *Convolutional Neural Network* (CNN), and an architecture that combines both of them (LSTM-CNN). We described how these architectures could be applied to multivariate time series data. Subsequently, we described how we implemented, trained, and evaluated the models.

LSTM models could have two input methods: *Single Time Step Input* (STSI) and *Multiple Time Step Input* (MTSI). When the first is used, information of previous time steps flows through the cell state to the current input. The MTSI method supplies for each prediction step a history of size h to each input. CNN and LSTM-CNN models can only work with MTSI inputs.

For the output we can make a similar distinction: *Direct Forecast Single Output* (DFSO) and *Direct Forecast Multiple Output* (DFMO). The first produces a single time step for the target variables at time step $\boldsymbol{x}_{t+s}$, and the latter produces a series of s time steps for the target variables. These time steps range from $\boldsymbol{x}_{t+1}$ to $\boldsymbol{x}_{t+s}$.

In addition to architectural choices, we also investigated if models trained on a specific patient group (age below 40 days at admission) produce better models. We assessed the predictive quality using four error-metrics: *Mean Squared Error* (MSE), *Root Mean Squared Error* (RMSE), *Mean Absolute Error* (MAE), *Mean Absolute Percentage Error* (MAPE). Besides these error-metrics, we also assessed the models by visually inspecting plots that compare the predictions with the true signal.

All the models, regardless of architecture, learned to mimic the naïve persistence model, which uses the last available measurement as its prediction. The *CNN-DFMO* model provided the best overall performance and provided a small improvement over the naïve persistence model, which was used as a baseline. The results of the other architectures only differ slightly from the results of the CNN-DFMO model. While this model performs slightly better, it still mimics the persistence model. In stable periods, the model performs well. However, it fails to predict unstable periods, which are the most interesting from a clinical perspective. In plots, the predictions seem to copy the last available measurement. Moreover, the error between the naïve model and the trained model is smaller than the trained model with the actual data. A model that is based on only the last measurement does not provide any added value to the caregivers. Therefore, these models are not clinically usable. When the best performing model is the naïve persistence model, this could be an indication that the model is a random walk.

Brigham et al. [7] performed similar experiments. Their results seem to correspond with ours; especially if we compare the plots of their predictions to ours, we see the same lagging behavior which is visible in our plots. While our methodology differs, their results support our conjecture that these time series are random walks.

6 Final Remarks

6.1 Introduction

In this thesis, we explored how we could apply machine learning techniques to ventilation related vital signs time series. The ultimate goal was to create a predictive model that could predict future vital sign values based on their historical values. The motivation of this work was to be able to leverage the enormous potential of Artificial Intelligence and new deep learning techniques for solving this complex problem. Caregivers at the PICU could greatly benefit from accurate predictions on vital signs; maybe even enabling them to mitigate complications before they occur. In section 1.4.3, we proposed the following two research objectives:

Research Objective 1. To describe the properties of the individual time series and possible (causal) relations between them. We will perform these tasks to gain more insight into the data generating process.

Research Objective 2. To create models that, after being trained on multiple time series of both measured variables and settings entered by the ventilator's operators, predict future values of the patient's oxygen saturation, end-tidal CO_2, expiratory tidal volume, and peak pressure.

Furthermore, we proposed the following three research questions:

Research Question 1. How do we prepare the raw dataset for further analysis?

Research Question 2. Will the creation of models that are trained on subsets of patients improve the model's accuracy?

Research Question 3. Which deep learning methods are suitable for making predictive models for Vital Signs Time Series?

In this project, we have accomplished both objectives. We have provided an extensive exploratory analysis describing several possible relations between the variables. We have proposed neural network-based architectures that can predict future values of these vital signs. However, the models' usefulness is up to discussion. In this chapter, we will discuss the main results and findings that were discussed in the individual chapters and discuss their limitations.

6.2 Discussion of the main results

6.2.1 Data preparation

The PICU dataset was not directly usable for further research; each ventilator type and ventilation mode used a different set of variables and parameters. Moreover, the dataset contained gaps and missing data. Our methodology was as follows: Initially, we adapted the dataset by partitioning it into ventilation mode-specific episodes of continuous measurements. After that, we imputed the remaining missing values. Now, the dataset was free of missing values, and we have a fixed schema for each ventilation mode. Because each ventilator mode still uses specific settings, we opted to focus our research on one ventilator type and one ventilation mode; we chose the Servo-i ventilator and the PC-IMV-Adaptive mode, as this is the most used mode on the PICU. However, the techniques that we used in this work are universal, in the sense that the analysis procedures only have to be adapted minimally to analyze other ventilation modes.

6.2.2 Vector Autoregression

In chapter 4, we used time series analysis techniques found in econometrics and psychological research for medical time series. We adapted the methodology of Bose et al. [5] to our dataset. There are some differences between their project and ours:

1. Our dataset contains more patients
2. More vital signs (variables) are recorded in our datasets (3 VSTS vs. 10)
3. Bose et al. [5] hand-picked their series and only considered series prior to a *cardiorespiratory instability* event, while in our dataset, no selection takes place.
4. The patient population differs (pediatric patients vs. adult step-down unit)

In addition to the increase in recorded variables, the Granger-causal tests we performed contained significantly more relations per patient than in Bose et al. [5]. Moreover,

the test only assesses causal relations between two variables; other variables are not assessed. This may lead to the following situation: If a spike in the $ETCO_2$-curve causes changes in first the HR variable and later the P^{PEAK} variable, we might wrongfully attribute causal relations if we assess Granger-causality between HR and P^{PEAK} (see figure 6.1). We may conclude that for a dataset of the size that we analyzed, the methodology as presented by Bose et al. [5] did not prove to be very useful.

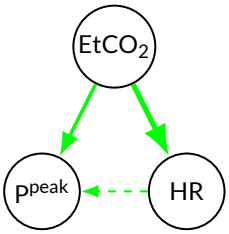


Figure 6.1: The Granger-causality tests that we performed only considered two variables. A third variable (in this example $EtCO_2$) which causes both changes is not taken into account.

6.2.3 Multilevel Graphical VAR models

By using Multilevel Graphical VAR models, we aimed to describe the relations between variables that are present in multiple patients; for both a sample of the whole dataset and a subset selected on age. These relations could be temporal but also contemporaneous ones. There were several observations that we made based on the fitted MLGVAR models:

The most significant correlations we found were between the variables' previous measurement and their current. In addition to the *self-loops*, there were also significant correlations found between different variables. These correlations persist up to a history of 30 minutes.

In addition to the temporal effects, we also discussed contemporaneous effects and between-subjects effects. The between-subjects network enabled us to visualize how variables predict each other on average. We found several correlations related to the lung's health in this network (e.g., correlations between P^{PEAK}, PEEP, and FIO_2).

We also fitted the MLGVAR model to a subset of the patients; this resulted in even more correlations between variables. This finding led us to the hypothesis that modeling for patient subsets might be beneficial for the model's predictive accuracy. A model has a limited capacity, and therefore, it may not be able to account for all the relations that are only valid for specific subsets of patients.

6.2.4 Predictive models

In addition to the exploratory analysis, we created predictive models for Vital Signs Time Series using deep learning methods, according to **Research Objective 2**. We pro-

posed several architectures for time series forecasting. These are the *Long Short-Term Memory network*, the *Convolutional Neural Network*, and an architecture that combines both of them. Subsequently, we fitted these networks on the PICU dataset and tested the performance of the created models. All the models, regardless of architecture, learned to mimic the naïve persistence model, which uses the last available measurement as its prediction. The *CNN-DFMO* model, a model that is based on the Convolutional Neural Network, provided the best overall performance and provided a small improvement over this naïve model. However, it still mimics the persistence model. In stable periods, the model performs well. However, it fails to predict unstable periods, which are clinically the most interesting, and therefore, the model is not clinically usable. The results of the other architectures only differ slightly from the results of the CNN-DFMO model.

Brigham et al. [7] performed similar experiments as we did. Our results seem to correspond with theirs; when we compare the plots of their predictions to ours, we see the same lagging behavior. We have to note that their methodology differs from ours, for example, their inclusion criteria are more strict than ours and the measurement frequency is significantly higher in their dataset (1.5 seconds vs. 1 minute). We might conclude from this that increasing the frequency of the measurements does not yield a better performance; however, further research on such a dataset is needed, as Brigham et al. [7] do not report the general (quantitative) performance of the models. Our results indicate that Vital Signs Time Series may be that unpredictable that the best model we can make is the naïve model.

6.3 Limitations

Throughout, we made certain choices that may influence the results and limit the usefulness of the models. In the following sections, we will discuss these choices and limitations of the scopes.

Linear exploratory models

The models we discussed in chapter 4 are linear models. However, between the time series, or even within a time series, non-linear relations can be present [7]. Linear models are relatively easy to interpret, especially the Multilevel Graphical VAR models. However, these models cannot capture non-linearities, as they can only model linear relations. The relations we discussed in section 4.6.2 may only be valid if the linearity assumption is correct.

Applicability to other ventilation modes

In the preparatory phase, we divided the dataset into episodes that are based on the ventilator type, ventilator mode. Furthermore, these episodes are gapless. While this division was necessary, this resulted in some loss of information; that is, events in prior

episodes are not available as predictors, while this information could be advantageous for the predictive quality.

The choice for the PC-IMV-Adaptive mode could have some influences on the results of our predictive model and the results of our exploratory analysis. A PC-CMV-Servo mode has different goals than a PC-IMV-Adaptive mode, which may result in different ranges and patterns in the recorded variables. Also, if we consider support (PC-CSV) or non-invasive ventilation methods, the number of artifacts and missing values present in the time series may be higher than in the episodes of PC-IMV-Adaptive series, for example, due to more leakage. The imputation method that we applied may not be appropriate for the other ventilation modes, because the MTSDI algorithm can only deal with a certain amount of missing data [22].

Imputation

We described in section 3.4.2 how we imputed missing values in the PICU dataset. The amount of missing data was fairly low; at most 3.5 %, so it is to be expected that the influence of the imputed values on our results is marginal. Our imputation method may not be directly applicable in a clinical setting. If a value is missing in a clinical setting, our method may not be suitable, as the imputed values may also be based on their future values and not only on their histories. An adequate method may be *Last Observation Carried Forward* [54] or a more sophisticated imputation method that considers previous values and the current values of other variables only.

Unbalanced dataset

The dataset that we used was not balanced in any way. Ideally, enough examples of several patient categories should be present in the dataset, as well as a balance between unstable and stable periods. Methods to divide the patients into categories like condition and in terms of disease were not available during our project. We investigated the effect of patient subsets based on age during our exploratory analysis, and this yielded more complex models than when the whole patient population was sampled. We suspect that even more specific (or better specified, for example on disease) subgroups may result in better models. Furthermore, as mentioned before, the patients remain mostly stable; therefore, the number of unstable periods in our dataset may be too low. Having a more balanced dataset, with more unstable training examples (or at least more balanced), may lead to a better predictive quality.

Hyperparameter tuning

In our work, we performed some hyperparameter tuning. However, due to time constraints, it was not possible to perform an extensive search for ideal hyperparameters. Our search was mainly concentrated on parameters related to the models' capacity. In

addition to capacity, we also tested several values for Dropout (regularization). We have seen some indications that the capacity of the models was high enough (we were able to create a model that overfits); however, the search was in no way exhaustive. While we think that the quality cannot improve drastically (test set error comparable with the training error of the overfitted model), there is a possibility that we could see some marginal improvement by further tuning of the hyperparameters.

Absence of comparative literature

In this work, we worked on several new techniques. The MLGVAR model was only recently introduced (2018) and since then only used in the psychological analysis of relatively small datasets (e.g., [1]). Therefore, we cannot compare our methodology and results in literature. Furthermore, working with a large multivariate time series dataset concerning multiple subjects is relatively new and difficult research topic, and some topics (e.g., imputation) have not been fully solved yet [52, 3]. This may be partly due to the computational complexity of the problem; moreover, this is still a problem as the MLGVAR model has difficulties processing large amounts of data [15].

For the predictive part, we could compare our results to the work of Brigham et al. [7]; however, the details on their exact implementation are minimal. Moreover, the discussion of their results is less extensive as ours, as no global statistics are given of the overall performance.

6.4 Conclusion

We had two objectives in this project: to perform an exploratory analysis and to create predictive models. We have learned what the strengths and limitations are of the several exploratory analysis tools. The *Multilevel Graphical VAR model* is an adequate tool to perform analysis on this type of data. The between-subjects model shows several relations which can be corroborated by clinical theory. Regardless of the validity of these relations, we have seen that the complexity of the models increases when the patient population becomes more specific. This led us to hypothesize the following:

> Models based on specific patient groups may perform better than more generic models.

Unfortunately, we could unfortunately not verify this hypothesis in our experiments with the predictive models (due to the mimicking). The predictive models that we created marginally outperformed the naïve persistence model, but they still largely follow its predictions. We have seen that the model performs well in stable periods, but does not foresee sudden changes in the vital signs, and therefore, the models are not clinically useful. This led us to the following conjecture:

> Creating clinically usable time series (regression) forecasting models is not possible with the PICU dataset.

We have several reasons to support this conjecture:

1. The predictive models all largely follow the naïve prediction after extensive training; this indicates that the time series may be a random walk. Moreover, the LSTM-CNN-DFMO model is able to overfit and seemingly having enough capacity to fit the entire training set. However, the best performing model on the test set (early stopping) still mimics the persistence model.
2. The PICU dataset does not contain enough observations of unstable periods. The naïve model performs well on the PICU dataset, which is only possible if the values of the vital signs remain stable for long periods of time. A more balanced dataset that contains more unstable periods is needed.

Selecting unstable periods was not possible during our project. In the following section, we will discuss some directions for further research. As selecting unstable periods is a problem that is not fully solved yet, and other datasets might be prone to this. We suggest that before attempting to train similar predictive models on similar datasets, more research is performed in addressing the limitations discussed in section 6.3, especially the unbalanced nature of the dataset. If the time series are truly random walks, which our results indicate, creating forecasting models may be futile. However, there may be other means through which we may be able to create predictive models based on clustering, which we will discuss in the next section.

6.5 Directions for further research

6.5.1 Clustering

During this project, we have learned how we can perform initial exploratory analysis on a multivariate time series. This analysis can be extended by clustering. By *clustering* we can group time series who share similar features or even identify patients belonging to a certain class. Among these groups, a group may be present that physicians would classify as unstable or of worrisome/critical condition. We could also predict the change between the different clusters, instead of predicting the actual scalar value. As a single scalar value of a vital sign is meaningless without its context, classifying time series fragments may even make the information more comprehensive to the caregivers. Džeroski et al. [12] describe a framework that can both clusters time series in homogenous groups and predict. The authors represent the clusters with symbolic notation. Lin et al. [27] describe another discretization approach (converting a series into symbolic notation), namely SAX-notation, which can be used to simplify the series and for clustering. Symbolic representations of time series could be used as features for these predictive models.

6.5.2 Performance metrics

In this work, we have described how we can perform vital signs forecasting using several neural network architectures, and we learned how we could assess the model's predictions. We proposed in section 5.4.5 several methods to assess the prediction of the models. It is sometimes difficult to compare the predictive quality of our models with those of other models. For example, the results of the prediction models of Brigham et al. [7] are in our opinion, underreported. When predictions are made on time series in a clinical setting; we consider it important to discuss the following criteria at least:

Overall performance. How does the model perform on the test set in terms of an error-metric? This test should give an overall indication of how the model performs in general. Using an error-metric, we can compare the model to other models. Besides reporting the error in the unit of measurement, it also is good to report a scale-invariant error-metric like *MAPE*, to make it easier to compare the results that work on different datasets. However, in our opinion, solely providing error-metrics is not enough, and visual inspection is needed.

