UNIVERSIDADE CATÓLICA PORTUGUESA

Structural Health Monitoring

A data-driven damage detection approach

Miguel Nogueira Rodrigues

Under the supervision of
Vera Miguéis

Católica Porto Business School, Universidade Católica Portuguesa
2020

UNIVERSIDADE CATÓLICA PORTUGUESA

Structural Health Monitoring

A data-driven damage detection approach

Final Assignment in the Dissertation modality
presented to Universidade Católica Portuguesa
to obtain the master's degree in MANAGEMENT
with specialization in Business Analytics

by

Miguel Nogueira Rodrigues

Under the supervision of
Vera Miguéis

Católica Porto Business School, Universidade Católica Portuguesa
March 2020

Acknowledgments

The author would like to acknowledge to the Instituto Superior de Engenharia do Porto for the opportunity to work on the S4Bridges project, as well as to the S4Bridges team for its hospitality and availability.

Also, a big thanks to the supervisor Vera Miguéis for her support throughout these six months, always showing availability to help when needed. This work could not have been done without her wisdom.

Abstract

Despite their importance, many Structural Health Monitoring (SHM) systems still rely on human inspection to verify the condition of the structure under analysis. Thus, the present work focus on creating an intelligent system that is able to detect damage automatically. This system is based on a real-life structure and the data collected in both undamaged and damaged states of the structure.

Two SHM approaches are proposed. First, explanatory models supported by machine learning algorithms (linear regression, random forest, support vector machines and neural network) are used to predict the values of the physical properties monitored in a regular condition. By comparing the predicted and observed values, a potential abnormal condition of the structure is detected by means of a Hotelling T^2 control chart. In the second approach, a time series analysis is adopted, using the cointegration properties of the series to compute the relationships between the variables monitored. These relationships are monitored with a X-bar control chart, where a potential change in the relationship indicate the presence of damage.

The two proposed approaches revealed to be capable of damage detection only when there is indeed a damage. More so, after the damage has been induced in the structure, both were able to signal an anomaly before 24 hours have passed. These results support the fact that SHM systems constitute a relevant tool to support the decision-makers in charge of monitoring the condition of the structures.

Keywords: Structural Health Monitoring; Bridge diagnosis; Damage detection; Linear regression; Random forest; Support vector machine; Neural network; Cointegration analysis; Johansen cointegration procedure; Temperature effect

Sumário

Apesar da sua importância, muitos sistemas de Monitorização da Saúde Estrutural (MSE) ainda dependem da inspeção humana para verificar a condição da estrutura em análise. Assim, o presente trabalho foca-se na criação de um sistema inteligente capaz de detetar dano de forma autónoma. Este sistema é baseado numa estrutura real em que os dados são captados nos estados com e sem dano da própria estrutura.

Duas abordagens para a MSE são propostas. Primeiro, modelos explicativos suportados por algoritmos de *machine learning* (regressão linear, *random forest*, redes neuronais e máquina de vetores de suporte) são usados para prever os valores das propriedades físicas monitorizadas numa condição normal. Comparando os valores previstos com os observados, uma potencial condição anormal da estrutura é detetada por meios de uma carta de controlo Hotelling T^2. Numa segunda abordagem, a análise de series temporais é adotada, usando as propriedades da cointegração das séries para encontrar as relações entre as variáveis monitorizadas. Estas relações são acompanhadas por uma carta de controlo X-bar, onde uma potencial mudança nas anteriores indica a presença de dano.

As duas abordagens propostas revelam ter a capacidade de detetar dano apenas quando realmente ele existe. Mesmo depois de o dano ter sido induzido na estrutura, ambas foram capazes de sinalizar uma anomalia antes de passarem 24 horas. Estes resultados apoiam o facto de os sistemas de monitorização da saúde estrutural revelarem ser ferramentas relevantes ao suporte à tomada de decisão no que toca à monitorização da condição de estruturas.

Palavras-chave: Monitorização da Saúde Estrutural; Deteção de dano; Regressão Linear; *Random Forest*; Máquina de vetores de suporte; Redes neuronais; Análise de cointegração; Procedimento de cointegração Johansen; Efeito temperatura

Index

Figure Index

Table Index

Introduction

The current society lives surrounded by a growing number of infrastructures that are used regularly. Thus, it is imperative to have constant revisions and quality monitoring of the mentioned infrastructures to ensure the minimum quality standards, avoiding any victims and accidents that result from infrastructures degradation and damages. In this perspective, monitoring systems have gained relevance, in particular because many infrastructures such as bridges are used beyond their life expectancy and are exposed to higher pressure from automobiles due to increased transport capacity (Neves, Gonz, & Leander, 2018).

In order to detect and report structural damages, sensitive systems have been developed to avoid permanent damages or an altogether collapse of a structure. These systems are called Structural Health Monitoring (SHM) systems and are responsible for collecting data and use it as input in a panoply of techniques that will detect if future data refer to potential anomalies and therefore a possible damage.

Although there are many infrastructures that throughout the previous decades were monitored, originating high volumes of data that characterize their condition, only in some cases the data produced was used to support decision-making process in what regards health assessment (Tomé, Pimentel, & Figueiras, 2019). The data recorded has usually a large data size, a high number of variables and low quality, making the use of Data Mining (DM) technology a must to

extract knowledge from the data (Duan & Zhang, 2006; Gordan, Razak, Ismail, & Ghaedi, 2017).

Throughout the last years, complex methods have been suggested and implemented in the SHM literature, which in turn are being slowly implemented in real SHM systems. These methods can mainly be divided in explanatory models and time series based models. In short, the explanatory models use one or more variables (explanatory variables) to explain the structure characteristics (dependent variables) (Farreras-Alcover, Chryssanthopoulos, & Andersen, 2015), while time series analysis use historic data to find a trend in the structure characteristics (Omenzetter & Brownjohn, 2006; Worden, Cross, & Barton, 2012).

In this work, both methods will be implemented in the development of a SHM system capable of damage detection. For explanatory models, four different machine learning algorithms will be used including linear regression, random forest, support vector machine and neural networks. After, they will be tested against each other by means of performance metrics comparison. Only the model with the best performance will be considered. As for the time series based models the cointegration properties of the series will be studied to compute the relationship between variables.

The damage detection process comes in the form of control charts. In the explanatory models, the difference between the predicted and observed values will be the input of a Hotelling T^2 based control chart. Meanwhile, the cointegration residuals that come from the relationship among variables will be applied to a X-bar based control chart.

All the practical procedures were developed in R language. In order to allow other users to learn and replicate the methods adopted, all the written code will be shared in the following work. Throughout **Section 3** and **Section 4**, references will be made to the respective parts of the code.

Chapter 1
Literature review

Studies on SHM usually follow one of two approaches, either a global approach or a local approach (Chang, Flatau, & Liu, 2003). The first focuses on data collection and analysis of the state of the infrastructure considering only the dynamic effects and compares it to its normal condition. The local approach focuses on the quantification of the damage in specific parts of the structure such as the cables of a bridge (Farreras-Alcover et al., 2015). The local approach could be seen as the next step of the global-based SHM stage, where the first step is to detect damage and the second step to locate and quantify such damage (Sharma & Sen, 2018; Tibaduiza, Mujica, & Rodellar, 2011).

The table in **Appendix 1** summarizes some studies on the topic of SHM, highlighting the common practices and algorithms that have been used recently to support them. These studies are supported by three main types of data. The first will be called virtual data approach and relies on using a virtual simulated structure to generate data from an undamaged and damaged state. Another, called virtual damage data approach, refers to the use of a real-life structure to record data in an undamaged state and then simulate the same structure virtually with damage to gather new data. Finally, there is the situation in which both undamaged and damaged state data come from a real-life structure. This is called physical data approach.

Example of studies following a virtual data approach are: (Dunia & Qin, 1998; Kim, Ryu, Cho, & Stubbs, 2003; Kromanis & Kripakaran, 2013; Neves et al., 2018; Posenato, Lanata, Inaudi, & Smith, 2008; Slišković, Grbić, & Hocenski, 2012; Yan, Kerschen, De Boe, & Golinval, 2005). In this approach, the insertion of damage in the structure is easier and without any real-life danger. This approach also promotes consistency on the data since the method of data collection in the undamaged and the damaged state is the same. However, some real-world variables may not be encompassed in it, making the models that come from this data less reliable. These models usually do not involve independent variables, being the principal component analysis the most used technique in this setting. However, the variables considered when modelling are generally different between studies.

All studies analyzed that adopt a virtual damage data approach (Tomé et al., 2019; Tomé, Pimentel, & Figueiras, 2020; Wipf, Phares, Doornink, Greimann, & Wood, 2007), use a real bridge to record data in an undamaged state. This allows for close view of the reality, resulting in models that have an accurate base to incorporate the behavior of the bridge in question.

When it comes to a physical data approach, there is a need to divide this approach in two categories. The first category uses a small structure to replicate the real structure (Barthorpe, 2010; Cross, Worden, & Chen, 2011; Farrar, Doebling, & Nix, 2001; Kesavan, John, & Herszberg, 2008; Pandey, Thostenson, & Heider, 2013; Park & Inman, 2007; Phares, Lu, Wipf, Greimann, & Seo, 2013; Rosales & Liyanapathirana, 2017; Tibaduiza et al., 2011; Yan et al., 2005). In this category, several models are used for SHM purposes. Some examples are principal component analysis and linear regression. Despite the different variables used in these models, the most frequent are strain and vibration. The second category refers to the studies that use a real structure to record both data in an undamaged and damaged states (Da Silva, 2017; Farreras-Alcover et al.,

2015; Reynders, Wursten, & de Roeck, 2014; Worden et al., 2012). It is rare to have the opportunity to record data of a bridge with damage in a controlled way, making these studies have a high value added when considering the accuracy that SHM systems can have. In this setting, the most common variables used to support the models are both temperature and vibration. The principal component is the most popular method in this branch of the literature.

Overall, the literature reveals a clear preference for the use of techniques that do not accommodate independent variables, i.e. other techniques than the explanatory ones. The most used one is principal component analysis (Cross et al., 2011; Da Silva, 2017; Dunia & Qin, 1998; Posenato et al., 2008; Reynders et al., 2014; Slišković et al., 2012; Tibaduiza et al., 2011; Tomé et al., 2019; Yan et al., 2005) and cointegration (Dao, 2013; Tomé et al., 2019, 2020; Worden et al., 2012). Meanwhile, only a third of the studies presented use machine learning, prioritizing the use of linear regression (Farreras-Alcover et al., 2015; Phares et al., 2013), autoregressive models with exogeneous inputs (Park & Inman, 2007; Rosales & Liyanapathirana, 2017) and neural networks (Da Silva, 2017; Neves et al., 2018). Regarding the variables monitored by the SHM systems, the most widely adopted variables are vibration, then strain and finally temperature.

Chapter 2
Methodology and Data

This study uses a real-life structure to support the development of two SHM models, i.e. one explanatory model, supported by machine learning techniques, and another based on time series analysis, supported by cointegration. In the next section it is detailed the data collected and the methodology proposed.

1. Structure

As mentioned before, a real structure is used to support this study and is consequently used to gather undamaged and damaged state datasets. This structure is a partial representation of a bridge, formed by two iron beams with the lower being thicker (see **Figure 1**). The structure is equipped with several sensors, such as an electrical resistance extension meters, accelerometers, inclinometers, displacement transducers, GPS and thermometers. These sensors collect data at a predetermined interval of time. Every sensor writes data every 15 minutes, which in turn generates 96 observations every 24 hours. These sensors enabled the collection of data from the end of April 2018 until the beginning of April 2019.

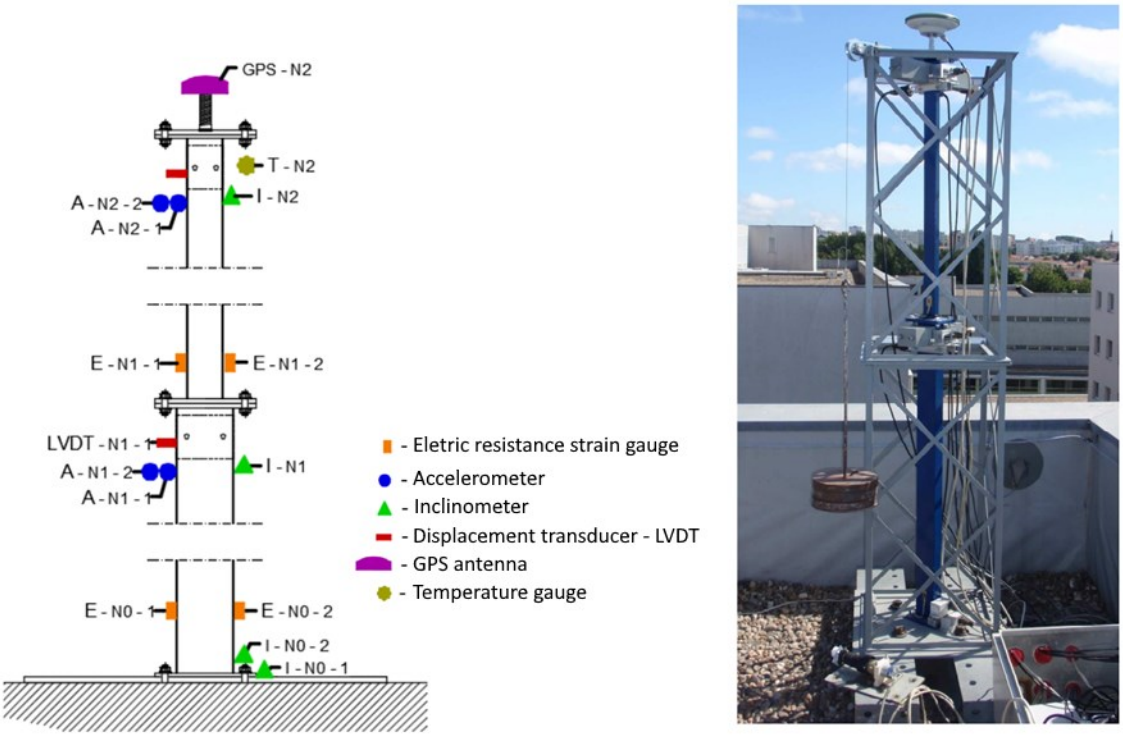

Figure 1: Visual representation of the structure

Throughout this period, the structure was exposed to two different situations (please refer to **Figure 2**). A period in which the structure is just being influenced by its dynamic and static properties which will therefore be used as a reference (between 27/04/2018 and 10/09/2018), and then a second state where some degree of damage is induced (from 10/09/2018 until the end of April 2019). The damage was induced by attaching a steel cable to the structure and connect it to the building where it is planted. The building is made of concrete and has an expansion joint. The cable is connected to the segment of the building that the structure is not placed. Since the cable is connected to the other segment of the building, a different response will be registered due to the different displacement caused by thermal contraction and expansion. Therefore, throughout the period, different tensions will be stimulated through the cable and these anomalies will be considered a case where the structure is under damage.

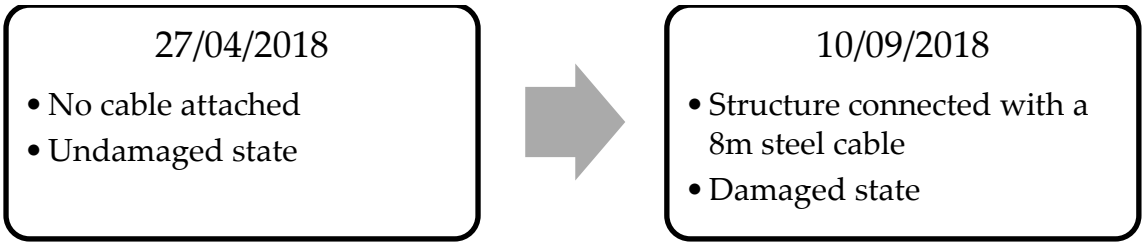

Figure 2: The two conditions the structure was subjected to

2. Data cleaning process

Since the accelerometers collect data only thrice per day, they were discarded from the dataset. This results in four group of variables being collected, i.e.: thermometers (7 sensors), strain gauge (4 sensors), displacement transducer (2 sensors) and inclinometer (3 sensors).

Due to the use of sensors, there will be inevitably some outliers that will need to be treated. Firstly, the dataset was divided into two, one containing the data that concerns the undamage period (stored in the *DataUD* object) and the other that refers to the damage period (stored in the *DataD* object). Only the former was considered in the data cleaning process. This was an attempt to mimic a real case scenario where there is access to data that is sure to be undamaged and then have data that may have outliers due to the existence of damage. In the case of data representing the damage period, having a data cleaning process could result in removing observations reflecting the damage and not anomalies in the sensors.

To identify outliers that should be removed from the undamage period, some techniques were tested. The process that provided better results was a procedure based on the absolute difference between past and future observations. First, the absolute difference between an observation and the previous observation was computed as well as the absolute difference between that observation and the subsequent observation. These two differences in the values of the variables monitored correspond to deviations observed in intervals of 15 minutes. If these

two differences were greater than the mean of the differences plus three standard deviations, the observation was removed. This procedure was also conducted for differences of 30 minutes, 45 minutes and so on, until an observation was compared to the homologous observation of the day before (96 observations before) and the homologous observation of the day after (96 observations after). A visual representation of this procedure can be found in **Figure 3** where a random observation in green is first compared to the observations 15 minutes before and after in red, then compared to the observations 30 minutes before and after in blue, and thirdly compared to the observation 45 minutes before and after in orange.

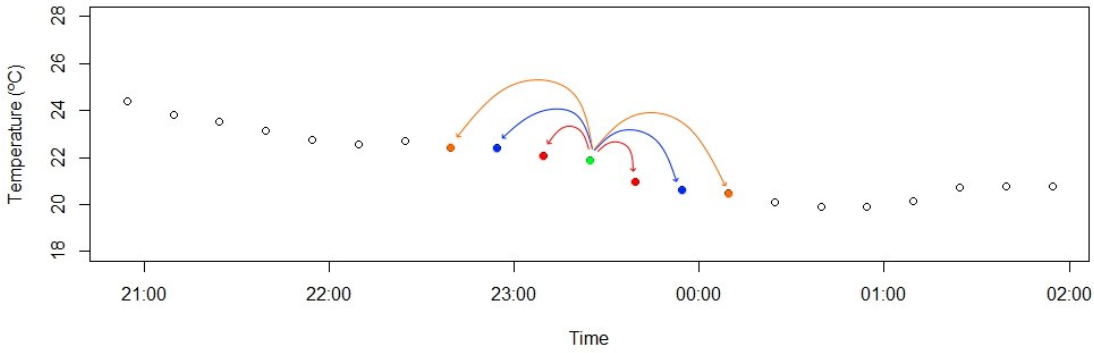

Figure 3: Sample of temperature data to demonstrate the outlier removal procedure

This procedure allowed the thoroughly detection and removal of outliers present in the dataset. Users interested in how this procedure was coded in R should follow **Appendix 2.1.** where an explanation of its mechanics is there fully described.

After computing the differences and classify them as being above or not the mean plus three standard deviations, they can finally be removed from the dataset altogether (see **Appendix 2.2.**). As mentioned before, this process was applied only to the data referring to the undamage period of the bridge. When it comes to the rest of the data, in the process of inserting damage, the data suffered

a shift in the recorded variables. The value of the shift was assumed to be equal to the difference between the mean of the last 672 observations of the undamage period and the mean of the first 672 observations of the damage period. This value was then applied to the damage data, removing the effect of the shift. The group of 672 observations considered was picked to represent a week of values.

3. Explanatory and Time series models

After a time consuming data processing stage, the data was used to support the damage detection system. In the development of the SHM system, two types of approaches were used and compared in order to define which model can better understand the data in question.

