University of Padova
department of mathematics

master thesis in
Data Science

Modeling the number of people in Venice during COVID-19: a State Space approach

Supervisor Prof. Bruno Scarpa
Department of Statistical Sciences

Master Candidate Ian Solagna
ID 2053345

Academic Year 2022/2023

Contents

Introduction

COVID-19 pandemic, caused by the novel coronavirus SARS-CoV-2, has had a strong impact on the world since its emergence in late 2019. This global health crisis has not only affected public health but has also disrupted daily life in unprecedented ways. The aim of this work is to assess and analyse COVID-19's impact on social interactions by focusing on the evolution of the number of people and their distribution at *Campo San Felice* in Venice. To accomplish this, a neural network architecture is employed to extract relevant information from daily photographs captured at the *Campo*. Subsequently, the extracted data are analyzed using *State Space Methods*, specifically *Dynamic Linear Models*, to gain insights into its progression and changes over time. State space methods are powerful tools used in time series analysis and forecasting that provide a flexible framework for modeling complex systems, accommodating both trend and seasonality components, as well as incorporating external factors and regressors.
The thesis is structured into four distinct chapters, each focusing on a specific aspect of the task.
The first chapter presents the original dataset containing the photographs, and provides an overview of the methodology employed to extract the relevant data from these images. Specifically, *You Only Look Once V7*, a neural network for object detection is introduced and described along its results on the pictures. Moreover, an explanatory data analysis is performed on the data extracted by the network.
The second chapter introduces *Dynamic Linear Models* and discusses the theoretical foundations of these models. Additionally, the theoretical aspects of *Prophet*, a time series forecasting tool developed by Facebook, are discussed. The objective is to present two different methodologies for the sake of comparison.
The third chapter presents an analysis of the number of people depicted in each photograph using both the Dynamic Linear Model and Prophet. A comparison between the two methods is conducted, and the results are discussed in detail.
The fourth chapter focuses on the analysis of the clusters observed in the photographs.

Density-based spatial clustering of applications with noise algorithm is introduced and applied to the images. Furthermore, the application of *Dynamic Linear Models* to different cluster characteristics provides insights into the evolution of these clusters during the COVID-19 pandemic.

Chapter 1

Dataset Description and Creation

1.1 The Dataset

The dataset consists of photos taken at *Campo San Felice* in Venice from the 8th of March 2020 to the 4th of March 2023. There could be multiple instances where several pictures were taken on the same day but at different times. On the other hand, some days in the dataset may not have any photos captured. For this work, only the period between the 8th of March 2020 and the 29th of September 2022 was considered. This is because the state of emergency for the COVID-19 pandemic officially ended on the 31st of March 2022, therefore five months seemed adequate to analyse the post-emergency situation, which was very stable and did not lead to any other significant protective measure from the government. As a result, the resulting dataset consists of 1421 photos which can be divided in three main categories: the pictures taken in the morning, the ones taken at noon and the ones taken in the evening. Figure 1.1 contains an example.

FIGURE 1.1: An example of a photo in the dataset.

The aim of the study is to extract the number of people and their position in the square and the road behind it. The people in the far back, however, are too small to be recognized in a consistent and robust way: to solve this problem each photo was cropped so that the road reaches the paper shop on the left of it (Fig. 1.2). This was done using *Adobe Lightroom*, a powerful photo editing software that has become a staple in the world of digital photography. After this step, every photo was manually checked to see if the result was satisfactory. The pictures underwent only this editing step, and the entire study was performed using this modified dataset. No additional pre-processing was conducted.

FIGURE 1.2: Pre-processed image.

1.2 Number of People and Position extraction

In recent years, computer vision technology has made tremendous strides in recognizing and detecting objects in images and videos. One of the most exciting and promising applications of this technology is the ability to identify and localize people in photos and videos. To achieve this, the go-to approach is to use deep learning models, such as *Convolutional Neural Networks*, that can learn to recognize and classify objects based on their visual features.

In this context, and since it was not the focus of this thesis, instead of building a custom model, a decision was made to choose a pre-trained network: YOLOv7, a state-of-the-art neural network model in object detection introduced in Wang et al. (2022). YOLOv7 is an advanced version of the *You Only Look Once* (YOLO) (Redmon et al., 2015) family of models that has achieved remarkable results in various object detection tasks, including people detection. By using it, we aim to accurately count the number of people and determine their positions in the photo.

1.2.1 YOLOv7 Architecture

First of all, the YOLO models are fully *Convolutional Neural Networks* which use a one-stage approach to object detection. This means that they use a single pass of the input image to make predictions about the presence and location of objects in the image. This makes them extremely efficient compared to other models which use a two-stage approach. Notice that YOLO is not a single architecture but a flexible research framework written in low-level languages. The framework has three main components: the head, neck, and backbone (Fig. 1.3) . Different sets of components and architecture are associated with the above-mentioned three components giving rise to different YOLO versions.

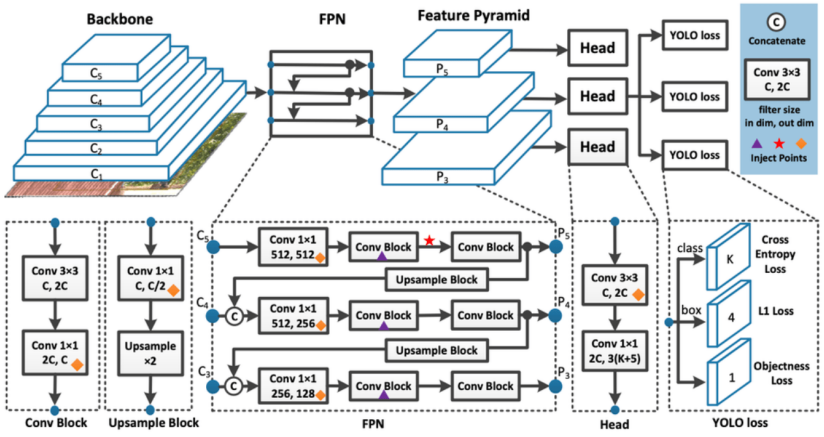

FIGURE 1.3: YOLO family architecture example composed of three main blocks: backbone, neck and head, Long et al. (2020)

Specifically, for YOLOv7 we have:

- Backbone network: The backbone network of YOLOv7 is based on CSPDarkNet, which is a modified version of the DarkNet architecture introduced in Bochkovskiy et al. (2020). This backbone network is used to extract features from the input image and generate a feature map that is then used for object detection.

- Neck network: YOLOv7 also includes a neck network, which is designed to refine the feature map and enhance the model's ability to detect small objects. The neck network consists of several convolutional layers and spatial pyramid pooling (SPP) modules.

- Detection heads: YOLOv7 includes three detection heads, which are responsible for detecting objects at different scales. These heads use anchor boxes, a set of predefined bounding boxes of specific height and width, to predict the final

bounding boxes, confidence scores, and class probabilities of the objects in the image.

Furthermore, the main new features added by this version of YOLO are:

- *Extended Efficient Layer Aggregation Networks* (E-ELAN): E-ELAN is what drives the computational block in the YOLOv7 backbone. It takes inspiration from previous research on network efficiency, such as Dollár et al. (2021), Lee et al. (2019), Ma et al. (2018), Wang et al. (2020). It has been designed by analyzing Memory-access cost, I/O channel ratio, Element wise operation, Activations and Gradient path. What it does is to enhance the learning capability of the network without changing the orginal gradient path.

- *Model Scaling*: Different applications require different models. Model scaling is performed to suit these requirements and make it fit in various computing devices. Model scaling is not something new, in YOLOv7 however, a compound scaling approach is used (Fig. 1.4). By doing this, the model can mantain the properties that the model had at the initial design and mantains the optimal structure.

- *Trainable Bag-of-Freebies*: Bag of Freebies are methods that increase the performance of a model without increasing the training cost. The ones introduced in YOLOv7 are the following: *Re-parametrized Convolution* and *Deep Supervision*. The former is a different type of convolutional block that the YOLOv7 researchers studied to find the best way to combine it with other networks. The latter is a technique that is often used in training deep networks. Its central concept is to add an extra auxiliary head in the middle layers of the network, and the shallow network weights with assistant loss as the guide. In YOLOv7, the head responsible for final output is called the Lead Head. The head used to assist training in the middle layers is called the Auxiliary Head. With the help of an assistant loss, the weights of the auxiliary heads are updated and the model learns better.

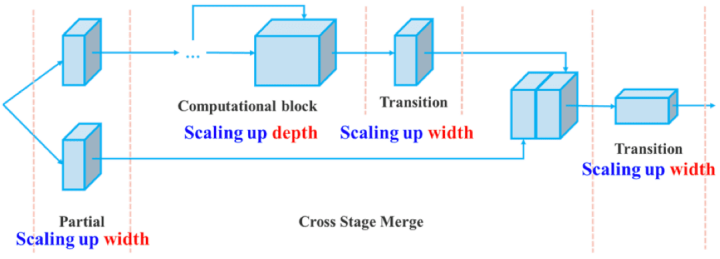

FIGURE 1.4: Compound scaling up depth and width, Wang et al. (2022).

1.2.2 Results of YOLOv7

YOLOv7 was trained on several large-scale object detection datasets, including COCO (Common Objects in Context) and Open Images. The COCO dataset contains over 330,000 images with more than 2.5 million labeled object instances across 80 different object categories. The Open Images dataset is even larger, with over 1.5 million images and more than 12 million labeled object instances across more than 600 categories. Additionally, the YOLOv7 authors also used various data augmentation techniques during training, such as random cropping, flipping, and color jittering, to further improve the robustness of the model. Unfortunately, when applied to the photos of our dataset the result are not very promising, as it can be seen in Fig. 1.5.

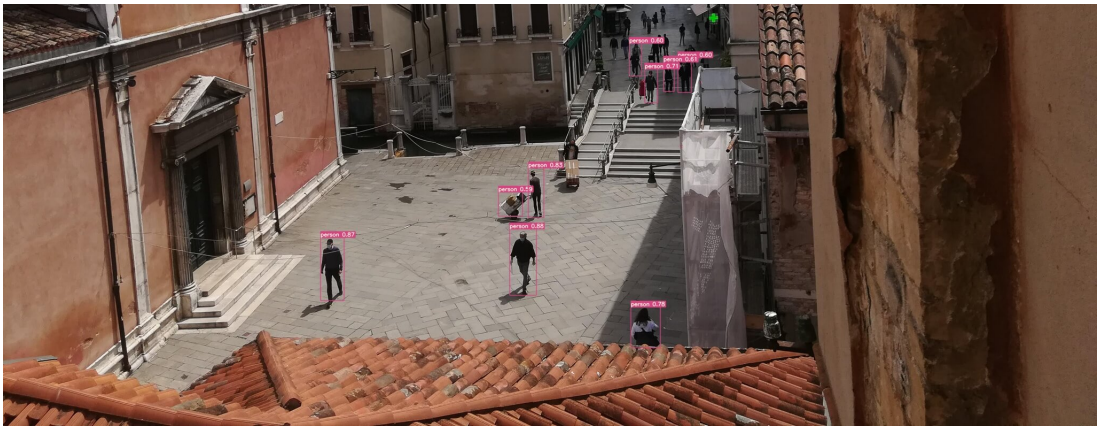

FIGURE 1.5: Results obtained by using the standard YOLOv7 weights

People on the main square and on the bridge are easily spotted, but the moment we move further away on the road the network is no longer able to recognize the people in the picture. Additionally, the model exhibits some errors, such as confusing the handcart positioned at the center of the *Campo* with a person. Luckily, YOLOv7 researchers made it possible to train the model on custom data to improve the results on specific tasks.

1.2.3 Custom Data

The photos after the 29th of September 2022 were not included in the final dataset, therefore we can use them to train YOLOv7 to recognize the people walking in *Campo San Felice*. Since all photos are very similar, only 146 were randomly picked and hand-labeled through *Roboflow*, a platform that provides tools for data preprocessing and management for computer vision projects like object detection, image classification,

and segmentation. After training the Neural Network on this custom data the accuracy in the results improved dramatically (Fig. 1.6)

FIGURE 1.6: Results obtained by training YOLOv7 on custom data.

The final results are not perfect in all photos: sometimes when the road is too crowded the model misses some people or over-estimates the number of them. On average, however, the accuracy achieved is remarkable, making YOLOv7 very consistent in locating humans in our photos. Notice that this custom version of the model is highly specific. Attempting to use it for recognizing people in pictures taken in different settings may result in poor performance of the model. For every photo, YOLOv7 produces a file in the format of a *.txt* file. Each row of this file represents an object detected in the photo and contains a set of five numerical values:

- The first number represents the class to which the object belongs to. In this case, the model only looks for people so the class is always 0.
- The second and third number represent the x and y coordinates of the center of the bounding box.
- The last two numbers represent the height and the width of the bounding box.

With this information it is now possible to build two datasets: one containing the number of people in each photo, and the second one including the individuals' coordinates within each picture.

1.3 Final Dataset

The date and time of each photo was extracted from their filename, for example, from *IMG_20200514_170022.jpg* we know that the photo was in year 2020, month 05, day 14

at 17:00:22. Furthermore, using open data found on the internet, new covariates that may affect the number of people in the streets of Venice were added to the two datasets, which, in their final version, contain:

- **ID**: an integer that symbolizes the photo to which the observation belongs to. This is particularly useful for the dataset containing the people's location since multiple observations may belong to the same picture. *1* is the 8th of March 2020 and *1421* is the 29th of September.
- **date**: The date of the observation.
- **time**: The time of the day when the photo was taken. *Morning* if captured from 5 AM to 10 AM, *Noon* if taken from 11 AM to 3 PM, and *Evening* for the rest of the day.
- **zona**: qualitative variable representing the government measures in act in Venice when the pictures were taken (data from *www.regione.veneto.it*). Italy implemented various restrictions throughout the pandemic, and each level of this variable represents the specific measures that were in effect during different periods of the year, specifically:
  - *lockdown*: During the lockdown, individuals were only permitted to leave their homes for work or essential grocery shopping. Restaurants and cinemas were closed and masks were mandatory to wear when walking outside. It covers the period between the 8th of March 2020 and the 3rd of May 2020.
  - *phase2*: During phase two, people were allowed to meet close relatives (the so called *congiunti*). Masks were mandatory and restaurants re-opened as delivery-only. It covers the period between the 4th of May 2020 and 14th of June, 2020.
  - *phase3*: In phase three, people could move freely as long as they kept their mask on. Outdoor activities were allowed. It covers the period between the 15th of June, 2020 and the 7th of October, 2020

  Form this point on, the *greenpass* was introduced and made mandatory for every indoor activity. Moreover, the concept of coloured zones with different restrictions was introduced, specifically:
  - *white*: the weakest color, people could move freely. Masks were not mandatory outdoor and non-essential activities such as cinemas and restaurants were open.

  - *yellow*: the same as the white zone but masks were mandatory outdoor too.
  - *orange*: the same as the previous colours but people could move freely only with the *super greenpass*.
  - *red*: people could not move freely. Restaurants were open just for delivery and all non-essential activities were closed.
  - *no-emergency*: all restrictions were removed.

- **mean_temperature**: the average temperature in Venice at the time the photo was captured (data from *www.3bmeteo.com*).
- **weekend_festive**: A binary value that indicates whether the day in Venice is a holiday or not (data from *www.scoprivenezia.com*). The value *1* denotes a festive day, while *0* represents a non-festive day.
- **rain**: the quantity of precipitation (in millimeters) in Venice on that particular day (data from *www.3bmeteo.com*).
- **season**: a categorical variable that indicates the season when the picture was captured. This variable can have four possible values: *Spring*, *Summer*, *Autumn*, and *Winter*, depending on the time of the year when the photo was taken.
- **number**: the number of people in the photo.
- **x** and **y**: the location of each person in the picture can be represented by two variables representing the two coordinates. The x coordinate ranges from 0 to 2048 to match the size of the photo in pixels, while the y coordinate ranges from 0 to 800 for the same reason.

1.4 Exploratory Data Analysis

In this section, we will conduct an Exploratory Data Analysis (EDA) to gain insights and understanding of the data. The primary focus of this EDA will be on the variable capturing the number of people recognized by YOLOv7 in each photo. By exploring this variable, we can gain insights into the social behavior and movement of people in Venice, as well as the impact of external factors such as weather and government measures on their mobility.

Given that the number of individuals present in photographs is the main focus of this thesis, and it will also serve as the response variable in chapter 3, it is useful to carry out an initial investigation to examine how this variable is distributed (Fig. 1.7).

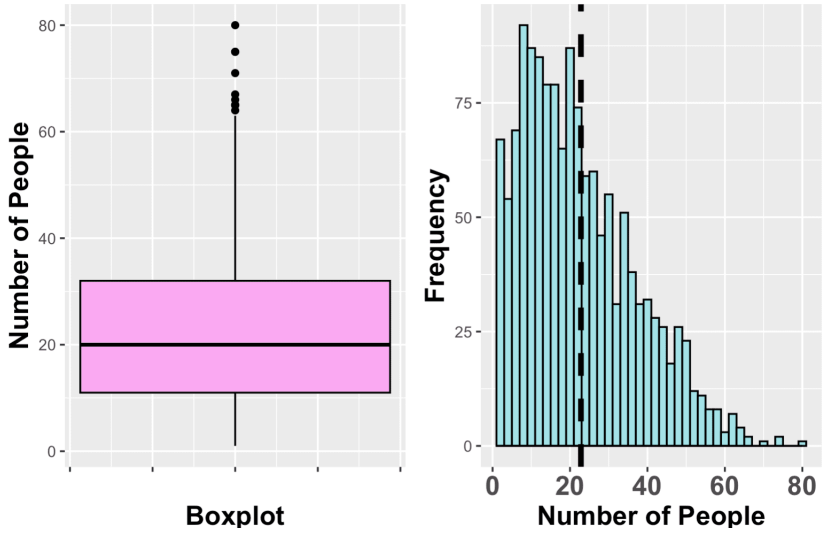

FIGURE 1.7: Distribution of the number of people

The graphical representations reveal that the distribution of the number of people in the photographs is right-skewed, with an asymmetrical shape. In fact, a lot of observations are very close to zero, but this is expected since many photos were taken early in the morning when there are not many people on the streets. Furthermore, it is worth noting that the median and mean (dashed line) values are quite similar and both hover around 20. Additionally, around 75% of the observations fall below 35 individuals, with the peak frequency occurring around 10 people.

Moving forward, an important aspect to consider is the effect of various predictors or independent variables on the response. In order to explore these relationships, the first step is to examine how the number of people in photographs is distributed based the time of the day and the season when the photographs were captured (Fig. 1.8).

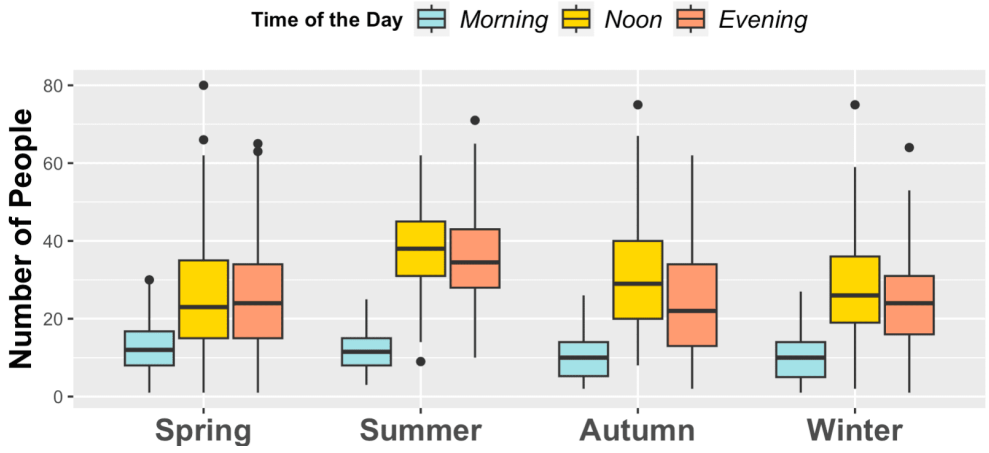

FIGURE 1.8: Boxplots of the number of people at different times of the day based on season.

Except summer, for which the number of people is, on average, slightly higher, there is little difference in the distributions across seasons. Specifically, the boxplots are very similar and they show a consistent pattern. However, there is a significant variation in the number of people at different times of the day. Typically, there are fewer people in the morning, and the number increases as the day progresses, with the peak usually observed at noon.

The next plot is useful to observe how the number of people at different times of the day changes over time in the considered period, to determine if there are any trends or patterns in the data (Fig. 1.9).

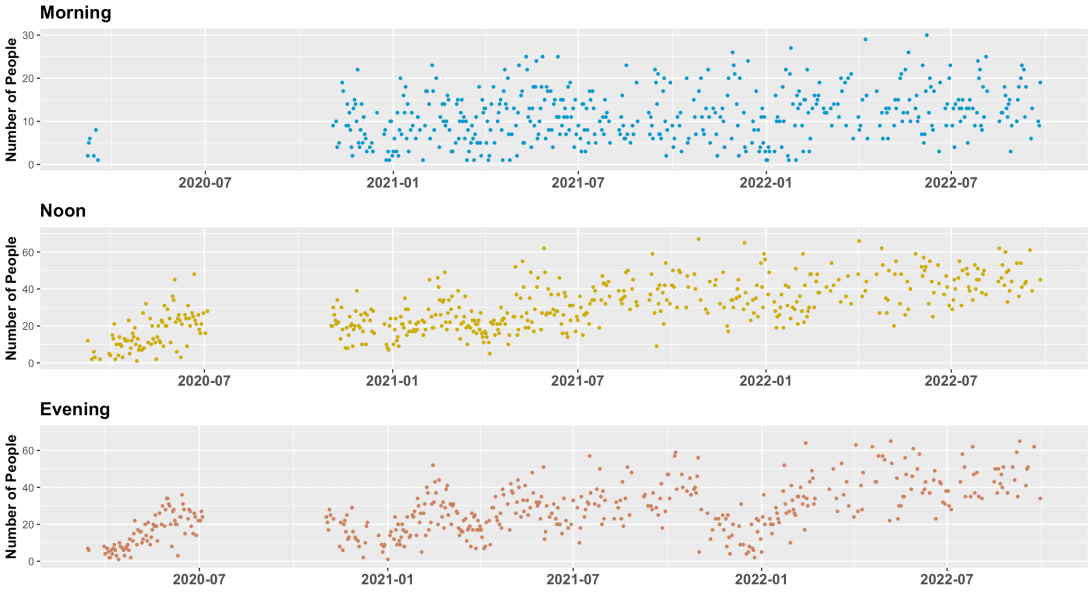

FIGURE 1.9: Time series of the number of people in the morning, at noon and in the evening.

From these plots it is clear that there are several gaps in the series, meaning that the number of missing values is quite high. Regardless, it is evident that all three time series exhibit an upward trend, with the trend being more prominent in the *Noon* and *Evening* plots. The reason behind this pattern is that the earliest observations were taken during the lockdown period, and as time progressed, the restrictions gradually eased, with some setbacks occurring along the way. For example, we can see that in December 2020 and January 2021 there is a considerable drop in the number of individuals captured in the photos. This is because, during this period, the city of Venice transitioned from *zona gialla* to *zona rossa*, meaning that individuals were only allowed to leave their homes for essential reasons such as grocery shopping or visiting close family members.

Related to these setbacks, another significant aspect to consider is the influence of the protective measures implemented by the Italian Government. Venice has experienced

several periods when the protective measures ranged from relaxed to strict. Therefore, the impact that they had on the number of people wandering around Venice should be substantial.

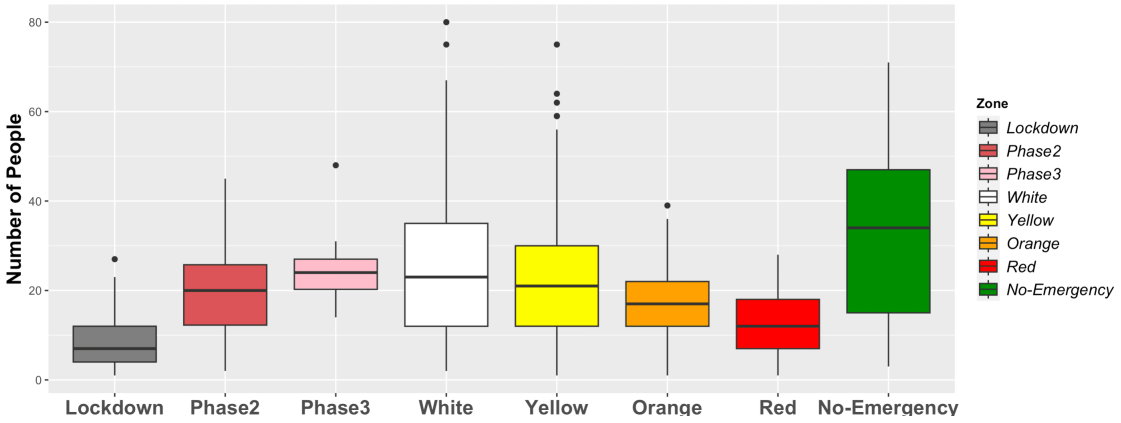

FIGURE 1.10: Distribution of the Number of People based on the government measures.