Visual inspection of the predictive behavior. What does the model actually predict for each time step? We can do this by plotting the predictions and actual data into the same graph and compare the results. In this step, we can assess if the model successfully predicts the real data, or if the model does something else entirely.

Comparison to naïve models. There are many data generating processes that produce random walks. To check that the model improves on the persistence model and does not mimic it, we should compare the predictions of the trained model with that of a naïve model. We test this in two ways: we can compare the predictions visually, or we can compare the error-metric for the actual data with the error-metric for the naïve model. If the latter is smaller than the first than the trained method has probably learned to mimic the naïve model.

However, we found that it is sometimes difficult to assess and characterize the visual predictions for the whole dataset as the visualizations are still patient-specific. Sometimes, the models make better or worse predictions than usual, and we have seen that in some cases, the model performs better than the naïve persistence model (for example, see figure 5.13). It is hard to verify if this behavior is present in other patients. Further research might be directed to create a protocol/methodology on how to assess time series forecasting methods, especially when it concerns a collection of multivariate time series.

Bibliography

[1] George Aalbers et al. "Social media and depression symptoms: a network perspective". In: *Journal of Experimental Psychology: General* (2018), p. 9.

[2] Alexander Amini. *Introduction to Deep Learning*. Jan. 2019. URL: http://introtodeeplearning.com/materials/2019_6S191_L1.pdf.

[3] Faraj Bashir and Hua-Liang Wei. "Handling missing data in multivariate time series using a vector autoregressive model-imputation (VAR-IM) algorithm". In: *Neurocomputing* 276 (2018), pp. 23–30.

[4] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166.

[5] Eliezer Bose, Marilyn Hravnak, and Susan M Sereika. "Vector Autoregressive (VAR) Models and Granger Causality in Time Series Analysis in Nursing Research: Dynamic Changes Among Vital Signs Prior to Cardiorespiratory Instability Events as an Example". In: *Nursing Research* 66.1 (2017), p. 12.

[6] Trevor S Breusch. "Testing for autocorrelation in dynamic linear models". In: *Australian Economic Papers* 17.31 (1978), pp. 334–355.

[7] Katharine Brigham, Samir Gupta, and John C Brigham. "Predicting responses to mechanical ventilation for preterm infants with acute respiratory illness using artificial neural networks". In: *International Journal for Numerical Methods in Biomedical Engineering* (2018).

[8] Robert L Chatburn, Mohamad El-Khatib, and Eduardo Mireles-Cabodevila. "A Taxonomy for Mechanical Ventilation: 10 Fundamental Maxims". In: *Respiratory Care* 59.11 (2014), pp. 1747–1763. ISSN: 0020-1324. DOI: 10.4187/respcare.03057. eprint: http://rc.rcjournal.com/content/59/11/1747.full.pdf. URL: http://rc.rcjournal.com/content/59/11/1747.

[9] Junyoung Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[10] Michel Dojat et al. "A knowledge-based system for assisted ventilation of patients in intensive care units". In: *International journal of clinical monitoring and computing* 9.4 (1992), pp. 239–250.

[11] Philip Drinker and Louis A Shaw. "An apparatus for the prolonged administration of artificial respiration: I. A design for adults and children". In: *The Journal of clinical investigation* 7.2 (1929), pp. 229–247.

[12] Sašo Džeroski et al. "Analysis of time series data with predictive clustering trees". In: *International Workshop on Knowledge Discovery in Inductive Databases*. Springer. 2006, pp. 63–80.

[13] Sacha Epskamp. *graphicalVAR: Graphical VAR for Experience Sampling Data*. R package version 0.2.2. 2018. URL: https://CRAN.R-project.org/package=graphicalVAR.

[14] Sacha Epskamp, Marie K. Deserno, and Laura F. Bringmann. *mlVAR: Multi-Level Vector Autoregression*. R package version 0.4.3. 2019. URL: https://CRAN.R-project.org/package=mlVAR.

[15] Sacha Epskamp et al. "The Gaussian Graphical Model in Cross-Sectional and Time-Series Data". In: *Multivariate Behavioral Research* (2018), pp. 1–28.

[16] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. "Real-valued (medical) time series generation with recurrent conditional gans". In: *arXiv preprint arXiv:1706.02633* (2017).

[17] Stan Franklin. "History, motivations, and core themes". In: *The Cambridge handbook of artificial intelligence* (2014), pp. 15–33.

[18] Diederik Gommers and Jeroen Rosmalen. *Beademing, een praktische handleiding*. Venticare, 2017.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[20] Clive WJ Granger. "Investigating causal relations by econometric models and cross-spectral methods". In: *Econometrica: Journal of the Econometric Society* (1969), pp. 424–438.

[21] Andrew C Harvey. *Forecasting, structural time series models and the Kalman filter*. Cambridge university press, 1990.

[22] WL Junger and A Ponce De Leon. "Imputation of missing data in time series for air pollutants". In: *Atmospheric Environment* 102 (2015), pp. 96–104.

[23] Yusuf Alper Kilic and Ilke Kilic. "A novel fuzzy logic inference system for decision support in weaning from mechanical ventilation". In: *Journal of medical systems* 34.6 (2010), pp. 1089–1095.

[24] Chang-Jin Kim, Charles R Nelson, et al. "State-space models with regime switching: classical and Gibbs-sampling approaches with applications". In: *MIT Press Books* 1 (1999).

[25] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[26] Zeshan Kurd, Tim Kelly, and Jim Austin. "Developing artificial neural networks for safety critical systems". In: *Neural Computing and Applications* 16.1 (2007), pp. 11–19.

[27] Jessica Lin et al. "Experiencing SAX: a novel symbolic representation of time series". In: *Data Mining and knowledge discovery* 15.2 (2007), pp. 107–144.

[28] Helmut Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

[29] Massimiliano Marcellino, James H Stock, and Mark W Watson. "A comparison of direct and iterated multistep AR methods for forecasting macroeconomic time series". In: *Journal of Econometrics* 135.1-2 (2006), pp. 499–526.

[30] Zelda Mariet and Vitaly Kuznetsov. "Foundations of Sequence-to-Sequence Modeling for Time Series". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. 2019, pp. 408–417.

[31] James H Maxwell. "The iron lung: halfway technology or necessary step?" In: *The Milbank Quarterly* (1986), pp. 3–29.

[32] James L McClelland, David E Rumelhart, and Geoffrey E Hinton. "The appeal of parallel distributed processing". In: *MIT Press, Cambridge MA* (1986), pp. 3–44.

[33] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133.

[34] Andrew V Metcalfe and Paul SP Cowpertwait. *Introductory time series with R*. 2009.

[35] Steffen Moritz and Thomas Bartz-Beielstein. "imputeTS: time series missing value imputation in R". In: *The R Journal* 9.1 (2017), pp. 207–218.

[36] Christopher Olah. "Conv Nets: A Modular Perspective". In: *colah's blog* (2014). URL: http://colah.github.io/posts/2014-07-Conv-Nets-Modular/.

[37] Christopher Olah. "Understanding LSTM Networks". In: *colah's blog* (2015). URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[38] Jostein Paulsen and Dag Tjøstheim. "On the estimation of residual variance and order in autoregressive time series". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 47.2 (1985), pp. 216–228.

[39] Bernhard Pfaff. *Analysis of integrated and cointegrated time series with R*. Springer Science & Business Media, 2008.

[40] Bernhard Pfaff. "VAR, SVAR and SVEC Models: Implementation Within R Package vars". In: *Journal of Statistical Software* 27.4 (2008). URL: http://www.jstatsoft.org/v27/i04/.

[41] Tom J Pollard et al. "The eICU Collaborative Research Database, a freely available multi-center database for critical care research". In: *Scientific data* 5 (2018).

[42] Donald B Rubin. "Inference and missing data". In: *Biometrika* 63.3 (1976), pp. 581–592.

[43] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[44] David E Rumelhart, Geoffrey E Hinton, James L McClelland, et al. "A general framework for parallel distributed processing". In: *Parallel distributed processing: Explorations in the microstructure of cognition* 1 (1986), pp. 45–76.

[45] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. "Learning representations by back-propagating errors". In: *Cognitive modeling* 5.3 (1988), p. 1.

[46] Jeffrey A. Ryan and Joshua M. Ulrich. *xts: eXtensible Time Series*. R package version 0.11-2. 2018. URL: https://CRAN.R-project.org/package=xts.

[47] Robert H. Shumway and David S. Stoffer. *Time series analysis and its applications: with R examples*. Springer, 2017.

[48] Michael Sigmund and Robert Ferstl. "Panel Vector Autoregression in R with the Package panelvar". In: *The Quarterly Review of Economics and Finance* (2019).

[49] Sunil K Sinha, Samir Gupta, and Steven M Donn. "Immediate respiratory management of the preterm infant". In: *Seminars in Fetal and Neonatal Medicine*. Vol. 13. 1. Elsevier. 2008, pp. 24–29.

[50] Ava Soleimany. *Introduction to Deep Learning*. Jan. 2019. URL: http://introtodeeplearning.com/materials/2019_6S191_L2.pdf.

[51] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[52] JH Stock. *Time series: economic forecasting, International Encyclopedia of the Social & Behavioral Sciences*. 2001.

[53] Adrian Trapletti and Kurt Hornik. *tseries: Time Series Analysis and Computational Finance*. R package version 0.10-47. 2019. URL: https://CRAN.R-project.org/package=tseries.

[54] Earo Wang, Dianne Cook, and Rob J Hyndman. "A new tidy data structure to support exploration and modeling of temporal data". In: *arXiv preprint arXiv:1901.10257* (2019).

[55] Hadley Wickham et al. *dplyr: A Grammar of Data Manipulation*. R package version 0.8.3. 2019. URL: https://CRAN.R-project.org/package=dplyr.

[56] Wikipedia contributors. *Granger causality — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Granger_causality&oldid=874892208. [Online; accessed 27-January-2019]. 2018.

[57] Wikipedia contributors. *Panel data — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Panel_data&oldid=898268405. [Online; accessed 27-February-2019]. 2019.

[58] Wikipedia contributors. *Predetermined variables — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Predetermined_variables&oldid=886225150. [Online; accessed 29-June-2019]. 2019.

Appendices

A

Visual inspection of eigenvalues

The following listing describes a method which can be used plot to assess the visually, conform the methodology of Bose et al. [5].

```
library(tidyverse)

circleFun ← function(center = c(0,0), r = 1, npoints = 100){
    tt ← seq(0,2*pi,length.out = npoints)
    xx ← center[1] + r * cos(tt)
    yy ← center[2] + r * sin(tt)
    return(data.frame(x = xx, y = yy))
}
# Given a vector of complex eigenvalues, it will plot them on the unit circle.
visualInspect ← function(rs){
    dat ← circleFun()
    ggplot(dat,aes(x,y)) +
        geom_path() +
        geom_point(
            data =  as.tibble(rs),
            aes(Re(rs), Im(rs))
        )
}
```

B

Example output of a VAR estimation of dSat

Below, the output a VAR(6) model is shown. There are 10 variables included in the analysis, so this results in 60 coefficients and one intercept (`const`). The probability coefficients show

```
Estimation results for equation dSat:
=====================================
dSat = dFreq.l1 + dEtCO2.l1 + dETV.l1 + ... + dSat.l6 + const

             Estimate Std. Error t value Pr(>|t|)
dFreq.l1    0.021272   0.024221   0.878 0.380208
dEtCO2.l1   0.124179   0.051998   2.388 0.017281 *
dETV.l1     0.018027   0.012399   1.454 0.146552
dFIO2.l1   -0.102030   0.017489  -5.834 9.39e-09 ***
dHF.l1     -0.163882   0.054984  -2.981 0.003009 **
dIE.l1     -1.124842   0.356480  -3.155 0.001693 **
dPEEP.l1    0.585646   0.151295   3.871 0.000122 ***
dPRES.l1    0.009215   0.024667   0.374 0.708856
dRF.l1      0.081982   0.026326   3.114 0.001944 **
dSat.l1    -0.149206   0.045171  -3.303 0.001020 **
dFreq.l2    0.010112   0.028440   0.356 0.722322
dEtCO2.l2   0.052219   0.054217   0.963 0.335915
dETV.l2     0.018595   0.016013   1.161 0.246061
```