First, explanatory models were adopted, using both the structure and ambient temperature to predict the structure characteristics. In this case, several machine learning techniques were tested. Having the predictions of the structure characteristics, it is possible to evaluate whether the actual characteristics are deviating from what was expected to happen, i.e. it is possible to detect an abnormal behavior.

Concerning the time series analysis, the cointegration properties were used to draw the relationships between the variables regarding the structure characteristics. These relationships come in the form of linear cointegration vectors that can be used to generate cointegration residuals. Since these residuals represent the relationship between the variables, any abrupt changes in them should indicate a change in this relationship and therefore an anomaly.

3.1 Explanatory models

Foremost, the explanatory and dependent variables need to be defined. On this case, the temperature related variables will be used as explanatory variables, while the variables coming from the other three groups of sensors will be used as dependent variables independently. This is, an explanatory model will be created for each variable related to strain gauge, displacement transducer and inclinometer, resulting in a total of 9 models.

The temperature related variables that came from the 7 thermometers were chosen as the explanatory variables with the assumption that the temperature will affect the variables representing the structure, since it is part of its dynamic properties.

By creating these explanatory models, there is the ability to predict the expected values of the variables in a regular condition, and likewise compare them to the observed values. The difference between the predicted and observed values is the error or residual. The error can also be seen as the part that the temperature cannot explain. Following this line of thought, the residuals produced by these models are the true behavior of the variables without the influence of the temperature. Finally, by using the residuals as the input of the Hotelling T^2 control chart, it is possible to detect when there is damage in the structure (Dunia & Qin, 1998; Slišković et al., 2012; Tibaduiza et al., 2011; Tomé et al., 2019, 2020). Thus, for the damage period, the model should predict values significantly different than the ones observed since this data reflect an abnormal behavior, which was not observed before. On the other hand, for the period without damage, the predictions are expected to be similar to the observed values.

The model proposed should detect anomalies but should not flag false positive situations, i.e. situations in which there is not a damage and the model classifies as an abnormal period. Therefore, both the data that refers to the damage period, as well as the undamage test period that consists on the last month of the

undamaged period were considered to evaluate the ability of the model proposed to detect a damage when it happened. This means that only the data referring to the period before the undamage test period was used to verify the ability of the SHM model. This period of data was considered to train (calibration period) and test (validation period) the four machine learning techniques used for prediction purposes. In **Figure 4**, a representation of the data division is illustrated. These machine learning algorithms were linear regression, random forests, support vector machines and neural networks. The respective parameters for each algorithm were computed and tested in unseen data. The parameters of the models were then tuned using a grid search approach and the final performance of the models was tested in unseen data.

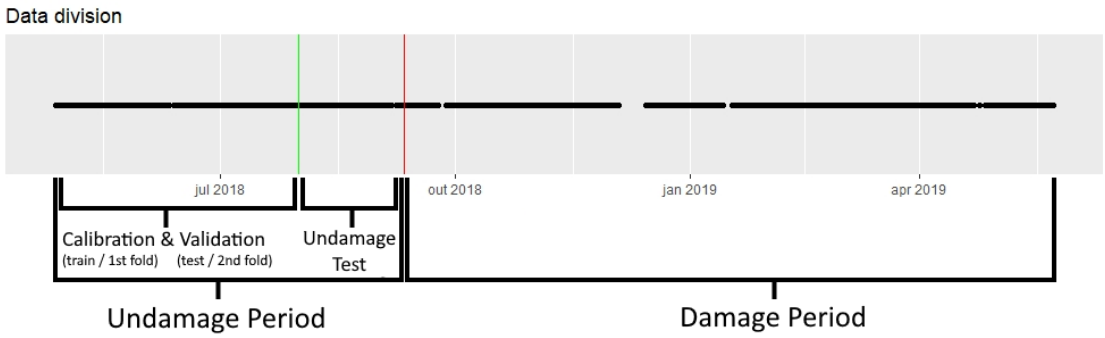

Figure 4: Visual representation of the data division for explanatory models

The model creation process went through randomly dividing the data in the undamage period, except for the undamage test period data, into two folds with the same number of observations (see **Appendix 2.3.**). Thereafter the cross-validation technique was used to evaluate the accuracy of the model and try to avoid overfitting problems. The first fold (stored in object *DataVal*) was divided into ten random samples, from which the algorithm takes nine of those and then tests its performance one the sample that was left out. This is done ten times, each time leaving a different sample out, and computing the ten sets of parameters.

Firstly, the linear regression was embraced. Although this is a simple algorithm, it has its advantageous for its low computational needs with the ability to create great predictive models (see **Appendix 2.4. and 2.5.**). Secondly, the Random Forest algorithm was used by referring to 1000 decision trees in order to create the regression parameters and decide how many variables are necessary to predict accurately (see **Appendix 2.6. and 2.7.**). Thirdly, the data was fed to a Support Vector Machine model with a variant cost and gamma parameter (see **Appendix 2.8. and 2.9.**). Finally, the Neural Network model was used with changes to the size and decay parameters (see **Appendix 2.10. and 2.11.**).

With the parameters estimated, they are to be applied to the second fold created earlier (stored in object *DataCal*). This fold will be referred as the validation period from now on. The validation period was also handled in a cross-section manner to avoid overfitting problems. The results that come from the validation period should already mirror what would be expected in the undamage period and will be applied to the undamage test and damage period (both stored in object *DataNew*). As said earlier, applying these parameters should wield an accurate prediction in the undamage test period but not to the in the damage period.

The performance metrics chosen to verify the performance of the machine learning models were the following: coefficient of determination (R^2), root mean square deviation (RMSE), mean absolute error (MAE) and mean absolute percentage error (MAPE).

Regarding the control charts, as mentioned before, the Hotelling T^2 control chart was used. This control chart can be used in multivariate settings and consequently can evaluate simultaneously whether the values of the strain gauge, displacement transducer and inclinometer variables reflect an anomaly.

In short, the Hotelling T^2 control chart considers the mean of each individual residual and a matrix of the covariance between each pair of residuals (Santos-

Fernández, 2012). This means that changes in the mean and variance of the residuals should result in a greater Hotelling T^2. An Upper Control Limit (UCL) is calculated and any values above of the UCL are out of control and represent an abnormal behavior in comparison to the rest of the data.

The Hotelling T^2 control chart is a two phased quality control chart. This means that in phase I one dataset is used to set a base mean and covariance matrix, while on phase II the mean and covariance matrix of the phase I are used as input along with a second dataset (Harris & Harris, 1995). In practice (see **Appendix 2.12.**), the validation period data is used for the Hotelling T^2 control chart in phase I, while both the undamage test and damage period data are used as input for the phase II.

3.2 Time series models

Time-series models take a set of observations ordered in a time order, registered in a defined pattern, and try to find a trend to fit it. These models do not accept datasets with missing values since it will violate the rule of being registered in a defined pattern, on this case every 15 minutes. In order to bypass this, there is a need to infer some missing values. This obviously has drawbacks since it will try to replicate the normal behavior of the series while the original data may have been abnormal. Nonetheless, it is a necessary step that should be taken into consideration when analyzing the results.

To interpolate the missing values, the Kalman filter was chosen. In simple terms, by feeding the algorithm a dataset and specifying the seasonal period, it will predict the missing value by replicating the trend of the series. For further details, please refer to Rudolf E. Kalman (Kalman, 1960, 1963; Kalman & Bucy, 1961). The process of interpolating the missing values in R is documented in the appendix (**Appendix 2.13.**).

Cointegration is a property of the time-series that defines the relationship between several variables. Variables are only said to be cointegrated if there is a linear relationship between them. This linear relationship can only be determined if the variables themselves are stationary. For a variable to be stationary, its parameters such as mean and variance cannot change over time (Chatfield, 1975). In case one or more variables are not stationary, the difference between an observation and p observations before must be done (Harris & Harris, 1995). Considering p to be 1, it is computed the difference between an observation and the one before, in this case it would be the differences in 15 minutes.

To check for stationarity, stationarity statistic test needs to be used, such as the augmented dickey-fuller (Dickey & Fuller, 1979, 1981). Nonetheless, before applying a stationarity test, the number of lags (p) need to be determined. To determine the optimal number of lags (p) some vectors autoregressive need to be created considering different p's and then compared, using some model selection criteria like AIC, HQ, SC, FPE (Liew, 2004, 2006).

After the series is deemed stationary, it is possible to explore the cointegration properties, computing the relationships between variables present in the series. In this work, the linear relationships between variables will be computed using the Johansen test (Johansen, 1988). The number of linear relationships that come from it can be at minimum zero and at maximum the number of variables minus one. If the number of linear relationships is zero, then the series are not cointegrated. The number of linear relationships is also tested in the Johansen test.

The determination of the optimal number of lags, the application of stationarity tests and the discovery of the cointegration vectors is present in the appendix (see **Appendix 2.14.**).

The parameters that define the relationships are called cointegration vectors, while the series produced by the cointegration vectors are called cointegration

residuals. Although the cointegration residuals values do not represent anything *per se*, any anomalies in them translates into a shift in the relationships between the variables. This shift is expected to signal a change in the behavior of the structure and the presence of damage. Therefore, by monitoring the cointegration residuals, it is possible to detect the presence of damage. For monitoring, an X-bar control chart is used, where the upper and lower limit are the mean plus or minus 3 standard deviations, respectively, in reference to the validation test period (**Appendix 2.15.**).

Chapter 3
Results

1. Explanatory models

In order to validate the use of explanatory models to detect anomalies, it is important to evaluate the performance of the predictive models proposed. This was performed in the calibration period. Moreover, the usage of different machine learning algorithms leads to different results. Thus, **Table 1** shows the performance of each model supported by Neural Networks model. **Appendix 3** presents the results for Linear Regression, Random Forest and Support Vector Machine models. Comparing the metrics, the models supported by Neural Networks seem to perform better, producing an average R^2 of 84% with a relatively low error variance in comparison the other models. Although the Random Forest model had some similar values, both the Linear Regression and Support Vector Machine Models had a worse overall performance. Based on these results, the neural networks were the algorithm adopted to perform the predictions.

Neural Networks (error metrics)									
	Strain F1 Middle	Strain F2 Middle	Strain F1 Lower	Strain F2 Lower	LVDT Upper	LVDT Middle	Inclinometer Upper	Inclinometer Middle	Inclinometer Lower
R2	0,84	0,79	0,85	0,87	0,87	0,70	0,92	0,89	0,88
RMSE	5,37	6,54	5,40	8,51	0,13	0,07	7,96	6,95	6,70
MAE	4,05	4,88	4,14	6,19	0,10	0,6	5,59	5,30	5,35
MAPE	0,02	0,03	0,02	0,18	0,10	0,65	0,00	0,00	0,00

Table 1: Error metrics for the Neural Network Model

Focusing on the *Inclinometer Upper* variable, the model provides accurate predictions. This can be verified in **Figure 5**, in particular when comparing the observed values in dark and the predicted values in blue. For the same period, the error observed in **Figure 6** also empathizes the quality of the predictions.

Having validated the performance of the neural networks in what regards their prediction ability in this setting, it is important to verify whether they are able to detect anomalies and whether they do not signal situations in which no anomaly occurred. Thus, it can be observed in green the predictions for a regular period (undamage test period) and conclude that the predictions seem to be aligned with observed values. The error for this same period also seems to be reasonable given the past error.

However, as soon as the damage is introduced (damage period), represented after the red line, the behavior of the structure changes and the model can no longer encompass it. This is also shown in the residuals, in **Figure 6,** which start to fluctuate a lot more. In fact, there is a significant change between the observed and predicted observations, from the moment the damage occurred, which is mirrored in the residual values. From this, an alarm system can be developed to detect this change automatically and more robustly.

While only the *Inclinometer Upper* variable was presented here, the graphics of the other variables can be found in the **Appendix 4**. The pattern

observed for *Inclinometer Upper* variable is similar to that observed for the other variables.

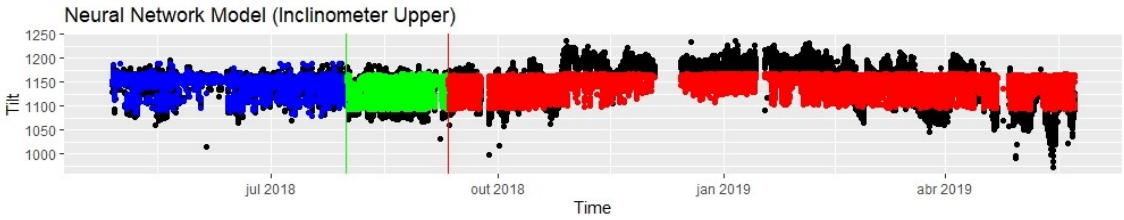

Figure 5: Observed and Predicted observations from the Neural Network model. Black dots refer to the observed values; Blue dots refer to the predicted values for the validation period; Green dots refer to the predicted values for the undamaged test period; Red dots refer to the predicted values for damage period

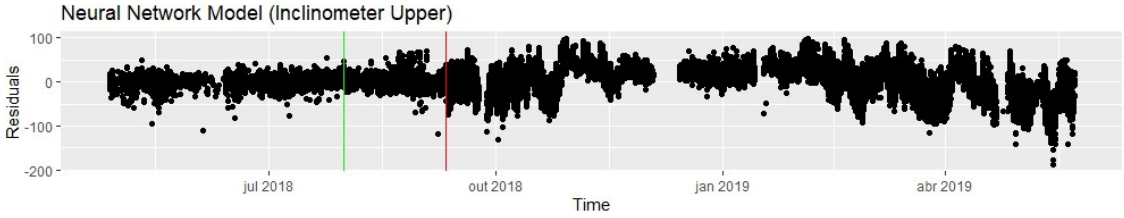

Figure 6: Difference between the Observed and Predicted observations from the Neural Network model

As a way to develop a tool that combines the residuals of the whole set of variables that are used to evaluate the condition of the structure, the Hotelling T^2 control chart is adopted. Looking at **Figure 7**, the validation period used to calibrate de control chart, only few values are above the control limit, more precisely 0,71% of the values. The same occurs for the undamage test period (see **Figure 8**) with 2,5% of the values above the UCL. As for the damage period (see **Figure 9**), there are much more values above the limit where 42,62% of the values are out of control, highlighting an abnormal behavior.

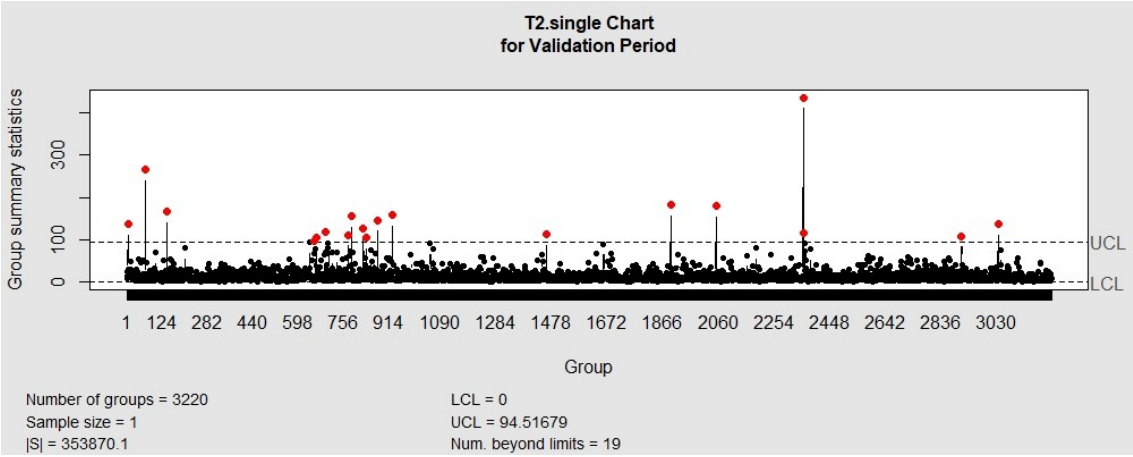

Figure 7: Hotelling T^2 control chart for the validation period

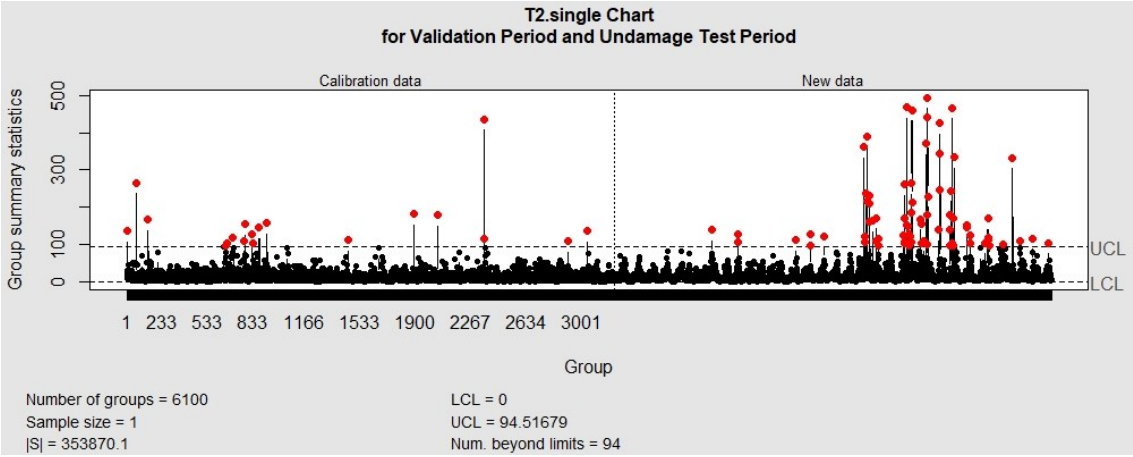

Figure 8: Hotelling T^2 control chart for the validation and undamaged test period. Vertical line separates these two periods

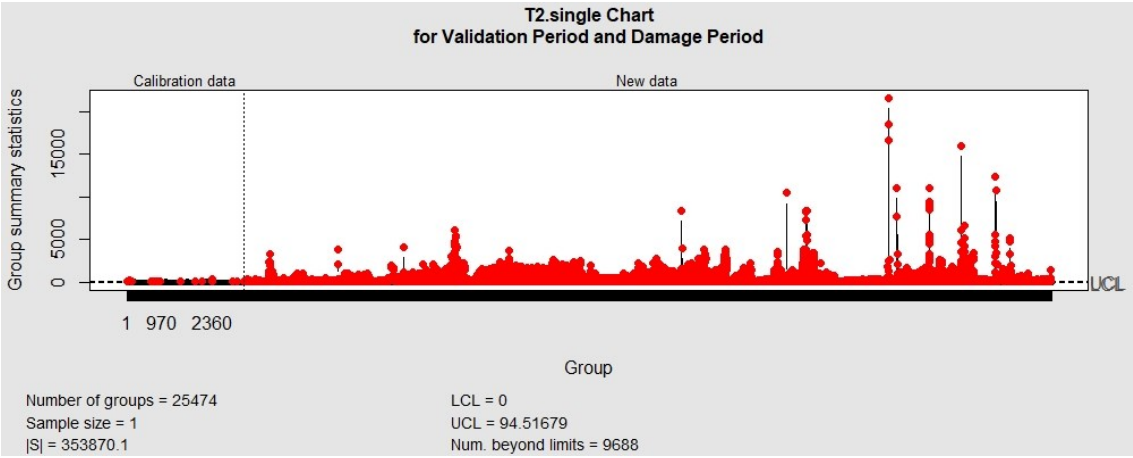

Figure 9: Hotelling T^2 control chart for the calibration and damage period. Vertical line separates these two periods

The number of successive observations out of the control limits was also analyzed. Thus, sets of 96 observations out of control were considered, in the sense that if in a period of 24h there were only values out of control, then there was with no doubt some damage in the structure. This approach showed to be promising, as this only occurred in the damage period. A similar analysis was developed considering only 2 hours and this revealed to be a better approach, as in case 8 observations in a row are out of control, there is a 100% confidence that there is something wrong with the structure in question (**Appendix 2.16.**). With this method false positives are completely removed, false negatives do not exist and the damage detection system is quicker.