It is clear from Figure 1.10 that the distribution of the number of people detected in the photos is related to the different levels of the *zona* variable. This relationship is evident in the fact that, on average, the highest number of people in the photos is observed in the zones with weak restrictions, such as *white*, *phase3*, and *no_emergency*. Conversely, the zones with stricter restrictions, such as *red* and *lockdown*, have the lowest average number of people detected.

During the period of *lockdown* and when restrictions were particularly stringent, individuals were only permitted to leave their homes for essential tasks such as purchasing groceries or going to work. Therefore, two boxplots were constructed to evaluate the effects of working days in contrast to weekends and holidays (Fig. 1.11).

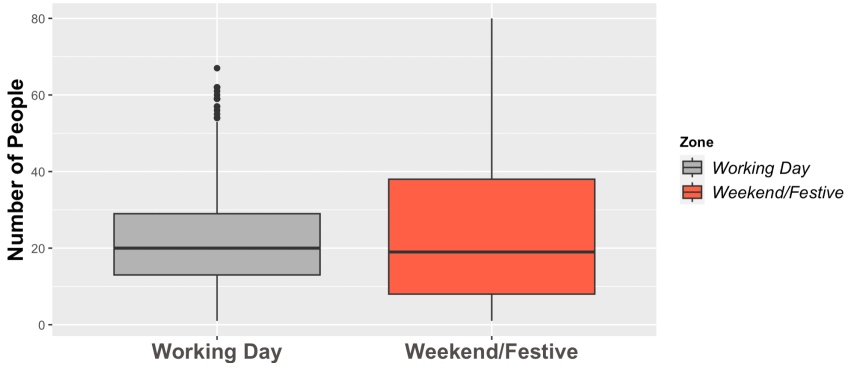

FIGURE 1.11: Distribution of the Number of People based on weekends and festive days.

The medians of the two categories, i.e., working days and holidays, are quite similar. However, there is a noticeable difference in the third quartile between the two groups, where the third quartile during holidays is substantially higher than that of working days. Based on these observations, we can conclude that during holidays, there is a higher probability of registering a significantly higher number of people as compared to working days.

Finally the last two variables that are considered in this analysis are those regarding the weather. Considering that Venice is a popular tourist destination, a pleasant and warm day would likely attract more people to the city, especially in the summer when the restrictions in Venice were not very strict. To conduct this analysis (Fig. 1.12), days with rainfall levels lower than 5 mm were classified as "No-Rain". Additionally, the *mean_temperature* variable was categorized into three groups: *Cold* for temperatures below 15° Celsius, *Hot* for temperatures ranging from 16° to 25° Celsius, and *Very-Hot* for temperatures exceeding 25° Celsius. These categorizations were made to simplify the data and to better understand how weather conditions affect the number of people in *Campo San Felice*.

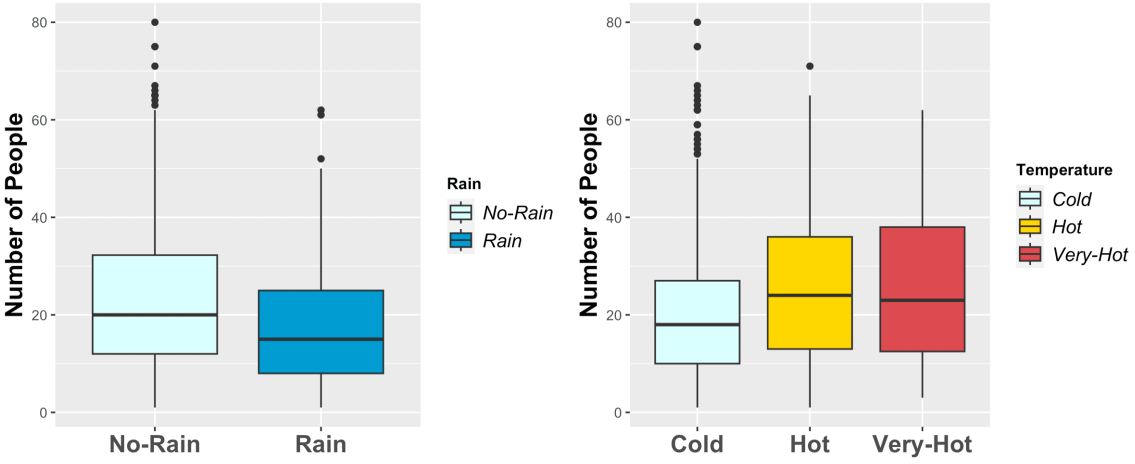

FIGURE 1.12: Distribution of the Number of People based on the weather conditions

Clearly, the weather conditions have a significant impact on the number of people in the square. Days with no precipitation tend to attract a larger number of people, while the same trend is observed for warm and hot days. The box plots for warm and hot days reveal higher median values as well as first and third quartiles, as compared to the box plot for cold days. This suggests that people are more likely to go out to the streets of Venice when temperatures are favorable. Conversely, lower temperatures may deter people from visiting the area, resulting in a lower number of visitors.

Regarding the variables x and y, which were introduced in the previous section, they will not be considered at this stage. Instead, they are the focus of chapter 4, which is completely dedicated to analysis of people's clusters based on their location in *Campo San Felice*.

Chapter 2

Dynamic Linear Models

Dynamic linear models (DLMs) are a popular tool used to model time series data and make forecasts. These models capture the complex dynamics and trends present in time series data by combining linear regression with state space models. This chapter will delve into the theory behind dynamic linear models and explore how they can be used for time series forecasting.

In addition to discussing the basics of DLMs, to asses their effectiveness, this chapter will also introduce another popular forecast model: *Prophet*, created by *Meta* (Taylor & Letham, 2018), which will be compared to DLMs in the next chapter.

2.1 Introduction

In this chapter Dynamic Linear Models are presented as a particular state space model with assumptions of being linear and gaussian. In recent years state space models have gained a lot of popularity in time-series analysis; Remarkably thanks to the work of Petris et al. (2009), Durbin & Koopman (2012), Shumway et al. (2000), and the references therein. As it will be clear in the following sections, these type of models offer a natural interpretation of a time series as a result of several components, such as trend, seasonal and regressive components. Moreover, they have a strong probabilistic structure and they are very flexible: they can model univariate and multivariate time series, even when there is non-stationarity, missing data and irregular patterns. Because of this they seem well-suited for dealing with the time series analysed in this Thesis.

2.2 State Space Models

Consider a time series $Y_t, t = 1,2,...$ where Y_t is an observable random vector that can be univariate as well as multivariate, to generalize $Y_t = (Y_{1,t}, ..., Y_{m,t})'$. In order to predict the value Y_{t+1}, we need to specify a dependence structure among the Y_t's variables.

State space models assume that the time series Y_t is a noisy and incomplete function of an unobservable process called state process $(\theta_t, t = 0,1,2,...)$. The benefit of this is that θ_t, has a much simpler *Markovian* dynamics. Therefore, it is easy to assume that the observation Y_t only depends on the state θ_t at that specific time. To formalize:

- θ_t is a Markov chain; i.e, θ_t only depends on past observations through θ_{t-1} (Fig. 2.1). Because of this, the probability law of θ_t is specified by simply assigning the density $p_0(\theta_0)$ and the transition densities $p(\theta_t|\theta_{t-1})$.
- Y_t's are independent and Y_t depends on θ_t only. Therefore, the joint conditional density can be written as $\prod_{t=1}^{n} f(y_t|\theta_t)$, for $n \geq 1$.

These assumptions allow us to write the probability law of the joint process $((\theta_t, Y_t), t = 1,2,...)$, from which we can deduce all the dependences among variables. For any $n \geq 1$,

$$\begin{aligned}
&(\theta_0, \theta_1, \ldots, \theta_n, Y_1, \ldots, Y_n) \sim p_0(\theta_0) \prod_{t=1}^{n} p(\theta_t, Y_t \mid \theta_0, \theta_1, \ldots, \theta_{t-1}, Y_1, \ldots, Y_{t-1}) \\
=&p_0(\theta_0) \prod_{t=1}^{n} f(Y_t \mid \theta_0, \ldots, \theta_t, Y_1, \ldots, Y_{t-1})\, p(\theta_t \mid \theta_0, \ldots, \theta_{t-1}, Y_1, \ldots, Y_{t-1}) \qquad (2.1) \\
=&p_0(\theta_0) \prod_{t=1}^{n} f(Y_t \mid \theta_t)\, p(\theta_t \mid \theta_{t-1}).
\end{aligned}$$

The process is clearly Markovian and the density of $Y_1, ..Y_n$ can be obtained by integrating out all the θ variables. Moreover, Y_t is conditionally independent from the past observations given the value of θ_t, which intuitively, gives θ_t the role of containing all the past history of the observable process. Moreover, state space models can be written in the form

$$\begin{aligned}
Y_t &= h_t(\theta_t, v_t) \\
\theta_t &= g_t(\theta_{t-1}, w_t)
\end{aligned} \qquad (2.2)$$

with arbitrary functions g_t and h_t.

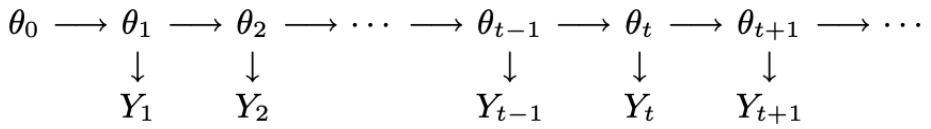

FIGURE 2.1: Dependancies of state space methods, Petris et al. (2009)

2.3 Dynamic Linear Models

Dynamic Linear Models are a class of state space models which are specified through the use of two equations:

$$\begin{aligned} Y_t &= F_t\theta_t + v_t, & v_t &\sim N_m(0, V_t) \\ \theta_t &= G_t\theta_{t-1} + w_t, & w_t &\sim N_p(0, W_t) \end{aligned} \tag{2.3}$$

where F_t and G_t are known matrices and v_t and w_t are two independent white noise sequences with mean zero V_t and W_t as covariance matrices. Of the two equations, the first one is known as *observation equation* while the second one is called *state equation*. Moreover, θ_0 is assumed to have a Gaussian distribution,

$$\theta_0 \sim N_p(m_0, C_0) \tag{2.4}$$

for some non-random vector m_0 and matrix C_0, and it is independent on v_t and w_t. Notice that this formulation satisfies the assumptions presented in the previous section with $Y_t|\theta_t \sim N(F_t\theta_t, V_t)$ and $\theta_t|\theta_{t-1} \sim N(G_t\theta_{t-1}, W_t)$. Furthermore, equation (2.3) is simply equation (2.2) with g_t and h_t linear.

2.3.1 Estimation

In this section, to present the estimation process for DLMs, $f(y_t, \theta_t)$ and $p(\theta_t|\theta_{t-1})$ are considered fully specified. Later in this chapter, these densities will depend on unknown parameters and their estimation will be presented.

In order to estimate the state vector, it is sufficient to compute the conditional densities $p(\theta_s|y_1, ..y_t)$. From this, three different problems can be distinguished:

- *Filtering*: when $s = t$
- *Smoothing*: when $s < t$
- *Forecasting*: when $s > t$

The first challenge involves receiving data in a sequential manner over time. In such situations, we require a method to estimate the current value of the state vector based on the available observations until that point, i.e. we compute $p(\theta_t|y_1, .., y_t)$. Conversely, *smoothing* is a form of retrospective analysis, where we estimate the state sequence at different time points from 1 to t, based on the data available from y_1 to y_t. The purpose of smoothing is to analyze the past behavior of the state vector to gain a better understanding of its functioning. Finally, forecasting is often the main task when dealing with time series. This involves computing $p(\theta_{t+1}|y_1, .., y_n)$, which is also referred to as one-step-ahead predictions. As we will explore further in this chapter, this calculation is dependent on the filtering density of θ_t. One can also be interested in computing the the state vector θ_{t+k} for some $k > 1$ to make k-steps-ahead forecast. Obviously, the higher the k the more uncertain the prediction based on $p(\theta_{t+k}|y_1, .., y_t)$ will be.

2.3.2 Filtering

Denote with $\mathcal{D}_t$ the information provided by the first t observations, $Y_1, ..., Y_t$, the markovian structure and the conditional independence of the observations allow us to compute the filtered and predictive densities by the means of a recursive algorithm. Starting from $\theta_0 \sim p_0(\theta_0) = p(\theta_0|\mathcal{D}_0)$, for $t = 1, 2, ...$:

- The one-step-ahead predictive density for the states can be computed from the filtered density $p(\theta_{t-1}|\mathcal{D}_{t-1})$ according to

$$p\left(\theta_t \mid \mathcal{D}_{t-1}\right) = \int p\left(\theta_t \mid \theta_{t-1}\right) p\left(\theta_{t-1} \mid \mathcal{D}_{t-1}\right) \mathrm{d}\nu\left(\theta_{t-1}\right) \tag{2.5}$$

- The one-step-ahead predicitive density for the observations can be computed from the predictive density for the states as

$$f\left(y_t \mid \mathcal{D}_{t-1}\right) = \int f\left(y_t \mid \theta_t\right) p\left(\theta_t \mid \mathcal{D}_{t-1}\right) \mathrm{d}\nu\left(\theta_t\right) \tag{2.6}$$

- The filtering density can be computed from the above densities as

$$p\left(\theta_t \mid \mathcal{D}_t\right) = \frac{f\left(y_t \mid \theta_t\right) p\left(\theta_t \mid \mathcal{D}_{t-1}\right)}{f\left(y_t \mid \mathcal{D}_{t-1}\right)} \tag{2.7}$$

To compute this quantities is tipically not an easy task, however in the case of DLMs these calculations simplify dramatically. Because of the assumption of normality, the random vector $(\theta_0, \theta_1, ..., \theta_t, Y_1, ..., Y_t)$ has a Gaussian distribution for any $t \geq 1$.

Therefore, the marginal and conditional distributions are also Gaussian and since all the relevant distributions are Gaussian it suffices to compute their means and covariances. In this case the solution of the filtering problem is given by the Kalman filter. In fact, for the DLM presented in equation (2.3), if

$$\theta_{t-1}|\mathcal{D}_{t-1} \sim N(m_{t-1}, C_{t-1}) \tag{2.8}$$

where $t \geq 1$, then the Kalman filter is defined as

- The one-step-ahead state predictive density of θ_t, given $\mathcal{D}_{t-1}$ is Gaussian, with parameters

$$\begin{aligned} a_t &= \mathrm{E}\left(\theta_t \mid \mathcal{D}_{t-1}\right) = G_t m_{t-1} \\ R_t &= \mathrm{Var}\left(\theta_t \mid \mathcal{D}_{t-1}\right) = G_t C_{t-1} G_t' + W_t \end{aligned} \tag{2.9}$$

- The one-step-ahead predictive density of Y_t given $\mathcal{D}_{t-1}$ is Gaussian, with parameters

$$\begin{aligned} f_t &= \mathrm{E}\left(Y_t \mid \mathcal{D}_{t-1}\right) = F_t a_t \\ Q_t &= \mathrm{Var}\left(Y_t \mid \mathcal{D}_{t-1}\right) = F_t R_t F_t' + V_t \end{aligned} \tag{2.10}$$

- The filtering density of θ_t given $\mathcal{D}_t$ is Gaussian, with

$$\begin{aligned} m_t &= \mathrm{E}\left(\theta_t \mid \mathcal{D}_t\right) = a_t + R_t F_t' Q_t^{-1} e_t \\ C_t &= \mathrm{Var}\left(\theta_t \mid \mathcal{D}_t\right) = R_t - R_t F_t' Q_t^{-1} F_t R_t \end{aligned} \tag{2.11}$$

  where $e_t = Y_t - f_t$ is the forecast error.

By using the Kalman filter, it becomes possible to recursively compute the predictive and filtering densities. This process starts from $\theta_0|\mathcal{D}_0 \sim N(m_0, C_0)$, and then computes $p(\theta_1|\mathcal{D}_1)$. This recursive process continues as new information becomes available over time. At this stage, we have computed the conditional density $\theta_t|\mathcal{D}_t$, however what we are often interested in, is a point estimate. One can use bayes theory to estimate θ_t based on the information in $\mathcal{D}_t$, using a quadratic loss function $L(\theta_t, a) = (\theta_t - a)'H(\theta_t - a)$, results in the conditional expected value $m_t = E(\theta_t|\mathcal{D}_t)$. Additionally, m_t holds an intuitive interpretation, in fact, filter mean equals to the prediction mean a_t plus a correction depending on how much the new observation differs from its prediction. The weight of the correction is given by the *gain matrix*

$$K_t = R_t F_t' Q_t^{-1} \tag{2.12}$$

In other words, the weight of the current information Y_t depends on the observation covariance matrix V_t (through Q_t) and on $R_t = Var(\theta_t|\mathcal{D}_{t-1}) = G_t C_{t-1} G_t' + W_t$.

2.3.3 Smoothing

As already mentioned, it may happen that one has some observations Y_t collected over a certain period of time and wants to analyse the behaviour of the whole system. To accomplish this, a backward recursive algorithm is utilized to compute the conditional densities of $\theta_t|\mathcal{D}_t$, for $t < T$. This process begins with the filtering density $p(\theta_T|\mathcal{D}_t)$ and involves estimating the entire history of the system's states by working backwards through time.

- Conditional on $\mathcal{D}_t$, the state sequence $(\theta_0, ..., \theta_T)$ has backward transition probabilities given by

$$p\left(\theta_t \mid \theta_{t+1}, \mathcal{D}_T\right) = \frac{p\left(\theta_{t+1} \mid \theta_t\right) p\left(\theta_t \mid \mathcal{D}_t\right)}{p\left(\theta_{t+1} \mid \mathcal{D}_t\right)} \tag{2.13}$$

- The smoothing densities of θ_t given $\mathcal{D}_t$ can be computed according to

$$p\left(\theta_t \mid \mathcal{D}_T\right) = p\left(\theta_t \mid \mathcal{D}_t\right) \int \frac{p\left(\theta_{t+1} \mid \theta_t\right)}{p\left(\theta_{t+1} \mid \mathcal{D}_t\right)} p\left(\theta_{t+1} \mid \mathcal{D}_T\right) d\mu\left(\theta_{t+1}\right) \tag{2.14}$$

For the Dynamic Linear Model presented in this chapter, these calculations reduce to the following, if

$$\theta_{t+1}|\mathcal{D}_t \sim N(s_{t+1}, S_{t+1}) \tag{2.15}$$

then $\theta_t|\mathcal{D}_t \sim N(s_t, S_t)$, where:

$$\begin{aligned} s_t &= m_t + C_t G_{t+1}' R_{t+1}^{-1} \left(s_{t+1} - a_{t+1}\right) \\ S_t &= C_t + C_t G_{t+1}' R_{t+1}^{-1} \left(S_{t+1} - R_{t+1}\right) R_{t+1}^{-1} G_{t+1} C_t \end{aligned} \tag{2.16}$$

The Kalman smoother enables the computation of densities of $\theta_t|\mathcal{D}_t$ by starting at $t = T - 1$, where $\theta_T|\mathcal{D}_t \sim N(s_T = m_T, S_T = C_T)$. From there, the process continues backwards to compute the densities of $\theta_t|\mathcal{D}_t$ for $t = T - 2, t = T - 3$, and so on.

2.3.4 Forecasting

In various scenarios, there is a need to look beyond just one step ahead and predict future states. In such cases, the filtering distribution at time t serves as an initial distribution for the model's future evolution. To be more specific, the joint distribution of present

and future states $(\theta_{t+k})_{k\geq 0}$ and future observations $(Y_{t+k})_{k\geq 1}$ can be modeled through a Dynamic Linear Model with an initial distribution of $p(\theta_t|\mathcal{D}_t)$. Since θ_t contains all the information up to time t, one simply starts from there and computes the filtering recursions.

By setting $E(\theta_t|\mathcal{D}_t) = m_t$ and $Var(\theta_t|\mathcal{D}_t) = C_t$, the recursive formulae are the following:

- The distribution of θ_{t+k} given $\mathcal{D}_t$ is Gaussian, with

$$\begin{aligned} E(\theta_{t+k}|\mathcal{D}_t) &= G_{t+k}E(\theta_{t+k-1}|\mathcal{D}_t) \\ Var(\theta_{t+k}|\mathcal{D}_t) &= G_{t+k}Var(\theta_{t+k-1}|\mathcal{D}_t)G'_{t+k} + W_{t+k} \end{aligned} \tag{2.17}$$

- The distribution of Y_{t+k} given $\mathcal{D}_t$ is Gaussian, with

$$\begin{aligned} E(Y_{t+k}|\mathcal{D}_t) &= F_{t+k}E(\theta_{t+k}|\mathcal{D}_t) \\ Var(Y_{t+k}|\mathcal{D}_t) &= F_{t+k}Var(\theta_{t+k}|\mathcal{D}_t)F'_{t+k} + V_t \end{aligned} \tag{2.18}$$

From equation (2.18) it is evident that the further into the future we predict, the variability increases. As a result, the predictions become more and more inaccurate since the available and valuable information is very far in the past.

2.3.5 Innovations

The Kalman filter provides the filtering estimate m_t by correcting the previous estimate m_{t-1} by a term which depends on the prediction error

$$e_t = Y_t - E(Y_t|\mathcal{D}_{t-1}) = Y_t - f_t \tag{2.19}$$

or, alternatively

$$\begin{aligned} e_t &= Y_t - F_t a_t = F_t\theta_t + v_t - F_t a_t \\ &= F_t\left(\theta_t - a_t\right) + v_t = F_t\left(\theta_t - G_t m_{t-1}\right) + v_t \end{aligned} \tag{2.20}$$

The sequence of prediction errors $(e_t, t \geq 1)$ has the following properties:

- The expected value of e_t is zero

- The random vector e_t is uncorrelated with any function of $Y_1, ..., Y_{t-1}$. Specifically, if $s < t$, then e_t and Y_s are uncorrelated. Let $Z = g(Y_1, ..., Y_{t-1})$, then

$$\begin{aligned} \mathrm{Cov}\,(e_t, Z) &= \mathrm{E}\,(e_t Z) = \mathrm{E}\,(\mathrm{E}\,(e_t Z \mid \mathcal{D}_{t-1})) \\ &= \mathrm{E}\,(\mathrm{E}\,(Y_t - f_t \mid \mathcal{D}_{t-1})\, Z) = 0 \end{aligned} \tag{2.21}$$

- for $s \neq t$, e_t and e_s are uncorrelated.
- e_t is a linear combination of $Y_1, ..., Y_{t-1}$
- $(e_t, t \geq 1)$ is a Gaussian process

$$e_t \sim N_m(0, Q_t), t = 1, 2, ... \tag{2.22}$$

  with Q_t as defined in equation 2.10

These errors are also called *innovations* because of the role they play in the model. In fact, Y_t can be thought as being the sum of a component which is predictable, f_t, and e_t which is independent from the past and contains new information about the process.

2.3.6 Model Checking

If the observations are univariate, then the standardized innovations, defined by $\tilde{e_t} = e_t/\sqrt{Q_t}$ follow a Gaussian white noise distribution. If the model is accurate, the sequence of standardized innovations $(\tilde{e_1}, ..., \tilde{e_t})$ should resemble a random sample of size t taken from a standard normal distribution. In the upcoming chapters, we will be evaluating the goodness of fit of the models we use by performing tests on the standardized innovations. These tests will serve two main purposes. Firstly, we will test the hypothesis that the sequence of standardized innovations is uncorrelated. Secondly, we will test whether the standardized innovations follow a standard normal distribution.

2.3.7 Trend

Within the framework of Dynamic Linear Models (DLMs), the trend component is regarded as the gradual and smooth evolution of the time series over time. Specifically, at a given time t, the expected behavior of the time series can be thought as the expected behavior of Y_{t+k} for $k \geq 1$, taking into account the information available up to time t. As a result, the expected trend can be represented by the function $f_t(k) = E(Y_{t+k}|\mathcal{D}_t)$.