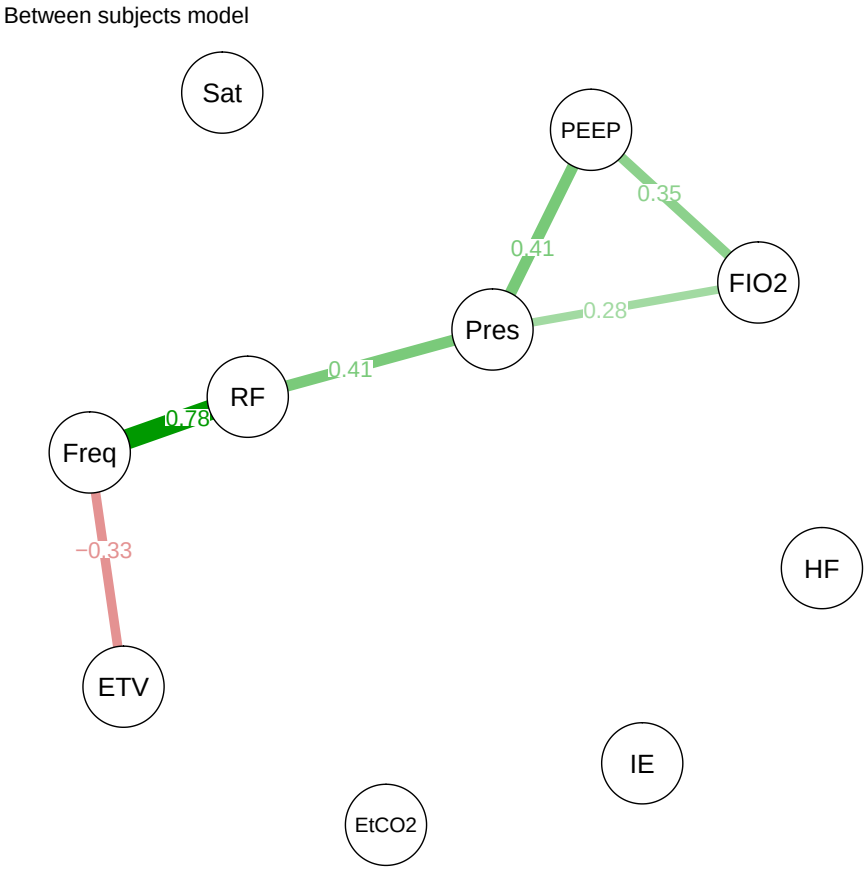
```
dFIO2.l2  -0.156908   0.019957  -7.862 2.10e-14 ***
dHF.l2    -0.152711   0.054566  -2.799 0.005318 **
dIE.l2    -0.854080   0.447272  -1.910 0.056730 .
dPEEP.l2   0.324414   0.201766   1.608 0.108455
dPRES.l2  -0.032095   0.030559  -1.050 0.294071
dRF.l2    -0.028291   0.031198  -0.907 0.364913
dSat.l2   -0.179121   0.045578  -3.930 9.61e-05 ***
dFreq.l3   0.018697   0.034211   0.547 0.584935
dEtCO2.l3  0.100291   0.054388   1.844 0.065739 .
dETV.l3    0.041253   0.017724   2.328 0.020307 *
dFIO2.l3  -0.004261   0.021764  -0.196 0.844846
dHF.l3    -0.102030   0.053819  -1.896 0.058526 .
dIE.l3    -1.191597   0.490609  -2.429 0.015478 *
dPEEP.l3   0.332950   0.206381   1.613 0.107276
dPRES.l3  -0.085488   0.033065  -2.585 0.009989 **
dRF.l3    -0.048841   0.035047  -1.394 0.164021
dSat.l3   -0.119052   0.045679  -2.606 0.009409 **
dFreq.l4  -0.070594   0.037115  -1.902 0.057708 .
dEtCO2.l4  0.068470   0.053225   1.286 0.198850
dETV.l4    0.070679   0.017792   3.973 8.09e-05 ***
dFIO2.l4  -0.013458   0.022399  -0.601 0.548215
dHF.l4    -0.044758   0.054250  -0.825 0.409721
dIE.l4     0.049955   0.497316   0.100 0.920026
dPEEP.l4   0.472376   0.206870   2.283 0.022797 *