The described procedure is illustrated in **Figure 10**. In this figure each dot represents a set of 8 consequent observations. If all the observations in a set are out of control, then *Alarm* equals 1. Observing the validation and undamage test period, it never happens to have 8 out of control observations for 2 hours straight. In the damage period, 12% of the sets include solely out of control observations which is quite alarming on itself. Nonetheless, the first damaged situation is flagged after 20h30m of damage being implemented. Also, as time passes, not only the number of sets where *Alarm* is 1 increase but the number of sets without any out of control observation decreases.

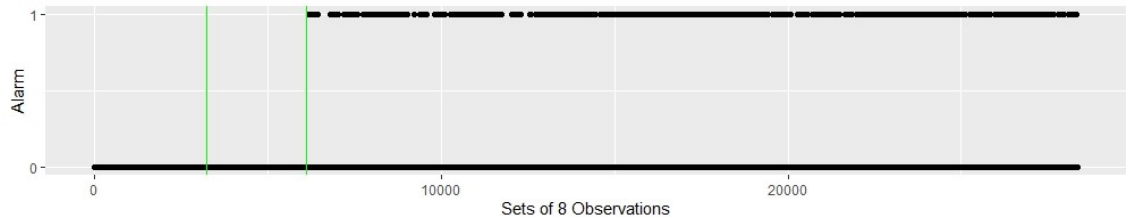

Figure 10: Graphical representation of which sets of 8 observations include only out of control observations

2. Time series analysis

After the data cleaning process referred in **Section 2.3.**, the series started to have some missing values, meaning that it was not possible to apply any kind of time series analysis. Therefore, as mentioned before, some values were interpolated with the Kalman filter. An example of interpolated values can be seen in **Figure 11**. After the missing values were interpolated, the series is finally complete in the sense that it can be used in time series analysis without violating any assumption. The data is then divided into 3 different datasets. In **Figure 12** it can be seen the data division, while **Table 2** presents the differences between this division and the one referred in **Section 3.1.** for the explanatory models.

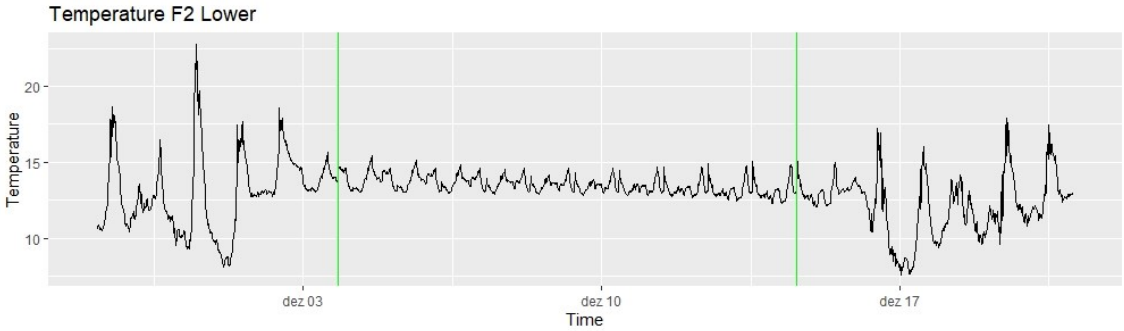

Figure 11: Sample of Temperature F2 Lower, where the values between the green lines have been interpolated using the Kalman Filter

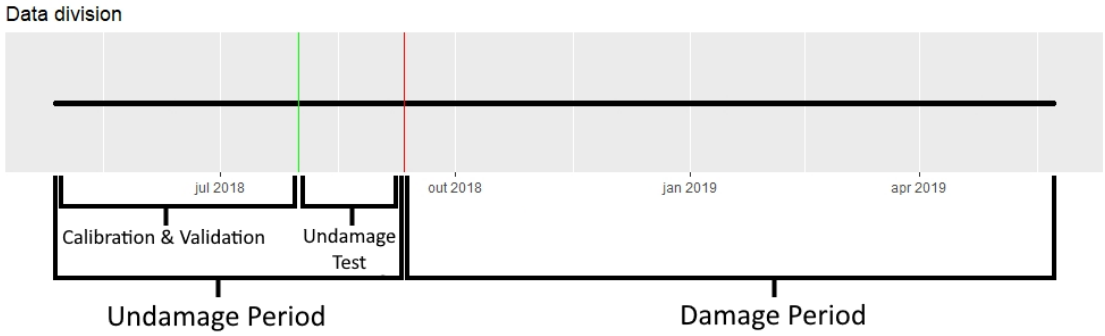

Figure 12: Visual representation of the data division for time series analysis

Explanatory models	Time series analysis
Calibration period	Calibration and Validation period
Validation period	
Undamage test period	Undamage test period
Damage period	Damage period

Table 2: Differences in the data division done for the explanatory models and time series analysis

As stated before, a stationarity test needs to be used. To do so, several vectors autoregressive were created for the validation period, considering p to be anything between 1 and 300. After doing so, the AIC, HQ, SC, FPE criteria were used. The optimal number of lags that resulted from them were 56, 15, 4 and 56 respectively. Therefore, the number of lags chosen was 56 since 2 different criterions suggested it.

Having the number of lags selected, it was possible to apply the augmented dickey-fuller test which states that the series is not stationary. This was expected since it is known the temperature affects the variables from the exploratory models and the temperature itself is not stationary. By differencing the series with 56 lags and test it once again for stationarity, the series is now stationary and therefore a series of order I(1) is enough.

After making sure the data in the calibration and validation period is stationary, the Johansen test was used to know how many cointegration vectors there were, and then to compute them. From this, 8 different cointegration vectors were calculated.

Afterwards, the data in the undamage test and damage period were also tested for stationary. By considering the data without integration, both series were not stationary. However, by differencing the series with 56 lags they started to present stationarity.

Since all the series present stationarity, they were all once again compiled into one. From these, the cointegration vectors computed for the calibration and validation period were applied to the whole series, generating 8 cointegration residuals. The first set of residuals is seen in **Figure 13**. By focusing on the calibration and validation period, an X-bar control chart was developed.

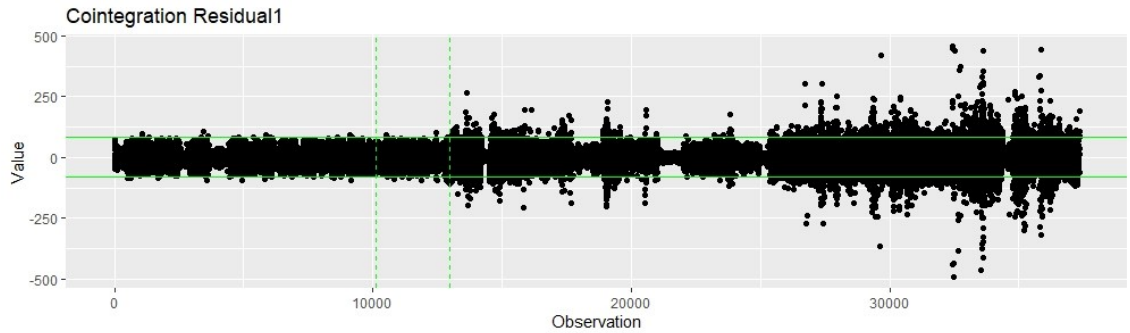

Figure 13: Cointegration residuals computed from the first cointegration vector. The horizontal lines relate to the upper and lower limits defined by the mean plus or minus 3 standard deviations, respectively, referenced to the calibration and validation period. The first vertical line divides the calibration and validation from the undamaged test period, while the second divides the undamaged test and damage period

Once again, both in the calibration and validation and the undamage test period, there are observations that fall outside the limits defined. To avoid this problem the method described in **Section 3.1** is adopted. This time, instead of testing for 2 hours of values outside of the control limits, it is possible to be more restrictive and consider only 45 minutes (see **Appendix 2.17.**). Doing so, allows this method to detect damage only in the damage period as seen **Figure 14**. Nonetheless, this system is only capable of flagging about 1% of the sets of observations in this period. Even so, this method is able detect damage 5 hours after the damage is introduced.

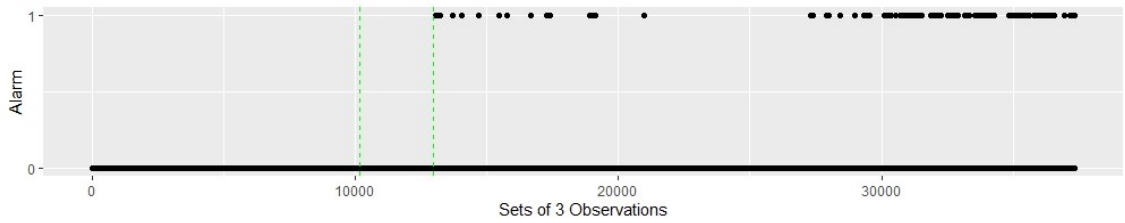

Figure 14: Graphical representation of which sets of 3 observations include only out of control observations

Conclusion

Keeping in mind the increase importance of a reliable SHM system nowadays, this work focused on the development of a system capable of detecting damage autonomously. The system should be robust enough to avoid false positives, while being sensitive enough to detect damage in a structure.

In the development of this system, a real-life partial experimental model of a bridge was used as case study. This structure was subjected to a period of free action and a period of damage. This allowed for very rich data capable to mirror what a real-life behavior would be, which in turn made the system that depend on the data more reliable.

The developed system took advantage of two different methods to test if it was possible to produce a model that could encompass the behavior of the structure. The first method relied on the use of machine learning algorithms (linear regression, random forest, support vector machine and neural network) to build explanatory models. While the second took advantage of time series analysis to draw the cointegration properties of the structure.

Finally, control charts were used to detect anomalies. By feeding the control charts data of a structure in a normal condition, they could detect atypical behaviors that signal damage. In the case of the explanatory models, the residuals that came from the difference between the predicted and observed values were used as input of Hotelling T^2 based control chart. Nevertheless, regarding time series analysis, the cointegration residuals were used in the development of a X-bar based control chart.

The creation of the control charts revealed that explanatory models have a greater damage detection capability than the results that come from time series analysis. While both these methods enabled the detection of damage only in the

damage period, their sensitivity difference is clear. Considering only the damage period, the cointegration residuals allowed for 1% of the observations to be correctly detected as damage, while the system developed around the explanatory models was capable of flagging 12% of the observations.

Although the explanatory model system is more reliable in damage detection, when it comes to how quickly a damage is detected the roles swap. Despite the explanatory model system detects a damage situation after 20 hours and 30 minutes of damage being induced in the structure, the cointegration approach signals damage only 5 hours after.

Even though there is a significant difference between the two systems developed, it should be considered that the one based on the cointegration properties of the series had some values interpolated, which could hinder its performance.

Bibliography

Barthorpe, R. J. (2010). *On Model and Data Based Approaches to Structural Health Monitoring*.

Chang, P. C., Flatau, A., & Liu, S. C. (2003). Review paper: Health monitoring of civil infrastructure. *Structural Health Monitoring*, *2*(3), 257–267. https://doi.org/10.1177/1475921703036169

Chatfield, C. (1975). *The Analysis of Time Series: Theory and Practice*.

Cross, E. J., Worden, K., & Chen, Q. (2011). Cointegration: A novel approach for the removal of environmental trends in structural health monitoring data. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *467*(2133), 2712–2732. https://doi.org/10.1098/rspa.2011.0023

Da Silva, M. F. M. (2017). *Machine learning algorithms for damage detection in structures under changing normal conditions*.

Dao, P. B. (2013). Cointegration method for temperature effect removal in damage detection based on lamb waves. *Diagnostyka*, *14*(3), 61–67.

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the Estimators for Autoregressive Time Series With a Unit Root. *Journal of the American Statistical Association*, *74*(366), 427. https://doi.org/10.2307/2286348

Dickey, D. A., & Fuller, W. A. (1981). *Likelihood Ratio Statistics for Autoregressive Time Series with a Unit Root*. *49*(4), 1057–1072. https://doi.org/10.1017/CBO9781107415324.004

Duan, Z., & Zhang, K. (2006). Data Mining Technology for Structural Health Monitoring. *Pacific Science Review*, *8*(1), 27–36.

Dunia, R., & Qin, S. J. (1998). Subspace Approach to Multidimensional Fault Identification and Reconstruction. *En*, *44*(8).

Farrar, C. R., Doebling, S. W., & Nix, D. A. (2001). Vibration-based structural

damage identification. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *359*(1778), 131–149. https://doi.org/10.1098/rsta.2000.0717

Farreras-Alcover, I., Chryssanthopoulos, M. K., & Andersen, J. E. (2015). Regression models for structural health monitoring of welded bridge joints based on temperature, traffic and strain measurements. *Structural Health Monitoring*, *14*(6), 648–662. https://doi.org/10.1177/1475921715609801

Gordan, M., Razak, H. A., Ismail, Z., & Ghaedi, K. (2017). Recent developments in damage identification of structures using data mining. *Latin American Journal of Solids and Structures*, *14*(13), 2373–2401. https://doi.org/10.1590/1679-78254378

Harris, R. I., & Harris, R. I. D. (1995). *Using Cointegration Analysis in Econometric Modelling*.

Johansen, S. (1988). Statistical analysis of cointegration vectors. *Journal of Economic Dynamics and Control*, *12*(2–3), 231–254. https://doi.org/10.1016/0165-1889(88)90041-3

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering, Transactions of the ASME*, *82*(1), 35–45. https://doi.org/10.1115/1.3662552

Kalman, R. E. (1963). Mathematical Description of Linear Dynamical Systems. *Journal of Society for Industrial and Applied Mathematic*. https://doi.org/10.1016/S0076-5392(08)63251-8

Kalman, R. E., & Bucy, R. S. (1961). New results in linear filtering and prediction theory. *Journal of Fluids Engineering, Transactions of the ASME*, *83*(1), 95–108. https://doi.org/10.1115/1.3658902

Kesavan, A., John, S., & Herszberg, I. (2008). Strain-based structural health monitoring of complex composite structures. *Structural Health Monitoring*, *7*(3), 203–213. https://doi.org/10.1177/1475921708090559

Kim, J. T., Ryu, Y. S., Cho, H. M., & Stubbs, N. (2003). Damage identification in beam-type structures: Frequency-based method vs mode-shape-based method. In *Engineering Structures* (Vol. 25). https://doi.org/10.1016/S0141-0296(02)00118-9

Kromanis, R., & Kripakaran, P. (2013). Advanced Engineering Informatics Support vector regression for anomaly detection from measurement histories. *Advanced Engineering Informatics*, *27*(4), 486–495. https://doi.org/10.1016/j.aei.2013.03.002

Liew, V. K. (2004). On Autoregressive Order Selection Criteria On Autoregressive Order Selection Criteria. *Computational Economics*, (March), 1–14. Retrieved from http://129.3.20.41/eps/comp/papers/0404/0404001.pdf

Liew, V. K. (2006). Which Lag Length Selection Criteria Should We Employ. *Economics Bulletin*, *3*(33), 1–9.

Neves, A. C., Gonz, I., & Leander, J. (2018). *Experimental Vibration Analysis for Civil Structures*. *5*(January), 72–84. https://doi.org/10.1007/978-3-319-67443-8

Omenzetter, P., & Brownjohn, J. M. W. (2006). Application of time series analysis for bridge monitoring. *Smart Materials and Structures*, *15*(1), 129–138. https://doi.org/10.1088/0964-1726/15/1/041

Pandey, G., Thostenson, E. T., & Heider, D. (2013). Electric time domain reflectometry sensors for non-invasive structural health monitoring of glass fiber composites. *Progress in Electromagnetics Research*, *137*(February), 551–564. https://doi.org/10.2528/PIER13020611

Park, G., & Inman, D. J. (2007). Structural health monitoring using piezoelectric impedance measurements. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *365*(1851), 373–392. https://doi.org/10.1098/rsta.2006.1934

Phares, B., Lu, P., Wipf, T., Greimann, L., & Seo, J. (2013). Field validation of a statistical-based bridge damage-detection algorithm. *Journal of Bridge*

Engineering, *18*(11), 1227–1238. https://doi.org/10.1061/(ASCE)BE.1943-5592.0000467

Posenato, D., Lanata, F., Inaudi, D., & Smith, I. F. C. (2008). Model-free data interpretation for continuous monitoring of complex structures. *Advanced Engineering Informatics*, *22*(1), 135–144. https://doi.org/10.1016/j.aei.2007.02.002

Reynders, E., Wursten, G., & de Roeck, G. (2014). Output-only structural health monitoring in changing environmental conditions by means of nonlinear system identification. *Structural Health Monitoring*, *13*(1), 82–93. https://doi.org/10.1177/1475921713502836

Rosales, M. J., & Liyanapathirana, R. (2017). Data driven innovations in structural health monitoring. *Journal of Physics: Conference Series*, *842*(1). https://doi.org/10.1088/1742-6596/842/1/012012

Santos-Fernández, E. (2012). *Multivariate Statistical Quality Control Using R*. https://doi.org/10.1007/978-81-322-0763-4

Sharma, S., & Sen, S. (2018). *Damage detection in presence of varying temperature through residual error modelling approach with dual neural network*. (5).

Slišković, D., Grbić, R., & Hocenski, Ž. (2012). *Multivariate Statistical Process Monitoring*.

Tibaduiza, D. A., Mujica, L. E., & Rodellar, J. (2011). Structural Health Monitoring based on principal component analysis: damage detection, localization and classification. *Advances in Dynamics, Control, Monitoring and Applications, Universitat Politècnica de Catalunya, Departament de Matemàtica Aplicada*, *3*(1), 8–17.

Tomé, E. S., Pimentel, M., & Figueiras, J. (2019). *SHM based damage detection using cointegration and linear multivariate data analysis: performance comparison based on a real case study*.

Tomé, E. S., Pimentel, M., & Figueiras, J. (2020). Damage detection under

environmental and operational effects using cointegration analysis – Application to experimental data from a cable-stayed bridge. *Mechanical Systems and Signal Processing*, *135*, 4–8. https://doi.org/10.1016/j.ymssp.2019.106386

Wipf, T. J., Phares, B., Doornink, J. D., Greimann, L., & Wood, D. L. (2007). *Evaluation of Steel Bridges , Volumes I & II*.

Worden, K., Cross, E., & Barton, E. (2012). Damage detection on the NPL footbridge under changing environmental conditions. *Proceedings of the 6th European Workshop - Structural Health Monitoring 2012, EWSHM 2012*, *2*, 1124–1131.