A polynomial model of order n in DLMs is characterized by the presence of fixed matrices $F_t = F$ and $G_t = G$, known matrices V_t and W_t, and a forecast function

$$f_t(k) = E(Y_{t+k}|\mathcal{D}_t) = a_{t,0} + a_{t,1}k + ... + a_{t,n-1}k^{n-1} \tag{2.23}$$

where $a_{t,0}, ..., a_{t,n-1}$ are linear functions of $E(\theta_t|\mathcal{D}_t)$.

Thus the forecast function is of order $n-1$. Every function can be approximated by choosing n sufficiently large, but in most cases, $n = 1$ (random walk plus noise) and $n = 2$ (linear growth model) give the best results. For example, a *Linear Growth* model , i.e. a random walk plus noise model which includes a time-varying slope in the dynamics of the level, can be written as

$$\begin{aligned} Y_t &= \lambda_t + v_t & v_t &\sim N(0, V) \\ \lambda_t &= \lambda_{t-1} + \beta_{t-1} + w_{1,t} & w_{1,t} &\sim N(0, \sigma^2_{w_1}) \\ \beta_t &= \beta_{t-1} + w_{2,t} & w_{2,t} &\sim N(0, \sigma^2_{w_2}) \end{aligned} \tag{2.24}$$

with λ_t being the estimated local level and β_t the dynamic slope, which together form the state vector $\theta_t = (\lambda_t, \beta_t)'$. The forecast function is then defined as

$$f_t(k) = \hat{\lambda}_t + k\hat{\beta}_t \tag{2.25}$$

which is a linear function of k, so the *linear growth* model is a polynomial DLM of order 2.

2.3.8 Seasonality

Here, to model seasonality we present the *Seasonal Factor* model, however, a second formulation using a *Fourier-form* seasonal model can be found in Petris et al. (2009).

Assume the series has zero mean, or in other words, assume that it is purely seasonal. Seasonality can be modeled by introducing seasonal deviations from the zero mean. This can be achieved by incorporating different coefficients, denoted as α_i, where each coefficient represents a specific season.

For example, in the case of quarterly data, α_1 may represent the first quarter, α_2 the second one and so on. This specific model can be written as a *Dynamic Linear Model* as follows. Let $\theta_{t-1} = (\alpha_1, \alpha_4, \alpha_3, \alpha_2)^T$ where $t-1$ refers to the first quarter of the year and $F_t = F = (1, 0, 0, 0)$ then the observation equation has the form of $Y_{t-1} = F\theta_{t-1} + v_{t-1} = \alpha_1 + v_{t-1}$. Then to represent the next quarter, θ_{t-1} has to change to $\theta_t = (\alpha_2, \alpha_1, \alpha_4, \alpha_3)^T$, so that $Y_t = F\theta_t + v_t = \alpha_2 + v_t$. The required permutation can

be represented by

$$G = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.26}$$

Then the state equation is

$$\theta_t = G\theta_{t-1} + w_t = (\alpha_2, \alpha_1, \alpha_4, \alpha_3)^T + w_t \tag{2.27}$$

In general, a time series with period s can be modeled through a state vector θ_t of dimension s, by specifying a DLM with $F = (1, 0, ..., 0)$ and G, a sxs permutation matrix. In this case, some constraints have to be imposed for identifiability problems. One common choice is to make $\sum_{j=1}^{s} \alpha_j = 0$. The constraint also implies that there are only $s - 1$ seasonal factors which lead to a more parsimonious representation that uses an $s-1$-dimensional state vector. In general, if the series has period s, one can consider the $(s-1)$-dimensional state space, with $F = (1, 0, ..., 0)$ and

$$G = \begin{bmatrix} -1 & -1 & ... & -1 & -1 \\ 1 & 0 & ... & 0 & 0 \\ 0 & 1 & ... & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & ... & 1 & 0 \end{bmatrix} \tag{2.28}$$

2.3.9 Regression

Regressors can also be easily implemented in a DLM. Imagine for example that Y_t depends on another variable x_t according to the following relationship

$$Y_t = a_t x_t + v_t \qquad v_t \sim N(0, V_t) \tag{2.29}$$

Usually $\alpha_t = \alpha$ in a simple regression model. In the context of DLMs, however, we let α change over time introducing a state equation

$$\alpha_t = \alpha_{t-1} + w_t \qquad w_t \sim N(0, W_t) \tag{2.30}$$

In general the *Dynamic Liner Regression model* is described as

$$\begin{aligned} Y_t &= X_t\theta_t + v_t, \qquad v_t \sim N(0, V_t) \\ \theta_t &= G_t\theta_{t-1} + w_t, \quad w_t \sim N(0, W_t) \end{aligned} \tag{2.31}$$

where X_t is the vector of explanatory variables for period t which is assumed to be known. This way, DLMs techniques may be used to sequentially update the estimates of the parameters of a regression model as new information become available.

2.3.10 Maximum likelihood estimation

For this study, in order to estimate the unknown values of F_t, G_t, V_t, and W_t, which in the general case are unknown, *Maximum Likelihood Estimation* was used.

Suppose we have a collection of z random vectors, denoted by $Y_1, Y_2, \ldots, Y_z$. The distribution of these vectors is dependent on an unknown parameter, denoted by ϕ. We can express the joint density of the observations for a specific value of ϕ as $p(y_1, y_2, \ldots, y_z; \phi)$, which allows us to define the likelihood function as $L(\phi) = const \cdot p(y_1, y_2, \ldots, y_z; \phi)$.

For a Dynamic Linear Model, it is more convenient to express the joint density of the observations in a different form. Specifically, we can write it as a product of conditional densities, which correspond to the probability of observing each data point y_t given the information available up to time $t-1$ (denoted by $\mathcal{D}_{t-1}$). More formally, we can write

$$p(y_1, y_2, \ldots, y_z; \phi) = \prod_{t=1}^{z} p(y_t|\mathcal{D}_{t-1}; \phi) \tag{2.32}$$

where $p(y_t|\mathcal{D}_{t-1}; \phi)$ represents the conditional density of y_t given the data up to time $t-1$, and it assumes that the value of the unknown parameter is ϕ. As it was previously explained, we know that the terms contained in the conditional density are Gaussian densities with mean f_t and variance Q_t. As a result, the likelihood can be written as

$$l(\phi) = -\frac{1}{2}\sum_{t=1}^{z} \log|Q_t| - \frac{1}{2}\sum_{t=1}^{z}(y_t - f_t)'Q_t^{-1}(y_t - f_t) \tag{2.33}$$

This expression can be maximized

$$\hat{\phi} = \arg\max_{\phi} \; l(\phi) \tag{2.34}$$

Moreover, by denoting with H the Hessian matrix $-l(\phi)$, evaluated at $\phi = \hat{\phi}$, then H^{-1} provides an estimate of the variance of the maximum likelihood estimator.

2.4 Prophet

Prophet is formed by a decomposable time series model consisting primarily of three components

- A *trend* $g(t)$ which models non periodic changes in the time series.
- A *seasonality* $s(t)$ which represents periodic changes.
- A *holiday* effect $h(t)$ to represents the effects of holidays which occur on irregular basis.

It is worth noting that the function $h(t)$ can be extended to include other regressors. While these additional regressors are not included in the basic notation and formulation presented in this chapter, the model can be readily expanded to incorporate them.

This formulation is similar to a *Generalized Additive Model* (GAM), for specifications see Azzalini & Scarpa (2012). This makes it advantageous because it decomposes easily and accomodates new components as necessary. By doing this, the problem becomes a curve-fitting exercise. This means that the model does not explicitly account for temporal dependence structure in the data (time is given simply as a regressor), but there are also several perks:

- Flexibility: seasonality can have multiple periods and the analyst can make different assumptions about the trend.
- Measurements do not need to be regularly spaced.
- Fitting is very fast.
- The parameters are very easy to interpret.

2.4.1 Trend

In the original paper, two trend models were implemented, here, however, we will present only the one we used in the analysis, i.e. the *Piecewise Linear Trend with Changepoints*. This model is defined as

$$g(t) = (k + a(t)^T\delta)t + (m + a(t)^T\gamma) \tag{2.35}$$

The growth rate, denoted by k, represents the expected rate of increase or decrease in a time series over time. Additionally, there is an offset parameter m, and a vector

of changepoints δ. Changepoints are specific points in time where the growth rate is allowed to change, resulting in a non-constant growth rate that can capture a variety of time series patterns. The growth rate at any given time t is calculated by adding the base rate k to the sum of all the changepoints up to that point, $k + \sum_{j:t>s_j} \delta_j$. This is represented more clearly by defining $a(t) \in \{0,1\}^S$, such that

$$a_j(t) = \begin{cases} 1, & \text{if } t \geq s_j \\ 0, & \text{otherwise} \end{cases} \tag{2.36}$$

The rate at time t is then $k+a(t)^T\delta$, like in equation 2.35. Moreover, these changepoints, defined by s_j can be specified or may be automatically selected. To do so, one needs to use Bayesian Statistics and put a prior on their distribution, resulting in

$$\delta_j \sim Laplace(0, \tau) \tag{2.37}$$

where τ controls the flexibility of the model. Finally in formula 2.35 there is γ where γ_j is set to $-s_j\delta_j$ to make the function continuous.

2.4.2 Seasonality

To effectively model and forecast seasonality in time series, it is crucial to specify a periodic function of time. This periodic function results in a seasonality model that is capable of capturing the seasonal patterns within the data. In the case of Prophet, Fourier series are utilized to provide a flexible model of periodic effects. By specifying the regular period (P) of the time series, the seasonal effects can be smoothed using

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right) \tag{2.38}$$

This requires estimating $2N$ parameters $\beta = [a_1, b_1, ..., a_N, b_N]^T$. This is done by creating a matrix of seasonality vectors for each value of t. The component is then

$$s(t) = X(t)\beta \tag{2.39}$$

In *Prophet*'s generative model, $\beta \sim N(0, \sigma^2)$ to impose a smoothing prior on the seasonality.

2.4.3 Holidays

The idea of having a *Holiday* effect in the standard formulation of the model, come from the fact that holidays ususally provide large and predictable shocks to many time series. To include holidays in the model, it is assumed that their effects are independent, making it a straightforward process. For each holiday, denoted as i, the set of dates in the past and future related to that holiday is defined as $\mathcal{D}_i$. An indicator function is then added to represent whether time t falls during holiday i, and a corresponding parameter κ_i is assigned to represent the change in the forecast during that holiday. This approach is similar to how seasonality is modeled, in fact, we define

$$Z(t) = [1(t \in \mathcal{D}_1), ..., 1(t \in \mathcal{D}_L)] \tag{2.40}$$

and take

$$h(t) = Z(t)\kappa \tag{2.41}$$

with $\kappa \sim Normal(0, \nu^2)$

2.5 Optimization

Since both of the models presented use the same optimization algorithm (at least in the implementation that it was used in this thesis), in this section the *Limited-memory Broyden-Fletcher-Goldfarb-Shanno* (L-BFGS) algorithm (Liu & Nocedal (1989)), will be introduced and explained.

2.5.1 L-BFGS

In the BFGS algorithm, each step has the form

$$x_{k+1} = x_k - \alpha_k H_k \nabla f(k) \tag{2.42}$$

where x_k is the vector of variables at iteration k, f_k is the objective function, α_k is the step length, and H_k is the approximation of the Hessian at iteration k. H_k is updated at every step by

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T \tag{2.43}$$

where

$$\rho_k = \frac{1}{y_k^T s_k}, \quad V_k = I - \rho_k y_k s_k^T, \tag{2.44}$$

and

$$s_k = x_{k+1} - x_k, \quad y_k = \nabla f_{k+1} - \nabla f_k \tag{2.45}$$

The storage and manipulation of the dense approximation H_k can be prohibitively expensive when dealing with a large number of variables. To address this issue, a modified version of H_k is stored by selecting only a limited number (e.g., m) of vector pairs $\{s_i, y_i\}$ used in the formulas 2.43-2.45. The product $H_k \nabla f_k$ can then be obtained by performing a series of inner products and vector summations involving ∇f_k and the selected vector pairs. The oldest vector pair $\{s_i, y_i\}$ is then replaced by the new pair $\{s_k, y_k\}$ obtained from the current step (2.45). This way, the set of vector pairs includes curvature information from only the most recent m iterations. Empirical evidence suggests that small values of m (typically between 3 and 20) often lead to satisfactory results.

More specifically, at iteration k we have x_k as the current value for the parameters and the set of vector pairs is given by $\{s_i, y_i\}$. Then starting from an initial approximation H_k^0, formula 2.43 is applied multiple times until H_k satisfies

$$\begin{aligned} H_k = &\left(V_{k-1}^T \cdots V_{k-m}^T\right) H_k^0 \left(V_{k-m} \cdots V_{k-1}\right) \\ &+ \rho_{k-m} \left(V_{k-1}^T \cdots V_{k-m+1}^T\right) s_{k-m} s_{k-m}^T \left(V_{k-m+1} \cdots V_{k-1}\right) \\ &+ \rho_{k-m+1} \left(V_{k-1}^T \cdots V_{k-m+2}^T\right) s_{k-m+1} s_{k-m+1}^T \left(V_{k-m+2} \cdots V_{k-1}\right) \\ &+ \cdots \\ &+ \rho_{k-1} s_{k-1} s_{k-1}^T. \end{aligned} \tag{2.46}$$

From this expression a recursive procedure to compute $H_k \nabla f_k$ efficiently can be derived (Algorithm 1).

Algorithm 1 L-BFGS two loop recursion

1: $q \leftarrow \nabla f_k$
2: **for** $(i = k-1, k-2, ..., k-m)$ **do**
3: $\quad \alpha_i \leftarrow \rho_i s_i^T q$;
4: $\quad q \leftarrow q - \alpha_i y_i$
5: $r \leftarrow H_k^0 q$;
6: **for** $(i = k-m, k-m+1, ..., k-1)$ **do**
7: $\quad \beta \leftarrow \rho_i y_i^T r$;
8: $\quad r \leftarrow r + s_i(\alpha_i - \beta)$
9: **stop** with result $H_k \nabla f_k = r$

The computation required for multiplying by H_k^0 is not very costly, as it only requires $4mn$ multiplications. Additionally, this multiplication step is separate from the other

computations, which means that H_k^0 can be selected freely and can be varied between iterations.

A method for choosing H_k^0 is to set $H_k^0 = \gamma_k I$ where

$$\gamma_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} \tag{2.47}$$

In this case, γ_k is the scaling factor that tries to estimate the size of the true Hessian matrix along the most recent search direction. Finally the L-BFGS algorithm can be defined as

Algorithm 2 L-BFGS

Choose starting point x_0, integer $m > 0$;
$k \leftarrow 0$;
repeat
    Choose H_k^0
    Compute $p_k \leftarrow -H_k \nabla f_k$ from Algorithm 1.
    Compute $x_{k+1} \leftarrow x_k + \alpha_k p_k$, where α_k is chosen to satisfy the Wolfe conditions;
    if $k > m$ **then**
        Discard the pair $\{s_{k-m}, y_{k-m}\}$ from storage;
    Compute and save $s_k \leftarrow x_{k+1} - x_k$, $y_k = \nabla f_{k+1} - \nabla f_k$;
    $k \leftarrow k + 1$;
until convergence

The strategy of keeping the m most recent correction pairs works well in practice and it is proven to outperform Hessian-free Newton methods, see Nocedal & Wright (1999). The main weakness of L-BFGS is that its performance can be limited in situations where the Hessian matrix is ill-conditioned. This is because the algorithm relies on approximating the Hessian matrix by a limited set of past gradient information. When the Hessian matrix contains a wide distribution of eigenvalues, the limited set of past gradient information may not be sufficient to capture the full range of curvature of the objective function. In such cases, the algorithm may converge slowly, leading to longer computation times. Despite, this L-BFGS remains one popular and efficient choice for optimization problems.

Chapter 3

Number of People Analysis

In this chapter, both the *Prophet* model and the *Dynamic Linear Model* will be applied to the dataset presented in Chapter 1. By conducting a comprehensive evaluation, the goal is to assess the efficacy of these models in forecasting the target variable. Moreover, we will explore the results obtained from applying both models and present a comparative analysis of their forecasting performance. By comparing the performance of the *Prophet* model and the Dynamic Linear Model on the same dataset, we aim to provide empirical evidence regarding the efficiency of state space methods and their suitability for time series forecasting tasks in a real world setting.

3.1 The Time Series

To begin, it is important to examine the time series to gain insights into the presence of any trend and potential seasonality. By visualizing the original dataset, we can obtain a clearer understanding of its characteristics. The resulting time series plot is depicted in Figure 3.1 where the colour of the background depends on the *zona* variable.

It is evident from the data that the number of people observed in each photo exhibits a gradual increase over time, though with occasional setbacks. However, there is an inherent issue with the dataset that needs to be acknowledged. As mentioned in Chapter 1, the collection of photos was done inconsistently, with multiple photos taken on certain days and no photos taken on others. Consequently, the time intervals between consecutive observations can vary significantly, ranging from hours to even weeks.

In order to address this issue, the dataset was divided into three distinct subsets of daily data: *Morning*, *Noon*, and *Evening*, corresponding to the respective timeframes when the photos were taken. Furthermore, for the days on which no pictures were taken, the corresponding observations were included in the datasets with the value NA

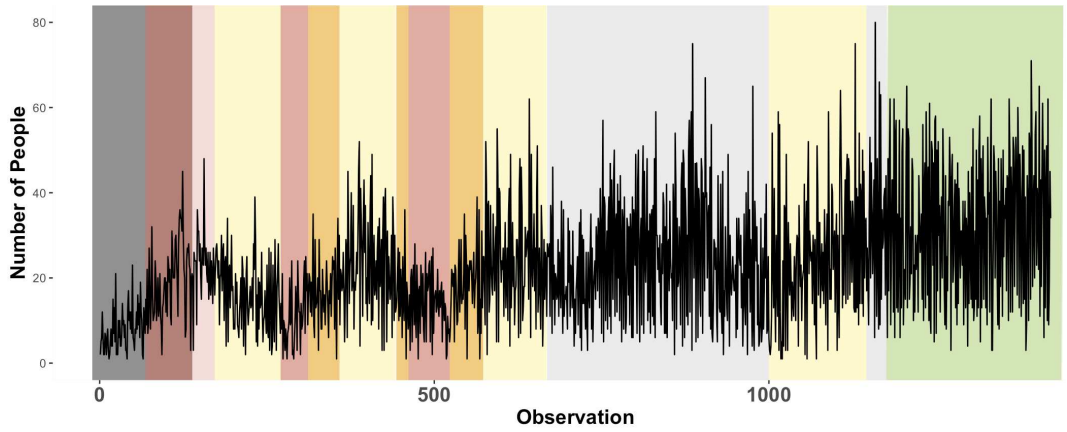

FIGURE 3.1: Time Series of the Number of People.

denoting the number of people detected. As a result, each dataset now consists of evenly spaced daily data, ensuring a consistent temporal structure across the observations.

The three times series are depicted in Figure 3.2. Each of the three series contains

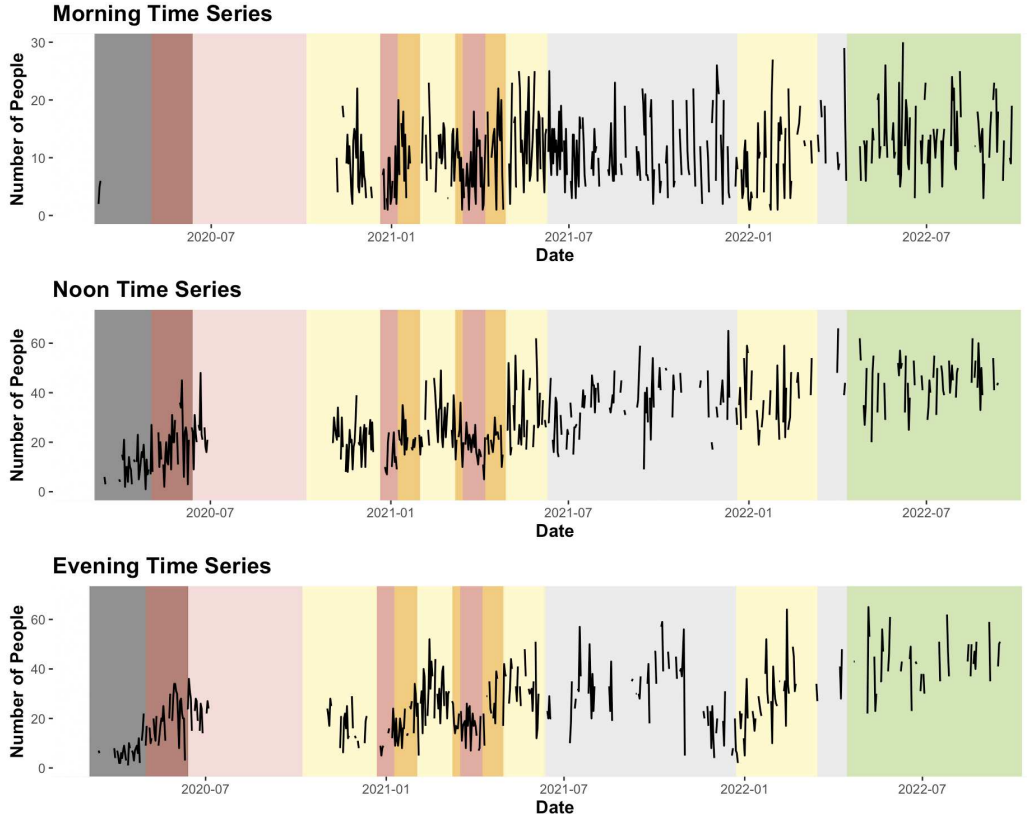

FIGURE 3.2: Time Series of the number of people at Morning, Noon and Evening (coloured by *zona* variable).

numerous missing observations. However, this does not pose a problem as the models presented in this work can handle missing data effectively. Additionally, it is worth

noting that all three time series exhibit an increasing trend, with occasional periods of low numbers that typically align with the Italian Government restrictions.

3.1.1 Autocorrelation Plots

Before starting with the modeling, it is necessary to take a look at the correlograms of these time series. A correlogram is the graphical representation of the autocorrelation function (ACF) values of a time series. ACF is a statistical tool used to analyze the presence of correlation or patterns in a time series. It measures the correlation between a time series and its lagged versions, revealing the relationship between past observations and the current observation. By examining the correlogram, we can gain insights into the temporal dependencies and the presence of trends, seasonality, and other patterns in the data (Fig. 3.3).

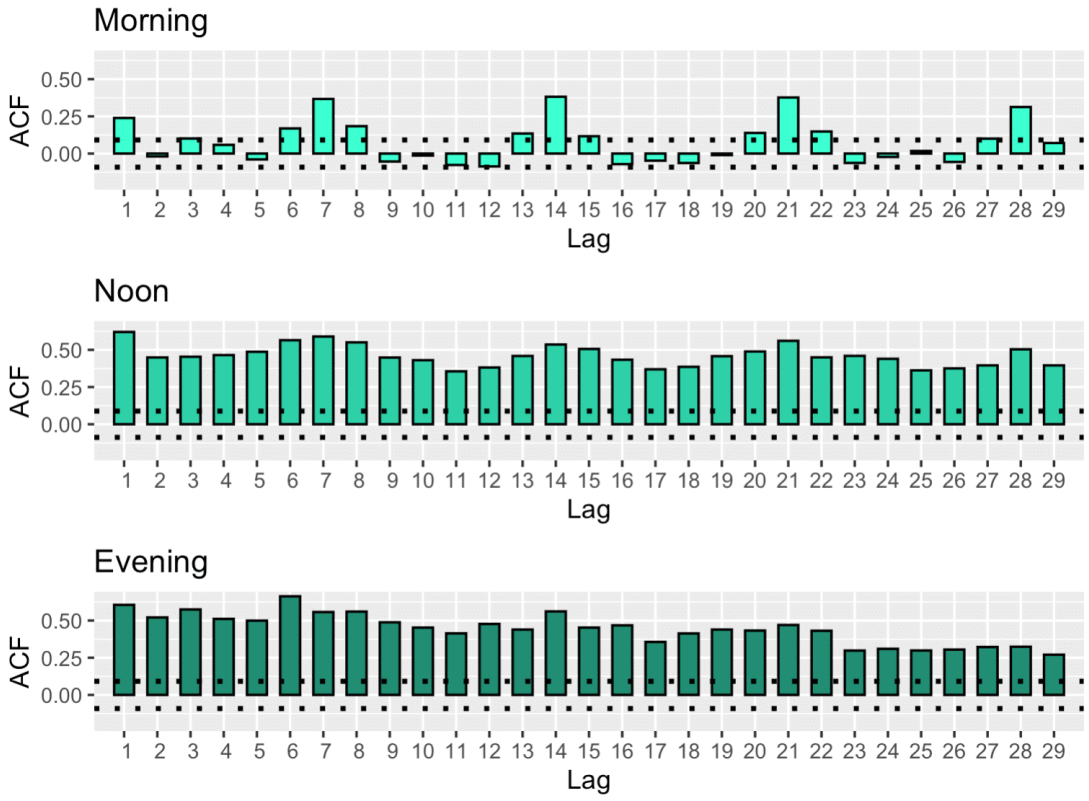

FIGURE 3.3: Correlograms of the Morning, Noon and Evening Time Series.