dPRES.l4  -0.083516   0.033802  -2.471 0.013795 *
dRF.l4    -0.023466   0.033888  -0.692 0.488960
dSat.l4   -0.074957   0.045005  -1.666 0.096394 .
dFreq.l5  -0.004816   0.034983  -0.138 0.890567
dEtCO2.l5  0.017123   0.049364   0.347 0.728827
dETV.l5    0.044837   0.016098   2.785 0.005539 **
dFIO2.l5  -0.024408   0.021832  -1.118 0.264067
dHF.l5    -0.051112   0.052215  -0.979 0.328086
dIE.l5     0.417482   0.453526   0.921 0.357716
dPEEP.l5  -0.010277   0.200570  -0.051 0.959152
dPRES.l5  -0.035582   0.031594  -1.126 0.260578
dRF.l5    -0.070295   0.030262  -2.323 0.020562 *
dSat.l5   -0.147972   0.041209  -3.591 0.000360 ***
dFreq.l6   0.005580   0.027655   0.202 0.840188
dEtCO2.l6 -0.026679   0.049128  -0.543 0.587324
dETV.l6    0.030847   0.012661   2.436 0.015157 *
dFIO2.l6  -0.045310   0.018566  -2.440 0.014994 *
```

```
60      dHF.l6     0.058904   0.050555   1.165 0.244473
61      dIE.l6     0.860855   0.366929   2.346 0.019336 *
62      dPEEP.l6  -0.139925   0.154039  -0.908 0.364090
63      dPRES.l6   0.030050   0.025257   1.190 0.234671
64      dRF.l6    -0.044620   0.025951  -1.719 0.086125 .
65      dSat.l6   -0.007802   0.040969  -0.190 0.849030
66      const     -0.038595   0.058030  -0.665 0.506287
67      ---