Yan, A. M., Kerschen, G., De Boe, P., & Golinval, J. C. (2005). Structural damage diagnosis under varying environmental conditions - Part I: A linear analysis. *Mechanical Systems and Signal Processing*, *19*(4), 847–864. https://doi.org/10.1016/j.ymssp.2004.12.002

Appendix

Appendix 1 – Studies summary

	Damage Location		Structure Undamaged State			Structure Damaged State		Variables			
Article	Global	Local	Physical	Virtual Simulation	Material	Physical	Virtual Simulation	Independent Variable(s)	Dependent Variables(s)	Model	Damage Detection Algorithm
Dunia 1998		x		process with reboilers and pre-heaters with control valves	x		x		flow	Principal Component Analysis	SPE and the T2 indices (control charts)
Farrar 2001	x		Columns		concrete	x			vibration	Auto-regressive estimation (linear predictive coding)	Fisher's discriminant
Kim 2003		x		beams	concrete and steel		x		strain	Finite element models	frequency-based damage detection (damage index)
									mode shapes		mode-shape-based damage detection (damage index)
Yan 2005	x			bridge	concrete		x		vibration	Principal Component Analysis	Mahalanobis distance
			wooden bridge		wood	x					

	Damage Location		Structure Undamaged State			Structure Damaged State		Variables			
Article	Global	Local	Physical	Virtual Simulation	Material	Physical	Virtual Simulation	Independent Variable(s)	Dependent Variables(s)	Model	Damage Detection Algorithm
Omenzetter 2006	x		Singapore-Malaysia Second Link		concrete and reinforced steel	x			strain	ARIMA (moving coefficients) with extended Kalman filter	Analysis of level shifts in coefficient values
Posenato 2008		x		beam	concrete		x		displacement	Moving Principal Component Analysis	eigenvector analysis
										Moving correlation analysis	correlation analysis
										Instance-based method	distance-from-training-set analysis
										Short-term Fourier transform	frequency and modulus analysis
										Continuous wavelet transform	coefficient analysis
Wipf 2007		x	US 30 bridge		steel, concrete and reinforced concrete		x		strain	Extrema event analysis (relationship between target and non-target sensors)	Analysis of % of values above or below manually set control limits
Park 2007		x	walls		reinforced concrete	x		frequency	impedance	Modified frequency-domain autoregressive model with exogenous inputs (ARX)	analysis of: outlier damage metric based on extreme value statistics and root mean square deviation damage metric
Kesavan 2008		x	T-joint specimens		glass fiber	x			strain	Global Neural network Algorithm for Sequential Processing of Internal sub Networks (GNAISPIN), which uses multiple ANNs in conjunction with a modified DRAT (Damage Relativity Analysis Technique)	delamination comparisson

	Damage Location		Structure Undamaged State			Structure Damaged State		Variables			
Article	Global	Local	Physical	Virtual Simulation	Material	Physical	Virtual Simulation	Independent Variable(s)	Dependent Variables(s)	Model	Damage Detection Algorithm
Barthorpe 2010	x		aluminium aicraft wing		aluminium baseplate	x			vibration	Mahalanobis squared distance (MSD)	ROC curve and AUC value
										Support vector machines	percentage of correctly classified, incorrectly classified and unclassified
Tibaduiza 2011		x	Aircraft turbine blade		similar to titanium	x			vibration time based signals and piezoelectric transducers	Principal Component Analysis	Self Organizing Maps (SOM) and the following indexes: Q-index (or SPE-index), the Hotelling's T2-statistic (D - statistic), Phi and I2
			Aluminium plate		smooth-raw aluminium	x					
Cross 2011	x		composite panel		carbon fibre-reinforced plastic	x			Lamb-wave signals	Principal Component Analysis	Multivariate outlier analyses of novelty index based on mean, covariance matrix and feature samples
Worden 2012	x		footbridge		concrete	x			structure local slope	Cointegration	X-bar chart (control charts)
Slišković 2012		x		liquid storage	x		x		input flow, flow and output flow	Principal Component Analysis and ICA	Hotelling's (T2), I2 and Q (SPE) (control charts)
Kromanis 2013	x			girder found in highway bridges	reinforced concrete		x	temperature	Strain	Support vector regression	moving fast Fourier transform (MFFT)

	Damage Location		Structure Undamaged State			Structure Damaged State		Variables			
Article	Global	Local	Physical	Virtual Simulation	Material	Physical	Virtual Simulation	Independent Variable(s)	Dependent Variables(s)	Model	Damage Detection Algorithm
Pandey 2013	x		ASTM D3039 tensile specimens		E-glass fibers and resin	x		dielectric constant and impedance	strain	Equation based on material dielectric and magnetic properties, specimen geometry and transmission line width Dielectrostriction phenomenon linear relationship Linear dependence of the dielectric constant on strain	Impedance change measurements Analysis
Phares 2013		x	specimens that replicate the US 30 bridge		steel, concrete and reinforced concrete	x		strain range of a sensor	strain range of another sensor	Linear regression	X-bar chart (control charts)
Dao 2013	x		plates		aluminium	x			Lamb-wave signals	Cointegration	analysis of the adf t-statistic value of variables and cointegration vectors
Reynders 2014	x		Z24 bridge		concrete	x			vibration	Linear and kernel Principal Component Analysis	visual analysis of the prediction error (not the focus of this work)
Alcover 2015	x		Great Belt Bridge (Denmark)		steel	(doesn't have)		daily-averaged pavement temperatures	daily-aggregated heavy traffic counts) and a strain-based performance indicator	Weighted least squares regression	Analysis of stress range residuals (control chart)

	Damage Location		Structure Undamaged State			Structure Damaged State		Variables			
Article	Global	Local	Physical	Virtual Simulation	Material	Physical	Virtual Simulation	Independent Variable(s)	Dependent Variables(s)	Model	Damage Detection Algorithm
Rosales 2015		x	Small three story building representation		aluminium baseplate	x		force from the shaker	vibration	Autoregression (AR) model; Autoregressive with Exogeneous (ARX) Inputs	Mahalanobis distance
Silva 2017	x		z-24 bridge		concrete	x			temperature, vibration	Linear Principal Componetne Analysis, Auto-Associative Neural Network, Mahalanobis squared distance, Gaussian mixture models	Number and percentage of Type I/II errors
	x		Tamar Bridge		concrete and steel	x			tensions on stays, vibration, wind velocity, temperature, deflection and tilt		
Neves 2018	x			single-track railway bridge	concrete and steel		x	train speed	vibration and axle loads	Artificial Neural Networks	Receiver Operating Characteristic curves
Tomé 2019		x	Corgo Bridge		concrete		x	concrete temperatures	stay-cable forces	Cointegration	ratio between the mean values of the T2 statistic in the damaged and undamaged states; ratio between the mean values of the T2 statistic in the damaged state and the UCL
										Multilinear Regression and Principal Component Analsysis (MLR-PCA)	
Tomé 2020		x	Corgo Bridge		concrete		x		vibration	Cointegration	ratio between the mean values of the T2 statistic in the damaged and undamaged states; ratio between the mean values of the T2 statistic in the damaged state and the UCL

Table 3: SHM related studies summary

Appendix 2 – Code

Note: *DataUD* refers to the undamaged data. This data has a total of 17 columns. The first has the time when the observation was recorded while the other 16 are in a numeric form.

2.1. Outlier detection

To do this in R, the data had to be first prepared for it. First, two variables were created. The first was an id variable, while the second was a binary one to determine if the observation at some point was detected as outlier and therefore not to be considered in the next difference. After, some objects were created to be later used.

When calculating the time difference between two observations in R, there needs to be extra attention. While a 15-minute difference is considered 15, an hour difference is 1 instead of 60. Also, if the time between two observations is one day, the result is 1 instead of 24. In order to work around this problem, three different loops were constructed in a way to be missing value proof. The first loop is designed to calculate the differences between 15, 30 and 45-minutes, the mean and standard deviation of the respective differences and finally compare the differences computed to the mean plus three standard deviations. The second loop is the same but designed for differences of 1 hour until differences of 23 hours and 45 minutes. Lastly, the third loop is used exclusively for differences of 24 hours.

When it comes to naming variables, the computed differences were stored and named B if they were being compared with the previous observations and named A if they were taking future observations into account. Also, in naming, all the time differences were considered in minutes.

If at any time, an observation had both differences above the mean plus three standard deviations, the variable *look* was set to 0. At the end of each loop, it is counted and stored in object *Difs* how many observations were removed so that the number outliers found in each time difference is known. In order to be missing value proof, the time difference between observations were computed and tested every loop to verify the right differences were being calculated.

Prepare data for the data cleaning process

```
DataUD$id <- 1:nrow(DataUD)
DataUD$id[1:9] <- paste("0000",DataUD$id[1:9],sep="")
DataUD$id[10:99] <- paste("000",DataUD$id[10:99],sep="")
DataUD$id[100:999] <- paste("00",DataUD$id[100:999],sep="")
DataUD$id[1000:9999] <- paste("0",DataUD$id[1000:9999],sep="")
DataUD$look <- 1 #if 1 the observation is not an outlier and should be
considered in the computation of the Mean + 3*SD.

shift <- c(0)
MeanSDB <- list()
MeanSDA <- list()
for (i in 1:96){
  MeanSDB[[i]] <- c(0,0)
  MeanSDA[[i]] <- c(0,0)
}
Difs <- c(0,0)
```

Compare differences between 15, 30 and 45 minutes

```
for (t in 1:3){

#create names for the differences: B for before and A for after
  for (i in c(2:17)) { #c(2:17) refers to the variables in the dataset
    name=paste(colnames(DataUD)[round(i)],"Dif",t*15,"B",sep="")
    DataUD$name <- 0
    colnames(DataUD)[grep("name",colnames(DataUD))] <- name
  }
  name=paste("Time",t*15,"B",sep="")
  DataUD$name <- 0
  colnames(DataUD)[grep("name",colnames(DataUD))] <- name
  for (i in c(2:17)) {
    name=paste(colnames(DataUD)[round(i)],"Dif",t*15,"A",sep="")
    DataUD$name <- 0
    colnames(DataUD)[grep("name",colnames(DataUD))] <- name
  }
  name=paste("Time",t*15,"A",sep="")
  DataUD$name <- 0
  colnames(DataUD)[grep("name",colnames(DataUD))] <- name

  #Compute the differences
  for (i in 1:(nrow(DataUD)-t)) { #loop to compute the time difference between
observations
    DataUD[i+t,(ncol(DataUD)-17)] <- DataUD$Time[i+t] - DataUD$Time[i]
  }
```

```
 for (k in 2:17){ #loop to compute the absolute differences for the observations
before
   DataUD[,(k-49+67*t)] <- c(shift, abs(DataUD[-c(1:t),k] - DataUD[-
c((nrow(DataUD)-t+1):nrow(DataUD)),k] ) )
 }
 for (i in (-47+67*t):(-31+67*t)){ #loop to compute the absolute differences for the
observations after
   DataUD[,i+17] <- c(DataUD[-c(1:t),i],shift)
 }

 #create variables to know if the differences are above or below the Means +
3*SD (binary)
 names <-
c("TempInfF2","TempIntF1","TempIntF2","IncInf","IncInt","IncSup","LVDTInt","
LVDTSup","SF1Inf","SF2Inf","SF1Int","SF2Int","TempSupF1","TempSupF2","Tem
pSupF2Despro","TempAmb")
 for (i in 1:16) {
   DataUD$name <- 0
   colnames(DataUD)[grep("name",colnames(DataUD))] <-
paste("Dif",t*15,"B",names[i],sep="")
 }
 for (i in 1:16) {
   DataUD$name <- 0
   colnames(DataUD)[grep("name",colnames(DataUD))] <-
paste("Dif",t*15,"A",names[i],sep="")
 }
```

```
  #Compute Mean + 3*SD and classify the differences as 1 is above the Mean +
3*SD
  for (i in 2:17){ #differences Before
    MeanSDB[[t]] <- rbind(MeanSDB[[t]],c((mean(DataUD[DataUD$look==1 &
DataUD[,(-31+67*t)]==(t*15),(i-49+67*t)])+3*sd(DataUD[DataUD$look==1 &
DataUD[,(-31+67*t)]==(t*15),(i-49+67*t)])),(mean(DataUD[DataUD$look==1 &
DataUD[,(-31+67*t)]==(t*15),(i-49+67*t)])-3*sd(DataUD[DataUD$look==1 &
DataUD[,(-31+67*t)]==(t*15),(i-49+67*t)])))
    look.for <- c((DataUD[DataUD$look==1 & DataUD[,(-31+67*t)]==(t*15) &
DataUD[,(i-49+67*t)]>MeanSDB[[t]][i,1],(i-49+67*t)]),(DataUD[DataUD$look==1
& DataUD[,(-31+67*t)]==(t*15) & DataUD[,(i-49+67*t)]<MeanSDB[[t]][i,2],(i-
49+67*t)]))
    DataUD[DataUD[,(i-49+67*t)] %in% look.for,(i-15+67*t)] <- 1
  }

  for (i in 2:17){ #differences After
    MeanSDA[[t]] <- rbind(MeanSDA[[t]],c((mean(DataUD[DataUD$look==1 &
DataUD[,(-14+67*t)]==(t*15),(i-32+67*t)])+3*sd(DataUD[DataUD$look==1 &
DataUD[,(-14+67*t)]==(t*15),(i-32+67*t)])),(mean(DataUD[DataUD$look==1 &
DataUD[,(-14+67*t)]==(t*15),(i-32+67*t)])-3*sd(DataUD[DataUD$look==1 &
DataUD[,(-14+67*t)]==(t*15),(i-32+67*t)])))
    look.for <- c((DataUD[DataUD$look==1 & DataUD[,(-14+67*t)]==(t*15) &
DataUD[,(i-32+67*t)]>MeanSDA[[t]][i,1],(i-32+67*t)]),(DataUD[DataUD$look==1
& DataUD[,(-14+67*t)]==(t*15) & DataUD[,(i-32+67*t)]<MeanSDA[[t]][i,2],(i-
32+67*t)]))
    DataUD[DataUD[,(i-32+67*t)] %in% look.for,(i+1+67*t)] <- 1
  }
```

```
  #Create a variable to classify an observation if both differences (Before and
After) are above the Mean + 3*SD. Variable is set to 1 if both differences are
abnormal and “look” is set to 0
  name=paste("Dif",t*15,sep="")
  DataUD$name <- 0
  colnames(DataUD)[grep("name",colnames(DataUD))] <- name
  for (i in 1:nrow(DataUD)){
    for (k in 2:17){
      if(sum(DataUD[i,(k-15+67*t)])>0 & sum(DataUD[i,(k+1+67*t)])>0){
        DataUD$look[i] <- 0
        DataUD[i,ncol(DataUD)] <- 1
      }
    }
  }


  Difs <- rbind(Difs,c(sum(DataUD[,ncol(DataUD)]),sum(DataUD$look))) #Table
that records how many observations are left out for 15, 30 and 45 minute
differences
  shift <- rbind(shift,c(0))
  print(t)
}

colnames(Difs) <- c("Observations Removed","Observations considered when
computing MeanSD")
```

Compare differences between 1 hour and 23h45

The code is the same as above. The only differences are:

Differences between 15, 30 and 45 minutes	Differences between 1 hour and 23h45
for (t in 1:3){	for (t in 4:95){
==(t*15)	==(t/4)

Table 4: Code difference between the loop designed for 15, 30 and 45 minutes differences and the loop designed for differences between 1 hour and 23h45

Compare differences for 24 hours

The code is the same as the first. The only differences are:

Differences between 15, 30 and 45 minutes	Differences between 24 hours
for (t in 1:3){	for (t in 96){
==(t*15)	==round(t*15/1440,4)

Table 5: Code difference between the loop designed for 15, 30 and 45 minutes differences and the loop designed for differences between 24 hours

2.2. Remove Outliers and correct shift in damage period data

```
DataUD <- DataUD[DataUD$Dif15==0 & DataUD$Dif30==0 &
DataUD$Dif45==0 & DataUD$Dif60==0 & DataUD$Dif75==0 &
DataUD$Dif90==0 & DataUD$Dif105==0 & DataUD$Dif120==0 &
DataUD$Dif135==0 & DataUD$Dif150==0 & DataUD$Dif165==0 &
DataUD$Dif180==0 & DataUD$Dif195==0 & DataUD$Dif210==0 &
DataUD$Dif225==0 & DataUD$Dif240==0 & DataUD$Dif255==0 &
DataUD$Dif270==0 & DataUD$Dif285==0 & DataUD$Dif300==0 &
```

```
DataUD$Dif315==0 & DataUD$Dif330==0 & DataUD$Dif345==0 &
DataUD$Dif360==0 & DataUD$Dif375==0 & DataUD$Dif390==0 &
DataUD$Dif405==0 & DataUD$Dif420==0 & DataUD$Dif435==0 &
DataUD$Dif450==0 & DataUD$Dif465==0 & DataUD$Dif480==0 &
DataUD$Dif495==0 & DataUD$Dif510==0 & DataUD$Dif525==0 &
DataUD$Dif540==0 & DataUD$Dif555==0 & DataUD$Dif570==0 &
DataUD$Dif585==0 & DataUD$Dif600==0 & DataUD$Dif615==0 &
DataUD$Dif630==0 & DataUD$Dif645==0 & DataUD$Dif660==0 &
DataUD$Dif675==0 & DataUD$Dif690==0 & DataUD$Dif705==0 &
DataUD$Dif720==0 & DataUD$Dif735==0 & DataUD$Dif750==0 &
DataUD$Dif765==0 & DataUD$Dif780==0 & DataUD$Dif795==0 &
DataUD$Dif810==0 & DataUD$Dif825==0 & DataUD$Dif840==0 &
DataUD$Dif855==0 & DataUD$Dif870==0 & DataUD$Dif885==0 &
DataUD$Dif900==0 & DataUD$Dif915==0 & DataUD$Dif930==0 &
DataUD$Dif945==0 & DataUD$Dif960==0 & DataUD$Dif975==0 &
DataUD$Dif990==0 & DataUD$Dif1005==0 & DataUD$Dif1020==0 &
DataUD$Dif1035==0 & DataUD$Dif1050==0 & DataUD$Dif1065==0 &
DataUD$Dif1080==0 & DataUD$Dif1095==0 & DataUD$Dif1110==0 &
DataUD$Dif1125==0 & DataUD$Dif1140==0 & DataUD$Dif1155==0 &
DataUD$Dif1170==0 & DataUD$Dif1185==0 & DataUD$Dif1200==0 &
DataUD$Dif1215==0 & DataUD$Dif1230==0 & DataUD$Dif1245==0 &
DataUD$Dif1260==0 & DataUD$Dif1275==0 & DataUD$Dif1290==0 &
DataUD$Dif1305==0 & DataUD$Dif1320==0 & DataUD$Dif1335==0 &
DataUD$Dif1350==0 & DataUD$Dif1365==0 & DataUD$Dif1380==0 &
DataUD$Dif1395==0 & DataUD$Dif1410==0 & DataUD$Dif1425==0 &
DataUD$Dif1440==0,1:17]
rownames(DataUD) <- NULL
```

```
#Remove shift in damage data
for (i in c(5:13)) {
  DataD[,i] <- DataD[,i] + (mean(DataUD[(nrow(DataUD)-672):nrow(DataUD),i])
- mean(DataD[,i][1:672]))
}
```

2.3. Fold creation (Calibration and Validation data)

```
DataUD$id <- 1:nrow(DataUD)
DataUD$id[1:9] <- paste("0000",DataUD$id[1:9],sep="")
DataUD$id[10:99] <- paste("000",DataUD$id[10:99],sep="")
DataUD$id[100:999] <- paste("00",DataUD$id[100:999],sep="")
DataUD$id[1000:7159] <- paste("0",DataUD$id[1000:7159],sep="")

DataD$id <- 1:nrow(DataD)
DataD$id[1:9] <- paste("0000",DataD$id[1:9],sep="")
DataD$id[10:99] <- paste("000",DataD$id[10:99],sep="")
DataD$id[100:999] <- paste("00",DataD$id[100:999],sep="")
DataD$id[1000:9999] <- paste("0",DataD$id[1000:9999],sep="")

DataUDT <- DataUD[(nrow(DataUD)-2879):nrow(DataUD),]
DataNew <- rbind(DataUDT,DataD)
DataCalVal <- DataUD[1:(nrow(DataUD)-2880),]

library(caret)
set.seed(13)
```

```
folds <- createFolds(DataCalVal$id, k=2)
DataCal <- DataCalVal[folds[[1]],]
DataVal <- DataCalVal[folds[[2]],]

set.seed(14)
folds_train <- createFolds(DataCal$id, k=10)
set.seed(15)
folds_test <- createFolds(DataVal$id, k=10)
```