Each of the three time series exhibits a robust temporal correlation between observations. Notably, with the exception of the *Morning* time series, the autocorrelation values surpass the confidence bands at all lag intervals. This indicates a strong and significant autocorrelation structure, suggesting an influence of past observations on the current values of the time series. Furthermore, there is a distinct pattern of seasonality with a

period of 7 within the time series. This is evident, because the bars corresponding to the lag intervals aligned with the time frames are consistently higher than the surrounding bars. This recurring pattern every 7 steps indicates a cyclic behavior within the data, suggesting the presence of weekly seasonality that needs to be taken into account when fitting the models.

3.2 Prophet Results

To establish a benchmark for comparison, we will begin by fitting the *Prophet* model as the initial modeling approach. This will serve as the baseline against which we will evaluate the results obtained from the Dynamic Linear Model. By employing the *Prophet* model as the starting point, we can assess its performance and subsequently analyze the improvements or differences achieved through the application of the Dynamic Linear Model.

3.2.1 Morning

In the case of the *Morning* time series, the setup employed is the following: As discussed in Chapter 2, a *Piecewise Linear* trend was selected as well as a weekly seasonality component. Additionally, all regressors were included in the model, except for *Season*, which was found to not be statistically significant.

First thing to look at is Figure 3.4, where the estimated trend was plotted. Notice that the highlighted periods are those when Venice was in yellow zone or worse (orange and red). This representation will be used throughout the thesis for better explainability.

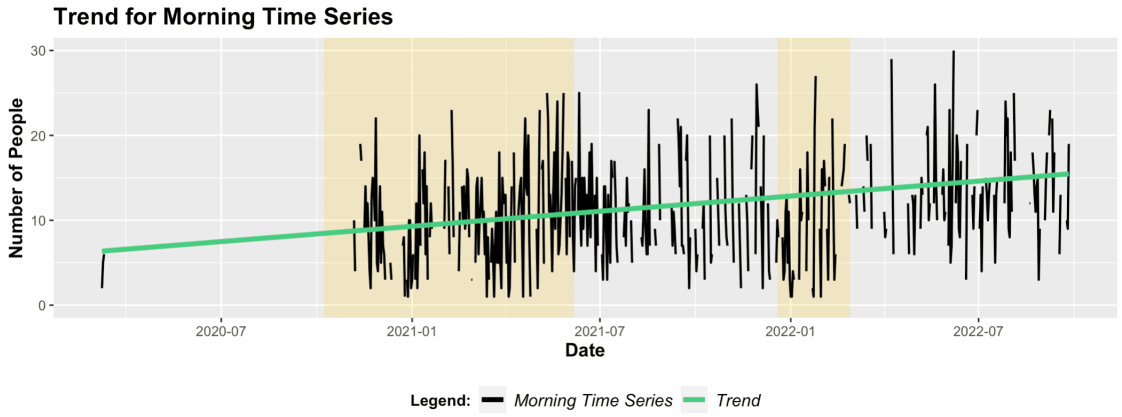

FIGURE 3.4: Trend estimated by Prophet for the Morning Time Series. The yellow background highlights the periods when Venice was in yellow zone or worse.

The fitted trend corresponds to a line with an upward slope over time, indicating that the model successfully captures the evident increasing pattern observed in the time series. However, it is unable to capture the temporary setbacks observed during the two years of COVID-19.

Next, in order to assess the model's performance, we plot the forecasted values, which incorporate the effects of seasonality and regressors (Fig. 3.5). This step is crucial as it provides insights into the model's predictive capabilities and allows us to evaluate its effectiveness as a predictor.

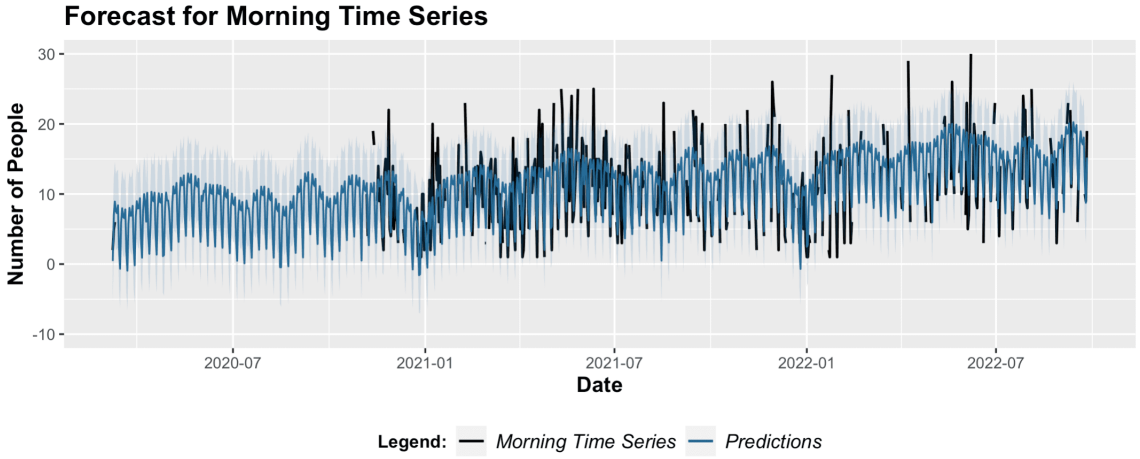

FIGURE 3.5: Predictions and intervals by Prophet for the Morning Time Series

The plot clearly illustrates the presence of seasonality in the time series, with a recurring pattern observed on a weekly basis. The vector of predicted values aligns reasonably well with the actual data, although there are instances where the observed values fall outside the confidence intervals. To obtain a more comprehensive understanding of the model's goodness of fit, it is crucial to analyze the residuals and determine if the model effectively eliminates autocorrelation. This can be accomplished by examining the correlograms computed on the residuals, which provide insights into the presence or absence of autocorrelation within the model. Figure 3.6 reveals that the distribution of residuals is slightly skewed to the right. However, the majority of residuals are clustered around zero, indicating a good fit of the model to the data. Additionally, the ACF plot displays no significant deviations beyond the confidence bands, suggesting the absence of autocorrelation. Furthermore, the *Ljung-Box test* conducted on the residuals yields a *p-value* of 0.24, indicating no evidence for the presence of residual autocorrelation. Based on these observations, it can be concluded that the model has effectively captured the underlying signal in the data and provide a satisfactory explanation for the observed time series.

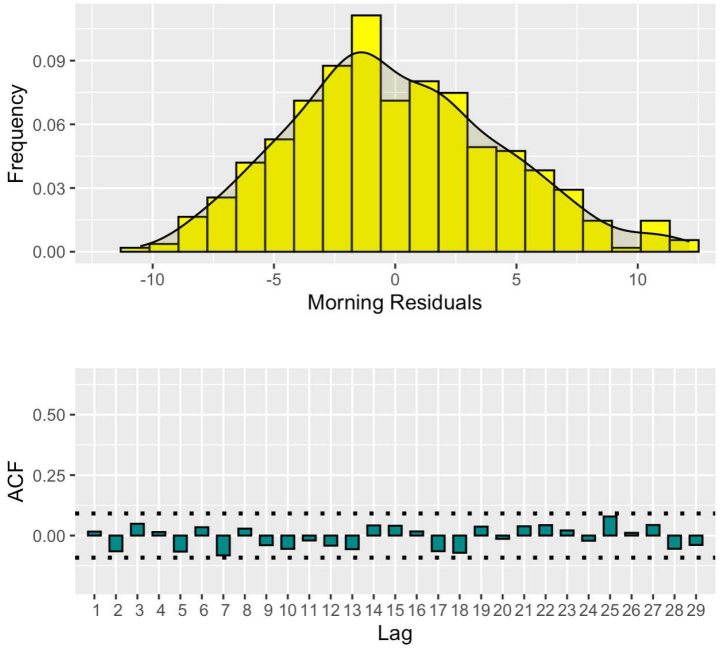

FIGURE 3.6: Prophet's residuals: histogram and ACF for the Morning Time Series

3.2.2 Noon

Continuing with the analysis on the *Noon* dataset, the *Prophet* model was configured with the same settings as in the previous case. This includes a *Piecewise Linear* trend, incorporation of a weekly seasonality component, and inclusion of all regressors except for the season variable.

As done earlier, the first thing to do is estimating the trend to see if the model is able to capture the the overall direction or tendency of the series.

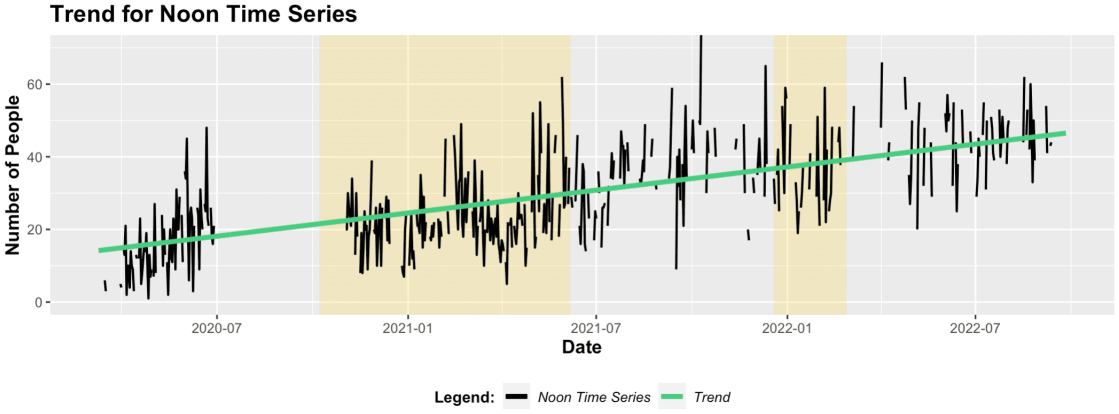

FIGURE 3.7: Trend estimated by Prophet for the Noon Time Series. The yellow background highlights the periods when Venice was in yellow zone or worse.

Upon examining Figure 3.7, it becomes evident that the *Noon* time series exhibits a more pronounced increasing trend compared to the *Morning* time series. The slope of

the trend line is even steeper in this case. However, despite the model fitting a piecewise linear trend and selecting multiple changepoints, the resulting line appears more like a single continuous line rather than a series of distinct segments able to capture the sudden changes in the series, which, in this case, are more prominent than in the previous series.

Now, the focus shifts to the forecasting ability of the model, which is plotted in Figure. 3.8. The incorporation of seasonality and regressors in the model greatly enhances its

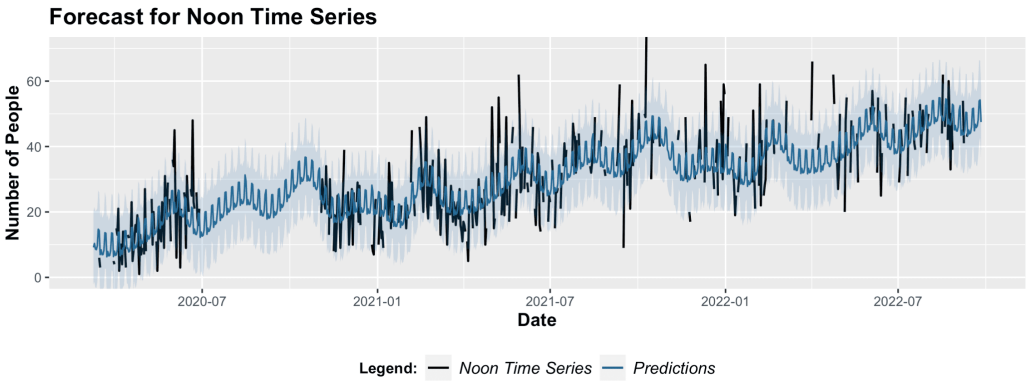

FIGURE 3.8: Predictions and intervals by Prophet for the Noon Time Series.

ability to capture the underlying behavior of the time series. The model effectively accounts for the setbacks observed during the pandemic and provides a satisfactory fit to the overall trend of the series. However, it is important to note that the fit is not perfect, as there are instances where high-value observations are not accurately predicted.

To have a more practical idea of how the model is predicting the residuals are plotted in Figure 3.9. In this case, although the distribution of errors exhibits a slight right-

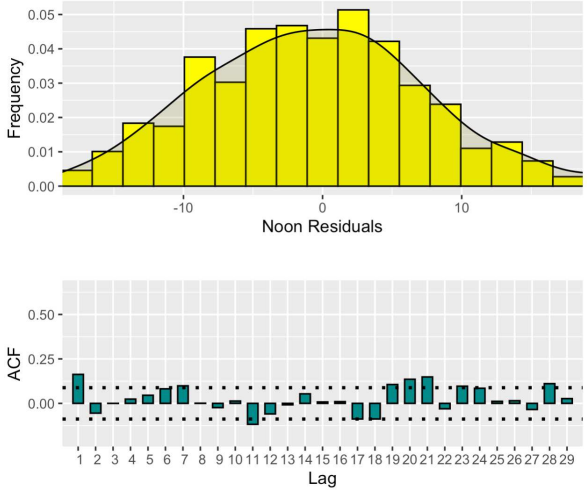

FIGURE 3.9: Prophet's residuals: histogram and ACF for the Noon Time Series.

skewness, it is not as prominent. In general, the majority the residuals are centered around zero, indicating that the model is fitted well to predict the values of the time series. However, the residuals still exhibit considerable autocorrelation, even at lag 1. Moreover, despite incorporating the seasonality when estimating the model, its effect is still visible in the correlogram. Additionally, several values of the Autocorrelation Function exceed the confidence bands, suggesting the presence of residual patterns or dependencies that persist in the time series data.

3.2.3 Evening

Lastly, we proceed with the analysis of the *Evening* time series, which exhibits a behavior similar to that of the *Noon* series. Because of this, similar results are to be expected. As for the previous cases, the model's setup is still the same: the *season* variable was removed from the regressors as it proved to be not significant. The trend was set to *Piece-Wise Linear* and a weekly seasonality effect was also added to the model.

Upon examining the plotted trend (Fig. 3.10), it is apparent once more that there is an upward trend observed in the time series. However, in this particular case, it becomes

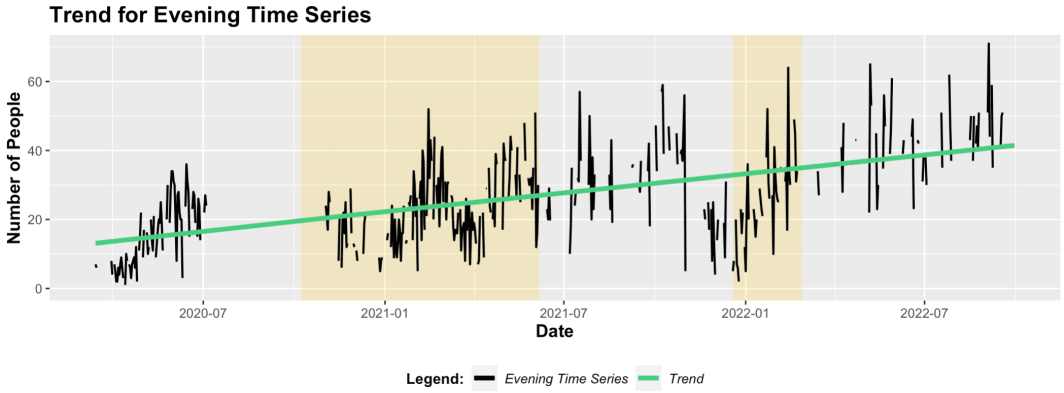

FIGURE 3.10: Trend estimated by Prophet for the Evening Time Series. The yellow background highlights the periods when Venice was in yellow zone or worse.

even more evident that the fitted trend may be overly simplistic, considering the high variability present in the observed data. Notably, significant drops in the series, such as those occurring in January 2021 and December 2021, are not adequately accounted for by the trend component of the model, highlighting a limitation in its ability to fully capture the dynamics and fluctuations of the time series once again.

The predicted values, on the other hand, seem to fit better to the observed data (Fig. 3.11). This means that the seasonality and regression components account for the

variability missing from the trend. A lot of the observed values, however, fall outside of the confidence bands which means that the predictions could be improved.

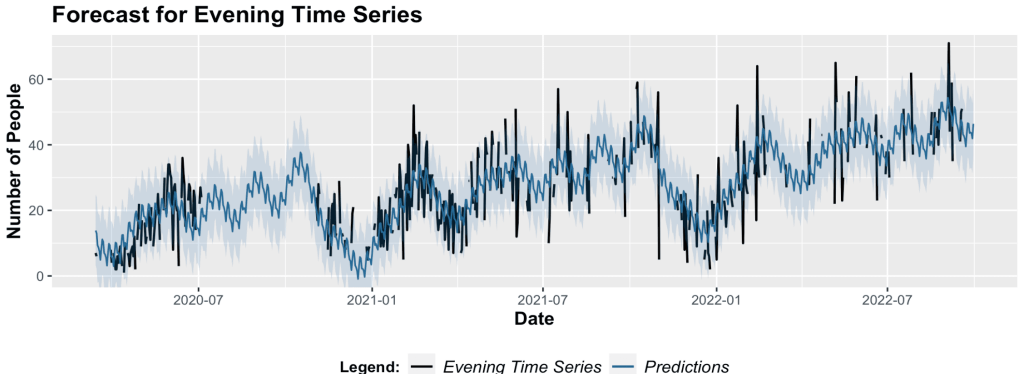

FIGURE 3.11: Predictions and intervals by Prophet for the Evening Time Series.

The residuals plots (Fig. 3.12) confirms that some autocorrelation still remains, as indicated by several values falling outside the confidence bands. The correlogram also reveals statistically significant differences from zero at lags 7, 14, 21, and 28, indicating the persistence of seasonality in the residuals. Furthermore, the distribution of errors exhibits asymmetry, with a mode around -5, suggesting a consistent pattern of overestimation in the model's predictions.

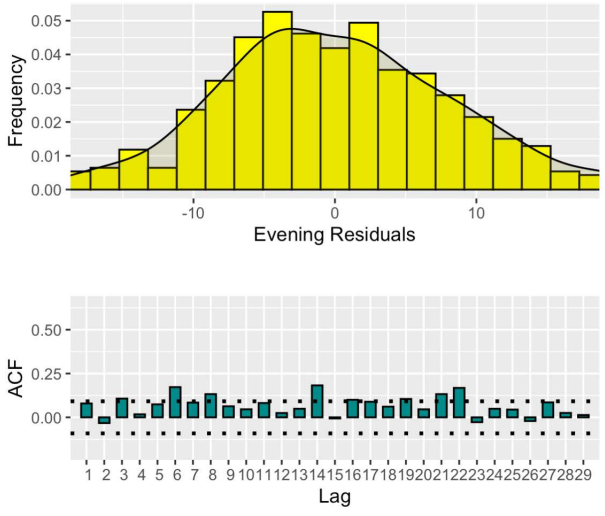

FIGURE 3.12: Prophet's residuals: histogram and ACF for the Noon Time Series.

3.3 Dynamic Linear Model Results

In this section, we will present and discuss the results obtained by applying the *Dynamic Linear Model* to the *Morning*, *Noon* and *Evening* time series. However, the comparison

with *Prophet*'s will not be presented here but in a later stage.

3.3.1 Morning

The first dataset on which the model was tested is the *Morning* dataframe. In this case, the series shows random fluctations around a certain level, therefore a local level model was tested and got the best results on the data. The setup is the following: the order of the polynomial for the trend component was set to 1, the seasonality was set to 7 and for the purpose of comparison, the regressors used were the same as those employed in *Prophet*: *mean_temperature*, *zona*, *rain*, and *weekend_festive*.

In Figure 3.13, the plots of the smoothed trend and the smoothed series are presented. Remember that, in the context of of DLMs, the smoother retrospectively reconstructs the behavior of the system under analysis by using all information up to time t.

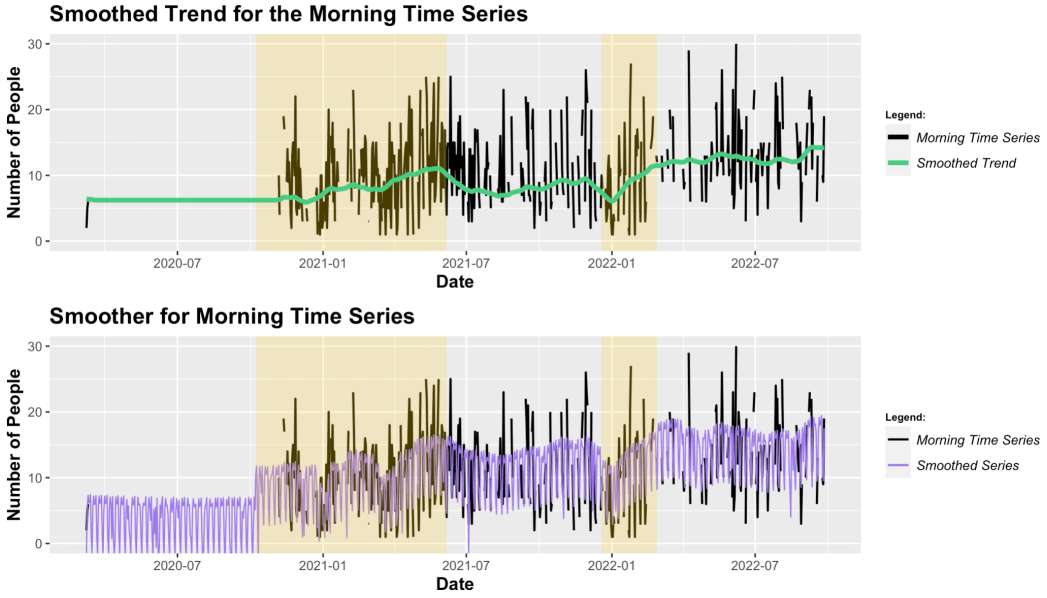

FIGURE 3.13: Smoothed Trend and Series for the Morning Time Series. The yellow background highlights the periods when Venice was in yellow zone or worse.

Clearly, the estimated trend effectively captures the behavior of the time series, as demonstrated by its accurate fitting of the setbacks observed in January 2021 and January 2022. Overall, the trend appears to closely align with the shape of the data, which indicates that the state space approach is very flexible. The smoothed series, obtained by combining the trend, seasonal, and regression components, exhibits a strong alignment with the observed data. On the other hand, there are some issues in the initial period between March 2020 and November 2020, these can be attributed to the presence of numerous missing values and the recursive nature of *Dynamic Linear Models*,

which rely on previous values for estimation. Furthermore, while the lows in the series are accurately fitted, there are instances where some of the peaks are not adequately represented, resulting in underestimation.

Next, in Figure 3.14 the one-step-ahead predictions for the observed values are plotted.

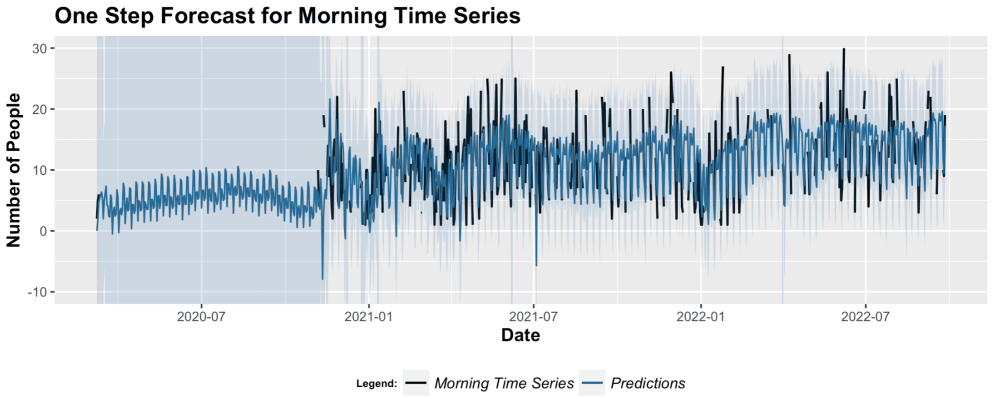

FIGURE 3.14: One-step-ahead predictions and intervals for the Morning Time Series.