68      Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

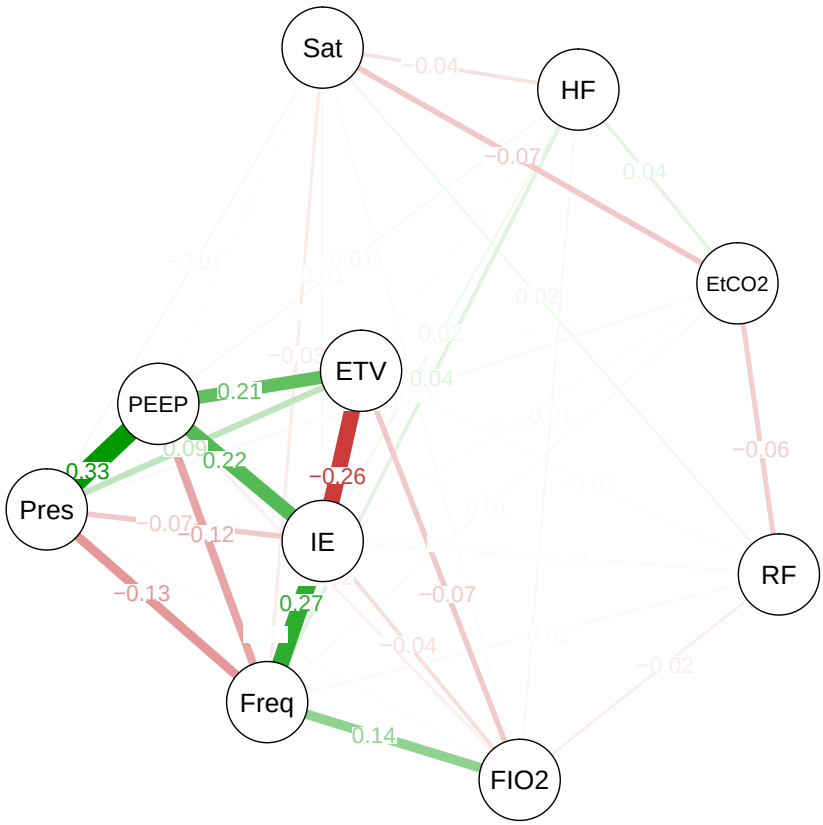
C

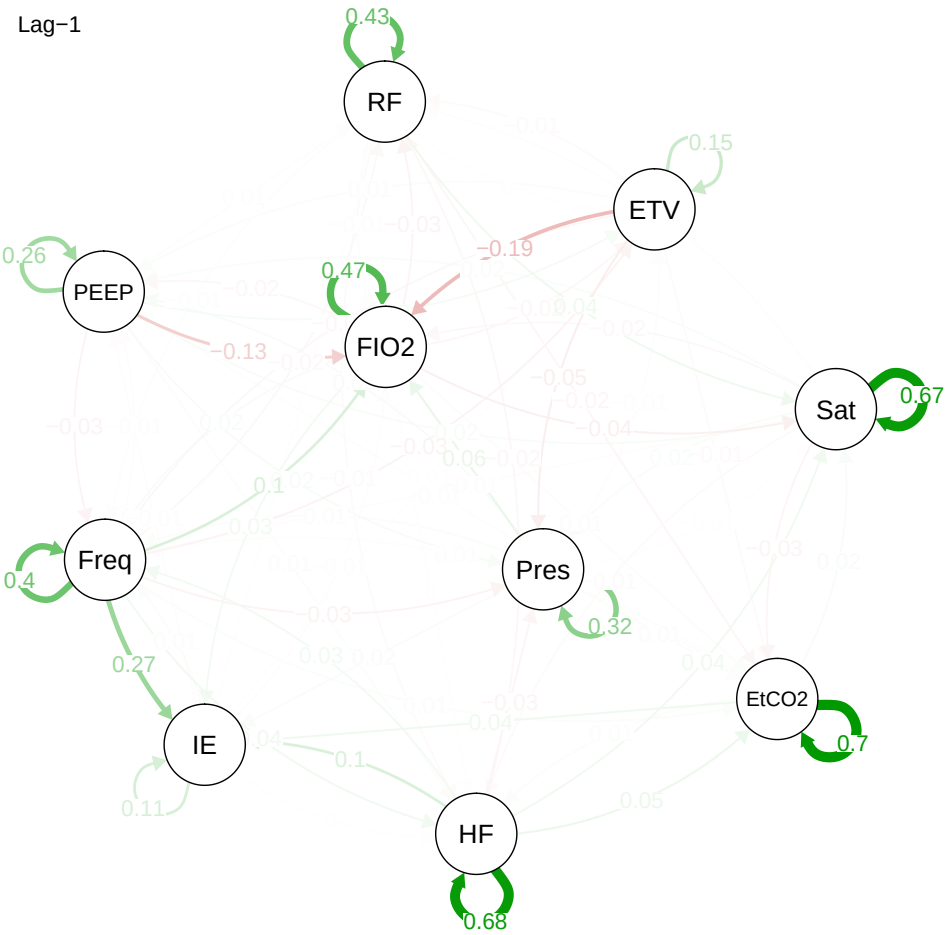
Multilevel Graphical VAR networks

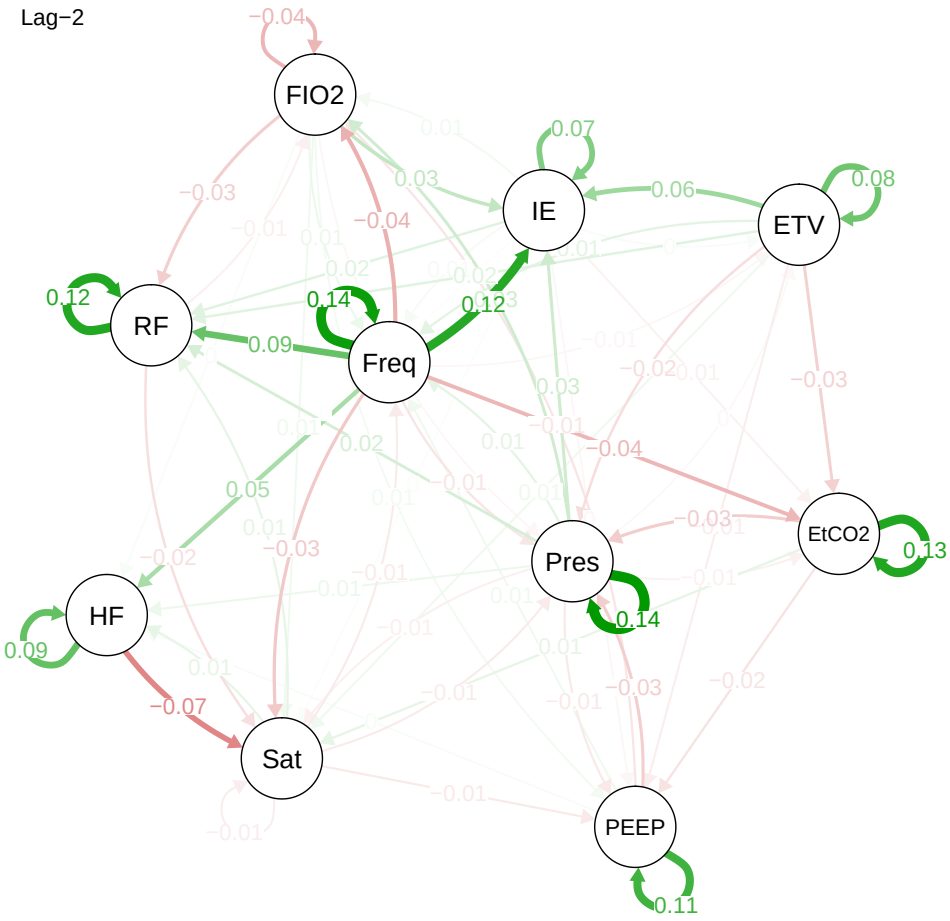
In the following pages, all the plots of all the 30 temporal networks are shown based on the whole dataset.

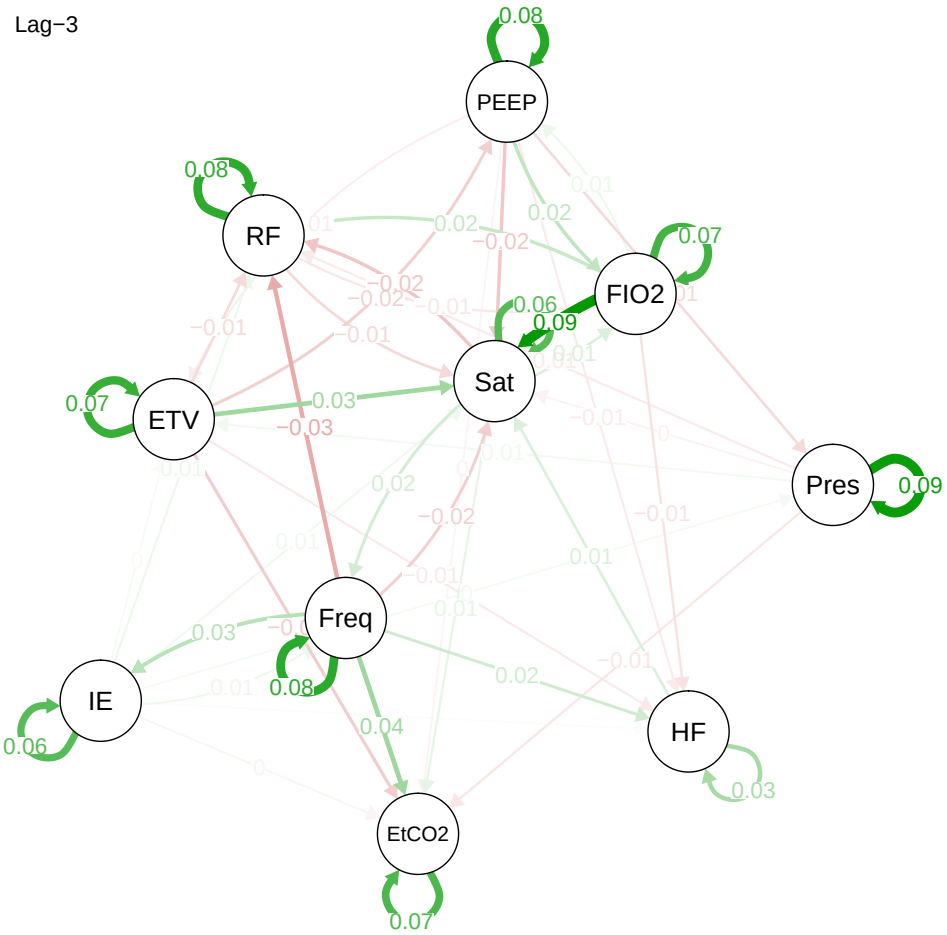

Between subjects model
Sat
PEEP
0.35
0.41
FIO2
Pres
0.28
0.41
RF
0.78
Freq
−0.33
HF
ETV
IE
EtCO2

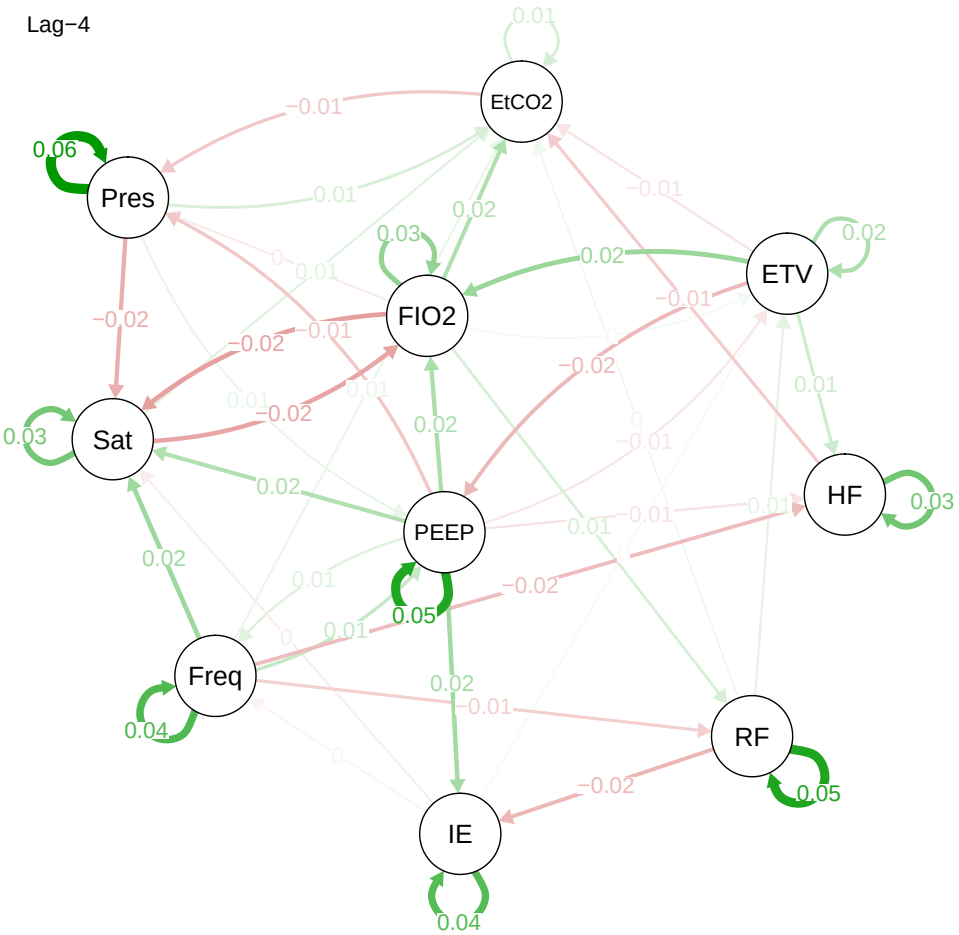
Contemporaneous model

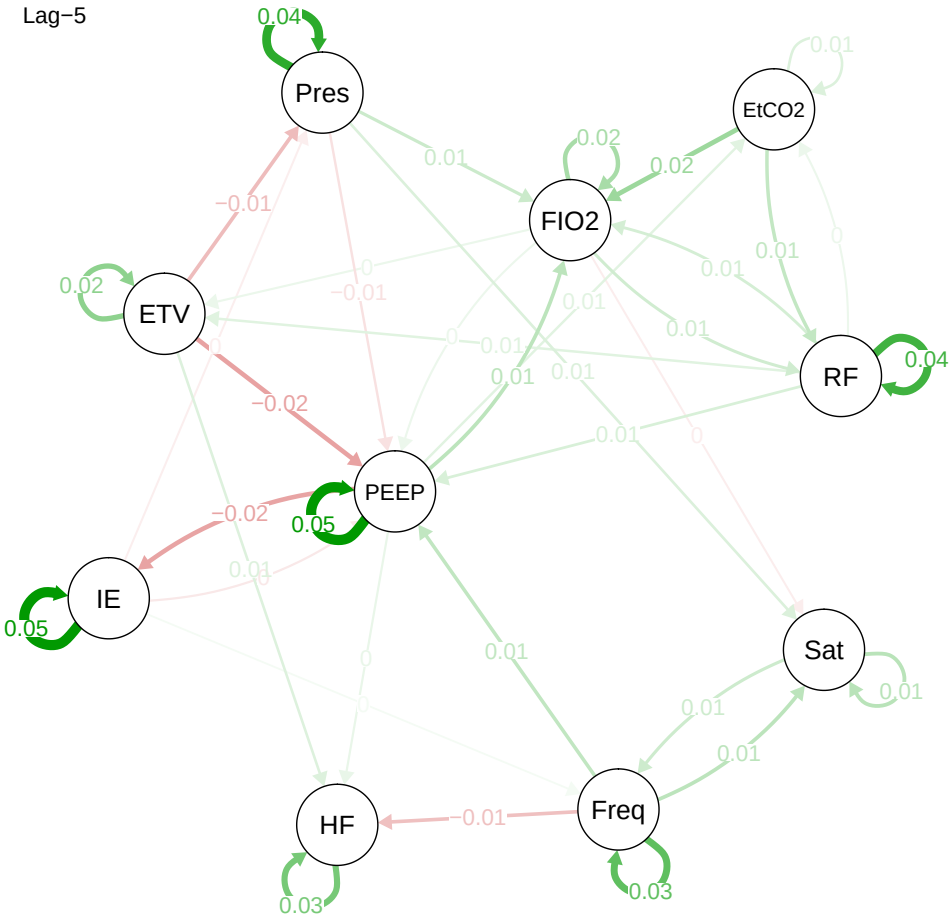



Lag−1
RF
ETV
PEEP
FIO2
Sat
Freq
Pres
IE
EtCO2
HF
0.43
0.15
0.26
0.47
0.67
0.4
0.32
0.27
0.7
0.11
0.68
−0.19
−0.13
0.1
0.06
0.05
0.04

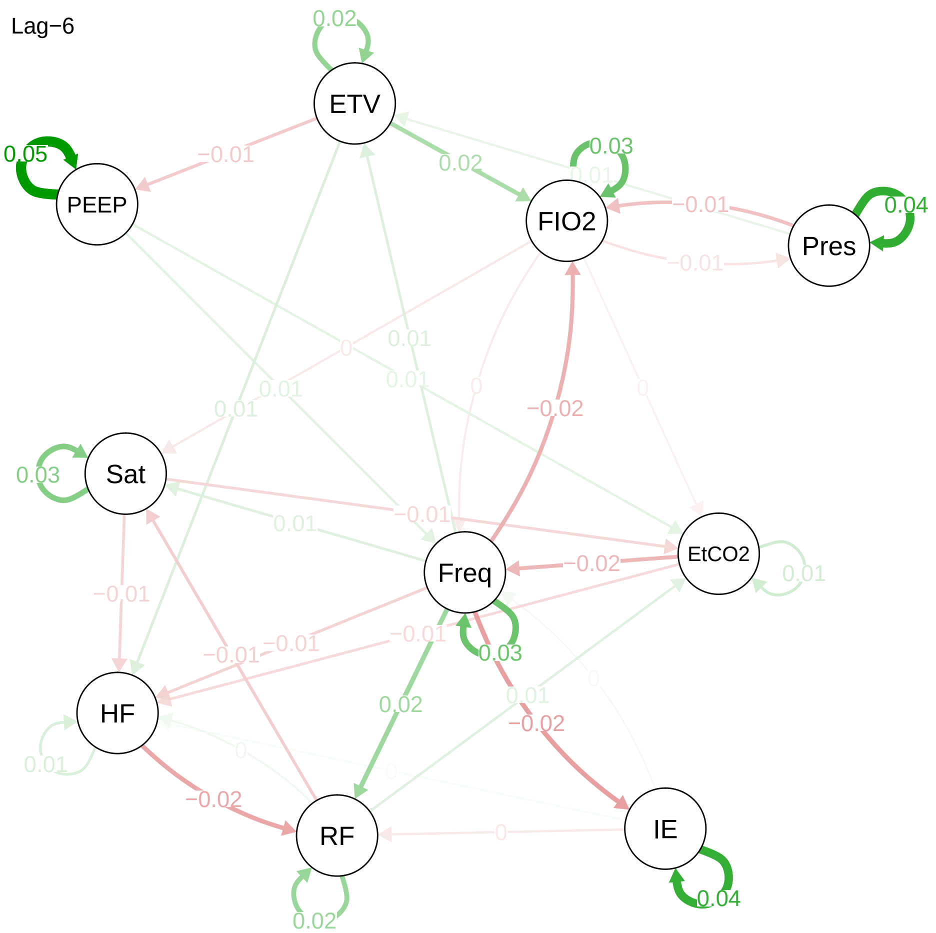

Lag−2
FIO2
IE
ETV
RF
Freq
Pres
EtCO2
HF
Sat
PEEP

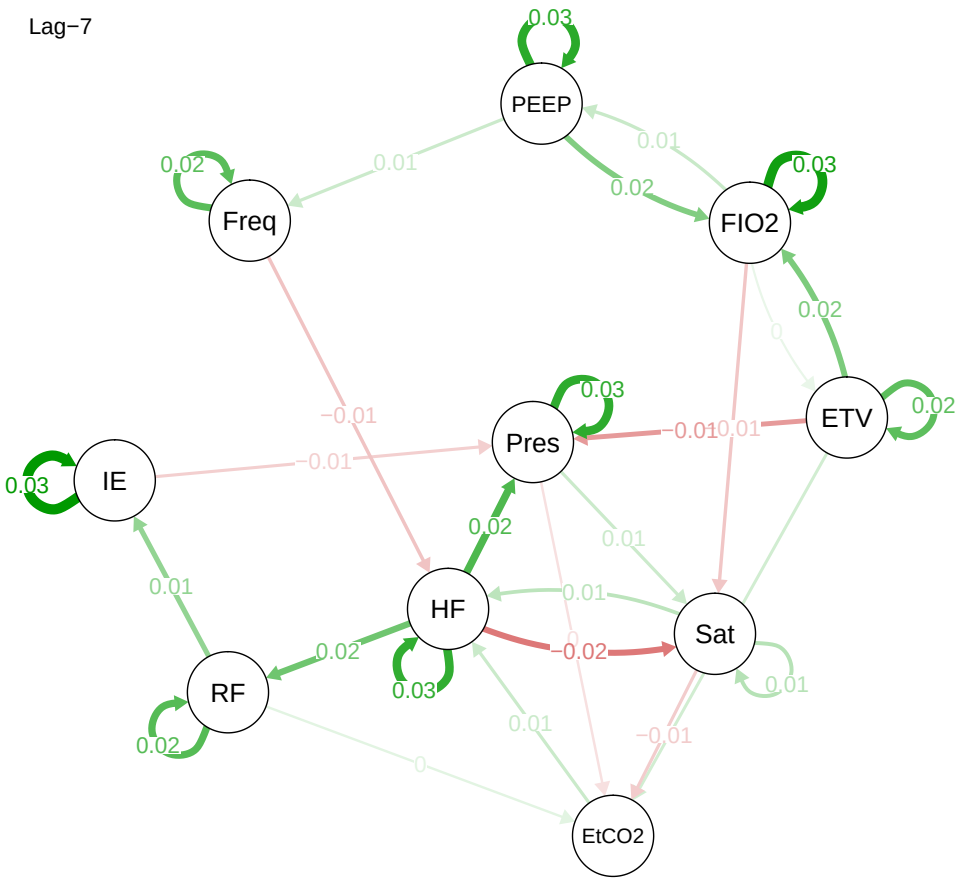

Lag−3
PEEP
RF
FIO2
Sat
ETV
Pres
Freq
IE
HF
EtCO2
0.08
0.08
0.07
0.06
0.09
0.07
0.03
−0.03
0.09
0.02
−0.02
0.03
0.08
0.04
0.06
0.02
0.03
0.07

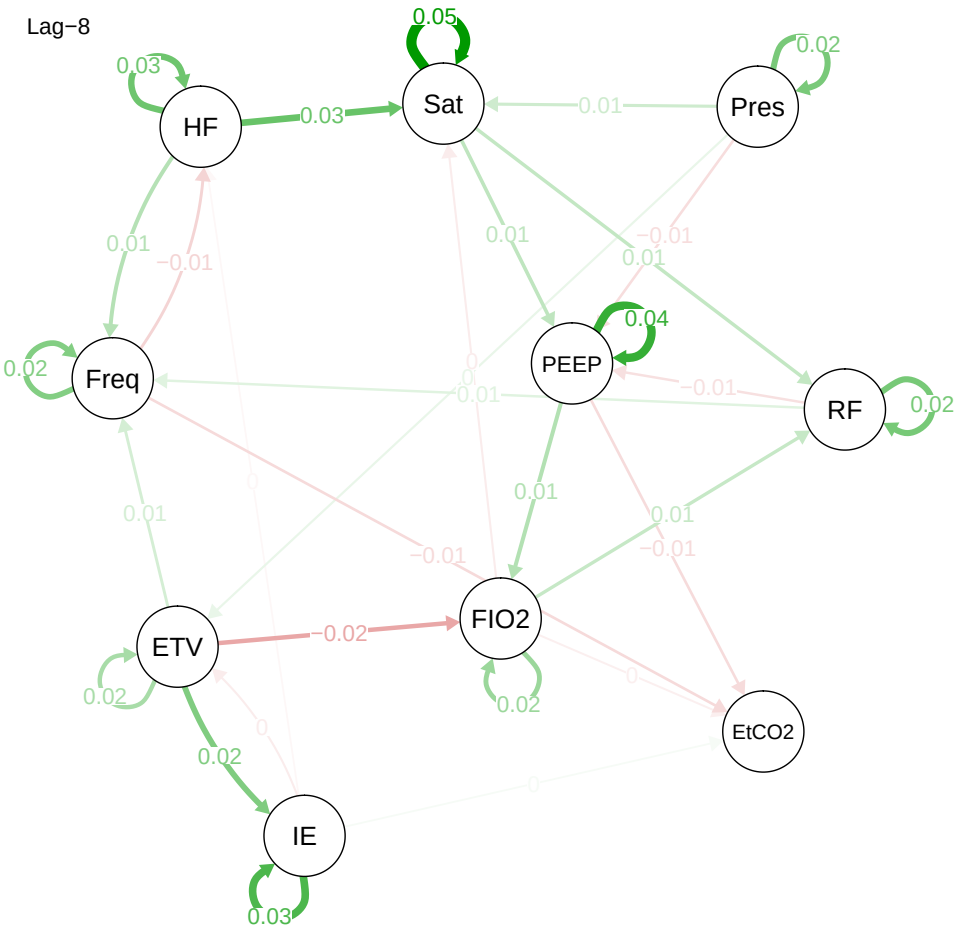

Lag−4
EtCO2
Pres
ETV
FIO2
Sat
HF
PEEP
Freq
RF
IE

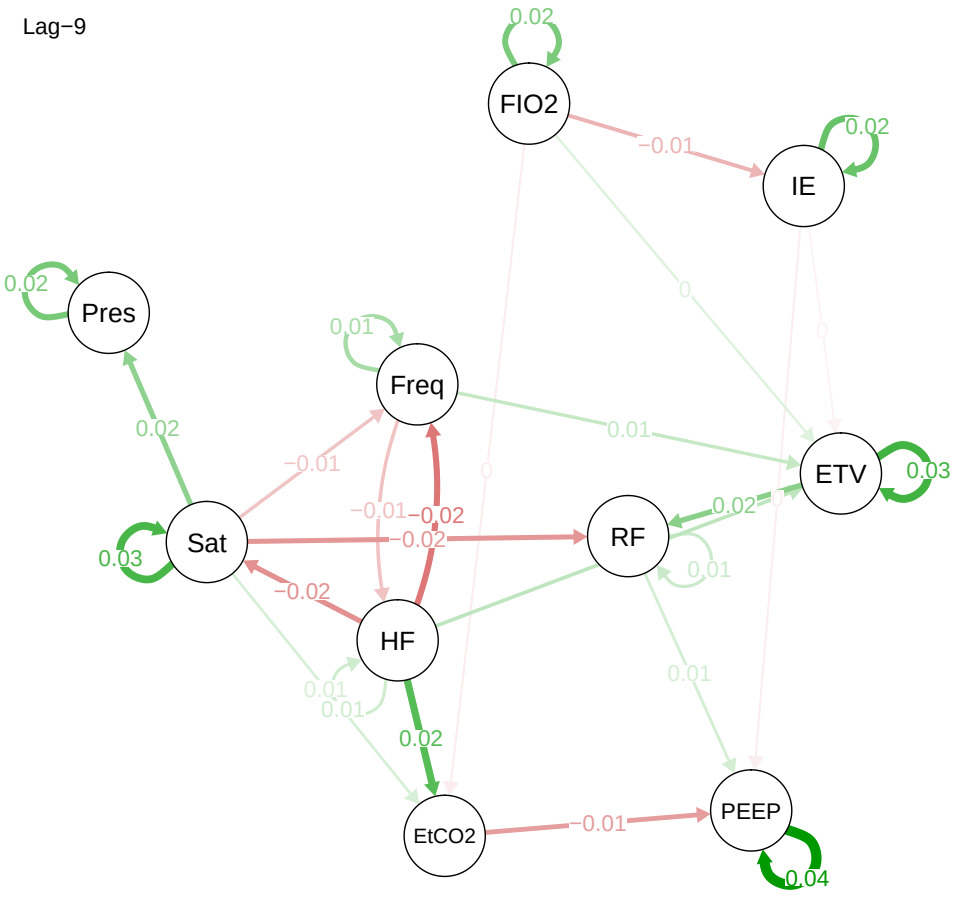

Lag−5
Pres
EtCO2
FIO2
ETV
RF
PEEP
IE
Sat
HF
Freq
0.04
0.01
0.02
−0.01
0
−0.02
0.05
0.03

Lag−6

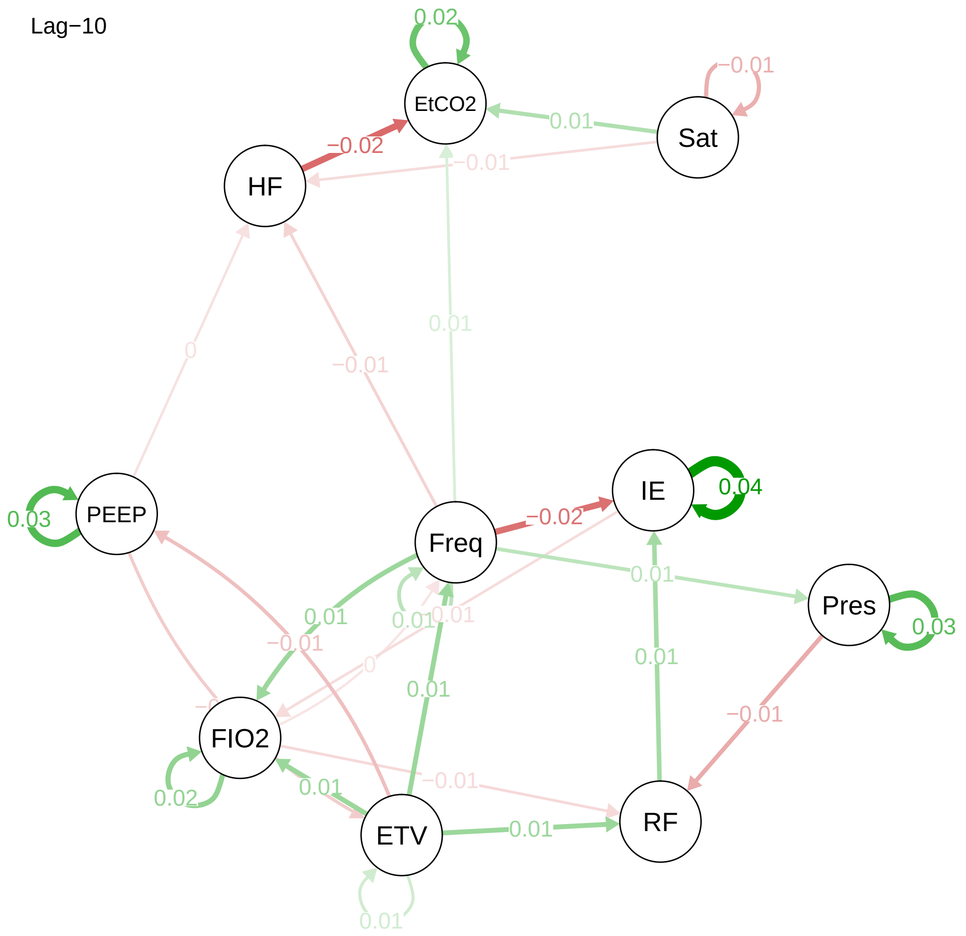



Lag−7
PEEP
Freq