2.4. Linear regression for calibration period data

The code below refers solely to one of the dependent variables (*Strain_F1_Middle*). This should be done as many times as dependent variables there are.

```
library(Metrics)

feats <- colnames(DataUD[c(2:4,14:17)]) # c(2:4,14:17) refers to the independent variables
f <- paste(feats,collapse=' + ')
f <- paste('Strain_F1_Middle ~',f) #"Strain_F1_Middle" refers to the name of the dependent variable
f <- as.formula(f) #Convert to formula

#Train
results_reg_sf1int <- data.frame()
predictions_summary <- data.frame()
fitsummary <- list()
```

```
for (k in 1:10){ #loop to compute the parameters and save the predictions
  fit <- lm(f, data=DataCal[-folds_train[[k]],])
  fitsummary[[k]] <- summary(fit)
  prediction <- predict(fit, newdata=DataCal[folds_train[[k]],])
  predictions_summary <- rbind(predictions_summary,
data.frame(Time=DataCal[folds_train[[k]],'Time'],
y=DataCal$Strain_F_Middle[folds_train[[k]]], prediction))
}

#Save metrics
R2 <- cor(predictions_summary$y, predictions_summary$prediction)
RMSE <- rmse(predicted = predictions_summary$prediction, actual =
predictions_summary$y)
MAE <- mae(predicted = predictions_summary$prediction, actual =
predictions_summary$y)
MAPE <- mape(predicted = predictions_summary$prediction, actual =
predictions_summary$y)
results_reg_sf1int=rbind(results_reg_sf1int,data.frame(R2, RMSE, MAE,
MAPE))
RegTrain_sf1int <- results_reg_sf1int
RegTrain_sf1int

for (i in 1:10){print(fitsummary[[i]])} #the variables that, on average, have a p-
value above 0.01 should be removed by changing the feats object at the start,
until all variables are significative
```

2.5. Linear regression for validation, undamaged test and damage period data

The code below refers solely to one of the dependent variables (*Strain_F1_Middle*). This should be done as many times as dependent variables there are.

```
feats <- colnames(DataUD[c(2:4,14,17)])
f <- paste(feats,collapse=' + ')
f <- paste('Strain_face1_intermedio ~',f)
f <- as.formula(f) #Convert to formula

predictions_summaryregsf1int <- data.frame()
results_reg_sf1int <- data.frame()

for (k in 1:10){
  fit <- lm(f, data=DataVal[-folds_test[[k]],])
  fitsummary[[k]] <- summary(fit)
  prediction <- predict(fit, newdata=DataVal[folds_test[[k]],])
  predictions_summaryregsf1int <- rbind(predictions_summaryregsf1int,
data.frame(Time=DataVal[folds_test[[k]],'Time'],
y=DataVal$Strain_face1_intermedio[folds_test[[k]]], pred=prediction))
}

#Save metrics for testing data
R2 <- cor(predictions_summaryregsf1int$y,
predictions_summaryregsf1int$pred)
```

```
RMSE <- rmse(predicted = predictions_summaryregsf1int$pred, actual =
predictions_summaryregsf1int$y)
MAE <- mae(predicted = predictions_summaryregsf1int$pred, actual =
predictions_summaryregsf1int$y)
MAPE <- mape(predicted = predictions_summaryregsf1int$pred, actual =
predictions_summaryregsf1int$y)
results_reg_sf1int=rbind(results_reg_sf1int,data.frame(R2, RMSE, MAE,
MAPE))
results_reg_sf1int

fit <- lm(f, data=DataVal)
predictionsf1int <- predict(fit, newdata=DataNew)
predictions_summaryregsf1int$dif <- predictions_summaryregsf1int$y -
predictions_summaryregsf1int$pred
predictions_summaryregsf1int <-
rbind(predictions_summaryregsf1int,data.frame(Time=DataNew$Time,y=Data
New$Strain_face1_intermedio,pred=predictionsf1int,dif=DataNew$Strain_face1
_intermedio-predictionsf1int))

#Undamage Test period
R2 <- cor(predictions_summaryregsf1int$y[3221:6100],
predictions_summaryregsf1int$pred[3221:6100])
RMSE <- rmse(predicted = predictions_summaryregsf1int$pred[3221:6100],
actual = predictions_summaryregsf1int$y[3221:6100])
MAE <- mae(predicted = predictions_summaryregsf1int$pred[3221:6100], actual
= predictions_summaryregsf1int$y[3221:6100])
MAPE <- mape(predicted = predictions_summaryregsf1int$pred[3221:6100],
actual = predictions_summaryregsf1int$y[3221:6100])
```

```
results=rbind(results,data.frame(R2, RMSE, MAE, MAPE))

#Damage period
R2 <- cor(predictions_summaryregsf1int$y[6101:28354],
predictions_summaryregsf1int$pred[6101:28354])
RMSE <- rmse(predicted = predictions_summaryregsf1int$pred[6101:28354],
actual = predictions_summaryregsf1int$y[6101:28354])
MAE <- mae(predicted = predictions_summaryregsf1int$pred[6101:28354],
actual = predictions_summaryregsf1int$y[6101:28354])
MAPE <- mape(predicted = predictions_summaryregsf1int$pred[6101:28354],
actual = predictions_summaryregsf1int$y[6101:28354])

RegTest_sf1int=rbind(results_reg_sf1int,data.frame(R2, RMSE, MAE, MAPE))
rownames(RegTest _sf1int) <- c("Validation Data","Undamage Test
Data","Damage Data")

#plot
library(ggplot2)
gridExtra::grid.arrange(ggplot(predictions_summaryregsf1int) +
geom_point(aes(Time,y)) +
geom_point(data=predictions_summaryregsf1int[1:3220,], aes(Time, pred),
colour = "blue") + geom_point(data=predictions_summaryregsf1int[3221:6100,],
aes(Time, pred), colour = "green") +
geom_point(data=predictions_summaryregsf1int[6101:28354,], aes(Time, pred),
colour = "red") + geom_vline(xintercept =
predictions_summaryregsf1int$Time[3220], colour = "green") +
geom_vline(xintercept = predictions_summaryregsf1int$Time[6101], colour =
"red") + labs(title="Linear Regression Model (Strain F1 Middle)") +
```

```
ylab("Strain"), ggplot(predictions_summaryregsf1int) +
geom_point(aes(Time,dif)) + ylab("Residuals"),  nrow = 2)
```

2.6. Random Forest for calibration data

The code below refers solely to one of the dependent variables (*Strain_F1_Middle*). This should be done as many times as dependent variables there are.

```
library(Metrics)
library(randomForest)

feats <- colnames(Dados[c(2:4,14:17)])
f <- paste(feats,collapse=' + ')
f <- paste('Strain_face1_intermedio ~',f)
f <- as.formula(f) #Convert to formula

results_forest<-data.frame()

for(i in c(2:7)){ #loop to compute the parameters and save the predictions
considering groups of 2 to 7 variables
  predictions_summarysf1int<-data.frame()

  for (k in 1:10){
    fit <- randomForest(f, data=DataCal[-folds_train[[k]],], mtry=i, ntree=1000)
    prediction <- predict(fit, newdata=DataCal[folds_train[[k]],])
```

```
   predictions_summarysf1int <- rbind(predictions_summarysf1int,
data.frame(Time=DataCal[folds_train[[k]],'Time'],
y=DataCal$Strain_face1_intermedio[folds_train[[k]]], prediction))
  }

  variables=i

  #Save metrics
  R2 <- cor(predictions_summarysf1int$y,
predictions_summarysf1int$prediction)
  RMSE <- rmse(predicted = predictions_summarysf1int$prediction, actual =
predictions_summarysf1int$y)
  MAE <- mae(predicted = predictions_summarysf1int$prediction, actual =
predictions_summarysf1int$y)
  MAPE <- mape(predicted = predictions_summarysf1int$prediction, actual =
predictions_summarysf1int$y)
  results_forest=rbind(results_forest,data.frame(R2, RMSE, MAE, MAPE,
variables))
}
ForestTrainsf1int <- results_forest
```

2.7. Random Forest for testing, validation and damaged data

The code below refers solely to one of the dependent variables (*Strain_F1_Middle*). This should be done as many times as dependent variables there are.

```
feats <- colnames(Dados[c(2:4,14:17)])
f <- paste(feats,collapse=' + ')
f <- paste('Strain_face1_intermedio ~',f)
f <- as.formula(f) #Convert to formula

variables=ForestTrainsf1int[which.max(ForestTrainsf1int$R2),5] #tendo em
conta os valores obtidos

predictions_summaryrfsf1int <- data.frame()
results_forest <- data.frame()

for (k in 1:10){ #loop to compute the parameters and save the predictions
considering the optimal number of variables
  fit <- randomForest(f, data=DataVal[-folds_test[[k]],],mtry=variables,
ntree=1000)
  fit$importance
  prediction <- predict(fit, newdata=DataVal[folds_test[[k]],])
  predictions_summaryrfsf1int <- rbind(predictions_summaryrfsf1int,
data.frame(Time=DataVal[folds_test[[k]],'Time'],
y=DataVal$Strain_face1_intermedio[folds_test[[k]]], pred=prediction))
}

#Save metrics for testing data
R2 <- cor(predictions_summaryrfsf1int$y, predictions_summaryrfsf1int$pred)
RMSE <- rmse(predicted = predictions_summaryrfsf1int$pred, actual =
predictions_summaryrfsf1int$y)
```

```
MAE <- mae(predicted = predictions_summaryrfsf1int$pred, actual =
predictions_summaryrfsf1int$y)
MAPE <- mape(predicted = predictions_summaryrfsf1int$pred, actual =
predictions_summaryrfsf1int$y)
results_forest=rbind(results_forest,data.frame(R2, RMSE, MAE, MAPE,
variables))
results_forest

#Save metrics
fit <- randomForest(f, data=DataVal,mtry=variables, ntree=1000)
predictionsf1int <- predict(fit, newdata=DataNew)
predictions_summaryrfsf1int$dif <- predictions_summaryrfsf1int$y -
predictions_summaryrfsf1int$pred
predictions_summaryrfsf1int <-
rbind(predictions_summaryrfsf1int,data.frame(Time=DataNew$Time,y=DataN
ew$Strain_face1_intermedio,pred=predictionsf1int,dif=DataNew$Strain_face1_i
ntermedio-predictionsf1int))

#Undamage Test period
R2 <- cor(predictions_summaryrfsf1int$y[3221:6100],
predictions_summaryrfsf1int$pred[3221:6100])
RMSE <- rmse(predicted = predictions_summaryrfsf1int$pred[3221:6100],
actual = predictions_summaryrfsf1int$y[3221:6100])
MAE <- mae(predicted = predictions_summaryrfsf1int$pred[3221:6100], actual
= predictions_summaryrfsf1int$y[3221:6100])
MAPE <- mape(predicted = predictions_summaryrfsf1int$pred[3221:6100],
actual = predictions_summaryrfsf1int$y[3221:6100])
results=rbind(results,data.frame(R2, RMSE, MAE, MAPE, variables))
```

```
#Damage period
R2 <- cor(predictions_summaryrfsf1int$y[6101:28354],
predictions_summaryrfsf1int$pred[6101:28354])
RMSE <- rmse(predicted = predictions_summaryrfsf1int$pred[6101:28354],
actual = predictions_summaryrfsf1int$y[6101:28354])
MAE <- mae(predicted = predictions_summaryrfsf1int$pred[6101:28354], actual
= predictions_summaryrfsf1int$y[6101:28354])
MAPE <- mape(predicted = predictions_summaryrfsf1int$pred[6101:28354],
actual = predictions_summaryrfsf1int$y[6101:28354])

RFTest_sf1int=rbind(results_rf_sf1int,data.frame(R2, RMSE, MAE, MAPE,
variables))
rownames(RFTest _sf1int) <- c("Data Teste","Data Validation","Data Damaged")

#plot
library(ggplot2)
gridExtra::grid.arrange(ggplot(predictions_summaryrfsf1int) +
geom_point(aes(Time,y)) +
geom_point(data=predictions_summaryrfsf1int[1:3220,], aes(Time, pred), colour
= "blue") + geom_point(data=predictions_summaryrfsf1int[3221:6100,],
aes(Time, pred), colour = "green") +
geom_point(data=predictions_summaryrfsf1int[6101:28354,], aes(Time, pred),
colour = "red") + geom_vline(xintercept =
predictions_summaryrfsf1int$Time[3220], colour = "green") +
geom_vline(xintercept = predictions_summaryrfsf1int$Time[6101], colour =
"red") + labs(title="Random Forest Model (Strain F1 Middle)") + ylab("Strain"),
```

```
ggplot(predictions_summaryrfsf1int) + geom_point(aes(Time,dif)) +
ylab("Residuals"),  nrow = 2)
```

2.8. Support Vector Machine for testing, validation and damaged data

The code below refers solely to one of the dependent variables (*Strain_F1_Middle*). This should be done as many times as dependent variables there are.

```
library(e1071)
library(Metrics)
library(randomForest)

feats <- colnames(Dados[c(2:4,14:17)])
f <- paste(feats,collapse=' + ')
f <- paste('Strain_face1_intermedio ~',f)
f <- as.formula(f) #Convert to formula

performance_metrics <- data.frame()
results_train_SVM_g=data.frame()

for (cost in 2^(seq(-5,15, by=2))){ #loop to compute the parameters and save the predictions with varying cost and gamma

  for (gamma in 2^(seq(-15,3, by=2))){
    predictions_summary <- data.frame()
```

```
   for (k in 1:10){
     svm.model <- svm(f, DataCal[-folds_train[[k]],], kernel="linear", type="eps-
regression", scale=TRUE, cost=cost, gamma=gamma)
     prediction <- predict(svm.model, newdata=DataCal[folds_train[[k]], ])
     predictions_summary<-rbind(predictions_summary,
data.frame(Time=DataCal[folds_train[[k]],'Time'],
y=DataCal$Strain_face1_intermedio[folds_train[[k]]], prediction))
   }

   R2 <- cor(predictions_summary$y, predictions_summary$prediction)
   RMSE <- rmse(predicted = predictions_summary$prediction, actual =
predictions_summary$y)
   MAE <- mae(predicted = predictions_summary$prediction, actual =
predictions_summary$y)
   MAPE <- mape(predicted = predictions_summary$prediction, actual =
predictions_summary$y)

   cost=cost
   gamma=gamma

   results_train_SVM_g=rbind(results_train_SVM_g,data.frame(R2, RMSE,
MAE, MAPE, cost, gamma))


  }
}
```

```
#Save metrics
results_train_SVM_g
SVMTrainsf1int <- results_train_SVM_g
unique(SVMTrainsf1int$cost)
unique(SVMTrainsf1int$R2)
```

2.9. Support Vector Machine for training data

The code below refers solely to one of the dependent variables (*Strain_F1_Middle*). This should be done as many times as dependent variables there are.

```
feats <- colnames(Dados[c(2:4,14:17)])
f <- paste(feats,collapse=' + ')
f <- paste('Strain_face1_intermedio ~',f)
f <- as.formula(f) #Convert to formula

results=data.frame()
predictions_summarysvmsf1int <- data.frame()
performance_metrics <- data.frame()
cost=SVMTrainsf1int[which.max(SVMTrainsf1int$R2),5]

for (k in 1:10){ #loop to compute the parameters and save the predictions
considering the optimal cost and gamma
  svm.model <- svm(f, DataVal[-folds_test[[k]],], kernel="linear", type="eps-
regression", scale=TRUE, cost=cost, gamma=gamma)
  prediction <- predict(svm.model, newdata=DataVal[folds_test[[k]], ])
```

```
  predictions_summarysvmsf1int<-rbind(predictions_summarysvmsf1int,
data.frame(Time=DataVal[folds_test[[k]],'Time'],
y=DataVal$Strain_face1_intermedio[folds_test[[k]]], pred=prediction))
}

R2 <- cor(predictions_summarysvmsf1int$y,
predictions_summarysvmsf1int$pred)
RMSE <- rmse(predicted = predictions_summarysvmsf1int$pred, actual =
predictions_summarysvmsf1int$y)
MAE <- mae(predicted = predictions_summarysvmsf1int$pred, actual =
predictions_summarysvmsf1int$y)
MAPE <- mape(predicted = predictions_summarysvmsf1int$pred, actual =
predictions_summarysvmsf1int$y)

results=rbind(results,data.frame(R2, RMSE, MAE, MAPE, cost, gamma))
results

fit <- svm(f, DataVal, kernel="linear", type="eps-regression", scale=TRUE,
cost=cost, gamma=gamma)
predictionsf1int <- predict(fit, newdata=DataNew)
predictions_summarysvmsf1int$dif <- predictions_summarysvmsf1int$y -
predictions_summarysvmsf1int$pred
predictions_summarysvmsf1int <-
rbind(predictions_summarysvmsf1int,data.frame(Time=DataNew$Time,y=Data
New$Strain_face1_intermedio,pred=predictionsf1int,dif=DataNew$Strain_face1
_intermedio-predictionsf1int))

#Undamage Test period
```

```
R2 <- cor(predictions_summarysvmsf1int$y[3221:6100],
predictions_summarysvmsf1int$pred[3221:6100])
RMSE <- rmse(predicted = predictions_summarysvmsf1int$pred[3221:6100],
actual = predictions_summarysvmsf1int$y[3221:6100])
MAE <- mae(predicted = predictions_summarysvmsf1int$pred[3221:6100],
actual = predictions_summarysvmsf1int$y[3221:6100])
MAPE <- mape(predicted = predictions_summarysvmsf1int$pred[3221:6100],
actual = predictions_summarysvmsf1int$y[3221:6100])
results=rbind(results,data.frame(R2, RMSE, MAE, MAPE, variables))

#Damage period
R2 <- cor(predictions_summarysvmsf1int$y[6101:28354],
predictions_summarysvmsf1int$pred[6101:28354])
RMSE <- rmse(predicted = predictions_summarysvmsf1int$pred[6101:28354],
actual = predictions_summarysvmsf1int$y[6101:28354])
MAE <- mae(predicted = predictions_summarysvmsf1int$pred[6101:28354],
actual = predictions_summarysvmsf1int$y[6101:28354])
MAPE <- mape(predicted = predictions_summarysvmsf1int$pred[6101:28354],
actual = predictions_summarysvmsf1int$y[6101:28354])

SVMTest_sf1int=rbind(results_svm_sf1int,data.frame(R2, RMSE, MAE, MAPE,
variables))
rownames(SVMTest _sf1int) <- c("Data Teste","Data Validation","Data
Damaged")

#plot
library(ggplot2)
```

```
gridExtra::grid.arrange(ggplot(predictions_summarysvmsf1int) +
geom_point(aes(Time,y)) +
geom_point(data=predictions_summarysvmsf1int[1:3220,], aes(Time, pred),
colour = "blue") + geom_point(data=predictions_summarysvmsf1int[3221:6100,],
aes(Time, pred), colour = "green") +
geom_point(data=predictions_summarysvmsf1int[6101:28354,], aes(Time,
pred), colour = "red") + geom_vline(xintercept =
predictions_summarysvmsf1int$Time[3220], colour = "green") +
geom_vline(xintercept = predictions_summarysvmsf1int$Time[6101], colour =
"red") + labs(title="Support Vector Machine Model (Strain F1 Middle)") +
ylab("Strain"), ggplot(predictions_summarysvmsf1int) +
geom_point(aes(Time,dif)) + ylab("Residuals"),  nrow = 2)
```