The fitted values demonstrate a strong alignment with the observed data: first of all, the overall trend is effectively captured by predictions. Second, the peaks that are not perfectly fitted by the model still fall within the confidence bands. Furthermore, in the initial period characterized by a high number of missing values, the fitting is reasonably performed, although the confidence intervals tend to be wide due to the limited data points available for prediction.

Finally, in order to evaluate the quality of the fit, it is essential to examine the standardized residuals, also known as *innovations*, as discussed in Chapter 2 (Fig. 3.15).

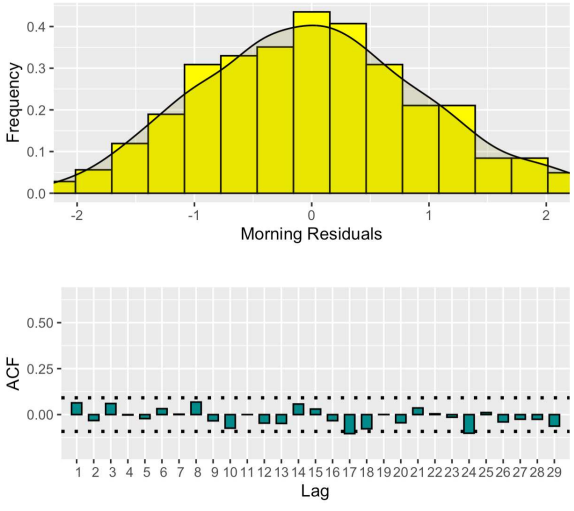

FIGURE 3.15: Residuals: histogram and ACF for the Morning Time Series.

The results are very promising, the residuals exhibit no autocorrelation and follow a symmetric distribution centered around zero with zero being the mode. As introduced in Chapter 2, it is expected for residuals to adhere to a standard normal distribution and display no autocorrelation. To evaluate these two properties, the *Shapiro* and *Ljung-Box* tests were conducted. The first one yielded a *p-value* of 0.11, while the second one resulted in a *p-value* of 0.34, confirming that both conditions are satisfied.

3.3.2 Noon

For the *Noon* time series two different models were tested for the trend: the local level model and the linear growth model. In this case, the latter was selected as it yielded the best results and a better *Akaike Information Criterion* (AIC) score. The setup, therefore, is the following: the order of the polynomial for the trend component was set to 2, a weekly seasonality was added and the matrix of regressors is the same as for the *Morning* time series.

The smoothed trend and the smoothed time series are plotted in Figure 3.16: the trend in the series effectively captures its behavior, with few exceptions. Apart from a slight decrease and subsequent relatively constant trend immediately following January 2021, the model adequately accounts for all other setbacks and sudden increases in the series. Regarding the smoothed series, the inclusion of seasonality and regressors significantly enhances the model's capabilities. The dynamic changes in seasonality over time are readily observable, illustrating its ability to adapt to the series' evolving patterns. In general, the smoother successfully represents the series' underlying characteristics.

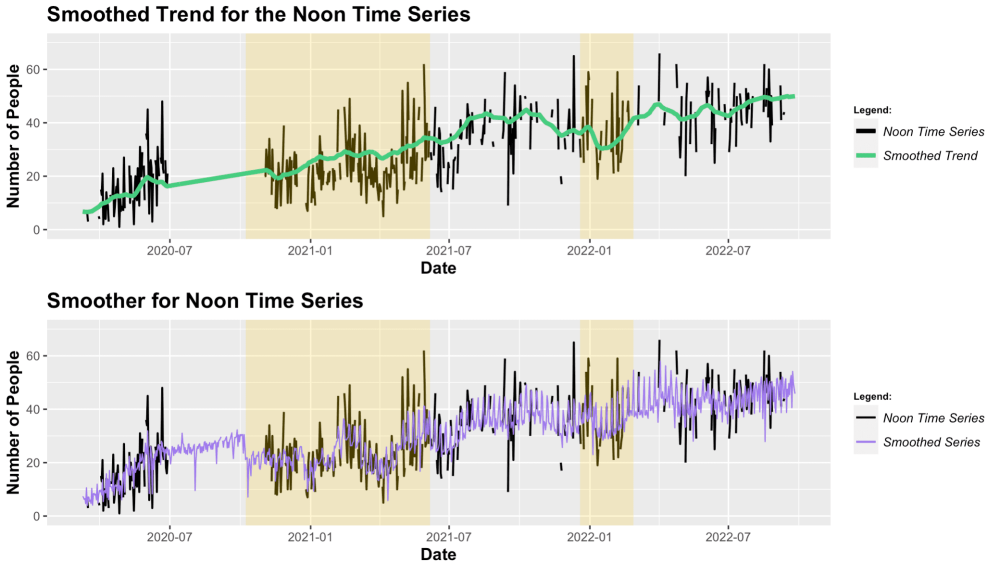

FIGURE 3.16: Smoothed Trend and Series for the Noon Time Series. The yellow background highlights the periods when Venice was in yellow zone or worse.

Next, in order to assess the model's performance when predicting future values, we plot the one-step-ahead prediction (Fig. 3.17).

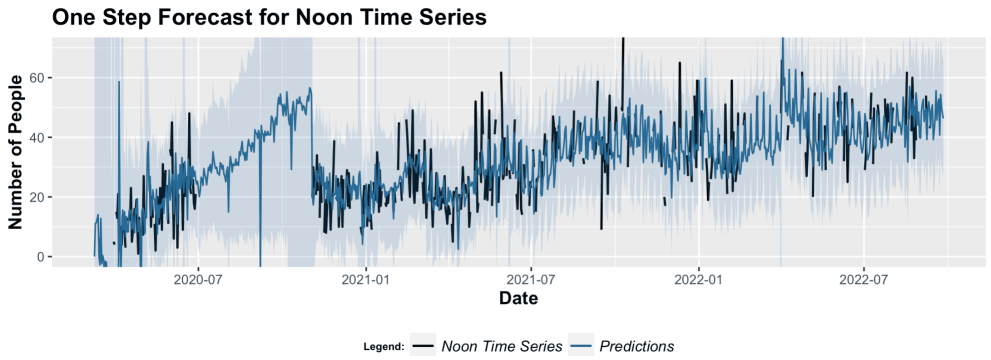

FIGURE 3.17: One-step-ahead predictions and intervals for the Noon Time Series.

During the initial 6 months, where the available data is limited, the predictions exhibit significant noise. This is evident from the wide confidence bands, indicating the uncertainty in the forecasted values. Moreover, between July 2020 and November 2020, the predicted trend shows a somewhat exaggerated constant increase that deviates from the actual observed behavior. As the number of observations stabilizes over time, the forecasting significantly improves and closely aligns with the observed data. The majority of the observations fall within the confidence intervals, indicating the accuracy of the predictions. Finally, it is necessary to analyse the *innovations*, whose distribution is plotted in Figure 3.18 alongside the correlogram.

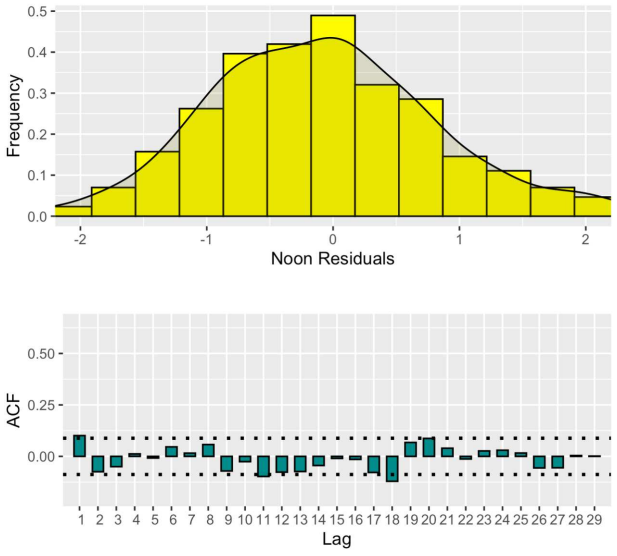

FIGURE 3.18: Residuals: histogram and ACF for the Noon Time Series.

The histogram displays a slight rightward asymmetry, although the majority of values are centered around zero, indicating a good fit. The correlogram exhibits a residual autocorrelation, but overall, the majority of the bars fall within the confidence intervals, with only a few, acceptable exceptions that are just outside of them. Conducting the Shapiro and Ljung-Box tests yield *p-values* of 0.000032 and 0.0117, respectively. As a result, the assumptions of normality and autocorrelation are violated. Nevertheless, considering the strength of these assumptions, these values can be deemed acceptable.

3.3.3 Evening

For the final dataset, the *Evening* one, the analysis is very similar to the *Noon* dataset. However, there is one main difference. The polynomial order that was found yielding the best results was actually a first order polynomial and not a second order. Everything else, i.e. the seasonality component and the design matrix, remained the same.

The trend component (Fig. 3.19) follows closely the observed data. In fact, it effectively captures the various oscillations occurring over the span of two and a half years, accurately representing the behaviour of the series. However, the smoother appears to slightly overestimate the values of the series.

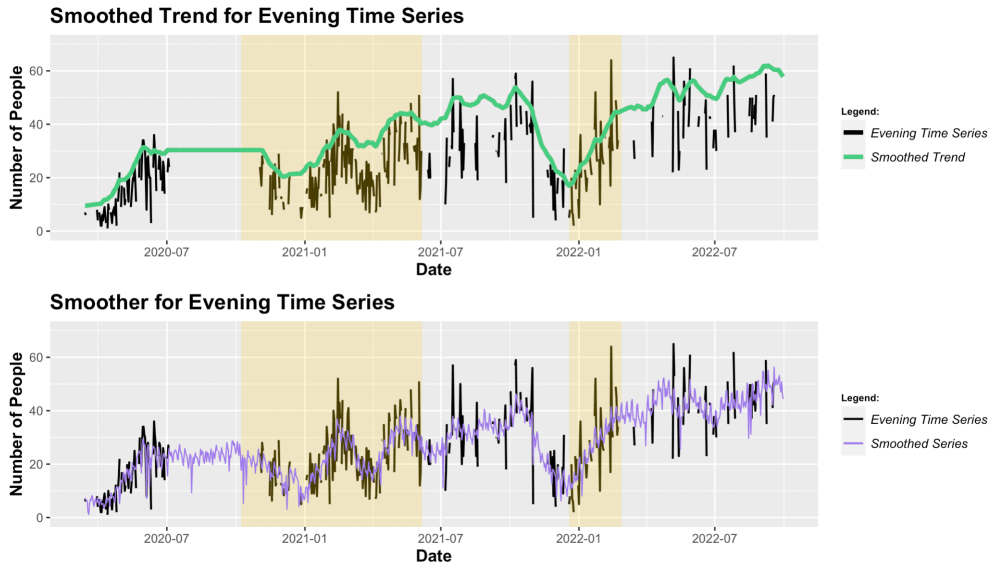

FIGURE 3.19: Smoothed Trend and Series for the Evening Time Series. The yellow background highlights the periods when Venice was in yellow zone or worse.

The smoothed series, which includes the seasonal component and the regressors, fixes the problem and achieves a close resemblance to the original series, indicating a good fit of the model. Once again, the dynamic nature of the seasonality is evident, showcasing

the flexibility of the model, which is able to accurately follow the series different patterns and variations.

As for the predictions, plotted in Figure 3.20, the behavior is very similar to the smoothed series. In the first six months, because of the missing values, there is high variability. On the other hand, the remaining part of the series does not exhibit any indications of poor fitting, as the majority of observations lie within the confidence intervals.

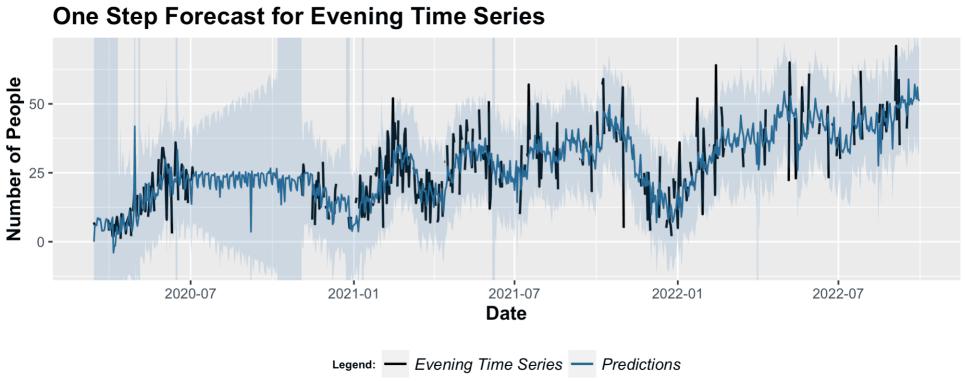

FIGURE 3.20: One-step-ahead predictions and intervals for the Evening Time Series.

The residuals exhibit a symmetrical distribution centered around zero, with the majority of values closely approaching zero (Fig. 3.21). Conversely, the correlogram is not flawless, as some values exceed the confidence intervals, despite the absence of seasonality. The violations of assumptions are confirmed by the Shapiro test and Ljung-Box test, whose *p-values* are respectively 0.00066 and 0.0087, respectively.

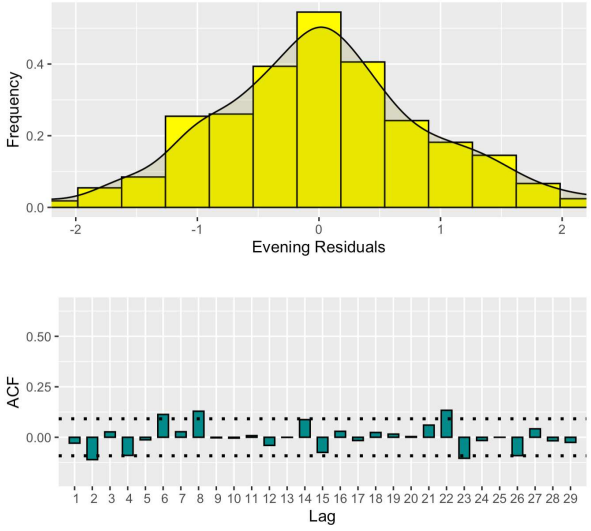

FIGURE 3.21: Residuals: histogram and ACF for the Evening Time Series.

3.4 Models Comparison

After presenting the results for both the *Prophet* model and the *Dynamic Linear Model*, we can derive some conclusions and assess the relative efficiency of these two models when applied to real-world time series data.

First of all, it is important to notice that the three time series presented in this work are not trivial. The primary challenge arises from the significant number of missing values. Both of the approaches we used are able to handle this type of data, nevertheless, dealing with a significant amount of missing data is a difficult task. Specifically, for the *Morning* and *Evening* datasets, 50% of the values are marked as NA, whereas for the *Noon* dataframe, the percentage decreases slightly to 46%. In addition, the number of people walking in the streets of Venice can be influenced by numerous unaccounted factors. Moreover, the data extraction process relied on a Neural Network, which, despite its overall effectiveness, faces challenges when handling highly crowded photos and images with reduced visibility. These two things combined lead to a high variability in the observations and thus a more complicated analysis. With these premises, the two models that have been employed, *Prophet* and *Dynamic Linear Model*, are able to give a satisfactory fit to the observed data, especially the latter, which outperforms *Prophet* in every aspect, offering a comprehensive analysis of the time series.

Firstly, Tables 3.1 and 3.2 provides a quantitative assessment of the performance of the two models by presenting the mean squared error (MSE) for both in-sample predictions and out-of-sample predictions. The in-sample predictions compare the predicted values of the *Prophet* model with the smoothed values of the *Dynamic Linear Model*. On the other hand, the out-of-sample predictions compare the one-step-ahead predictions of *Prophet* with the ones of the DLM for the last thirty observations in the three series.

In-sample MSE		
Dataset	Prophet	Dynamic Linear Model
Morning	18.56	17.06
Noon	80.34	51.49
Evening	72.60	49.68

Table 3.1: In-sample MSE for the two models.

Out-of-sample MSE		
Dataset	Prophet	Dynamic Linear Model
Morning	26.04	21.08
Noon	113.66	88.72
Evening	140.99	135.22

TABLE 3.2: Out-of-sample MSE for the two models

In both metrics across all three datasets, the DLM outperforms *Prophet*. This superiority is particularly pronounced in the *Noon* dataset, where the results show a significant improvement compared to those of *Meta*'s model. These results underscore the strength of the *Dynamic Linear Model* in capturing the underlying patterns and dynamics of the data, enabling it to generate more precise predictions.

Secondly, for all three series the trend captured by the DLM is way more accurate than the one estimated by *Prophet*. Despite *Prophet*'s capability to detect different changepoints and approximate the trend using piecewise linear functions, the result appears as a shallow linear increase in all three cases. On the other hand, the smoothed trend provided by *Dynamic Linear Model* successfully captures all the patterns and the general behaviour of the response variable. Indeed, throughout the observed period, there were numerous falls in the number of people driven by the pandemic, followed by subsequent periods of recovery and regrowth. It is essential to consider these setbacks and fluctuations rather than being constrained by an informative, but still simplistic linear increase.

Next, aside from the first months, where the large number of missing values plus the sequential development of the Kalman filter throught time, make the predictions very variable, the forecasted values provided by the DLM look much better. Not only they offer a better fit to the observed data, but they also fall within the confidence intervals, with only a few exceptions. Moreover, the residuals of the DLM exhibit lower levels of autocorrelation, with only one case having a slightly lower *p-value* than 0.01, indicating a good fit to the data. In contrast, the residuals of Prophet, aside from the *Morning* dataset, demonstrate signs of autocorrelation and seasonality.

Finally, it is important to focus on how these models achieve the results. As introduced in Chapter 2, *Prophet* is, at its core, an additive model composed of different components. As a result of its additive nature, whenever new data becomes available and updated estimates are required, it is necessary to refit the entire model. In the context described in this thesis, this is not a limitation, as the number of observations

is relatively small, and the computational time required to fit the model is minimal. However, in many applications, especially in recent years with the exponential growth of data, this may not hold true. On the contrary, thanks to their recursive nature, *Dynamic Linear Models* have several advantages:

- Real-time analysis: With sequential updating, DLMs can analyze time series data in real-time as new observations are received.
- Efficiency: By updating the model incrementally, sequential updating avoids the need to reestimate the entire model with each new data point.
- Adaptive modeling: DLMs can adapt and adjust to changing patterns or dynamics in the data over time. As new observations are incorporated, the model's estimates can be refined, allowing it to capture evolving trends, seasonality, or other underlying patterns more accurately.

Overall, sequential updating of DLMs offers a flexible and efficient approach to model time series data, allowing for adaptive analysis and real-time insights as new observations are received.

3.5 Interpretation

Having established the superior fit of the DLM compared to *Prophet* on the three datasets presented in this thesis, this section aims to delve further into the model to analyze the behavior of the series and examine the influence of COVID-19 on the number of people in the streets of Venice.

In fact, one of the key characteristics of *Dynamic Linear Models* is their ability to generate interpretable results, allowing analysts to gain insights into the relationships between variables and the evolution of a system over time. Unlike other algorithms, DLMs provide a clear and transparent representation of how the data evolves over time. By explicitly modeling the underlying state variables and their relationships, DLMs allow for meaningful interpretation and understanding of the observed data. Furthermore, DLMs allow for the interpretation of various components of the model, such as the trend, seasonality, and exogenous variables. By decomposing the time series into these distinct components, analysts can gain insights into the individual contributions of each component and their impact on the overall behavior of the system.

To begin with, the trend, plotted in Figure 3.13, 3.16, and 3.19, appears similar the three series. The only noticeable distinction lies in the *Morning* series, which exhibits a

more consistent trend centered around the value of 10. On the contrary, the remaining two series demonstrate a consistent upward trajectory in their values, which appears to persist until around July 2022. Consequently, with some occasional setbacks, notably in January 2021 and January 2022 due to strengthened restrictions during holiday periods, the number of people walking in the streets continued to rise until the impact of COVID-19 diminished as a threat.

Next, we shift our focus on the smoothed seasonality component, plotted in Figure 3.22, to analyse the pattern and the impact it has on the series.

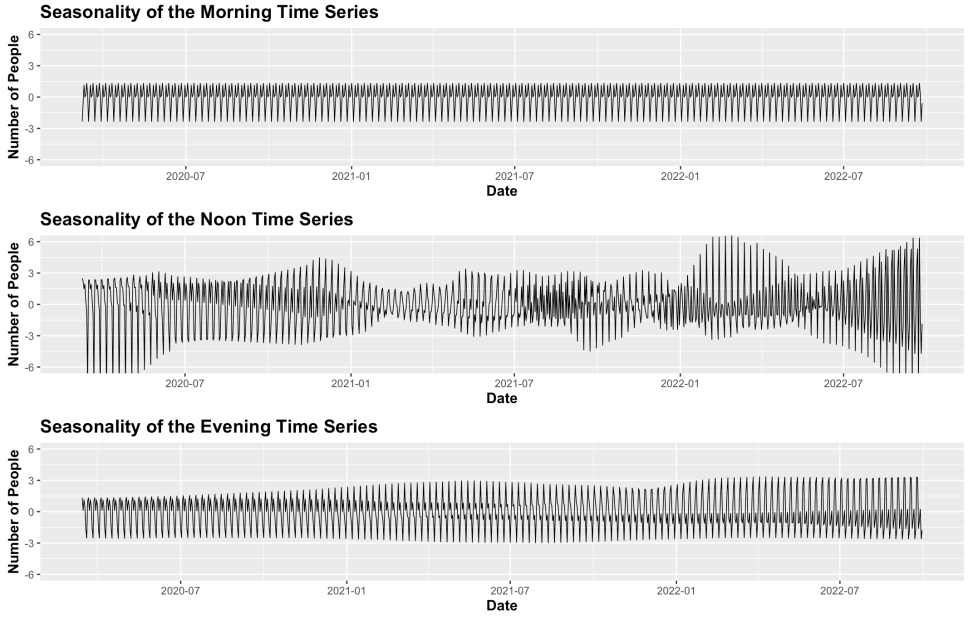

FIGURE 3.22: Seasonality estimated by the DLM for the three series.

Clearly, the seasonality is estimated dynamically by the model. This is noticeable in the *Noon* and *Evening* time series, where we can see it constantly changing across different periods. The former, however, presents the most variability, which significantly affects the number of individuals, with substantial negative and positive values, particularly during the initial and final nine months. Furthermore, the pattern remains consistent across different times of the day: the count consistently increases significantly during the weekends and subsequently declines at the beginning and throughout the week.

Finally, the regressors, along with other components, are dynamically estimated. To examine their behavior, we will plot their values estimated by the Kalman filter at each time step. However, it's worth noting that the first six months predominantly contain missing values, resulting in highly variable estimates. For the sake of clarity, we will only plot the values starting from November 2020 onwards. The first two variables are

the ones related to the wheather conditions on the day the pictures were taken, plotted in Figure 3.23.

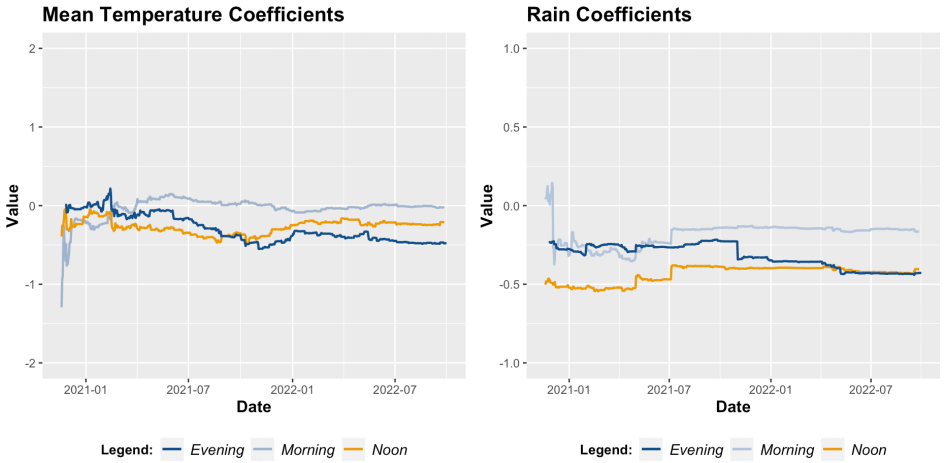

FIGURE 3.23: Values estimated for the parameters of the variable *mean_temperature* and *rain* respectively.

When examining the temperature variable, we observe that in the morning, the coefficients' values are nearly zero. This makes sense since the photos are captured early in the day when the streets are primarily occupied by workers who go work. Conversely, during the afternoon and evening, there is a negative effect, albeit not significantly large, and during the summer, the values rise closer to zero. It is likely that the impact of various seasons and temperatures is already accounted for by the trend component, which captures the long-term relationships within the series. In contrast, the coefficients related to rainfall are highly intuitive. In the morning, the coefficient value remains near zero for the same reason as earlier. However, during other times of the day when people venture out for a walk, the impact on the number of individuals is negative, indicating that rainfall has a noticeable influence in deterring people from being outdoors.