FIO2
Pres
ETV
IE
HF
Sat
RF
EtCO2
0.03
0.02
0.01
−0.01
0
−0.02

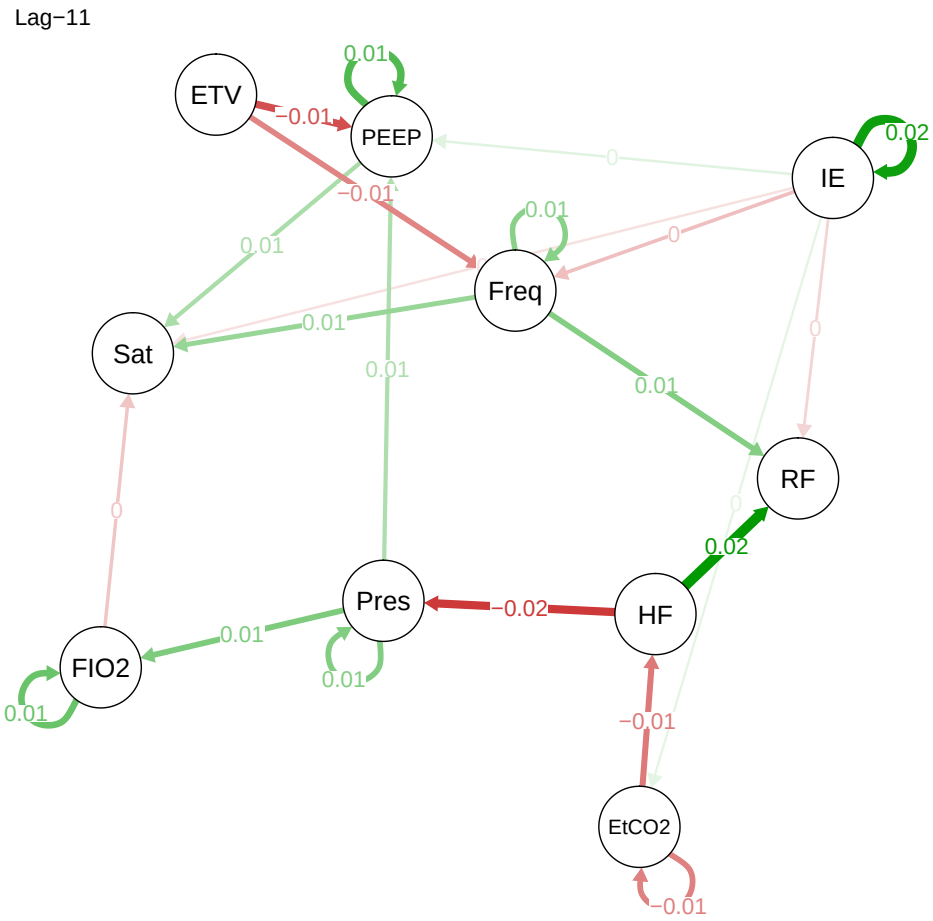

Lag−8
HF
Sat
Pres
Freq
PEEP
RF
ETV
FIO2
EtCO2
IE
0.05
0.03
0.02
0.03
0.01
0.02
0.01
−0.01
0.01
−0.01
0.04
0.02
0
0
−0.01
−0.01
0.02
0.01
0.01
0.01
−0.01
−0.01
−0.02
0.02
0
0.02
0
0
0.03

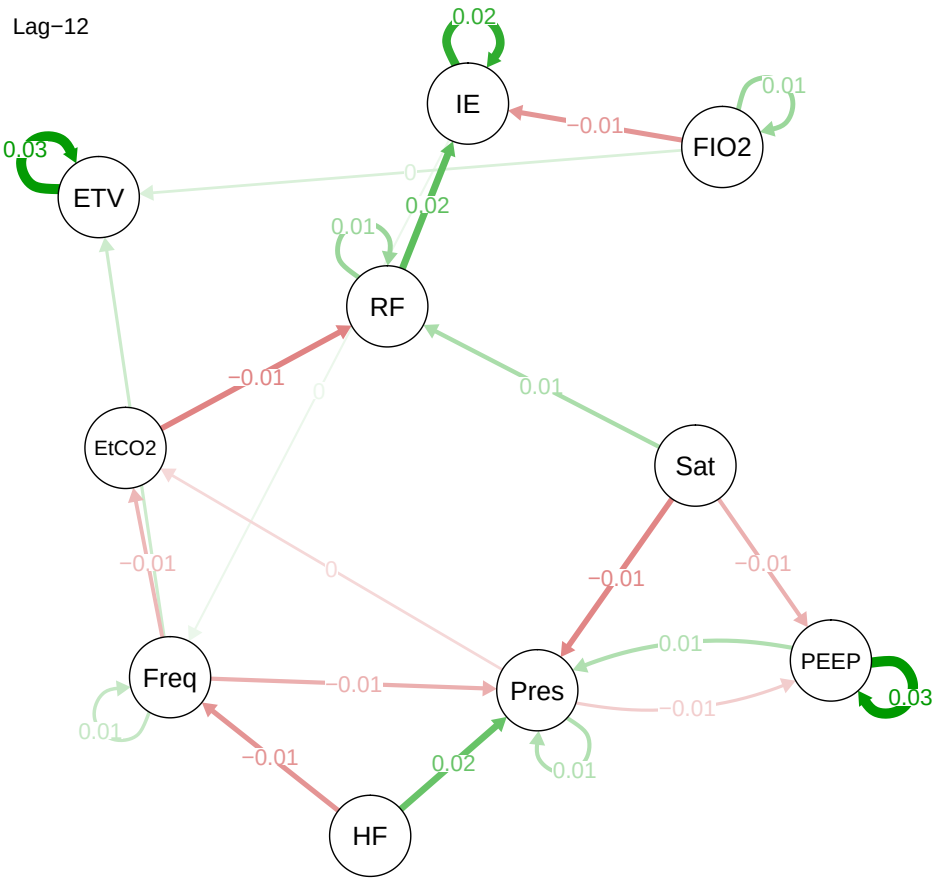

Lag−9
FIO2
IE
Pres
Freq
ETV
Sat
RF
HF
EtCO2
PEEP
0.02
−0.01
0.02
0.02
0
0
0.01
0.01
0.02
−0.01
0.03
−0.01
−0.02
−0.02
0.02
0.03
−0.02
0.01
0.01
0.01
0.01
0.02
−0.01
0.04

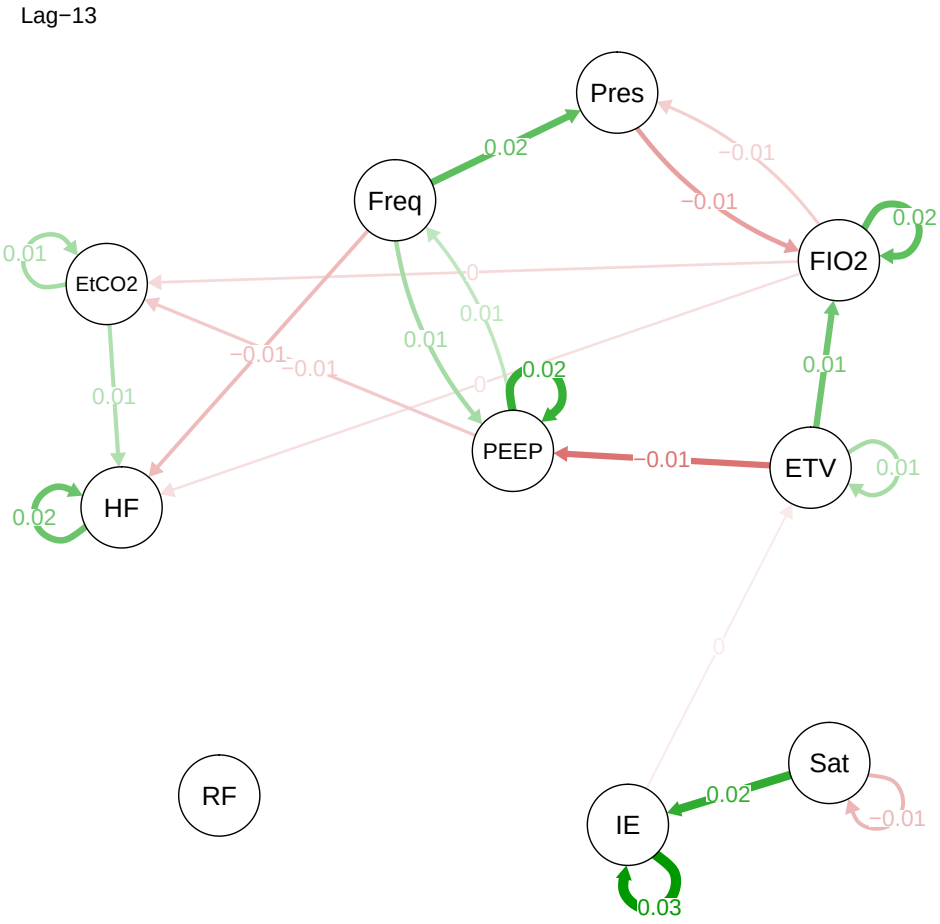
Lag−10

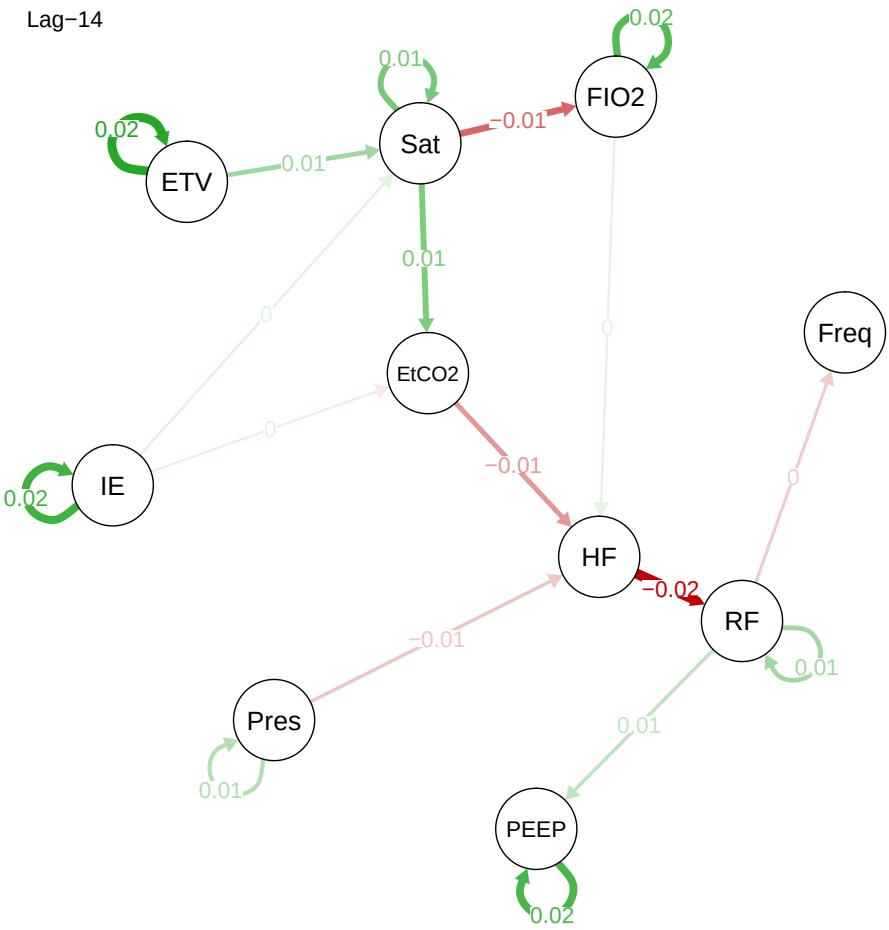


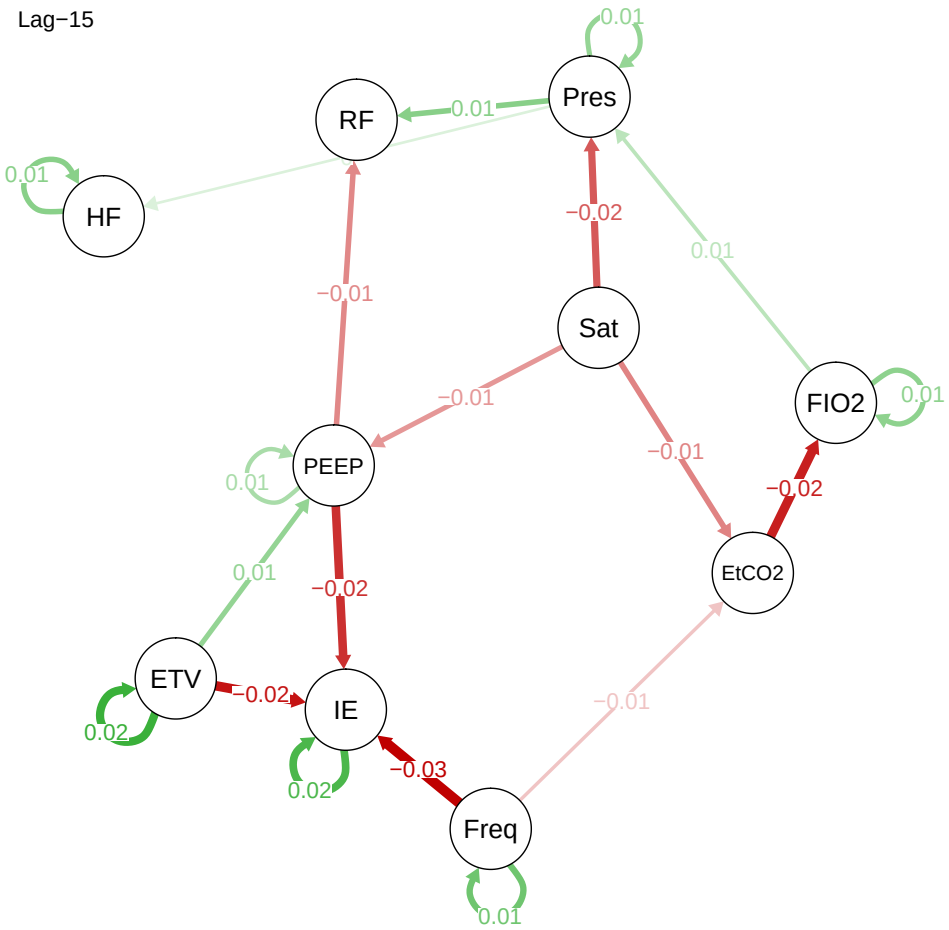
EtCO2
Sat
HF
PEEP
Freq
IE
Pres
FIO2
ETV
RF
0.02
−0.01
0.01
−0.02
−0.01
0.01
0
−0.01
0.03
−0.02
0.04
0.01
0.03
0.01
0.01
0.01
−0.01
0
0.01
0.01
−0.01
0.02
0.01
−0.01
0.01
0.01

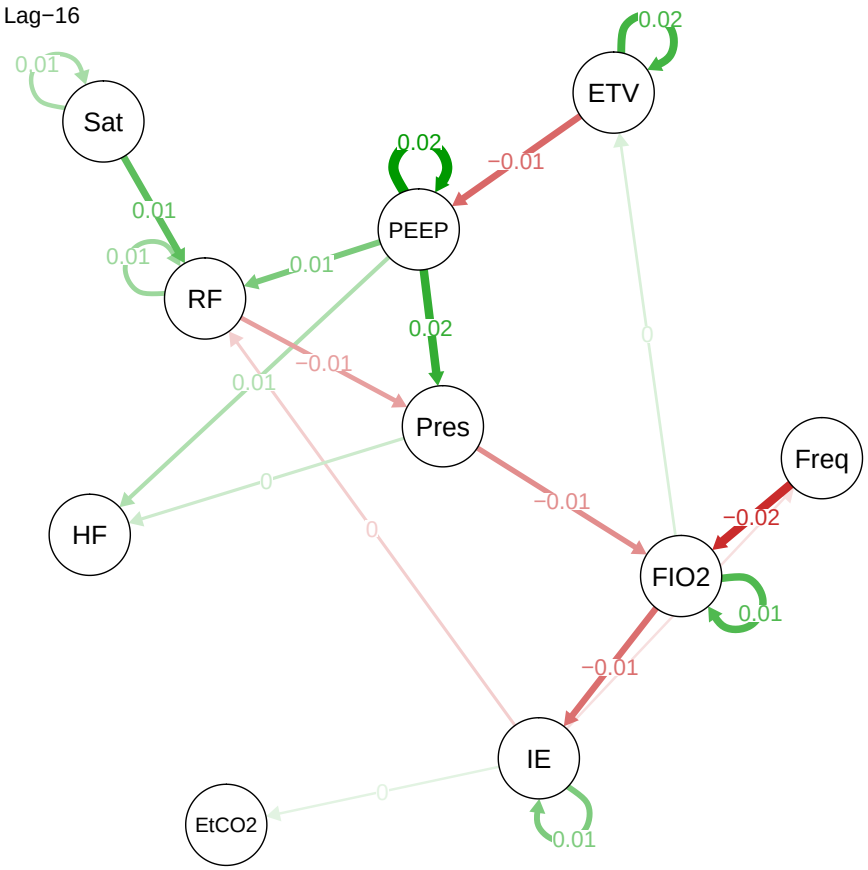

Lag−11
ETV
PEEP
IE
Freq
Sat
RF
Pres
HF
FIO2
EtCO2
0.01
−0.01
0.02
0
−0.02

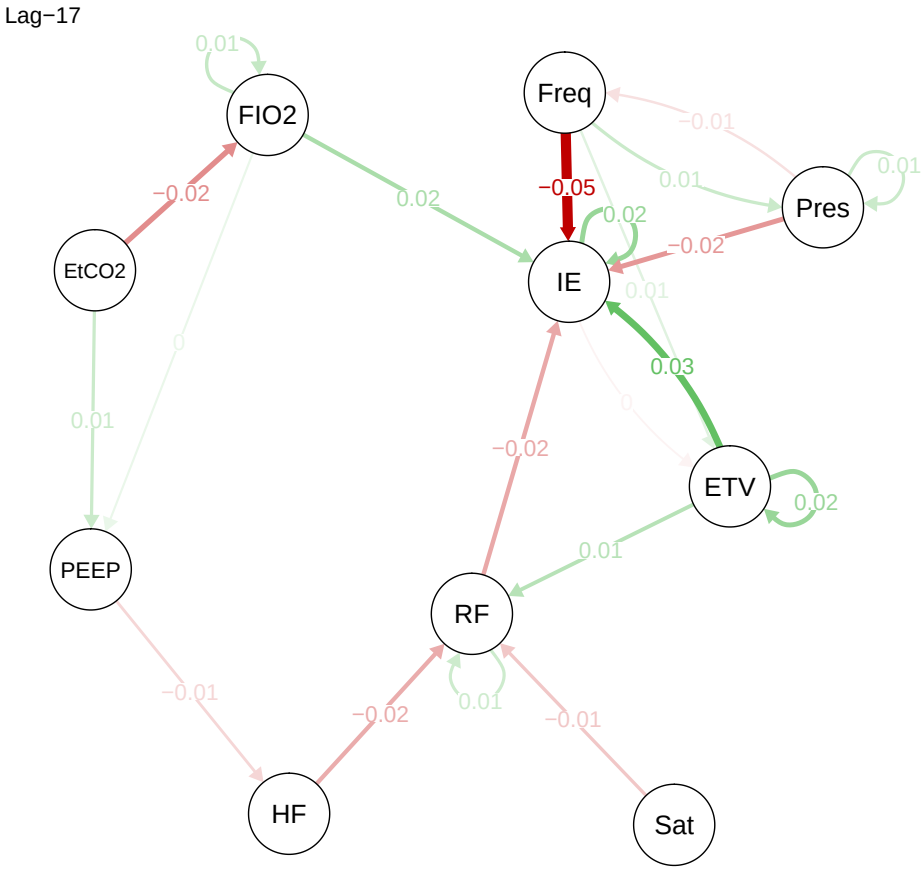

Lag−12
IE
FIO2
ETV
RF
EtCO2
Sat
Freq
Pres
PEEP
HF
0.02
−0.01
0.01
0.03
0
0.02
0.01
−0.01
0
0.01
−0.01
0
−0.01
−0.01
0.01
0.03
−0.01
0.01
−0.01
0.02
0.01

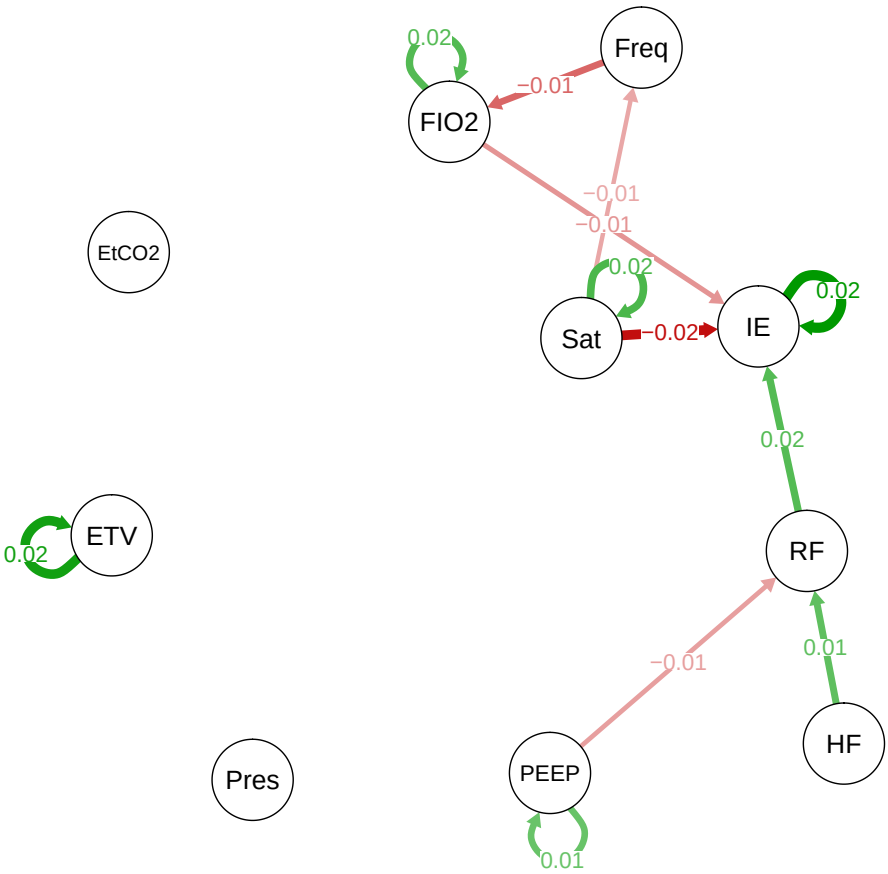

Lag−13
Pres
Freq
FIO2
EtCO2
PEEP
ETV
HF
RF
IE
Sat
0.02
−0.01
0.02
0.01
0
0.01
0.01
−0.01