2.10. Neural Networks for training data

The code below refers solely to one of the dependent variables (*Strain_F1_Middle*). This should be done as many times as dependent variables there are.

```
library(nnet)
library(Metrics)
library(randomForest)

learning_rate=0
hidden_layers=0

feats <- colnames(Dados[c(2:4,14:17)])
f <- paste(feats,collapse=' + ')
```

```
f <- paste('Strain_face1_intermedio ~',f)
f <- as.formula(f) #Convert to formula

NeuralTrainsf1int=data.frame()
set.seed(13)
performance_metrics <- data.frame()

for(H in seq(10,80,10)){ #loop to compute the parameters and save the
predictions with varying decay and size

  for(L in 10^seq(-3, 0, length = 10)){
    predictions_summary <- data.frame()

    for (k in 1:10){
      nn <- nnet(f,data=DataCal[-folds_train[[k]],],size=H, decay=L, linout=TRUE,
trace = FALSE, maxit=100, MaxNWts=7000)
      prediction <- predict(nn,DataCal[folds_train[[k]],])
      predictions_summary <- rbind(predictions_summary,
data.frame(Time=DataCal[folds_train[[k]],'Time'],
y=DataCal$Strain_face1_intermedio[folds_train[[k]]], prediction))
    }

    decay=L
    size=H

    R2 <- cor(predictions_summary$y, predictions_summary$prediction)
    RMSE <- rmse(predicted = predictions_summary$prediction, actual =
predictions_summary$y)
```

```
   MAE <- mae(predicted = predictions_summary$prediction, actual =
predictions_summary$y)
   MAPE <- mape(predicted = predictions_summary$prediction, actual =
predictions_summary$y)
   NeuralTrainsf1int=rbind(NeuralTrainsf1int,data.frame(R2, RMSE, MAE,
MAPE, decay, size))
  }
}

#Save metrics
NeuralTrainsf1int
NeuralTrainsf1int[which.max(NeuralTrainsf1int$R2),]
```

2.11. Neural Networks for testing, validation and damaged data

The code below refers solely to one of the dependent variables (*Strain_F1_Middle*). This should be done as many times as dependent variables there are.

```
size=NeuralTrainsf1int[which.max(NeuralTrainsf1int$R2),6]
decay=NeuralTrainsf1int[which.max(NeuralTrainsf1int$R2),5]

feats <- colnames(Dados[c(2:4,14:17)])
f <- paste(feats,collapse=' + ')
f <- paste('Strain_face1_intermedio ~',f)
f <- as.formula(f) #Convert to formula
```

```
results=data.frame()
predictions_summarynnsf1int <- data.frame()
performance_metrics <- data.frame()

for (k in 1:10){
  nn <- nnet(f,data=DataVal[-folds_test[[k]],], size=size, decay=decay,
linout=TRUE, trace = FALSE, maxit=100, MaxNWts=7000)
  prediction <- predict(nn,DataVal[folds_test[[k]],])
  predictions_summarynnsf1int <- rbind(predictions_summarynnsf1int,
data.frame(Time=DataVal[folds_test[[k]],'Time'],
y=DataVal$Strain_face1_intermedio[folds_test[[k]]], pred=prediction))
}
predictions_summarynnsf1int$dif <- predictions_summarynnsf1int$y -
predictions_summarynnsf1int$pred

R2 <- cor(predictions_summarynnsf1int$y, predictions_summarynnsf1int$pred)
RMSE <- rmse(predicted = predictions_summarynnsf1int$pred, actual =
predictions_summarynnsf1int$y)
MAE <- mae(predicted = predictions_summarynnsf1int$pred, actual =
predictions_summarynnsf1int$y)
MAPE <- mape(predicted = predictions_summarynnsf1int$pred, actual =
predictions_summarynnsf1int$y)
results=rbind(results,data.frame(R2, RMSE, MAE, MAPE, size, decay))

fit <- nnet(f, data=DataVal, size=size, decay=decay, linout=TRUE, trace = FALSE,
maxit=100, MaxNWts=7000)
validationsf1int <- predict(fit, newdata=DataNew)
```

```
validation_summarynn <-
data.frame(Time=DataNew$Time,y=DataNew$Strain_face1_intermedio,pred=v
alidationsf1int,dif=DataNew$Strain_face1_intermedio-validationsf1int)
predictions_summarynnsf1int <-
rbind(predictions_summarynnsf1int,validation_summarynn)

#Undamage Test period
R2 <- cor(predictions_summarynnsf1int$y[3221:6100],
predictions_summarynnsf1int$pred[3221:6100])
RMSE <- rmse(predicted = predictions_summarynnsf1int$pred[3221:6100],
actual = predictions_summarynnsf1int$y[3221:6100])
MAE <- mae(predicted = predictions_summarynnsf1int$pred[3221:6100], actual
= predictions_summarynnsf1int$y[3221:6100])
MAPE <- mape(predicted = predictions_summarynnsf1int$pred[3221:6100],
actual = predictions_summarynnsf1int$y[3221:6100])
results=rbind(results,data.frame(R2, RMSE, MAE, MAPE, size, decay))

#Damage period
R2 <- cor(predictions_summarynnsf1int$y[6101:28354],
predictions_summarynnsf1int$pred[6101:28354])
RMSE <- rmse(predicted = predictions_summarynnsf1int$pred[6101:28354],
actual = predictions_summarynnsf1int$y[6101:28354])
MAE <- mae(predicted = predictions_summarynnsf1int$pred[6101:28354],
actual = predictions_summarynnsf1int$y[6101:28354])
MAPE <- mape(predicted = predictions_summarynnsf1int$pred[6101:28354],
actual = predictions_summarynnsf1int$y[6101:28354])
```

```
NNTest_sf1int=rbind(results_nn_sf1int,data.frame(R2, RMSE, MAE, MAPE,
size, decay))
rownames(NNTest_sf1int) <- c("Data Teste","Data Validation","Data Damaged")

#plot
library(ggplot2)
gridExtra::grid.arrange(ggplot(predictions_summarynnsf1int) +
geom_point(aes(Time,y)) +
geom_point(data=predictions_summarynnsf1int[1:3220,], aes(Time, pred),
colour = "blue") + geom_point(data=predictions_summarynnsf1int[3221:6100,],
aes(Time, pred), colour = "green") +
geom_point(data=predictions_summarynnsf1int[6101:28354,], aes(Time, pred),
colour = "red") + geom_vline(xintercept =
predictions_summarynnsf1int$Time[3220], colour = "green") +
geom_vline(xintercept = predictions_summarynnsf1int$Time[6101], colour =
"red") + labs(title="Neural Network Model (Strain F1 Middle)") + ylab("Strain"),
ggplot(predictions_summarynnsf1int) + geom_point(aes(Time,dif)) +
ylab("Residuals"),  nrow = 2)
```

2.12. Hotelling T^2 Control chart

```
library(MSQC)
library(qcc)
library(MASS)

#Data preparation
"Validation Period" <-
data.frame(cbind(predictions_summarynnsf1int$dif[1:3220],predictions_summ
```

```
arynnsf2int$dif[1:3220],predictions_summarynnsf1inf$dif[1:3220],predictions_s
ummarynnsf2inf$dif[1:3220],predictions_summarynnLVDTSup$dif[1:3220],pre
dictions_summarynnLVDTInt$dif[1:3220],predictions_summarynnIncSup$dif[1
:3220],predictions_summarynnIncInt$dif[1:3220],predictions_summarynnIncInf
$dif[1:3220]))
colnames(`Validation Period`) <-
c("SF1int","SF2Int","SF1Inf","SF2Inf","LVDTSup","LVDTInt","IncSup","IncInt","In
cInf")
"Undamage Test Period" <-
data.frame(cbind(predictions_summarynnsf1int$dif[3221:6100],predictions_su
mmarynnsf2int$dif[3221:6100],predictions_summarynnsf1inf$dif[3221:6100],pr
edictions_summarynnsf2inf$dif[3221:6100],predictions_summarynnLVDTSup$
dif[3221:6100],predictions_summarynnLVDTInt$dif[3221:6100],predictions_su
mmarynnIncSup$dif[3221:6100],predictions_summarynnIncInt$dif[3221:6100],p
redictions_summarynnIncInf$dif[3221:6100]))
colnames(`Undamage Test Period`) <-
c("SF1int","SF2Int","SF1Inf","SF2Inf","LVDTSup","LVDTInt","IncSup","IncInt","In
cInf")
"Damage Period" <-
data.frame(cbind(predictions_summarynnsf1int$dif[6101:28354],predictions_su
mmarynnsf2int$dif[6101:28354],predictions_summarynnsf1inf$dif[6101:28354],
predictions_summarynnsf2inf$dif[6101:28354],predictions_summarynnLVDTS
up$dif[6101:28354],predictions_summarynnLVDTInt$dif[6101:28354],predictio
ns_summarynnIncSup$dif[6101:28354],predictions_summarynnIncInt$dif[6101:
28354],predictions_summarynnIncInf$dif[6101:28354]))
colnames(`Damage Period`) <-
c("SF1int","SF2Int","SF1Inf","SF2Inf","LVDTSup","LVDTInt","IncSup","IncInt","In
cInf")
```

```
#T2-Hotelling#
rob <- cov.rob(`Validation Period`)

#Phase I
q <- mqcc(`Validation Period`, type = "T2.single", confidence.level =
0.9999999999999999, plot=TRUE, center = rob$center, cov = rob$cov)
summary(q)
(length(q$statistics[q$statistics>q$limits[2]])/nrow(Before))*100

#Phase II (Validation)
qval <- mqcc(`Validation Period`, type = "T2.single", confidence.level =
0.9999999999999999, newdata = `Undamage Test Period`, plot=TRUE, center =
rob$center, cov = rob$cov)
summary(qval)
(length(qval$newstats[qval$newstats>qval$limits[2]])/nrow(Validation))*100

#Phase II (Damage)
qq <- mqcc(`Validation Period`, type = "T2.single", confidence.level =
0.9999999999999999, newdata = `Damage Period`, plot=TRUE, center =
rob$center, cov = rob$cov)
summary(qq)
```

2.13. Missing values interpolation

The present dataset registered the hour, minute and second of when an observation was recorded. Since there were some intervals where the system stopped recording observations, the data had to be normalized. This was done

by first considering the observations were always recorded at zero seconds. Next, the minute part had to be either 0, 15, 30 or 45. To do so, every observation recorded between minute 0 and minute 14 was to be considered recorded in minute 0. This leads to the *Time* variable to only be presented in the xxh00m00s, xxh15m00s, xxh30m00s and xxh45m00s.

After normalizing the data a sequence of time in multiples of 15 minutes from 27-04-2018 14h15m00s to 23-05-2019 07h00m00s was created. From these a match was made from this sequence and the *dataset*.

Next the Kalman filter was used. Applying it to the whole series creates object *DataTS1* which has very good results except for the last day of the undamaged data. To correct this, in parallel, the Kalman filter was used feeding it only the undamaged data creating *DataTS2*. From this, a new *dataset* is created (*DataTS3*), considering the observations in *DataTS1* and the day of observations in *DataTS2*. From this, the process of removing the shift after the insertion of damage is also dealt with.

```
#Tranform time to always be in multiples of 15 minutes
DataTS <- rbind(DataCalVal,DataD)
DataTS$Hour <- substr(DataTS$Time,15,16)
DataTS$Hour <- replace(DataTS$Hour, DataTS$Hour > 0 & DataTS$Hour < 15,
"00")
DataTS$Hour <- replace(DataTS$Hour, DataTS$Hour > 15 & DataTS$Hour < 30,
15)
DataTS$Hour <- replace(DataTS$Hour, DataTS$Hour > 30 & DataTS$Hour < 45,
30)
DataTS$Hour <- replace(DataTS$Hour, DataTS$Hour > 45 & DataTS$Hour < 60,
45)
library(lubridate)
```

```
DataTS$Time <-
ymd_hm(paste(substr(DataTS$Time,1,14),DataTS$Hour,sep=""))
DataTS$Hour <- NULL

#Create a sequence of time in multiples of 15 minutes from 27-04-2018
14h15m00s to 23-05-2019 07h00m00s
allDates <- seq(ISOdate(2018,4,27,14,15), ISOdate(2019,5,23,7,0), by = "15 min")
DataTS <- merge(data.frame(Time=allDates),DataTS,all.x=TRUE)
rownames(DataTS) <- NULL

DataTS$id <- 1:nrow(DataTS)
DataTS$id[1:9] <- paste("0000",DataTS$id[1:9],sep="")
DataTS$id[10:99] <- paste("000",DataTS$id[10:99],sep="")
DataTS$id[100:999] <- paste("00",DataTS$id[100:999],sep="")
DataTS$id[1000:9999] <- paste("0",DataTS$id[1000:9999],sep="")

library(imputeTS)

DataTS1 <- DataTS
DataTS1[,-c(1,18,19)] <- na_seasplit(DataTS[,-c(1,18,19)], algorithm = "kalman",
find_frequency=TRUE, type="level")

DataTS2 <- DataTS
DataTS2[1:13079,-c(1,18,19)] <- na_seasplit(DataTS[1:13079,-c(1,18,19)],
algorithm = "kalman", find_frequency=TRUE, type="level")
DataTS2[13080:37508,-c(1,18,19)] <- na_seasplit(DataTS[13080:37508,-c(1,18,19)],
algorithm = "kalman", find_frequency=TRUE, type="level")
```

```
DataTS3 <-
rbind(DataTS1[1:13017,],DataTS2[13018:13979,],DataTS1[13980:37508,])

DataCalVal <- DataTS3[1:13079,c(1:17,19)]
DataD <- DataTS3[13080:37508,c(1:17,19)]

#Correct shift in damage period data
for (i in c(5:13)) {
  DataD[,i] <- DataD[,i] + (mean(DataCalVal[12408:13079,i],na.rm=TRUE) -
mean(DataD[,i][1:671],na.rm=TRUE))
}

DataTS <- rbind(DataCalVal,DataD)
```