Moving on with the *weekend_festive* variable, its values are plotted in Figure 3.24.

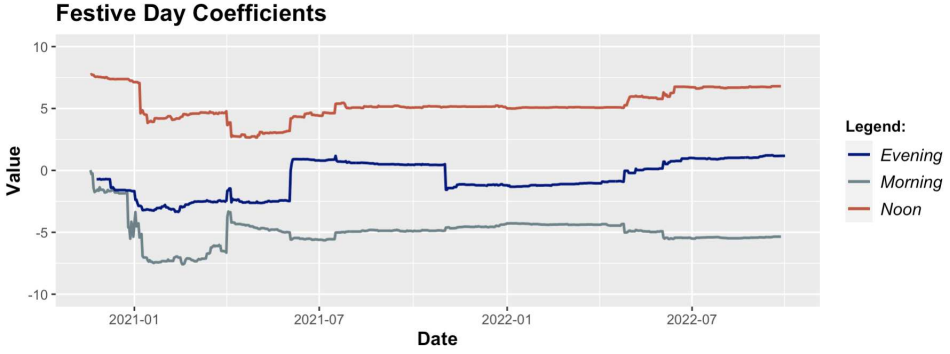

FIGURE 3.24: Values estimated for the parameters of the *weekend_festive* variable.

Clearly, during the morning hours, festive days have a negative effect on pedestrian numbers. This can be attributed to the fact that on festive days, people are not working, resulting in significantly emptier streets during the early hours. On the contrary, during the afternoon, there is a strongly positive effect, indicating a significant impact on the number of people in the streets of Venice on days of rest. In the evening, when both workers and non-workers have the opportunity to go out, the coefficient values are close to zero, suggesting a minimal or negligible effect. The last variable, the one related to the government constrictions, has also a big impact on the target variable, however, not in the way one would think (Fig. 3.25).

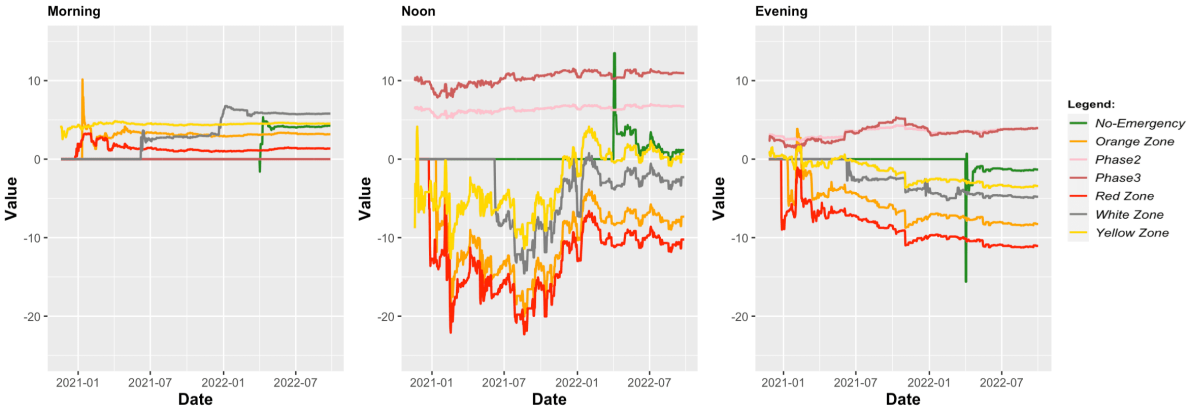

FIGURE 3.25: Values estimated for the parameters of the *zona* variable.

The baseline level is set as *lockdown*, and except for the *Morning* dataset, the impact of different zones appears to be contrary to what was observed during the COVID-19 pandemic. Observations during *Phase2* and *Phase3* are not reliable since they are very limited but they exhibit a notable positive effect. On the other hand, other zones, including *no-emergency*, seem to have a more negative impact on pedestrian numbers compared to the *lockdown* period when people were predominantly at home. However, the apparent inconsistency can be explained by the fact that, similar to the mean temperature, a portion of the variability and information contained in the *zona* variable is already captured by the trend component. As previously mentioned, in the "Evening" dataset, the trend component slightly overestimates the observed values. Hence, it is logical for the regressors to compensate for this by exerting a negative effect, even for the more restrictions-free coefficients. Furthermore, a simple *Linear Model* was fitted with only the variable *zona* as a dependent variable and *lockdown* as the reference level. The results are summarized in Table 3.3.

. **Morning**

Parameter	Value	P-value
Intercept	4.0	0.08
No-Emergency	9.7	5.5e-05
Orange	6.5	0.01
Phase 2	NA	NA
Phase 3	NA	NA
Red	3.0	0.23
White	7.5	0.001
Yellow	7.0	0.003

Noon

Parameter	Value	P-value
Intercept	10.0	9.0e-09
No-Emergency	33.9	2.0e-16
Orange	10.7	7.8e-06
Phase 2	9.6	0.0001
Phase 3	14.7	4.2e-06
Red	7.3	0.003
White	25.9	2.0e-16
Yellow	20.0	2.0e-16

Evening

Parameter	Value	P-value
Intercept	7.4	0.0001
No-Emergency	36.4	2.0e-16
Orange	13.4	2.0e-07
Phase 2	13.0	5.5e-07
Phase 3	15.6	3.1e-06
Red	3.0	0.004
White	22.4	2.0e-16
Yellow	19.1	2.0e-16

TABLE 3.3: Parameter estimates of a linear model for *zona* variable.

In this scenario, with only the *zona* variable as a predictor, we can observe that the effect aligns with our expectations. The reference level, corresponding to the lockdown period, has the lowest value, while all other zones increase this value proportionally based on the degree of restrictions imposed in that period. As a result, the Red zone consistently exhibits the lowest value among all the other zones. After that, the Orange zone and Phase 2, characterized by relatively stringent restrictions, show slightly higher values. Finally, the remaining periods are relatively comparable, with the *no-emergency* period exhibiting the highest increase in value.

This concludes the analysis of the results of the *Dynamic Linear Model*. As we have seen, this methodology offers flexibility and interpretability, allowing us to gain valuable insights into the data. By incorporating various components such as trends, regressors, and reference levels, we were able to examine the relationships and impacts within the time series. More specifically, the impact of COVID-19 on pedestrian activity was evident in various time periods and can be observed through the analysis of the time series data, with factors like temperature, rainfall, and zones (reflecting different stages of restrictions) all playing a different role in shaping the fluctuations in pedestrian numbers during the pandemic.

Chapter 4

Cluster Analysis

After successfully implementing the *Dynamic Linear Models* and conducting an analysis on the number of people at *Campo San Felice* in Venice, the focus now shifts towards examining the spatial distribution of these individuals. The analysis is of particular interest due to the implementation of various restrictions during the pandemic concerning public gatherings. Throughout the majority of the studied period, individuals were prohibited from being in close proximity, with a minimum requirement of 1.5 metres, and clusters were strongly discouraged. Hence, the objective of this chapter is to examine the alterations in the clustering behavior of individuals. The focus is on understanding how people's tendency to form groups or clusters has changed over time and under different constrictions. To begin with, a significant issue concerning the perspective captured in the photos will be presented, followed by a proposed solution to address it. Subsequently, the *DBSCAN* clustering algorithm will be introduced and applied to the coordinates of individuals within the photos. Finally, with these clusters various analysis will be conducted using *Dynamic Linear Models*.

4.1 Perspective Transformation

As discussed in Chapter 1, YOLOv7 has the capability not only to detect the number of people in a photo but also to locate their positions within the image. Consequently, a dataset was created containing the coordinates of each person in each photo. However, it is important to note that this dataset suffers from a significant flaw. In fact, by taking a quick look at any of the pictures, it is clear that were not captured from an overhead, bird's eye perspective of the square. Therefore, the coordinates derived from the neural network are inaccurate. While this discrepancy may have a negligible impact on people in the foreground, it becomes more pronounced for individuals situated farther away. In

such cases, the estimated positions are significantly closer than their actual distances. As a result, any clustering algorithm relying on distance-based calculations would fail in this biased environment. Because of this, a solution is required to address this aspect and incorporate it into the analysis. It turns out that the problem can be solved by a transformation called *Homography*, which consists of a simple matrix multiplication.

4.1.1 Homography

Through *Homography* it is possible to shift from one view to another view of the same scene. This is achieved by multiplying the *Homography Matrix* with the points in one viewpoint, thus determining their corresponding locations in the alternative viewpoint.

By denoting with P and R the matrices containing the original locations and the projected locations respectively, the task is to find H such that

$$\begin{aligned} R &= HP \\ R &= \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} P \end{aligned} \tag{4.1}$$

In this study, we will only provide a brief overview of how the *Homography matrix* (H) is estimated. However, to understand this estimation process, we need to introduce two fundamental concepts: Homogenous Coordinates, Bloomenthal & Rokne (1994) and the Pin Hole Camera Model, Sturm (2014). Homogeneous coordinates are a mathematical representation used in projective space. They allow us to represent points, lines, and other geometric entities in a uniform manner. Unlike Cartesian coordinates, which use three values (x, y, z) to represent a point in three-dimensional space, homogeneous coordinates use four values (x, y, z, w). This extra dimension, represented by the scaling factor w, allows for the representation of points at infinity and supports transformations involving perspective. It enables us to represent parallel lines as intersecting at a point at infinity and perform perspective transformations such as projecting 3D objects onto a 2D image plane.

Homogeneous coordinates and projective space make it possible to develop the pin hole camera model, which, in simple terms, is a simplified representation of how a camera captures an image. It is based on the principle that light from the scene passes through a small opening (known as the pinhole) and forms an inverted image on a photosensitive surface or image plane. Thanks to this model it is possible to project a scene in the 3D space onto the image plane (2D image). The equation is the following:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.2}$$

The two matrix multiplying $[X, Y, Z, 1]^T$, the *Camera Intrinsic Matrix* and the *Camera Extrinsic Matrix* respectively, can be combined to obtain the *Camera Matrix*

$$C = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & t_{34} \end{bmatrix} \tag{4.3}$$

which can be used to perform the transformation. In our case, however, we are interested in projecting a 2D image into a different 2D space. Consequently, the matrix in 4.3 can be updated to obtain the *Homography Matrix*.

$$\begin{aligned} \begin{bmatrix} u \\ v \\ w \end{bmatrix} &= \begin{bmatrix} c_{11} & c_{12} & c_{14} \\ c_{21} & c_{22} & c_{24} \\ c_{31} & c_{32} & c_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \\ &= H \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \end{aligned} \tag{4.4}$$

The matrix tells us that a shift from one view to another view of the same scene is essentially a transform from one projective plane to another. Notice that there is no h_{33} value in H since we can assume it to be 1 for normalization purposes.

Next, to obtain the estimates it is sufficient to find some points that related to each other between the two desired planes. Specifically, we need four points, and by

combining all the equations together we obtain

$$\begin{bmatrix} x^{(1)} & y^{(1)} & 1 & 0 & 0 & 0 & -\hat{x}^{(1)}x^{(1)} & -\hat{x}^{(1)}y^{(1)} \\ 0 & 0 & 0 & x^{(1)} & y^{(1)} & 1 & -\hat{y}^{(1)}x^{(1)} & -\hat{y}^{(1)}y^{(1)} \\ & & & & \dots & & & \\ x^{(4)} & y^{(4)} & 1 & 0 & 0 & 0 & -\hat{x}^{(4)}x^{(4)} & -\hat{x}^{(4)}y^{(4)} \\ 0 & 0 & 0 & x^{(4)} & y^{(4)} & 1 & -\hat{y}^{(4)}x^{(4)} & -\hat{y}^{(4)}y^{(4)} \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} \hat{x}^{(1)} \\ \hat{y}^{(1)} \\ \hat{x}^{(2)} \\ \hat{y}^{(2)} \\ \hat{x}^{(3)} \\ \hat{y}^{(3)} \\ \hat{x}^{(4)} \\ \hat{y}^{(4)} \end{bmatrix} \tag{4.5}$$

This form makes it possible to solve the problem through the least square method.

4.1.2 Results

Since all photos are taken from the same window, the perspective was assumed to be the same for all pictures. The four reference points used for the estimate are the two bottom corners of *Campo San Felice* and the two top corners of the street. Moreover, to better resemble the dimensions of the real square, the points were projected in a space of size 1300x5000. An example of a projected photo can be seen in Figure 4.1.

FIGURE 4.1: Original Image (left) and the Projected Image (right) obtained using *Homography*.

Clearly, the obtained images are not perfect, but they are a good approximation of a real bird's eye view. To make it even clear the plots in Figure 4.2 show a comparison of the original and adjusted coordinates of the photo.

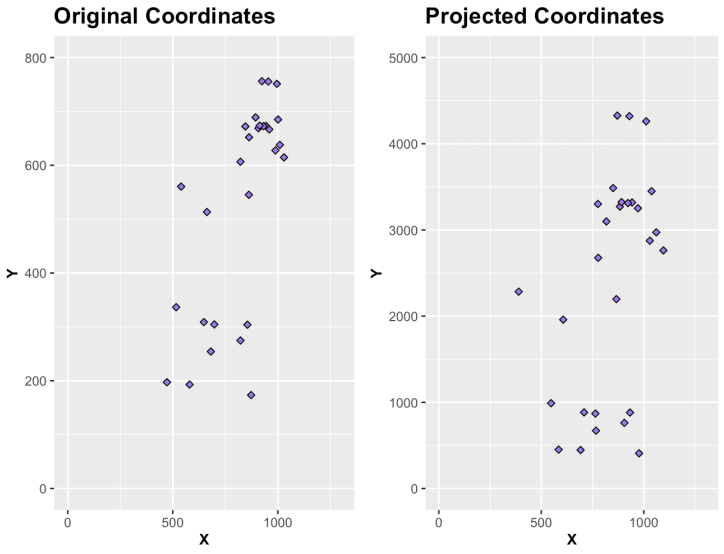

FIGURE 4.2: Original Coordinates and Projected Coordinates.

After projecting the original coordinates to a new one of different size, the y coordinates of the two graphs cannot be directly compared. However, if we consider the boundaries of the plots as representative of the borders of *Campo San Felice*, the improvements are noticeable. People are now distributed like they should be, especially on the y-axis. On the other hand, the horizontal spacing between individuals in the far background may be slightly underestimated, but overall, the final results are satisfactory.

It is important to notice that due to the limited view in the original perspective, the second half of the bridge remains hidden and cannot be taken into account. As a result, individuals standing at the end of the bridge are considered equivalent to those in the second half. This is particular evident in Figure 4.3 where individuals positioned at the top of the bridge are mistakenly placed at the end of it. Because of this, the distances between people on the bridge and people on the road behind is underestimated, moreover, people on the far back are pulled closer to the bridge. These, however, do not pose a problem since the clusters of people remain consistent, the only issue is that the coordinates do not exactly match those in the real square, but the analysis remains unaffected.

FIGURE 4.3: Map of the people in Campo San Felice on the 17th of May, 2020 in the evening.

4.2 Clustering

Now that the actual coordinates of the people in the photo have been extracted and the distance between them has been adjusted according to the prospective of the different pictures, it is possible to compute the clusters of pedestrian in the different pictures. To achieve this, *Density-Based Spatial Clustering of Applications with Noise* will be used. DBSCAN is a popular clustering algorithm widely used in data mining and machine learning introduced in 1996. Here, we will give only a brief introduction to how it works, for the details, see Ester et al. (1996). Note that DBSCAN can be employed in data spaces of any dimensionality. Additionally, any form of distance metric can be effectively utilized without sacrificing generality. Because of this, for proper visualization, all examples will be in 2D space using Euclidean distance.

4.2.1 DBSCAN

DBSCAN requires only two parameters: *Eps* and *MinPts*. The former is defined as the radius of the neighborhood, while the latter is the minimum number of points in an *Eps-neighborhood*.

Definition 1. The *Eps-neighborhood* of a point p, denoted by $N_{Eps}(p)$, is defined by

$$N_{Eps}(p) = \{q \in D | dist(p, q) \leq Eps\} \tag{4.6}$$

Definition 2. A point p is *directly density-reachable* from a point q with regard to Eps, $MinPts$ if

- $p \in N_{Eps}(q)$ and
- $|N_{Eps}| \geq MinPts$.

Definition 3. A point p is *density-reachable* from a point q with regard to Eps, $MinPts$ if there is a chain of points $p_1, ..., p_n$, $p_1 = q, p_n = p$ such that p_{i+1} is directly-reachable from p_i (Fig. 4.4).

Definition 4. A point p is *density-connected* to a point q with regard to Eps, $MinPts$ if there is a point o such that both, p and q are density-reachable from o with regard to Eps and $MinPts$ (Fig. 4.4).

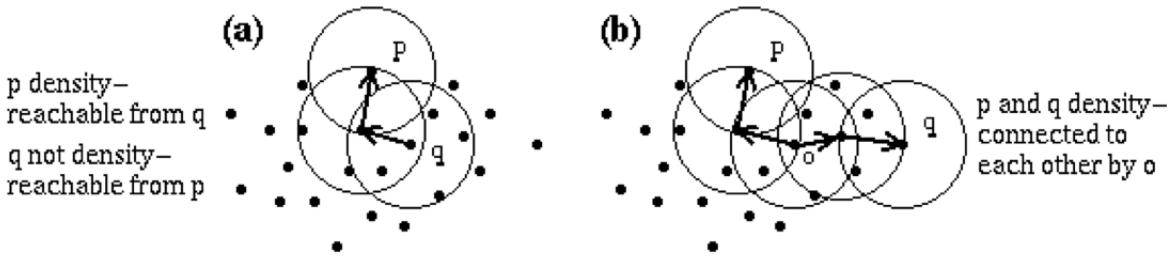

FIGURE 4.4: Density-reachability and density-connectivity. Image by Ester et al. (1996)

From these definitions we can distinguish

- *Region around a point*: within a distance *Eps*.
- *Density of a region*: more than *MinPts* in a region
- *Core point*: a point whose neighborhood contains at least *MinPts* points (Fig 4.5).
- *Border point*: a point whose neighborhood contains less than *MinPts* points, but it is density reachable from a core point (Fig 4.5).

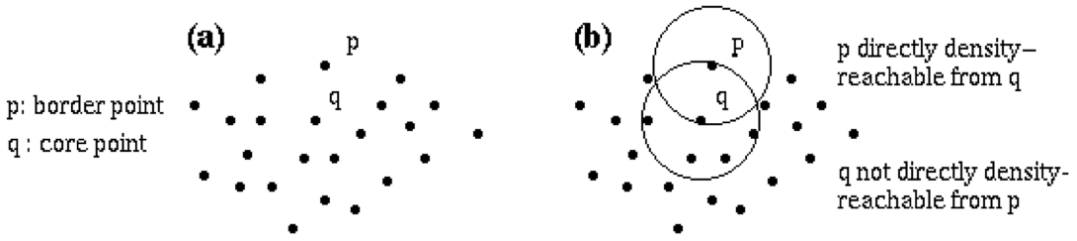

FIGURE 4.5: Core points and border points. Image by Ester et al. (1996)

Definition 5. Let D be a database of points. A *Cluster* C with regard to Eps and $MinPts$ is a non-empty subset of D satisfying the following conditions:

- $\forall p, q$: if $p \in C$ and q is density-reachable from p with regard to Eps and $MinPts$, then $q \in C$.
- $\forall p, q$: p is density-connected to q with regard to Eps and $MinPts$.

Definition 6. Let $C_1, ..., C_k$ be the clusters of the database D with regard to parameters Eps_i and $MinPts_i$, $i = 1, ..., k$. Then *noise* is defined as the set of points in the database D not belonging to any cluster C_i, i.e. noise $= \{p \in D | \forall i : p \notin C_i\}$.

Lemma 1. Let p be a point in D and $|N_{Eps}(p)| \geq MinPts$. Then the set $0 = \{o | o \in D$ and o is density-rachable from p with regard to Eps and $Minpts\}$ is a cluster with regard to Eps and $MinPts$.

Lemma 2. Let C be a cluster with regard to Eps and $MinPts$ and let p be any point in C with $|N_{Eps}(p)| \geq MinPts$. Then C equals to the set $O = \{o | o$ is density-reachable from p with regard to Eps and $MinPts\}$.

With this information, to find a cluster, DBSCAN starts with a point p chosen at random and retrieves all density-reachable points from p with regard to Eps and $MinPts$. If p is a core point, this procedure yields a cluster. If p is a border point, no points are density-reachable from p and DBSCAN visits the next point of the database. For the algorithm specifications, see Ester et al. (1996).

4.2.2 DBSCAN Results

As mentioned earlier, DBSCAN requires only two parameters. In this thesis, the parameter $MinPts$ was set to 1, as each individual walking down the street is considered as a separate cluster. The parameter Eps was set to approximately 1.4 meters, representing the radius within which two points are considered neighbors. The task is obviously not trivial, distance is not the only factor when considering groups of people, but in most

cases, it is a good approximation. For each photo contained in the dataset, the clusters were calculated (Fig. 4.6).

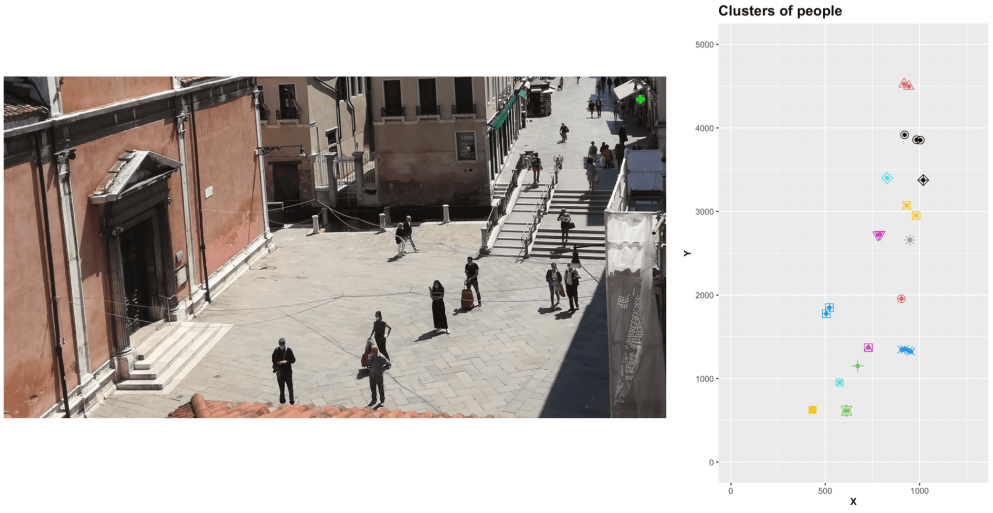

FIGURE 4.6: Example of the clusters detected in a photo.

Moreover, some additional features such as the cluster count and the average variance within value were computed for each photo. The first allows to examine the variation in the number of clusters throughout the pandemic (Fig. 4.7).

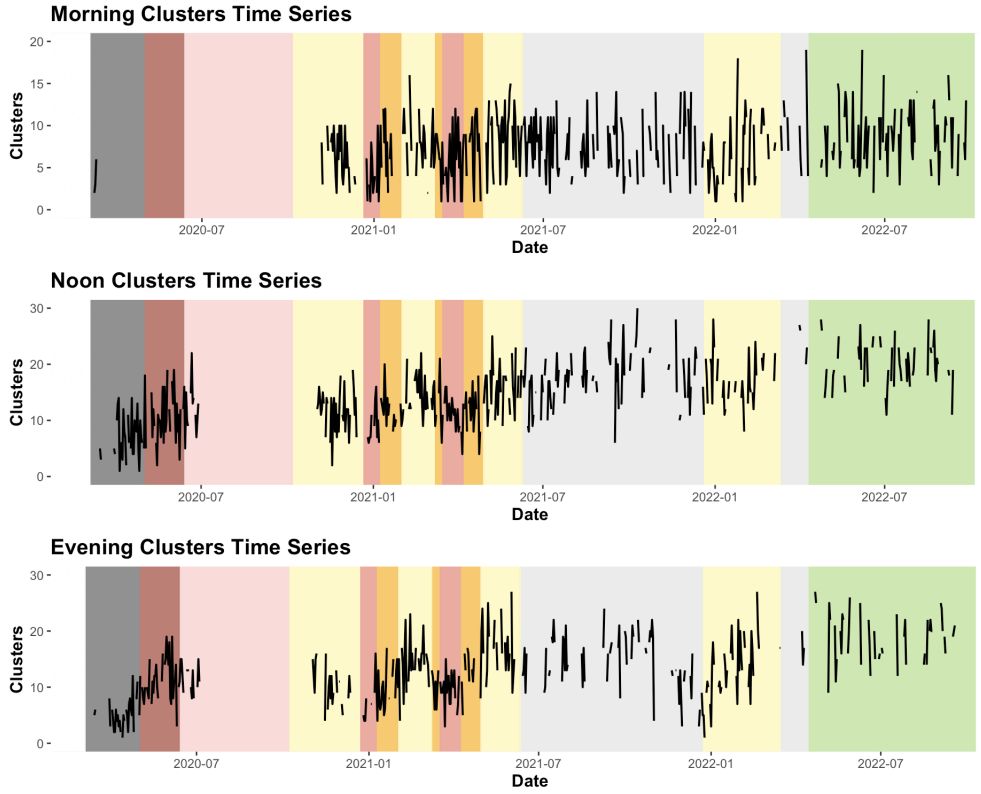

FIGURE 4.7: Time series of the number of clusters at different times of the day.