−0.01
0.02
0.01
0.01
−0.01
0.01
0.02
0
0.02
−0.01
0.03

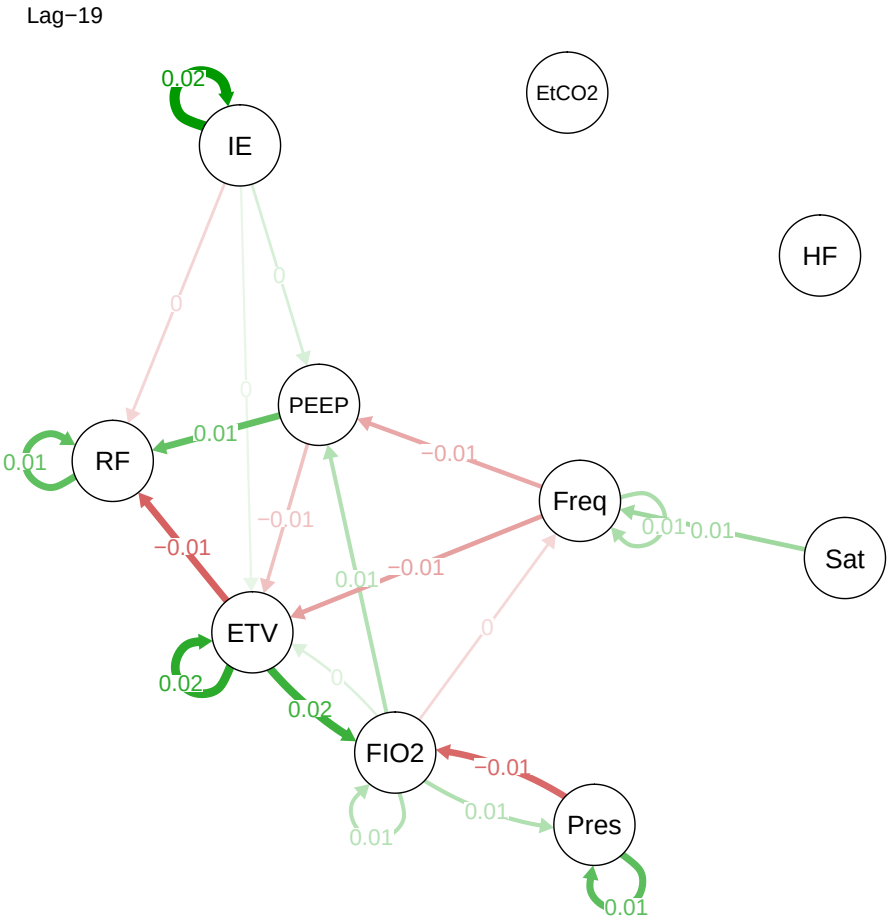

Lag−14
0.02
0.01
FIO2
0.02
−0.01
Sat
ETV
0.01
0.01
0
0
Freq
EtCO2
0
−0.01
IE
0.02
HF
0
−0.02
RF
−0.01
0.01
Pres
0.01
0.01
PEEP
0.02

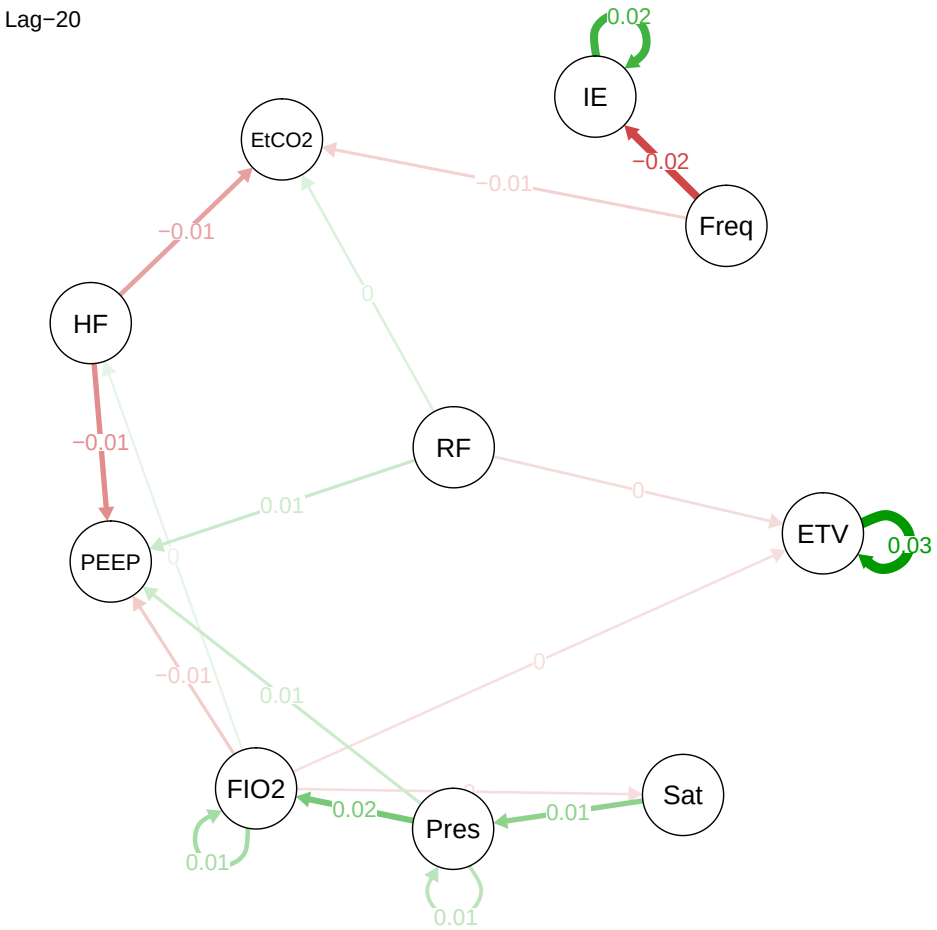

Lag−15
Pres
RF
HF
Sat
FIO2
PEEP
EtCO2
ETV
IE
Freq
0.01
0.01
0.01
−0.02
0.01
−0.01
−0.01
0.01
−0.01
−0.02
0.01
0.01
−0.02
−0.02
0.02
0.02
−0.03
−0.01
0.01

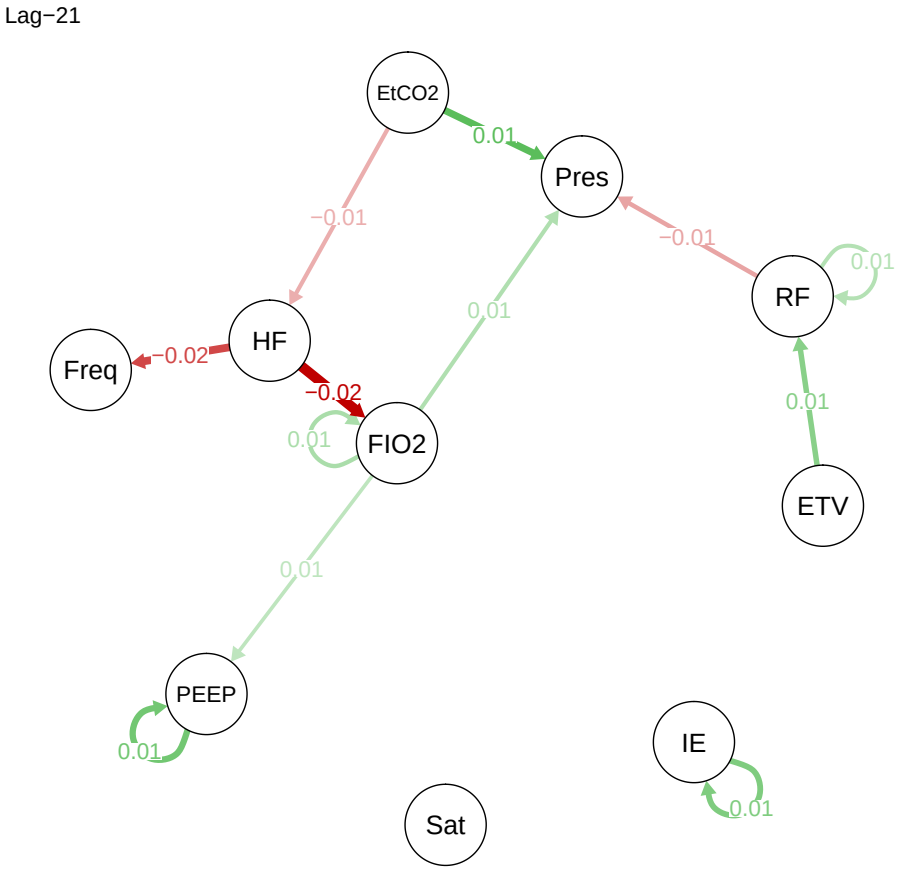

Lag−16
Sat
ETV
PEEP
RF
Pres
HF
Freq
FIO2
IE
EtCO2
0.01
0.02
−0.01
0.01
0.01
0.01
0.02
−0.01
0.01
0
0
−0.01
−0.02
0.01
−0.01
0
0.01

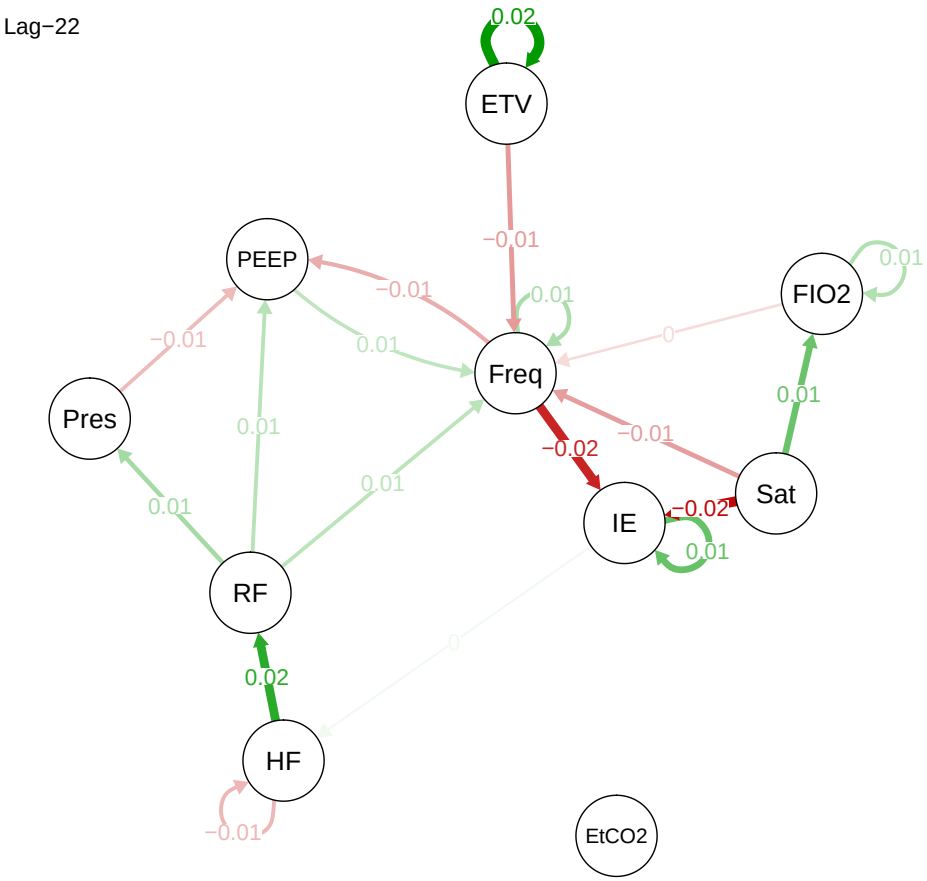

Lag−17
FIO2
Freq
Pres
EtCO2
IE
ETV
PEEP
RF
HF
Sat
0.01
−0.02
0.02
−0.05
−0.01
0.01
0.02
−0.02
0.01
0.03
0
0.01
−0.02
0.02
0.01
−0.01
−0.02
0.01
−0.01

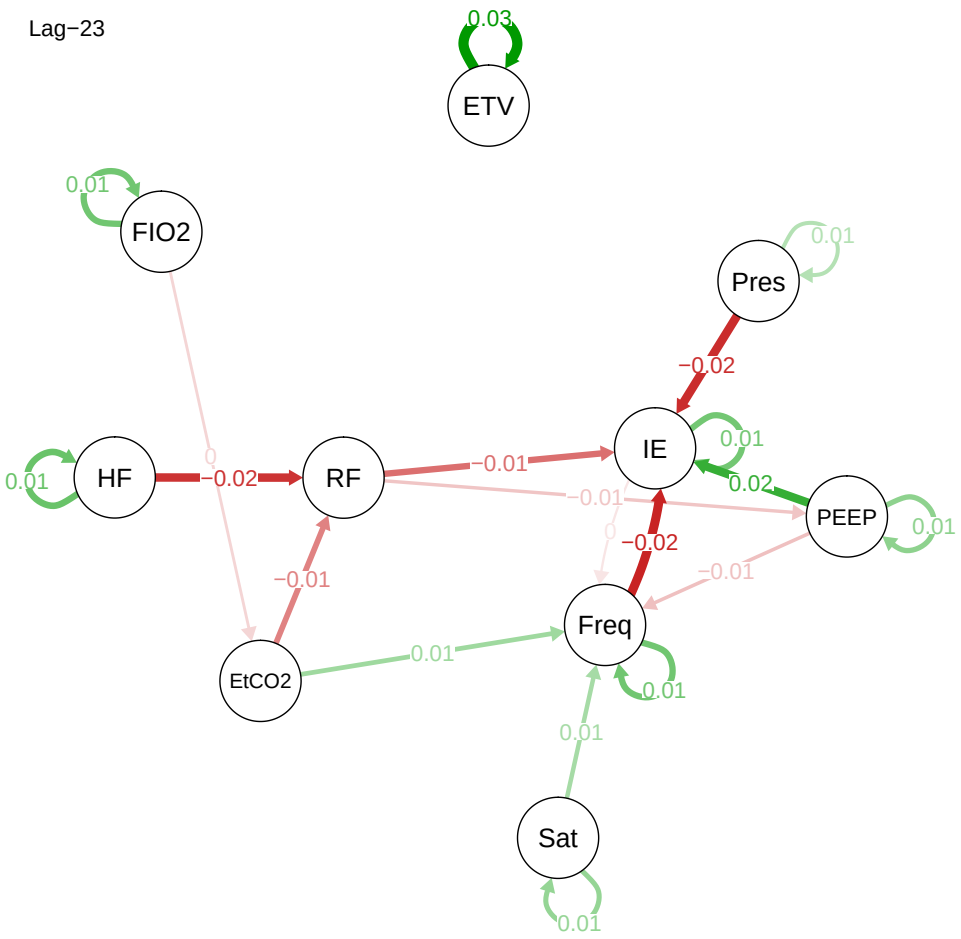

Lag−18
Freq
0.02
−0.01
FIO2
−0.01
−0.01
EtCO2
0.02
Sat
−0.02
IE
0.02
0.02
RF
ETV
0.02
−0.01
0.01
HF
Pres
PEEP
0.01


Lag−19
IE
EtCO2
HF
PEEP
RF
Freq
Sat
ETV
FIO2
Pres
0.02
0
0.01
−0.01
0.02

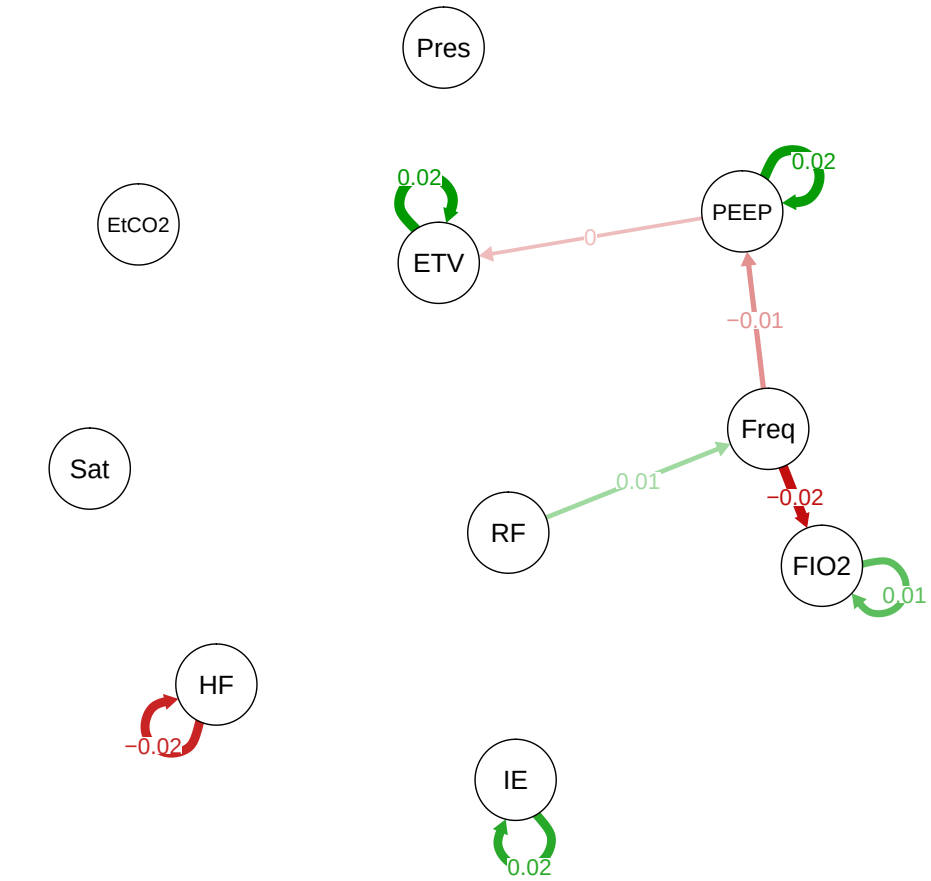

Lag−20
IE
EtCO2
Freq
HF
RF
PEEP
ETV
FIO2
Sat
Pres
0.02
−0.02
−0.01
−0.01
0
−0.01
0.01
0
0.03
−0.01
0.01
0
0.02
0.01
0.01
0.01

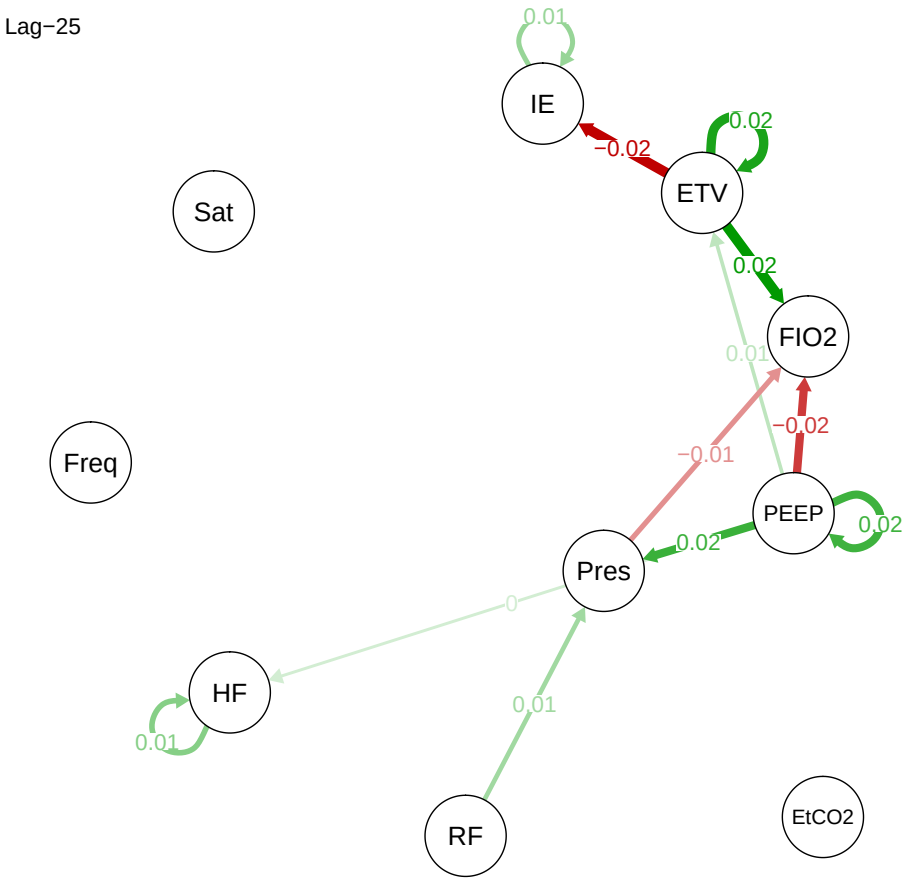

Lag−21
EtCO2