2.14. Cointegration

```
library(tseries)
library(forecast)
library(urca)
library(vars)
library(tsDyn)

#Data Undamage####
IncInf <- ts(DataCalVal[,5], start=1.15625, frequency=96)
IncInt <- ts(DataCalVal[,6], start=1.15625, frequency=96)
IncSup <- ts(DataCalVal[,7], start=1.15625, frequency=96)
LVDTInt <- ts(DataCalVal[,8], start=1.15625, frequency=96)
LVDTSup <- ts(DataCalVal[,9], start=1.15625, frequency=96)
```

```
sf1Inf <- ts(DataCalVal[,10], start=1.15625, frequency=96)
sf2Inf <- ts(DataCalVal[,11], start=1.15625, frequency=96)
sf1Int <- ts(DataCalVal[,12], start=1.15625, frequency=96)
sf2Int <- ts(DataCalVal[,13], start=1.15625, frequency=96)

#Create dataset and select the number of lags (data test)
DataCalValT <-
(data.frame((IncInf[1:10199]),(IncInt[1:10199]),(IncSup[1:10199]),(LVDTInt[1:101
99]),(LVDTSup[1:10199]),(sf1Inf[1:10199]),(sf2Inf[1:10199]),(sf1Int[1:10199]),(sf2I
nt[1:10199])))
VARselect <- VARselect(DataCalValT, lag.max=300)$selection #optimal number
of lags=56

#Verify the stationarity of the series before and after integration of order 1: I(1)
ur.df(IncInf[1:10199], type = "none", lags=56 ,selectlags = "Fixed") %>%
summary()
ur.df(diff(IncInf[1:10199],lag=56), type = "none", lags=56, selectlags = "Fixed")
%>% summary()
ur.df(IncInt[1:10199], type = "none", lags=56 ,selectlags = "Fixed") %>%
summary()
ur.df(diff(IncInt[1:10199],lag=56), type = "none", lags=56, selectlags = "Fixed")
%>% summary()
ur.df(IncSup[1:10199], type = "none", lags=56 ,selectlags = "Fixed") %>%
summary()
ur.df(diff(IncSup[1:10199],lag=56), type = "none", lags=56, selectlags = "Fixed")
%>% summary()
ur.df(LVDTInt[1:10199], type = "none", lags=56 ,selectlags = "Fixed") %>%
summary()
```

```
ur.df(diff(LVDTInt[1:10199],lag=56), type = "none", lags=56, selectlags = "Fixed")
%>% summary()
ur.df(LVDTSup[1:10199], type = "none", lags=56 ,selectlags = "Fixed") %>%
summary()
ur.df(diff(LVDTSup[1:10199],lag=56), type = "none", lags=56, selectlags =
"Fixed") %>% summary()
ur.df(sf1Inf[1:10199], type = "none", lags=56 ,selectlags = "Fixed") %>%
summary()
ur.df(diff(sf1Inf[1:10199],lag=56), type = "none", lags=56, selectlags = "Fixed")
%>% summary()
ur.df(sf2Inf[1:10199], type = "none", lags=56 ,selectlags = "Fixed") %>%
summary() #sai
ur.df(diff(sf2Inf[1:10199],lag=56), type = "none", lags=56, selectlags = "Fixed")
%>% summary()
ur.df(sf1Int[1:10199], type = "none", lags=56 ,selectlags = "Fixed") %>%
summary()
ur.df(diff(sf1Int[1:10199],lag=56), type = "none", lags=56, selectlags = "Fixed")
%>% summary()
ur.df(sf2Int[1:10199], type = "none", lags=56 ,selectlags = "Fixed") %>%
summary()
ur.df(diff(sf2Int[1:10199],lag=56), type = "none", lags=56, selectlags = "Fixed")
%>% summary()



#Johansen Test
jotest=ca.jo(DataCalValT, type="trace", K=56, ecdet="none", spec = 'longrun')
summary(jotest)
```

```
#Generate Cointegration residuals for Validation data
BeforeV1 <-
as.numeric(jotest@V[1,1]*diff(IncInf[1:10199],lag=56)+jotest@V[2,1]*diff(IncInt[1:
10199],lag=56)+jotest@V[3,1]*diff(IncSup[1:10199],lag=56)+jotest@V[4,1]*diff(LV
DTInt[1:10199],lag=56)+jotest@V[5,1]*diff(LVDTSup[1:10199],lag=56)+jotest@V[6
,1]*diff(sf1Inf[1:10199],lag=56)+jotest@V[7,1]*diff(sf2Inf[1:10199],lag=56)+jotest@
V[8,1]*diff(sf1Int[1:10199],lag=56)+jotest@V[9,1]*diff(sf2Int[1:10199],lag=56))
BeforeV2 <-
as.numeric(jotest@V[1,2]*diff(IncInf[1:10199],lag=56)+jotest@V[2,2]*diff(IncInt[1:
10199],lag=56)+jotest@V[3,2]*diff(IncSup[1:10199],lag=56)+jotest@V[4,2]*diff(LV
DTInt[1:10199],lag=56)+jotest@V[5,2]*diff(LVDTSup[1:10199],lag=56)+jotest@V[6
,2]*diff(sf1Inf[1:10199],lag=56)+jotest@V[7,2]*diff(sf2Inf[1:10199],lag=56)+jotest@
V[8,2]*diff(sf1Int[1:10199],lag=56)+jotest@V[9,2]*diff(sf2Int[1:10199],lag=56))
BeforeV3 <-
as.numeric(jotest@V[1,3]*diff(IncInf[1:10199],lag=56)+jotest@V[2,3]*diff(IncInt[1:
10199],lag=56)+jotest@V[3,3]*diff(IncSup[1:10199],lag=56)+jotest@V[4,3]*diff(LV
DTInt[1:10199],lag=56)+jotest@V[5,3]*diff(LVDTSup[1:10199],lag=56)+jotest@V[6
,3]*diff(sf1Inf[1:10199],lag=56)+jotest@V[7,3]*diff(sf2Inf[1:10199],lag=56)+jotest@
V[8,3]*diff(sf1Int[1:10199],lag=56)+jotest@V[9,3]*diff(sf2Int[1:10199],lag=56))
BeforeV4 <-
as.numeric(jotest@V[1,4]*diff(IncInf[1:10199],lag=56)+jotest@V[2,4]*diff(IncInt[1:
10199],lag=56)+jotest@V[3,4]*diff(IncSup[1:10199],lag=56)+jotest@V[4,4]*diff(LV
DTInt[1:10199],lag=56)+jotest@V[5,4]*diff(LVDTSup[1:10199],lag=56)+jotest@V[6
,4]*diff(sf1Inf[1:10199],lag=56)+jotest@V[7,4]*diff(sf2Inf[1:10199],lag=56)+jotest@
V[8,4]*diff(sf1Int[1:10199],lag=56)+jotest@V[9,4]*diff(sf2Int[1:10199],lag=56))
BeforeV5 <-
as.numeric(jotest@V[1,5]*diff(IncInf[1:10199],lag=56)+jotest@V[2,5]*diff(IncInt[1:
10199],lag=56)+jotest@V[3,5]*diff(IncSup[1:10199],lag=56)+jotest@V[4,5]*diff(LV
```

```
DTInt[1:10199],lag=56)+jotest@V[5,5]*diff(LVDTSup[1:10199],lag=56)+jotest@V[6
,5]*diff(sf1Inf[1:10199],lag=56)+jotest@V[7,5]*diff(sf2Inf[1:10199],lag=56)+jotest@
V[8,5]*diff(sf1Int[1:10199],lag=56)+jotest@V[9,5]*diff(sf2Int[1:10199],lag=56))
BeforeV6 <-
as.numeric(jotest@V[1,6]*diff(IncInf[1:10199],lag=56)+jotest@V[2,6]*diff(IncInt[1:
10199],lag=56)+jotest@V[3,6]*diff(IncSup[1:10199],lag=56)+jotest@V[4,6]*diff(LV
DTInt[1:10199],lag=56)+jotest@V[5,6]*diff(LVDTSup[1:10199],lag=56)+jotest@V[6
,6]*diff(sf1Inf[1:10199],lag=56)+jotest@V[7,6]*diff(sf2Inf[1:10199],lag=56)+jotest@
V[8,6]*diff(sf1Int[1:10199],lag=56)+jotest@V[9,6]*diff(sf2Int[1:10199],lag=56))
BeforeV7 <-
as.numeric(jotest@V[1,7]*diff(IncInf[1:10199],lag=56)+jotest@V[2,7]*diff(IncInt[1:
10199],lag=56)+jotest@V[3,7]*diff(IncSup[1:10199],lag=56)+jotest@V[4,7]*diff(LV
DTInt[1:10199],lag=56)+jotest@V[5,7]*diff(LVDTSup[1:10199],lag=56)+jotest@V[6
,7]*diff(sf1Inf[1:10199],lag=56)+jotest@V[7,7]*diff(sf2Inf[1:10199],lag=56)+jotest@
V[8,7]*diff(sf1Int[1:10199],lag=56)+jotest@V[9,7]*diff(sf2Int[1:10199],lag=56))
BeforeV8 <-
as.numeric(jotest@V[1,8]*diff(IncInf[1:10199],lag=56)+jotest@V[2,8]*diff(IncInt[1:
10199],lag=56)+jotest@V[3,8]*diff(IncSup[1:10199],lag=56)+jotest@V[4,8]*diff(LV
DTInt[1:10199],lag=56)+jotest@V[5,8]*diff(LVDTSup[1:10199],lag=56)+jotest@V[6
,8]*diff(sf1Inf[1:10199],lag=56)+jotest@V[7,8]*diff(sf2Inf[1:10199],lag=56)+jotest@
V[8,8]*diff(sf1Int[1:10199],lag=56)+jotest@V[9,8]*diff(sf2Int[1:10199],lag=56))
BeforeV <-
cbind(BeforeV1,BeforeV2,BeforeV3,BeforeV4,BeforeV5,BeforeV6,BeforeV7,Befor
eV8)


#Data Undamaged (Undamage Test period)
#Generate cointegration residuals
```

```
BeforeUT1 <-
as.numeric(jotest@V[1,1]*diff(IncInf[10200:13079],lag=56)+jotest@V[2,1]*diff(IncI
nt[10200:13079],lag=56)+jotest@V[3,1]*diff(IncSup[10200:13079],lag=56)+jotest@V
[4,1]*diff(LVDTInt[10200:13079],lag=56)+jotest@V[5,1]*diff(LVDTSup[10200:130
79],lag=56)+jotest@V[6,1]*diff(sf1Inf[10200:13079],lag=56)+jotest@V[7,1]*diff(sf2I
nf[10200:13079],lag=56)+jotest@V[8,1]*diff(sf1Int[10200:13079],lag=56)+jotest@V[
9,1]*diff(sf2Int[10200:13079],lag=56))
BeforeUT2 <-
as.numeric(jotest@V[1,2]*diff(IncInf[10200:13079],lag=56)+jotest@V[2,2]*diff(IncI
nt[10200:13079],lag=56)+jotest@V[3,2]*diff(IncSup[10200:13079],lag=56)+jotest@V
[4,2]*diff(LVDTInt[10200:13079],lag=56)+jotest@V[5,2]*diff(LVDTSup[10200:130
79],lag=56)+jotest@V[6,2]*diff(sf1Inf[10200:13079],lag=56)+jotest@V[7,2]*diff(sf2I
nf[10200:13079],lag=56)+jotest@V[8,2]*diff(sf1Int[10200:13079],lag=56)+jotest@V[
9,2]*diff(sf2Int[10200:13079],lag=56))
BeforeUT3 <-
as.numeric(jotest@V[1,3]*diff(IncInf[10200:13079],lag=56)+jotest@V[2,3]*diff(IncI
nt[10200:13079],lag=56)+jotest@V[3,3]*diff(IncSup[10200:13079],lag=56)+jotest@V
[4,3]*diff(LVDTInt[10200:13079],lag=56)+jotest@V[5,3]*diff(LVDTSup[10200:130
79],lag=56)+jotest@V[6,3]*diff(sf1Inf[10200:13079],lag=56)+jotest@V[7,3]*diff(sf2I
nf[10200:13079],lag=56)+jotest@V[8,3]*diff(sf1Int[10200:13079],lag=56)+jotest@V[
9,3]*diff(sf2Int[10200:13079],lag=56))
BeforeUT4 <-
as.numeric(jotest@V[1,4]*diff(IncInf[10200:13079],lag=56)+jotest@V[2,4]*diff(IncI
nt[10200:13079],lag=56)+jotest@V[3,4]*diff(IncSup[10200:13079],lag=56)+jotest@V
[4,4]*diff(LVDTInt[10200:13079],lag=56)+jotest@V[5,4]*diff(LVDTSup[10200:130
79],lag=56)+jotest@V[6,4]*diff(sf1Inf[10200:13079],lag=56)+jotest@V[7,4]*diff(sf2I
nf[10200:13079],lag=56)+jotest@V[8,4]*diff(sf1Int[10200:13079],lag=56)+jotest@V[
9,4]*diff(sf2Int[10200:13079],lag=56))
```

```
BeforeUT5 <-
as.numeric(jotest@V[1,5]*diff(IncInf[10200:13079],lag=56)+jotest@V[2,5]*diff(IncI
nt[10200:13079],lag=56)+jotest@V[3,5]*diff(IncSup[10200:13079],lag=56)+jotest@V
[4,5]*diff(LVDTInt[10200:13079],lag=56)+jotest@V[5,5]*diff(LVDTSup[10200:130
79],lag=56)+jotest@V[6,5]*diff(sf1Inf[10200:13079],lag=56)+jotest@V[7,5]*diff(sf2I
nf[10200:13079],lag=56)+jotest@V[8,5]*diff(sf1Int[10200:13079],lag=56)+jotest@V[
9,5]*diff(sf2Int[10200:13079],lag=56))
BeforeUT6 <-
as.numeric(jotest@V[1,6]*diff(IncInf[10200:13079],lag=56)+jotest@V[2,6]*diff(IncI
nt[10200:13079],lag=56)+jotest@V[3,6]*diff(IncSup[10200:13079],lag=56)+jotest@V
[4,6]*diff(LVDTInt[10200:13079],lag=56)+jotest@V[5,6]*diff(LVDTSup[10200:130
79],lag=56)+jotest@V[6,6]*diff(sf1Inf[10200:13079],lag=56)+jotest@V[7,6]*diff(sf2I
nf[10200:13079],lag=56)+jotest@V[8,6]*diff(sf1Int[10200:13079],lag=56)+jotest@V[
9,6]*diff(sf2Int[10200:13079],lag=56))
BeforeUT7 <-
as.numeric(jotest@V[1,7]*diff(IncInf[10200:13079],lag=56)+jotest@V[2,7]*diff(IncI
nt[10200:13079],lag=56)+jotest@V[3,7]*diff(IncSup[10200:13079],lag=56)+jotest@V
[4,7]*diff(LVDTInt[10200:13079],lag=56)+jotest@V[5,7]*diff(LVDTSup[10200:130
79],lag=56)+jotest@V[6,7]*diff(sf1Inf[10200:13079],lag=56)+jotest@V[7,7]*diff(sf2I
nf[10200:13079],lag=56)+jotest@V[8,7]*diff(sf1Int[10200:13079],lag=56)+jotest@V[
9,7]*diff(sf2Int[10200:13079],lag=56))
BeforeUT8 <-
as.numeric(jotest@V[1,8]*diff(IncInf[10200:13079],lag=56)+jotest@V[2,8]*diff(IncI
nt[10200:13079],lag=56)+jotest@V[3,8]*diff(IncSup[10200:13079],lag=56)+jotest@V
[4,8]*diff(LVDTInt[10200:13079],lag=56)+jotest@V[5,8]*diff(LVDTSup[10200:130
79],lag=56)+jotest@V[6,8]*diff(sf1Inf[10200:13079],lag=56)+jotest@V[7,8]*diff(sf2I
nf[10200:13079],lag=56)+jotest@V[8,8]*diff(sf1Int[10200:13079],lag=56)+jotest@V[
9,8]*diff(sf2Int[10200:13079],lag=56))
```

```
BeforeUT <-
cbind(BeforeUT1,BeforeUT2,BeforeUT3,BeforeUT4,BeforeUT5,BeforeUT6,Befor
eUT7,BeforeUT8)


#Data Damage####
IncInfD <- ts(DataD[,5], start=1.7083333, frequency=96)
IncIntD  <- ts(DataD[,6], start=1.7083333, frequency=96)
IncSupD  <- ts(DataD[,7], start=1.7083333, frequency=96)
LVDTIntD  <- ts(DataD[,8], start=1.7083333, frequency=96)
LVDTSupD  <- ts(DataD[,9], start=1.7083333, frequency=96)
sf1InfD  <- ts(DataD[,10], start=1.7083333, frequency=96)
sf2InfD  <- ts(DataD[,11], start=1.7083333, frequency=96)
sf1IntD  <- ts(DataD[,12], start=1.7083333, frequency=96)
sf2IntD  <- ts(DataD[,13], start=1.7083333, frequency=96)

#Generate cointegration residuals
After1 <-
as.numeric(jotest@V[1,1]*diff(IncInfD,lag=56)+jotest@V[2,1]*diff(IncIntD,lag=56)
+jotest@V[3,1]*diff(IncSupD,lag=56)+jotest@V[4,1]*diff(LVDTIntD,lag=56)+jotest
@V[5,1]*diff(LVDTSupD,lag=56)+jotest@V[6,1]*diff(sf1InfD,lag=56)+jotest@V[7,1
]*diff(sf2InfD,lag=56)+jotest@V[8,1]*diff(sf1IntD,lag=56)+jotest@V[9,1]*diff(sf2In
tD,lag=56))
After2 <-
as.numeric(jotest@V[1,2]*diff(IncInfD,lag=56)+jotest@V[2,2]*diff(IncIntD,lag=56)
+jotest@V[3,2]*diff(IncSupD,lag=56)+jotest@V[4,2]*diff(LVDTIntD,lag=56)+jotest
@V[5,2]*diff(LVDTSupD,lag=56)+jotest@V[6,2]*diff(sf1InfD,lag=56)+jotest@V[7,2
```

```
]*diff(sf2InfD,lag=56)+jotest@V[8,2]*diff(sf1IntD,lag=56)+jotest@V[9,2]*diff(sf2In
tD,lag=56))
After3 <-
as.numeric(jotest@V[1,3]*diff(IncInfD,lag=56)+jotest@V[2,3]*diff(IncIntD,lag=56)
+jotest@V[3,3]*diff(IncSupD,lag=56)+jotest@V[4,3]*diff(LVDTIntD,lag=56)+jotest
@V[5,3]*diff(LVDTSupD,lag=56)+jotest@V[6,3]*diff(sf1InfD,lag=56)+jotest@V[7,3
]*diff(sf2InfD,lag=56)+jotest@V[8,3]*diff(sf1IntD,lag=56)+jotest@V[9,3]*diff(sf2In
tD,lag=56))
After4 <-
as.numeric(jotest@V[1,4]*diff(IncInfD,lag=56)+jotest@V[2,4]*diff(IncIntD,lag=56)
+jotest@V[3,4]*diff(IncSupD,lag=56)+jotest@V[4,4]*diff(LVDTIntD,lag=56)+jotest
@V[5,4]*diff(LVDTSupD,lag=56)+jotest@V[6,4]*diff(sf1InfD,lag=56)+jotest@V[7,4
]*diff(sf2InfD,lag=56)+jotest@V[8,4]*diff(sf1IntD,lag=56)+jotest@V[9,4]*diff(sf2In
tD,lag=56))
After5 <-
as.numeric(jotest@V[1,5]*diff(IncInfD,lag=56)+jotest@V[2,5]*diff(IncIntD,lag=56)
+jotest@V[3,5]*diff(IncSupD,lag=56)+jotest@V[4,5]*diff(LVDTIntD,lag=56)+jotest
@V[5,5]*diff(LVDTSupD,lag=56)+jotest@V[6,5]*diff(sf1InfD,lag=56)+jotest@V[7,5
]*diff(sf2InfD,lag=56)+jotest@V[8,5]*diff(sf1IntD,lag=56)+jotest@V[9,5]*diff(sf2In
tD,lag=56))
After6 <-
as.numeric(jotest@V[1,6]*diff(IncInfD,lag=56)+jotest@V[2,6]*diff(IncIntD,lag=56)
+jotest@V[3,6]*diff(IncSupD,lag=56)+jotest@V[4,6]*diff(LVDTIntD,lag=56)+jotest
@V[5,6]*diff(LVDTSupD,lag=56)+jotest@V[6,6]*diff(sf1InfD,lag=56)+jotest@V[7,6
]*diff(sf2InfD,lag=56)+jotest@V[8,6]*diff(sf1IntD,lag=56)+jotest@V[9,6]*diff(sf2In
tD,lag=56))
After7 <-
as.numeric(jotest@V[1,7]*diff(IncInfD,lag=56)+jotest@V[2,7]*diff(IncIntD,lag=56)
```

```
+jotest@V[3,7]*diff(IncSupD,lag=56)+jotest@V[4,7]*diff(LVDTIntD,lag=56)+jotest
@V[5,7]*diff(LVDTSupD,lag=56)+jotest@V[6,7]*diff(sf1InfD,lag=56)+jotest@V[7,7
]*diff(sf2InfD,lag=56)+jotest@V[8,7]*diff(sf1IntD,lag=56)+jotest@V[9,7]*diff(sf2In
tD,lag=56))
After8 <-
as.numeric(jotest@V[1,8]*diff(IncInfD,lag=56)+jotest@V[2,8]*diff(IncIntD,lag=56)
+jotest@V[3,8]*diff(IncSupD,lag=56)+jotest@V[4,8]*diff(LVDTIntD,lag=56)+jotest
@V[5,8]*diff(LVDTSupD,lag=56)+jotest@V[6,8]*diff(sf1InfD,lag=56)+jotest@V[7,8
]*diff(sf2InfD,lag=56)+jotest@V[8,8]*diff(sf1IntD,lag=56)+jotest@V[9,8]*diff(sf2In
tD,lag=56))
After <- cbind(After1,After2,After3,After4,After5,After6,After7,After8)
```

2.15. X-bar Control chart – time series analysis

```
CointResiduals <- as.data.frame(rbind(BeforeV,BeforeUT,After))
CointResiduals <- cbind(c(1:nrow(CointResiduals)),CointResiduals)
colnames(CointResiduals) <-
c("Observation","Residual1","Residual2","Residual3","Residual4","Residual5","R
esidual6","Residual7","Residual8")

library(ggplot2)
ggplot(CointResiduals) +
geom_point(data=CointResiduals,aes(Observation,Residual1)) +
  geom_vline(xintercept=nrow(BeforeV),linetype="dashed",color="green") +

geom_vline(xintercept=(nrow(BeforeV)+nrow(BeforeUT)),linetype="dashed",col
or="green") +
```

```
  geom_hline(yintercept = mean(CointResiduals$Residual1[1:nrow(BeforeV)])-
3*sd(CointResiduals$Residual1[1:nrow(BeforeV)]), colour = "green") +
  geom_hline(yintercept =
mean(CointResiduals$Residual1[1:nrow(BeforeV)])+3*sd(CointResiduals$Resid
ual1[1:nrow(BeforeV)]), colour = "green") +
  ylab("Value") + labs(title="Cointegration Residual1", caption="Vertical lines
separate validation, undamaged test and damage period ; Horizontal lines are
mean+-3*sd limits based on the validation period")
```

2.16. Hotelling T^2 based Control chart – explanatory models

```
#Phase I
t2I <- as.data.frame(q$statistics)
t2I$UCL <- 0
t2I$UCL[t2I$`q$statistics`>q$limits[2]] <- 1
t2I$L8 <- 0
for (i in 8:3220){
  t2I$L8[i] <- sum(t2I$UCL[(i-7):i])
}
length(t2I$L8[t2I$L8==8])



#Phase II (Validation)
t2Val <- as.data.frame(qval$newstats)
t2Val$UCL <- 0
t2Val$UCL[t2Val$`qval$newstats`>qval$limits[2]] <- 1
```

```
t2Val$L8 <- 0
for (i in 8:2880){
  t2Val$L8[i] <- sum(t2Val$UCL[(i-7):i])
}
length(t2Val$L8[t2Val$L8==8])

#Phase II (Damage)
t2II <- as.data.frame(qq$newstats)
t2II$UCL <- 0
t2II$UCL[t2II$`qq$newstats`>qq$limits[2]] <- 1
t2II$L8 <- 0
for (i in 8:22254){
  t2II$L8[i] <- sum(t2II$UCL[(i-7):i])
}
length(t2II$L8[t2II$L8==8])-4
(length(t2II$L8[t2II$L8==8])-4)/22254

#plot
T28 <- as.data.frame(c(t2I$L8,t2Val$L8,t2II$L8))
T28 <- cbind(c(1:28354),T28)
colnames(T28) <- c("Observation","Frequency")
library(ggplot2)
ggplot(T28) + geom_point(data=T28,aes(Observation,Frequency)) +
  geom_vline(xintercept=nrow(t2I),color="green") +
geom_vline(xintercept=(nrow(t2I)+nrow(t2Val)),color="green") +
  scale_y_continuous(breaks = seq(0, 8, len = 5))

T28$Alarm <- 0
```

```
T28$Alarm[T28$Frequency==8] <- 1
ggplot(T28) + geom_point(data=T28,aes(Observation,Alarm)) +
  geom_vline(xintercept=nrow(t2I),color="green") +
geom_vline(xintercept=(nrow(t2I)+nrow(t2Val)),color="green") +
  scale_y_continuous(breaks = seq(0, 1)) +
  xlab("Sets of 8 Observations")
```