The second, instead, serves as an indicator of the dispersion or spread of data points within each cluster relative to the cluster's center. To obtain this metric for a single photo with K clusters, one needs to compute W_k, the variance within each cluster C_k :

$$W_k = \sum_{x_i \in C_k} ||x_i - \bar{x}_k||^2 \tag{4.7}$$

where x_i represents an element and $\bar{x}_k$ represents the center of cluster k.

Then, with the K values of W_k, one simply performs a weighted average to obtain W, i.e. the average variance within of the photo:

$$W = \frac{1}{n} \sum_{k=1}^{K} n_k W_k \tag{4.8}$$

with n and n_k indicating the total number of people in the photo and the number of people in cluster k respectively.

In this particular context, the higher value of average variance within the clusters mainly indicates that there are more clusters and that they are more numerous. This observation comes from the fact that the number of people inside the clusters tends to be low, and that most individuals are often found by themselves resulting in zero variance.

4.3 Cluster Time Series Analysis

In this chapter, the focus will be exclusively on the time series analysis of the average variance within metric. The analysis of the number of clusters can be found in the Appendix. This decision was made based on two primary reasons:

- Since the number of clusters strongly depends on the number of people in the photo, the series exhibit the same behaviour as those analysed in chapter 3 and yield similar and comparable results.
- The average variance within gives a more interesting insight on how the clusters evolved during the pandemic.

Furthermore, due to the redundancy between the plots and analysis, only the results from the DLM will be presented in this chapter. The results from the *Prophet* model can be found in the Appendix.

The time series are depicted in Figure 4.8 . It is important to note that these series contain a significant number of zero values, indicating that all individuals in a picture

are not clustered with others. In the case of the *Morning* time series, there are typical fluctuations, but the overall level appears to remain consistent over the two-year period.

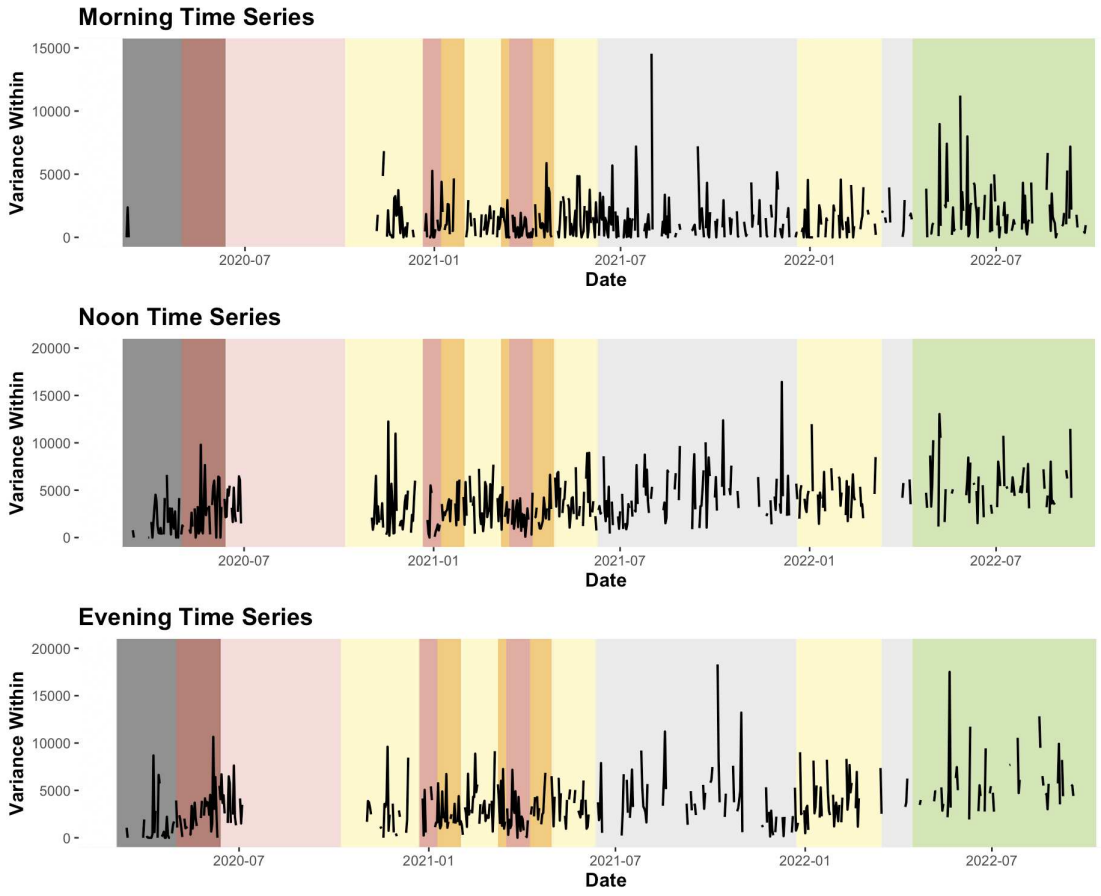

FIGURE 4.8: Time series of the average variance within in each photo.

Conversely, the other two time series exhibit an upward trend, indicating that the clusters got bigger and more numerous as the restrictions eased. This can be attributed to the fact that during the morning hours, the individuals present on the streets are predominantly workers. Consequently, it is unlikely that people gather with others at this time of the day.

To see if there actually is temporal correlation between the observations the correlograms are plotted in Figure 4.9. In contrast to the time series related to the number of people, the presence of autocorrelation is significantly lower in the current scenario. In particular, the *Morning* series exhibits minimal signs of autocorrelation. On the other hand, the other two series are comparable, displaying some level of autocorrelation and a subtle weekly seasonality, although not strongly pronounced.

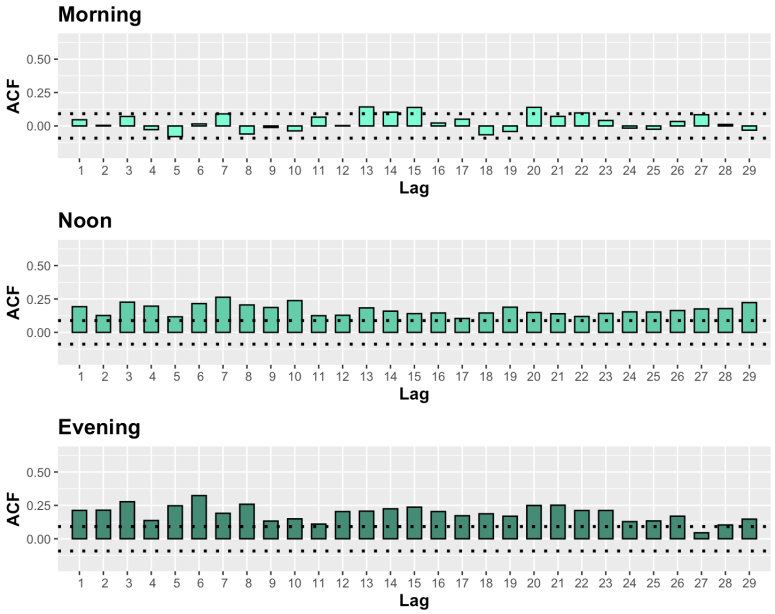

FIGURE 4.9: Correlograms of the three time series.

4.3.1 Morning

As for the previous analysis, the DLM setup for the *Morning* time series is the same: since the data do not show any particular pattern, the order of the polynomial component was set to 1. A weekly seasonality component was added and the matrix of regressors is composed by *mean_temperature*, *zona*, *rain* and *weekend_festive* variables. The first thing to look at is the smoothed trend (Fig. 4.10).

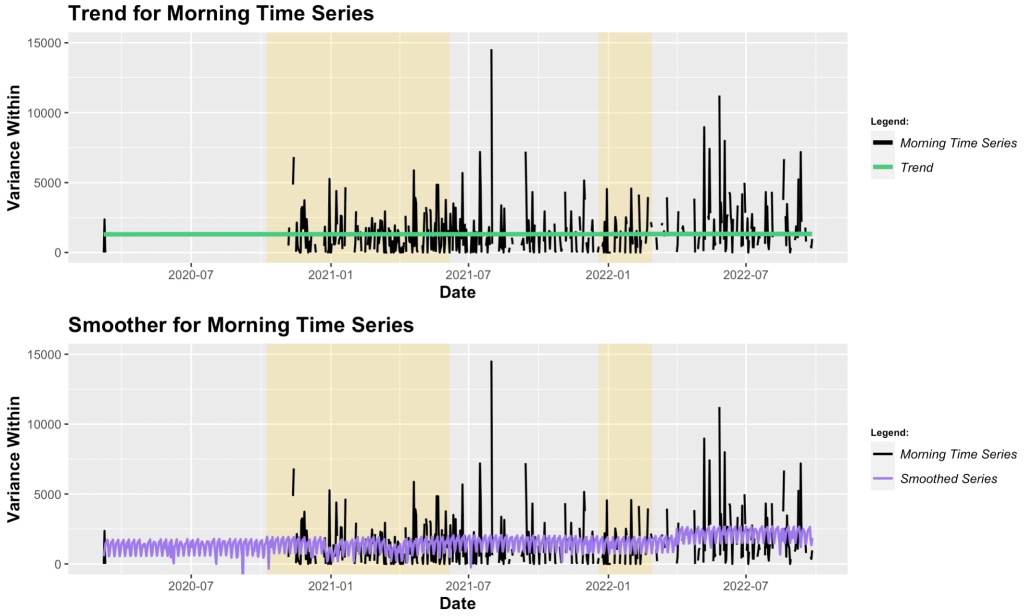

FIGURE 4.10: Trend of the average variance within for the Morning Time Series. The yellow background highlights the periods when Venice was in yellow zone or worse.

As anticipated, the model did not capture any specific pattern. The estimated trend appears as a nearly horizontal line slightly above zero, suggesting that the fluctuations observed are merely due to variability rather than a systematic pattern. Based on the smoothed series, it is evident that the impact of seasonality and regressors is not particularly significant. The estimated values do not deviate significantly from the estimated trend, indicating that the seasonal patterns and the influence of regressors have minimal effect on the overall trend.

Next, it is necessary to take a look at the one-step-ahead predictions in order to assess how the model performs in the forecasting task. The predicted values are plotted in Figure 4.11.

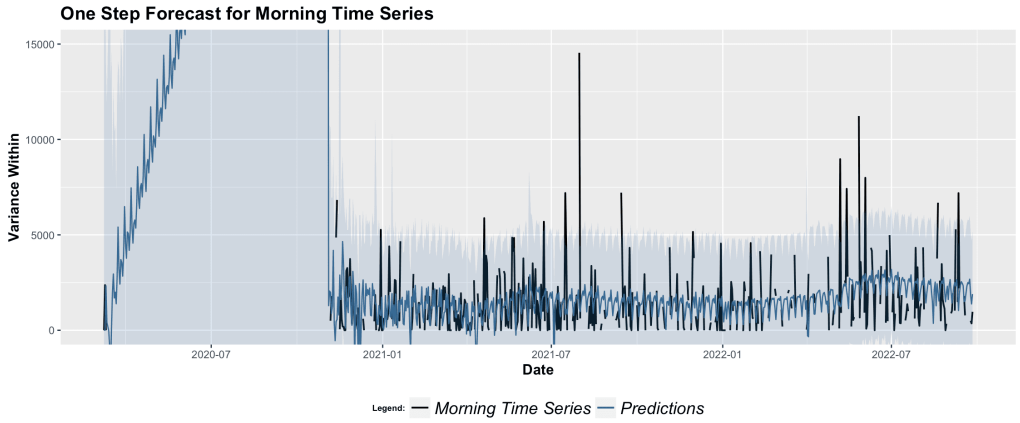

FIGURE 4.11: One-step-ahead predictions and intervals of the variance within for the Morning time series.

Aside from the first period, when the lack of observed values makes the model predict a constant increasing trend with huge variability, the predictions approximately follow the fluctations of the data. The majority of the actual data points fall within the confidence intervals, with only a few exceptions during the summer period when the average variance within reaches unusually high levels.

Finally, Figure 4.12 shows the distribution of the standardized residuals, along with the correlogram. The histogram exhibits a skewed distribution towards the right and even though the majority of values are close to zero, the mode is slightly under zero. As observed from the one-step-ahead predictions, the model tends to underestimate certain values. However, considering the relatively low number of people in each photo, it is expected to have a higher degree of variability. The correlogram does not show any significant indications of autocorrelation. However, it is worth noting that the original series did not display much autocorrelation either, making it difficult to assess the extent of improvements in this regard.

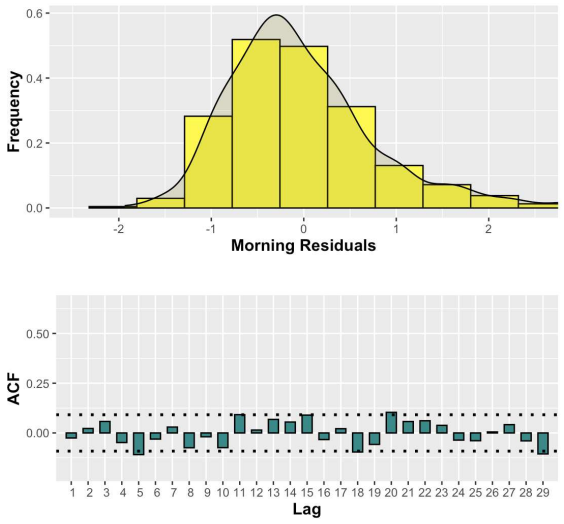

FIGURE 4.12: Residuals: histogram and ACF for the Morning Time Series.

4.3.2 Noon

Moving on with the *Noon* dataset, there are slight changes in the setup. While the matrix of regressors and the seasonal component remain unchanged, it was found that the polynomial component of order 2 provided the best results.

The trend plotted in Figure 4.13 demonstrates an overall increasing pattern in the series values over time. However, after the initial year and a half of the COVID-19 pandemic, the values appear to stabilize at a constant level around July 2021. The

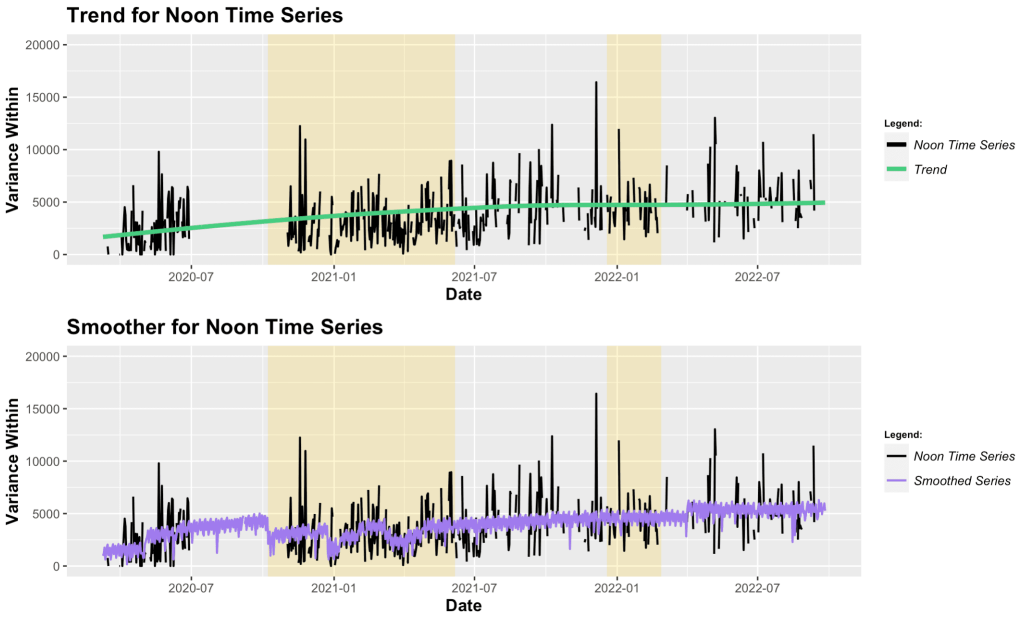

FIGURE 4.13: Trend of the average variance within for the Noon Time Series. The yellow background highlights the periods when Venice was in yellow zone or worse.

smoothed series reveals that occasional setbacks are captured by the seasonal and regressive components. However, their impact on the overall series is relatively limited since the values exhibit only minor deviations from the estimated trend. Furthermore, while the final result does not capture all the variability of the series, it successfully captures the general behavior of the data.

The forecasted values (Fig. 4.14) show a good fit, with most of the observed values falling inside the confidence intervals. Like in the *Morning* series, some of the peaks of the series are not well represented, however, considering the rarity of these values, they can be attributed to variability and randomness. In the first six months, as usual, the model predicts an increasing pattern in the series with large confidence intervals.

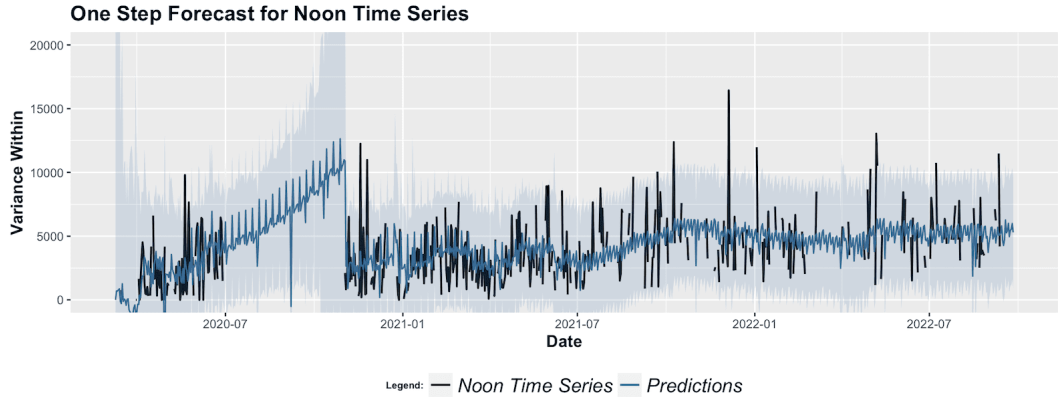

FIGURE 4.14: One-step-ahead predictions and intervals of the variance within for the Noon time series.

Finally, from the plotted *innovations* (Fig. 4.15) we can see that the behaviour is very similar to the previous series.

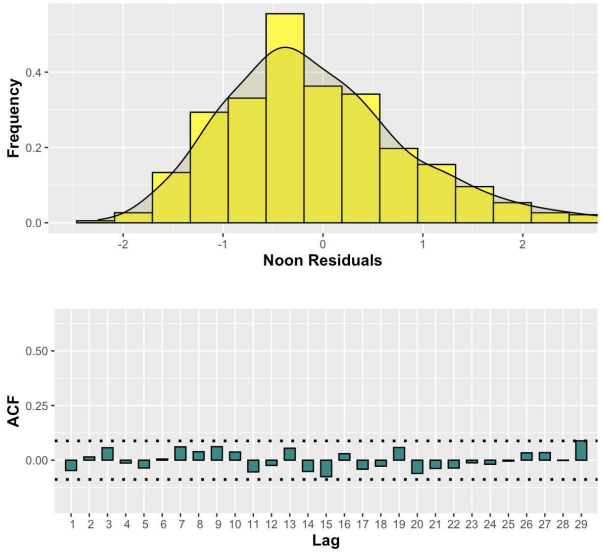

FIGURE 4.15: Residuals: histogram and ACF for the Noon Time Series.

The histogram clearly exhibits a noticeable right asymmetry, which aligns with the expectation after observing Figure 4.14 where certain values are significantly underestimated. Additionally, it is worth noting that the mode deviates significantly from zero, indicating that the model's predictive capabilities may not be optimal. On the other hand, the residuals exhibit a complete absence of correlation that was originally present in the data. None of the values at any lag demonstrate significance, suggesting that the model has effectively captured the underlying patterns within the data.

4.3.3 Evening

The final series to analyse is the one related to the pictures taken in the evening or a little later. As already seen in the previous chapter, this series is probably the hardest to model, given its high variability. The setup used for the *Dynamic Linear Model* is identical to the one used for the *Noon* dataset: the order of the trend component was set to two, the seasonality's period was set to 7, and the matrix of regressors contains the usual variables.

In this scenario, the results of the smoother differ significantly from the other two time series (Fig. 4.16).

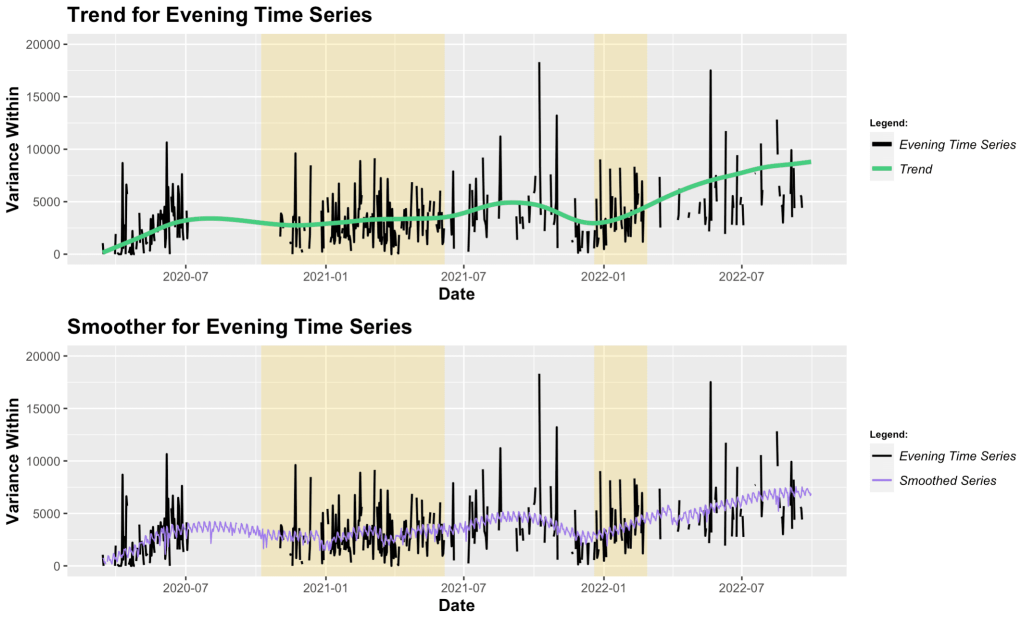

FIGURE 4.16: Trend of the average variance within for the Evening Time Series. The yellow background highlights the periods when Venice was in yellow zone or worse.

The estimated trend no longer follows a simple linear pattern, but instead effectively captures the diverse behavior and distinct patterns present in the observed data. In the final months the trend becomes a logarithmic-shaped line, but this can be attributed

to the low number of observed values. The seasonal and regressive components, on the other hand, appear to have minimal impact on the smoothed series like in the previous cases.

The one-step-ahead predictions, plotted in Figure 4.16, are comparable to those of the *Noon* series. The majority of observed values are well within the confidence intervals, indicating that the DLM captures the overall behavior quite accurately.