Pres
RF
HF
Freq
FIO2
ETV
PEEP
IE
Sat
0.01
−0.01
−0.01
0.01
−0.02
−0.02
0.01
0.01
0.01
0.01
0.01
0.01

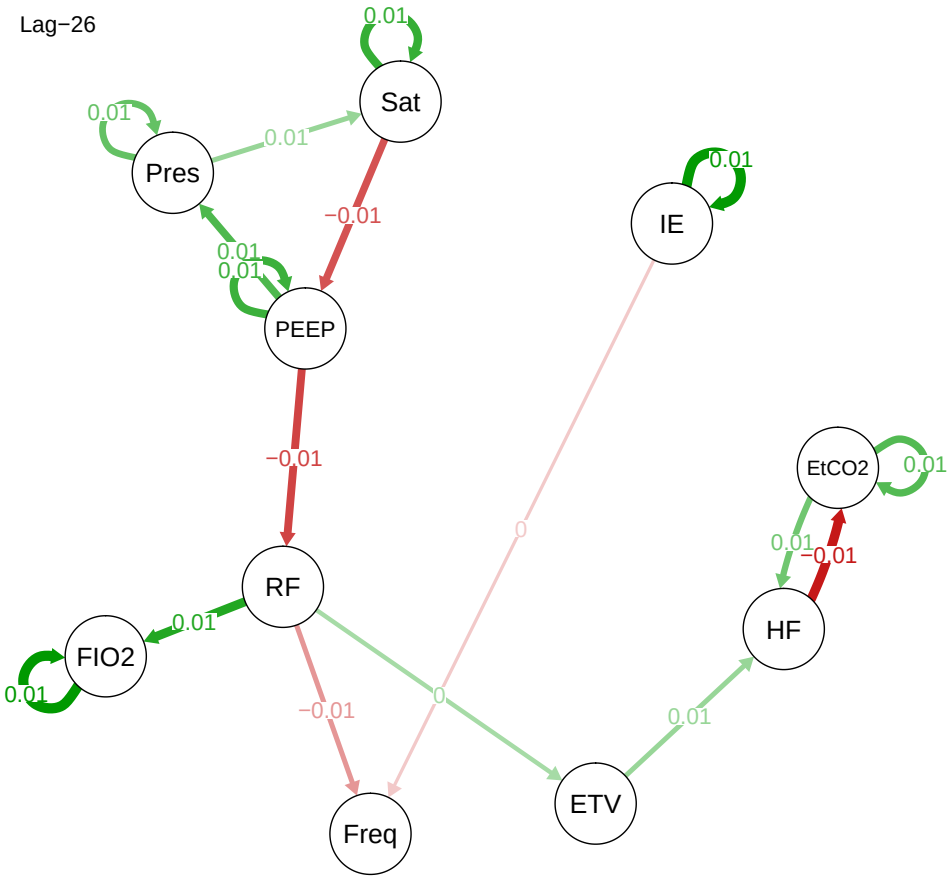

Lag−22
ETV
PEEP
FIO2
Pres
Freq
Sat
IE
RF
HF
EtCO2
0.02
−0.01
0.01
0
−0.02

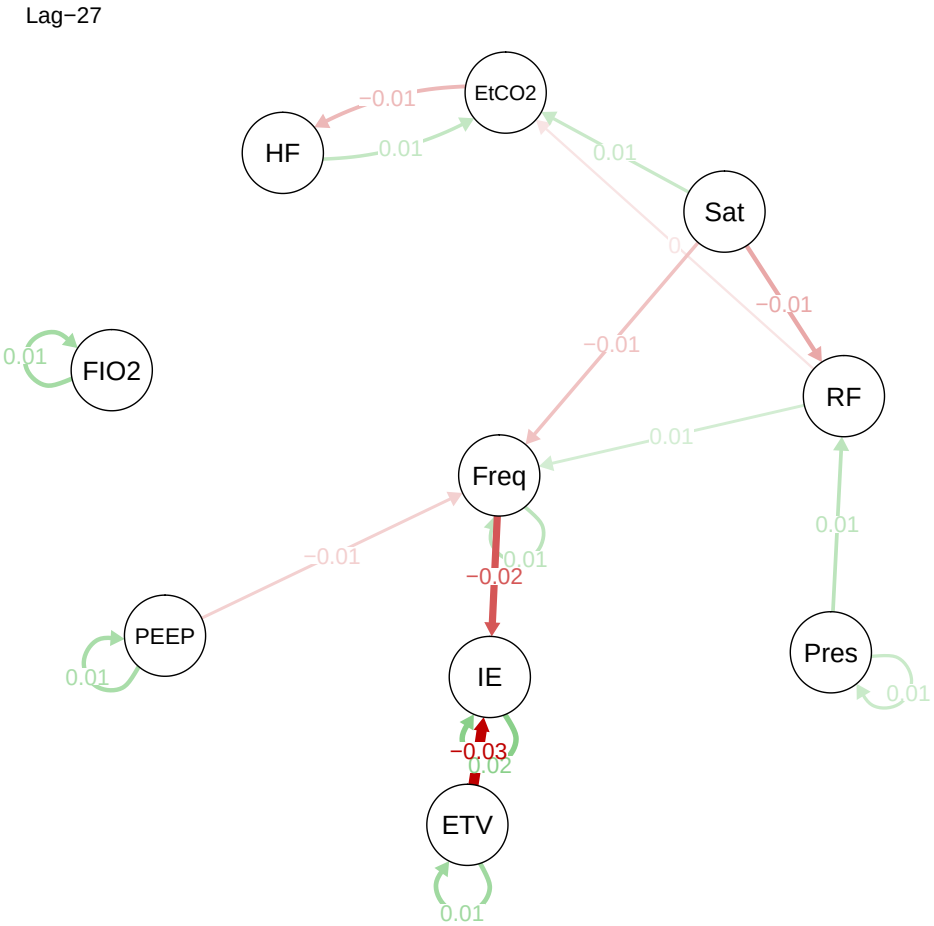

Lag−23
0.03
ETV
0.01
FIO2
0.01
Pres
−0.02
0.01
IE
0.01
HF
0
−0.02
RF
−0.01
0.02
PEEP
0.01
−0.01
0
−0.02
−0.01
−0.01
Freq
EtCO2
0.01
0.01
0.01
Sat
0.01

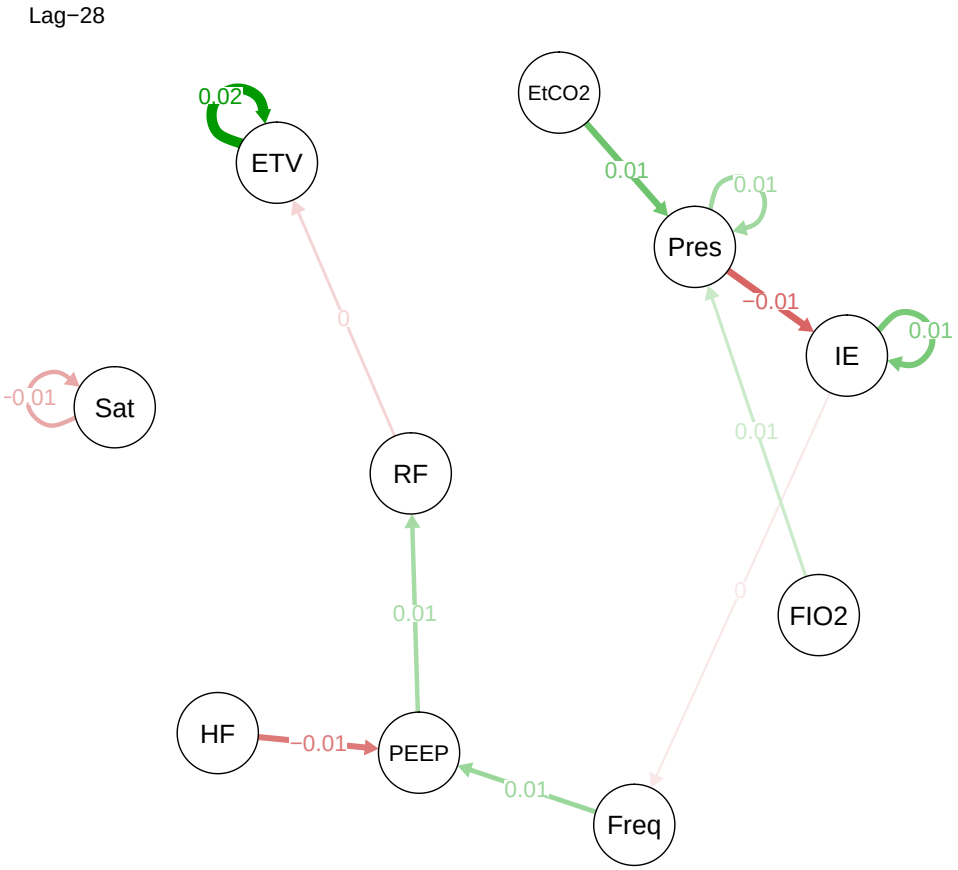
Lag−24

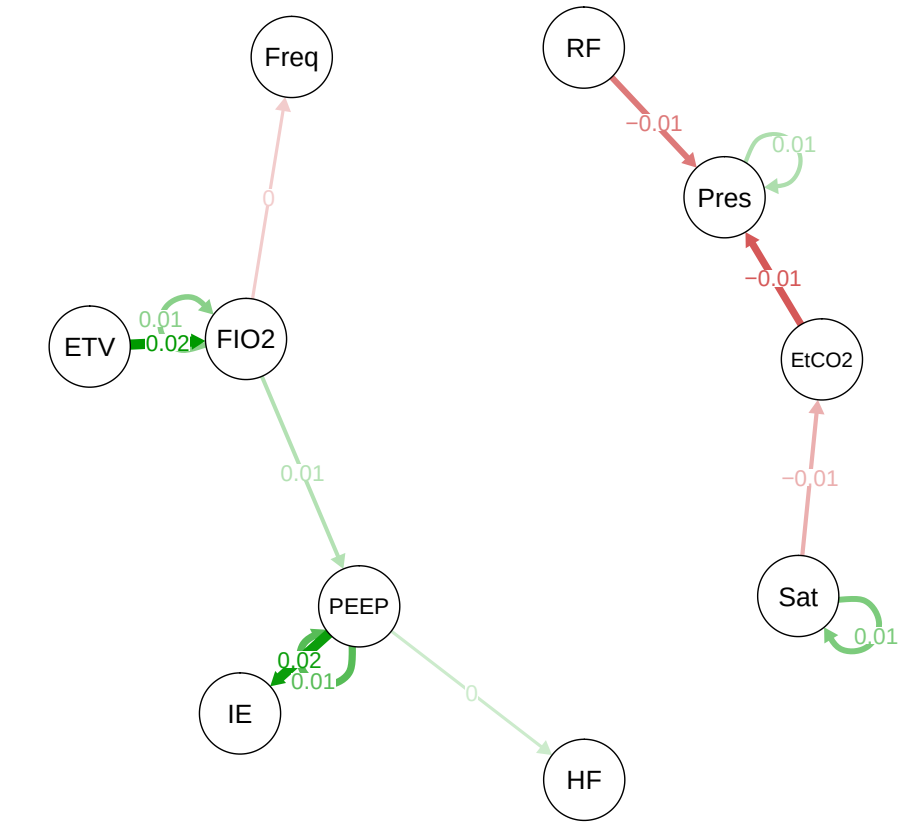

Pres
EtCO2
0.02
ETV
0
PEEP
0.02
−0.01
Freq
Sat
0.01
−0.02
RF
FIO2
0.01
HF
−0.02
IE
0.02

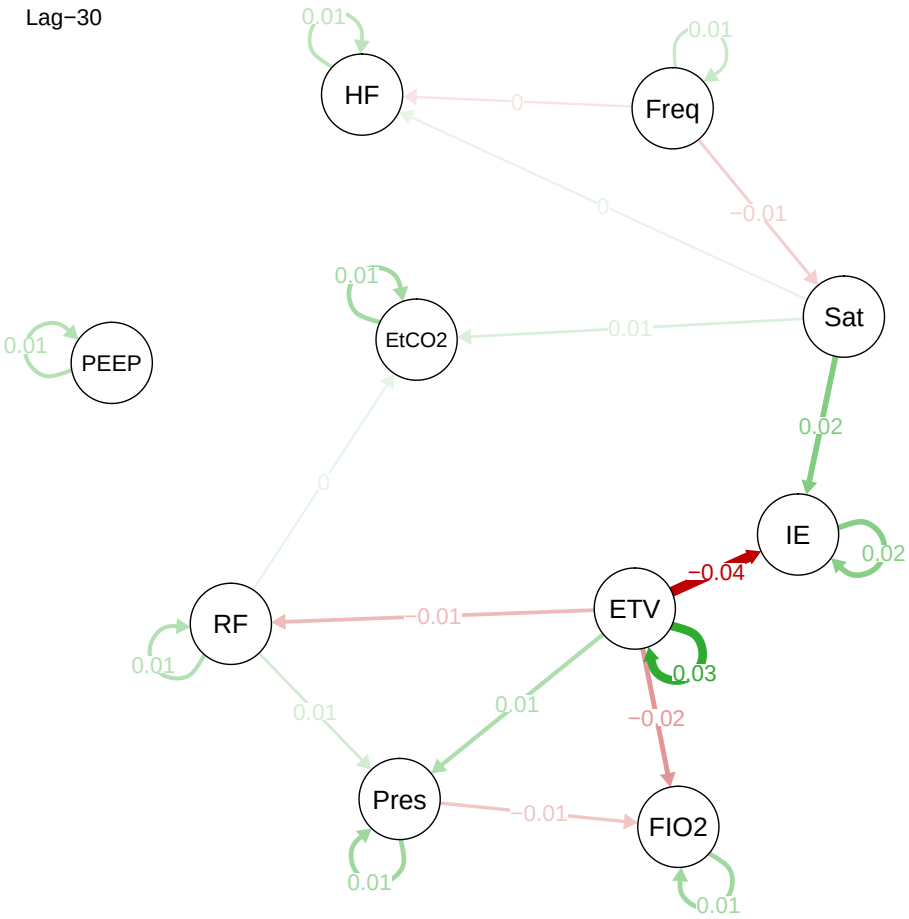

Lag−25
0.01
IE
−0.02
0.02
ETV
Sat
0.02
FIO2
0.01
−0.02
Freq
−0.01
PEEP
0.02
0.02
Pres
0
HF
0.01
0.01
RF
EtCO2


Lag−26
Sat
Pres
PEEP
IE
RF
FIO2
Freq
ETV
HF
EtCO2
0.01
−0.01
0


Lag−27
EtCO2
HF
Sat
FIO2
RF
Freq
PEEP
IE
Pres
ETV
−0.01
0.01
−0.02
−0.03
0.02


Lag−28
0.02
ETV
EtCO2
0.01
0.01
Pres
−0.01
IE
0.01
0
−0.01
Sat
0.01
RF
FIO2
0
0.01
HF
−0.01
PEEP
0.01
Freq


Lag−29
Freq
RF
−0.01
0.01
Pres
0
−0.01
ETV
0.01
0.02
FIO2
EtCO2
0.01
−0.01
PEEP
Sat
0.02
0.01
0.01
IE
0
HF


Lag−30
HF
Freq
Sat
EtCO2
PEEP
IE
ETV
RF
Pres
FIO2
0.01
0.01
0
−0.01
0
0.01
0.01
0.02
0
0.02
−0.04
−0.01
0.01
0.03
0.01
0.01
−0.02
−0.01
0.01
0.01