2.17. X-bar based Control chart – time series analysis

```
XbarI <- as.data.frame(CointResiduals[,1:2])
XbarI$CL <- 0
XbarI$CL[XbarI$Residual1>mean(CointResiduals$Residual1[1:nrow(BeforeV)])
+3*sd(CointResiduals$Residual1[1:nrow(BeforeV)])] <- 1
XbarI$CL[XbarI$Residual1<mean(CointResiduals$Residual1[1:nrow(BeforeV)])
-3*sd(CointResiduals$Residual1[1:nrow(BeforeV)])] <- 1
XbarI$OC <- 0
for (i in 3:37340){
  XbarI$OC[i] <- sum(XbarI$CL[(i-2):i])
}

XbarOC <- XbarI[,c(1,4)]
colnames(XbarOC) <- c("Observation","Frequency")

library(ggplot2)
ggplot(XbarOC) + geom_point(data= XbarOC,aes(Observation,Frequency)) +
```

```
  geom_vline(xintercept=nrow(BeforeV),linetype="dashed",color="green") +
geom_vline(xintercept=(nrow(BeforeV)+nrow(BeforeUT)),linetype="dashed",col
or="green") +
  scale_y_continuous(breaks = seq(0, 3, len = 4))

XBAROC$Alarm <- 0
XBAROC$Alarm[XBAROC$Frequency==3] <- 1
ggplot(XBAROC) + geom_point(data=XBAROC,aes(Observation,Alarm)) +
  geom_vline(xintercept=nrow(BeforeV),linetype="dashed",color="green") +
geom_vline(xintercept=(nrow(BeforeV)+nrow(BeforeUT)),linetype="dashed",col
or="green") +
  scale_y_continuous(breaks = seq(0, 1)) +
  xlab("Sets of 3 Observations")
```

Appendix 3 – Explanatory Models Results

	Linear Regression								
	Strain F1 Middle	Strain F2 Middle	Strain F1 lower	Strain F2 lower	LVDT upper	LVDT middle	Inclinometer upper	Inclinometer middle	Inclinometer lower
R2	0,76	0,69	0,77	0,82	0,53	0,62	0,90	0,85	0,87
RMSE	6,45	7,70	6,65	9,99	0,22	0,08	9,06	8,16	6,92
MAE	5,07	5,96	5,20	7,55	0,16	0,06	6,56	6,16	5,53
MAPE	0,03	0,04	0,03	0,20	0,16	0,80	0,01	0,01	0,00

Table 6: Error metrics for the Linear Regression Model

	Random Forest								
	Strain F1 Middle	Strain F2 Middle	Strain F1 lower	Strain F2 lower	LVDT upper	LVDT middle	Inclinometer upper	Inclinometer middle	Inclinometer lower
R2	0,81	0,75	0,83	0,85	0,82	0,62	0,88	0,86	0,87
RMSE	5,88	7,04	5,83	9,10	0,15	0,8	9,90	7,88	6,91
MAE	4,45	5,24	4,41	6,68	0,10	0,06	6,38	5,76	5,51
MAPE	0,02	0,03	0,02	0,19	0,11	0,79	0,01	0,00	0,00

Table 7: Error metrics for the Random Forest Model

	Support Vector Machine								
	Strain F1 Middle	Strain F2 Middle	Strain F1 lower	Strain F2 lower	LVDT upper	LVDT middle	Inclinometer upper	Inclinometer middle	Inclinometer lower
R2	0,76	0,69	0,77	0,82	0,80	0,62	0,90	0,84	0,87
RMSE	6,47	7,75	6,67	9,98	0,15	0,08	9,13	8,22	6,94
MAE	5,06	5,93	5,19	7,52	0,12	0,06	6,53	6,12	5,52
MAPE	0,03	0,04	0,03	0,20	0,12	0,77	0,01	0,01	0,00

Table 8: Error metrics for the Support Vector Machine Model

Appendix 4 - Neural Network Model Prediction

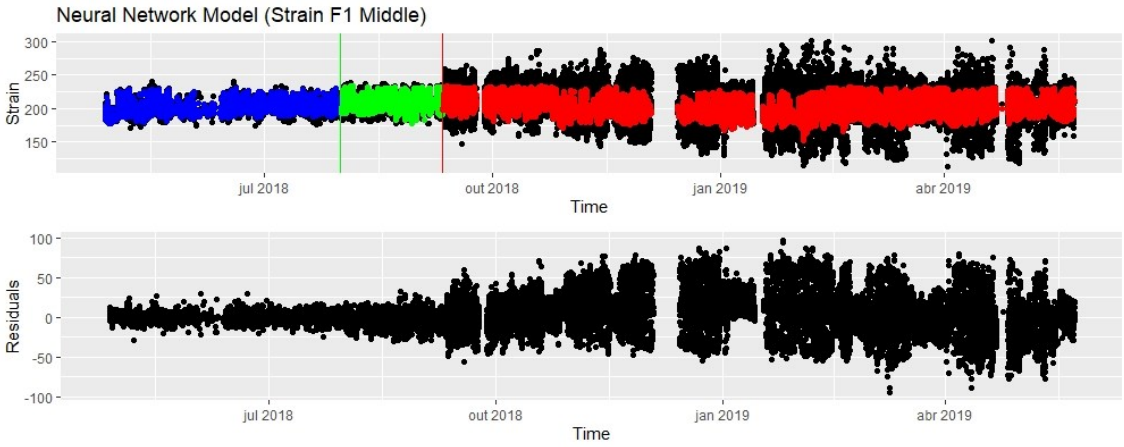

Figure 15: Top graph refers to Observed and Predicted observations from the Neural Network model. Black dots refer to the observed values; Blue dots refer to the predicted values for the validation period; Green dots refer to the predicted values for the undamaged test period; Red dots refer to the predicted values for damage period. Bottom graph represents the difference between the Observed and Predicted observations

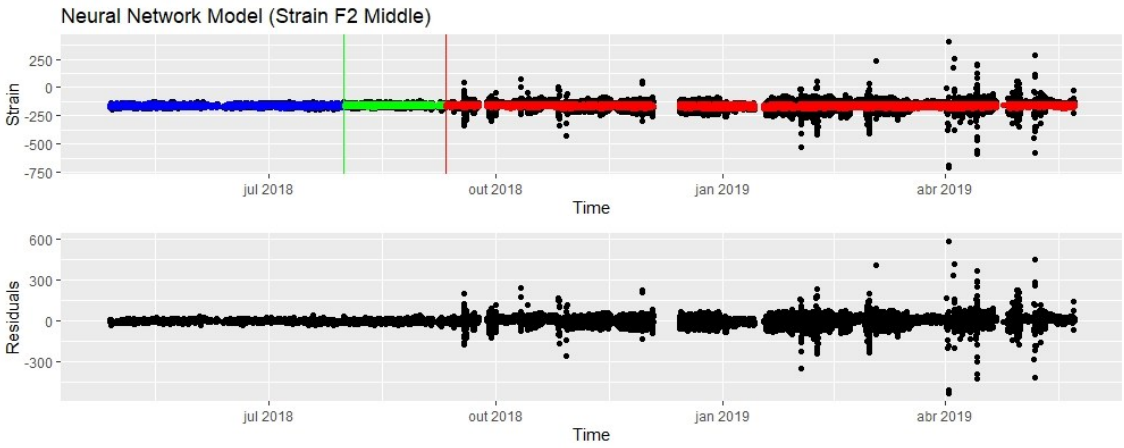

Figure 16: Top graph refers to Observed and Predicted observations from the Neural Network model. Black dots refer to the observed values; Blue dots refer to the predicted values for the validation period; Green dots refer to the predicted values for the undamaged test period; Red dots refer to the predicted values for damage period. Bottom graph represents the difference between the Observed and Predicted observations

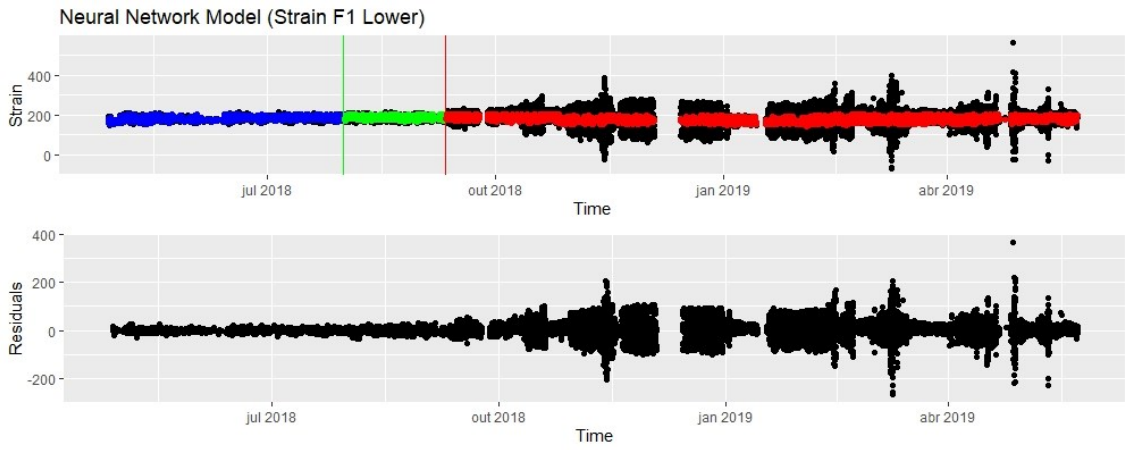

Figure 17: Top graph refers to Observed and Predicted observations from the Neural Network model. Black dots refer to the observed values; Blue dots refer to the predicted values for the validation period; Green dots refer to the predicted values for the undamaged test period; Red dots refer to the predicted values for damage period. Bottom graph represents the difference between the Observed and Predicted observations

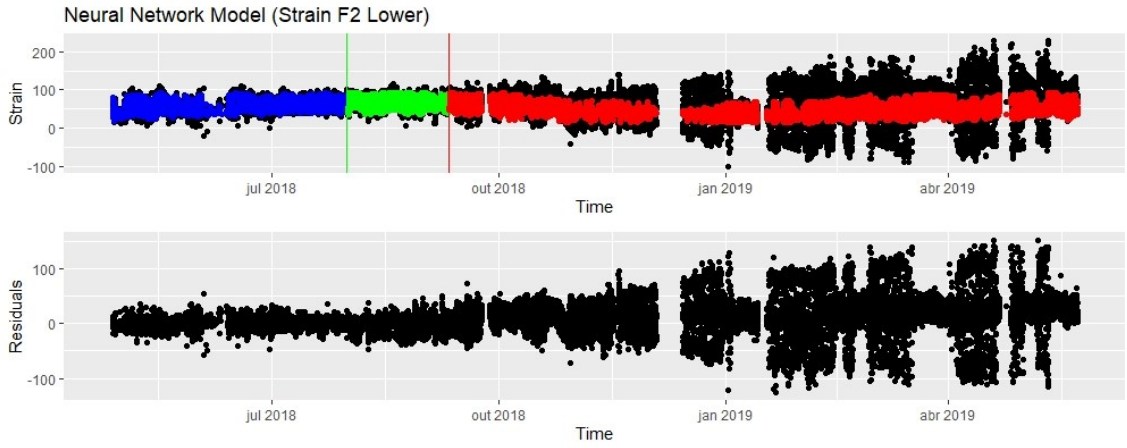

Figure 18: Top graph refers to Observed and Predicted observations from the Neural Network model. Black dots refer to the observed values; Blue dots refer to the predicted values for the validation period; Green dots refer to the predicted values for the undamaged test period; Red dots refer to the predicted values for damage period. Bottom graph represents the difference between the Observed and Predicted observations

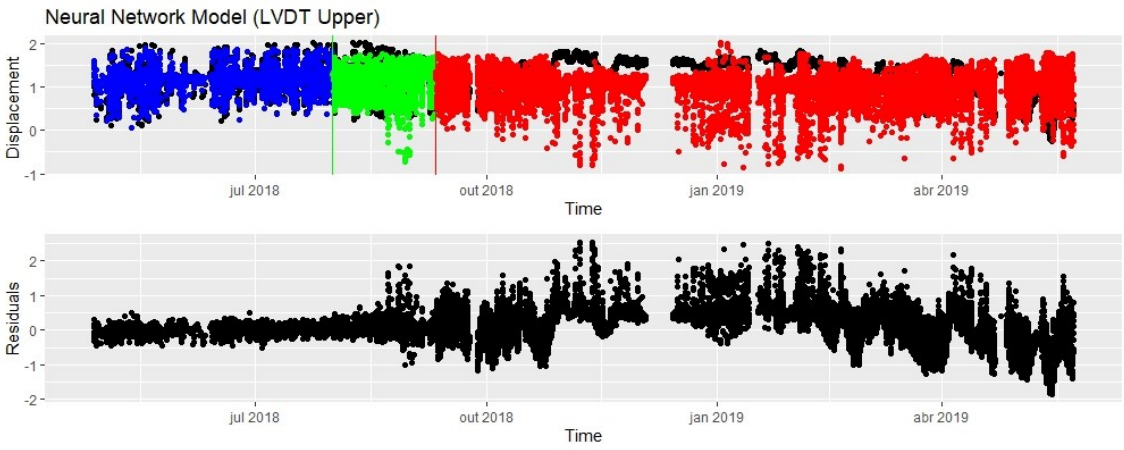

Figure 19: Top graph refers to Observed and Predicted observations from the Neural Network model. Black dots refer to the observed values; Blue dots refer to the predicted values for the validation period; Green dots refer to the predicted values for the undamaged test period; Red dots refer to the predicted values for damage period. Bottom graph represents the difference between the Observed and Predicted observations

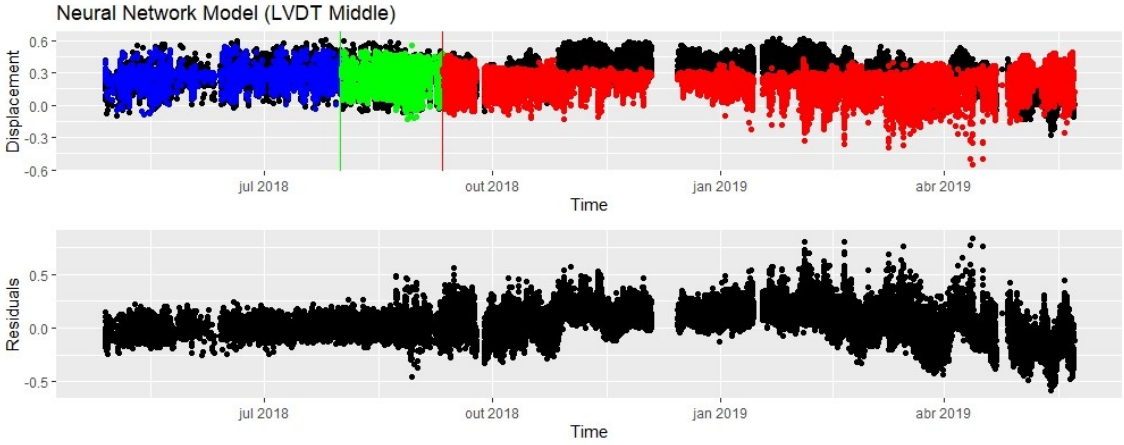

Figure 20: Top graph refers to Observed and Predicted observations from the Neural Network model. Black dots refer to the observed values; Blue dots refer to the predicted values for the validation period; Green dots refer to the predicted values for the undamaged test period; Red dots refer to the predicted values for damage period. Bottom graph represents the difference between the Observed and Predicted observations

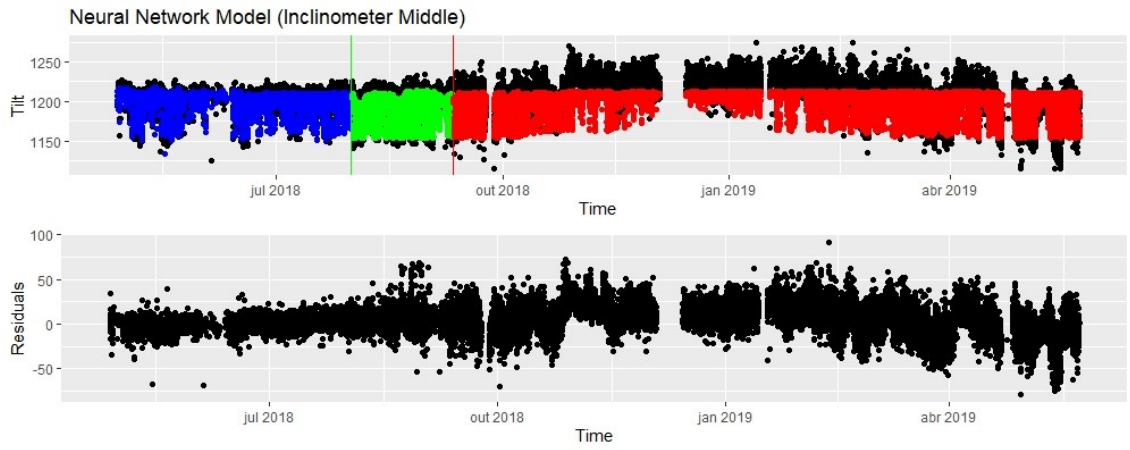

Figure 21: Top graph refers to Observed and Predicted observations from the Neural Network model. Black dots refer to the observed values; Blue dots refer to the predicted values for the validation period; Green dots refer to the predicted values for the undamaged test period; Red dots refer to the predicted values for damage period. Bottom graph represents the difference between the Observed and Predicted observations

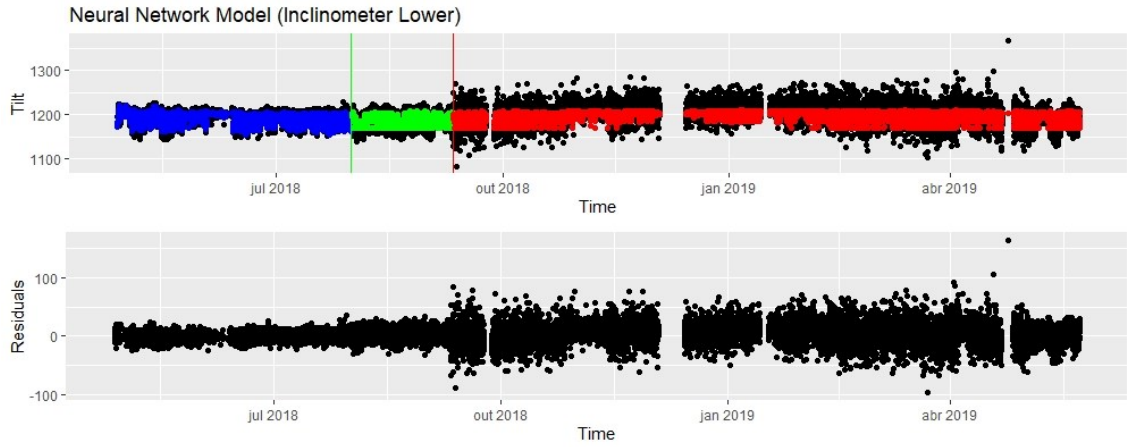

Figure 22: Top graph refers to Observed and Predicted observations from the Neural Network model. Black dots refer to the observed values; Blue dots refer to the predicted values for the validation period; Green dots refer to the predicted values for the undamaged test period; Red dots refer to the predicted values for damage period. Bottom graph represents the difference between the Observed and Predicted observations