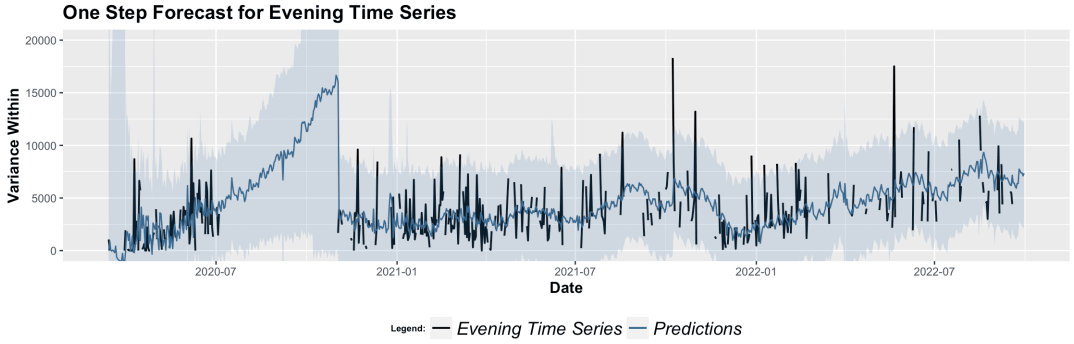

FIGURE 4.17: One-step-ahead predictions and intervals of the variance within for the Evening time series.

Finally, the standardized residuals display the typical right asymmetry, and there is no significant evidence of autocorrelation except for a lone value at lag five (Fig. 4.18).

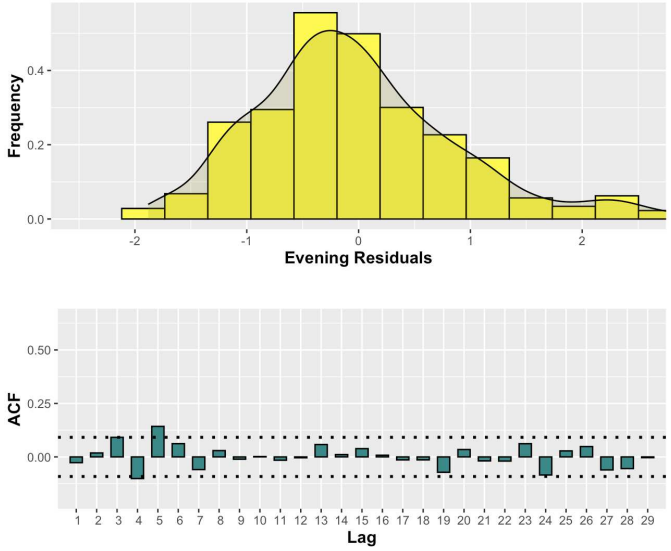

FIGURE 4.18: Residuals: histogram and ACF for the Evening Time Series.

4.4 Observations

Upon analyzing the three distinct series and achieving satisfactory outcomes, this section aims to provide an interpretation of the results to better understand what happened to the clusters of people at *Campo San Felice* in Venice during the COVID-19 pandemic.

First of all, it is important to notice that the three series show a high level of variability, this is clearly caused by the way the average variance within metric is built. In many instances, due to the prevalence of individuals walking alone, the variance within the computed clusters often results in a value of zero. Consequently, when these values are aggregated and averaged with the values from other clusters, it can lead to a wide range of outcomes. It is not uncommon, particularly in the *Morning* dataset, to encounter entire pictures whose metric is zero. Moreover, most of the clusters that are not composed by a single individual contain only a few people, mainly two or three, with only some unusual high values when lots of people gather on the bridge.

This is the main reason why the smoothed series, in all three cases, show a low impact of the seasonal and the regressive components, which are not fully able to predict the peaks and the setbacks of the observed data. It suggests that factors such as festive days, rain, and temperature do have an impact on the series but may not be strongly correlated with its sudden changes. As for the *zona* variable, its effect aligns with the findings presented in chapter 3, where the trend component already accounts for most of the variability. Nonetheless, the general behaviour of the three series is well captured by the DLM and by taking a look at the smoothed series, some conclusions can be drawn. Indeed, with the exception of the *Morning* series that exhibits a consistent trend with only minimal variations in values at the beginning and at the end of the pandemic, an upward trend in the average variance within can be observed over the years. For the reasons stated above this means that, in general, when the restrictions eased, people tendencies changed. Specifically, people went out more, formed more clusters and these clusters were bigger than those formed in the first months. Additionally, in the latter half of the considered period, there were no pictures in the *Noon* and *Evening* datasets that solely depicted clusters formed by a single person. Conversely, the number of such photos in the first and second halves are more comparable in the *Morning* dataset, due to the low presence of people in the photos, likely attributed to individuals being predominantly workers during those hours.

In terms of the forecasting ability of the *Dynamic Linear Models*, all three models perform quite well. They effectively capture the general pattern of the series, and the majority of observed values fall within the corresponding confidence intervals. However,

it should be noted that the estimated values tend to underestimate the actual values of the series, especially in cases where unusual high values occur. As already mentioned, this is due to the high variability in the data which makes it hard to predict changes that are so sudden. This is not necessarily a problem, in fact, while predicting the number of people can be valuable, for example for mobility-related purposes, the value of the average variance within is a more abstract number that tells about how the people at *Campo San Felice* are distributed. Therefore, achieving a perfect prediction is not necessarily crucial. However, it is still interesting to observe the evolution of the filter and assess the goodness of fit of the model.

Connected to the predictions are the residuals. Due to the reasons explained above, the residuals exhibit some positive high values, indicating an underestimation of certain values by the model. However, apart from the *Morning* dataset where no autocorrelation was present in the observed values, the DLM successfully eliminates any remaining autocorrelation in the residuals. This suggests that the model adequately accounts for the observed variation in the time series, leaving no unexplained patterns or information. Consequently, all three models pass the *Ljung-Box* test, with only the evening series getting a *p-value* of 0.02. Clearly, the *Shapiro* test for the normality is failed for the three series.

Conclusion

This thesis explored the application of *Dynamic Linear Models*, for the modeling of time series data. Particularly, this type of model was applied to data regarding the affluence of people at *Campo San Felice* in Venice during the COVID-19 pandemic. In this context, characterized by the presence of missing values and abrupt pattern changes in the series, DLMs have demonstrated their effectiveness in capturing the underlying patterns and dynamics of the various series. With their recursive nature, *Dynamic Linear Models* offer an efficient and flexible approach for modeling such data, making them preferable over other forecasting models like Prophet, as discussed in this study. Moreover, DLMs offer an interpretive framework that allows for a better understanding of the impact of various components on the observed values. Specifically, several *Dynamic Linear Models* were employed. These models were fitted both with and without the inclusion of various variables, and their *Akaike Information Criterion* was computed. The results are summarised in Table 1.

. **AIC improvements**

Dataset	*mean_temperature*	*rain*	*weekend_festive*	*zona*
Morning	−26.80	142.76	4248.01	15787.93
Noon	9.35	1061.99	6643.75	91285.26
Evening	1735.86	980.51	2266.89	61499.02

TABLE 1: Improvements in terms of AIC of the different regressors

Clearly, there are significant changes in the number of people gathering at *Campo San Felice* over time, with factors such as weather conditions, weekdays, holidays, and notably, the COVID-19-related restrictions imposed by the Italian government, all playing a role. Specifically, temperature and rainfall have a minimal effect on the *Morning* dataset since individuals are probably workers, and, as such, they are not affected by the weather conditions. On the other hand, the *weekend_festive* regressor demonstrates a substantial impact on all datasets, indicating that weekends have a distinct distribution of people compared to weekdays. The most significant difference, however, is observed

when including the *zona* variable. This suggests that considering the government restrictions greatly enhances the model's ability to explain the variation in the number of people at Campo San Felice, providing further confirmation of the profound influence that COVID-19 had on mobility and social gatherings. The same methodology was also employed to examine the clusters formed by individuals, confirming the aforementioned findings.

Further improvements in this field may regard the implementation of *Generalized Dynamic Linear Models* (G-DLMs). Although the theoretical aspects of G-DLMs have been discussed in the literature, there is a need for their intuitive implementation in popular programming languages such as R and Python. This would enable researchers and practitioners to apply G-DLMs more effectively and explore their potential in various domains. While the models employed in this thesis demonstrated good performance, it is important to note that they assume the response variable follows a Gaussian distribution. However, considering the context of this study, a Poisson distribution would have been more suitable for modeling the data accurately.

The analysis conducted in this thesis could also be examined from a Bayesian perspective. While the current study employed a frequentist approach, adopting a Bayesian framework offers an alternative viewpoint that may yield distinct insights into the data. By employing Bayesian methods, one can incorporate prior knowledge or beliefs about the data and model parameters. This prior information acts as an initial reference point and can influence the posterior distribution, allowing for a more nuanced understanding of the underlying patterns and uncertainties present in the data.

An alternative approach for improvement could involve employing multivariate analysis techniques to study the interdependencies and dynamics among the *Morning*, *Noon*, and *Evening* time series collectively. By considering these series as a multivariate dataset, one can explore the relationships and interactions between the variables observed at different times of the day.

Bibliography

Azzalini, A. & Scarpa, B. (2012). *Data analysis and data mining: An introduction*. OUP USA.

Bloomenthal, J. & Rokne, J. (1994). Homogeneous coordinates. *The Visual Computer* **11**, 15–26.

Bochkovskiy, A., Wang, C. & Liao, H. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *CoRR* **abs/2004.10934**.

Dollár, P., Singh, M. & Girshick, R. B. (2021). Fast and accurate model scaling. *CoRR* **abs/2103.06877**.

Durbin, J. & Koopman, S. J. (2012). *Time series analysis by state space methods*, vol. 38. OUP Oxford.

Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96. AAAI Press.

Lee, Y., Hwang, J., Lee, S., Bae, Y. & Park, J. (2019). An energy and gpu-computation efficient backbone network for real-time object detection. *CoRR* **abs/1904.09730**.

Liu, D. C. & Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming* **45**, 503–528.

Long, X., Deng, K., Wang, G., Zhang, Y., Dang, Q., Gao, Y., Shen, H., Ren, J., Han, S., Ding, E. & Wen, S. (2020). Pp-yolo: An effective and efficient implementation of object detector.

Ma, N., Zhang, X., Zheng, H. & Sun, J. (2018). Shufflenet V2: practical guidelines for efficient CNN architecture design. *CoRR* **abs/1807.11164**.

Nocedal, J. & Wright, S. J. (1999). *Numerical optimization*. Springer.

Petris, G., Petrone, S. & Campagnoli, P. (2009). *Dynamic linear models with R*. Springer Science & Business Media.

Redmon, J., Divvala, S. K., Girshick, R. B. & Farhadi, A. (2015). You only look once: Unified, real-time object detection. *CoRR* **abs/1506.02640**.

Shumway, R. H., Stoffer, D. S. & Stoffer, D. S. (2000). *Time series analysis and its applications*, vol. 3. Springer.

Sturm, P. (2014). *Pinhole Camera Model*. Boston, MA: Springer US, pp. 610–613.

Taylor, S. J. & Letham, B. (2018). Forecasting at scale. *The American Statistician* **72**, 37–45.

Wang, C., Bochkovskiy, A. & Liao, H. M. (2020). Scaled-yolov4: Scaling cross stage partial network. *CoRR* **abs/2011.08036**.

Wang, C.-Y., Bochkovskiy, A. & Liao, H.-Y. M. (2022). Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.

Appendix

Number of Clusters Analysis

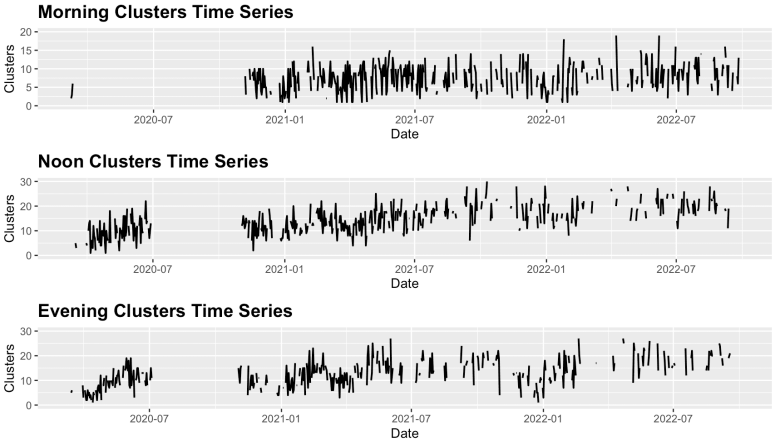

FIGURE .1: Number of clusters time series

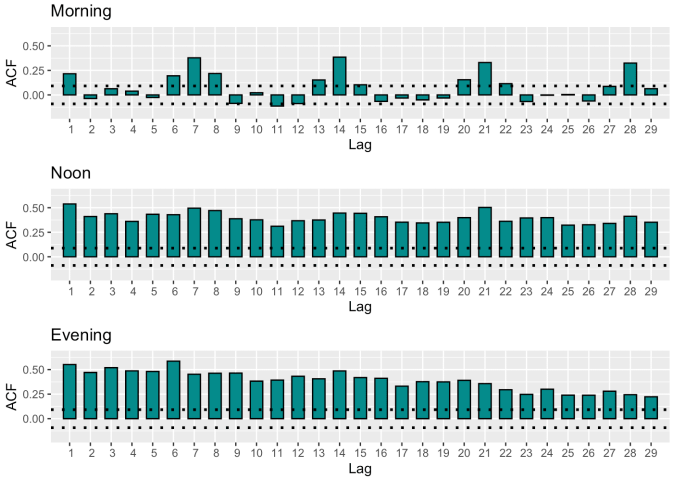

FIGURE .2: Correlograms for the number of clusters time series

Prophet Results

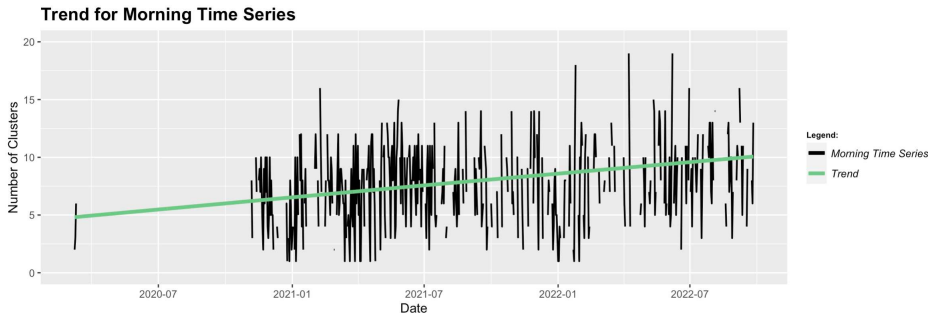

FIGURE .3: Estimated trend for Morning time series.

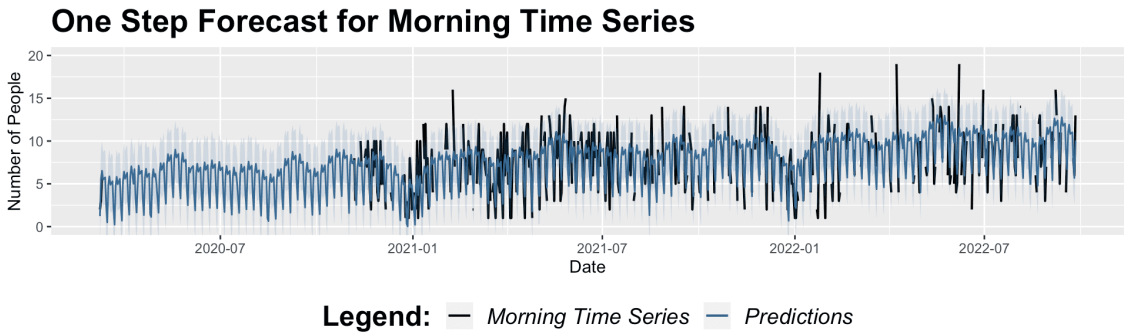

FIGURE .4: Predictions for Morning time series.

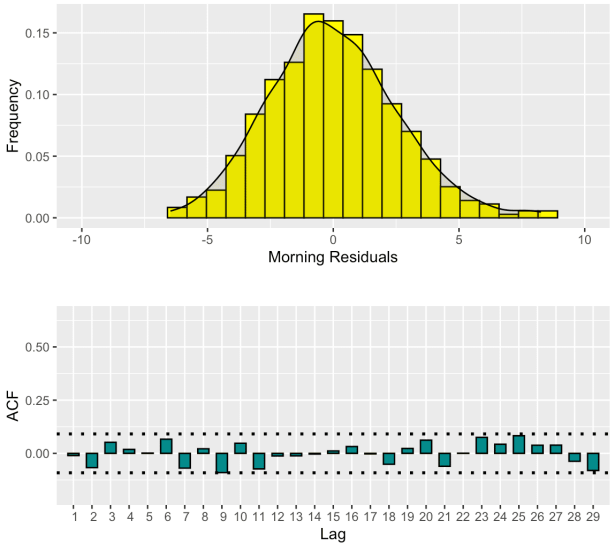

FIGURE .5: Residuals for Noon time series.

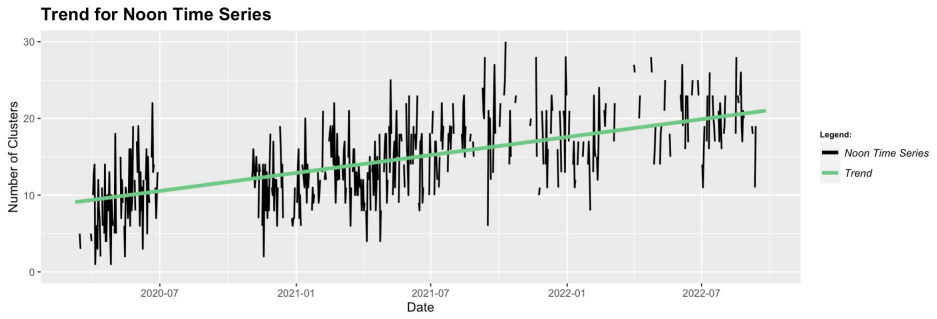

FIGURE .6: Estimated trend for Morning time series.

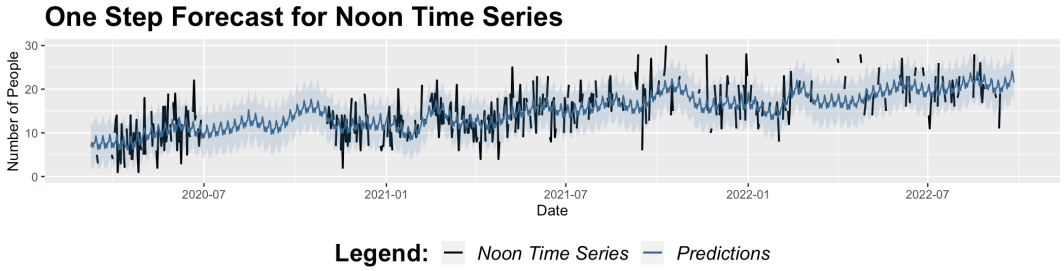

FIGURE .7: Predictions for Noon time series.

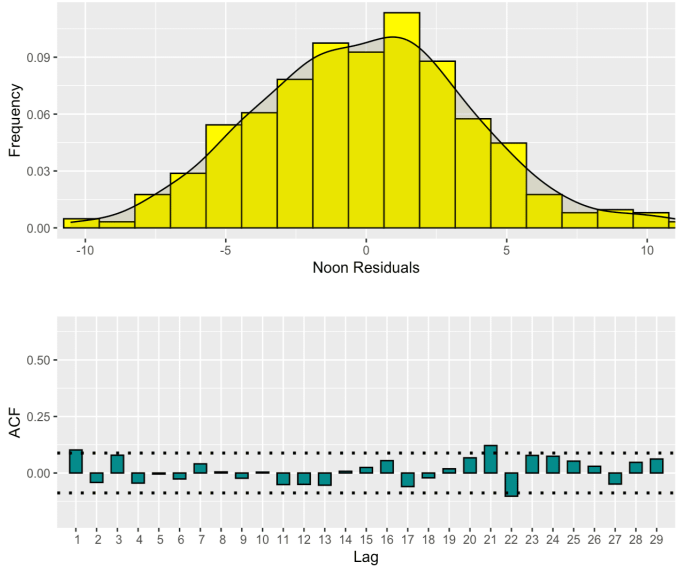

FIGURE .8: Residuals for Noon time series.

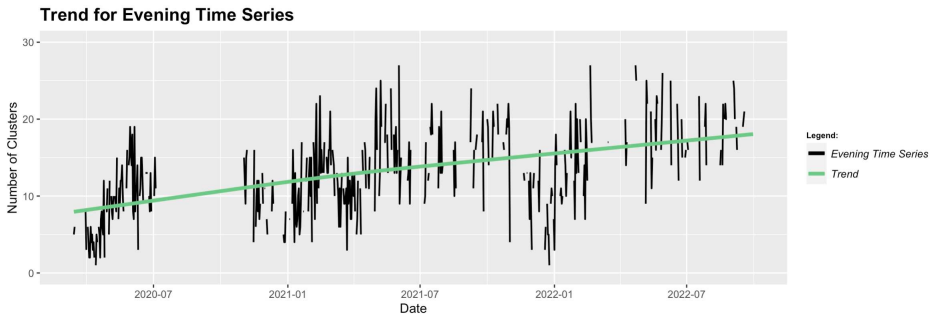

FIGURE .9: Estimated trend for Evening time series.

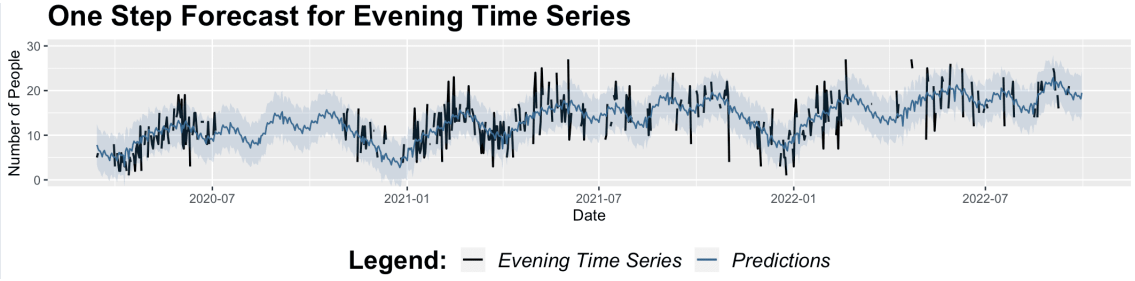

FIGURE .10: Predictions for Evening time series.

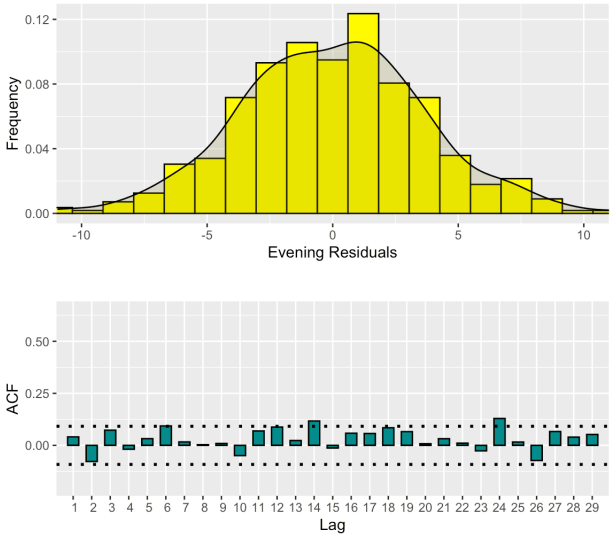

FIGURE .11: Residuals for Evening time series.

DLM results

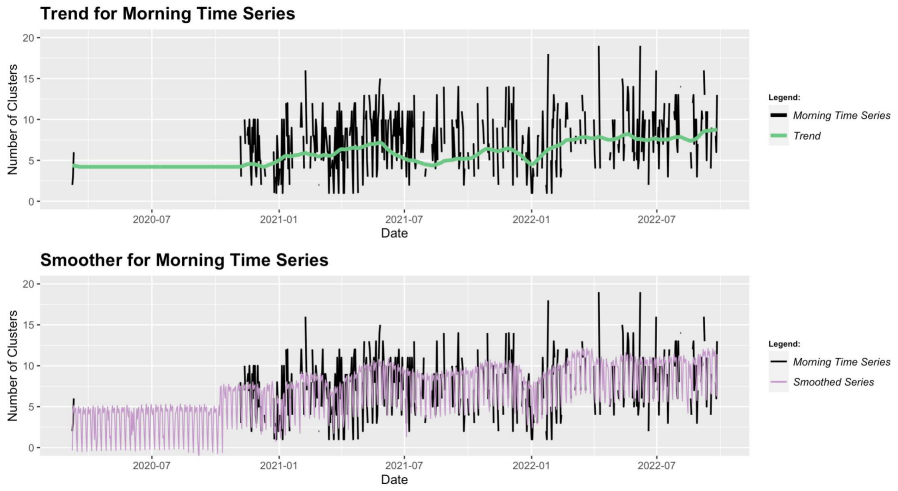

FIGURE .12: Estimated trend for Morning time series.

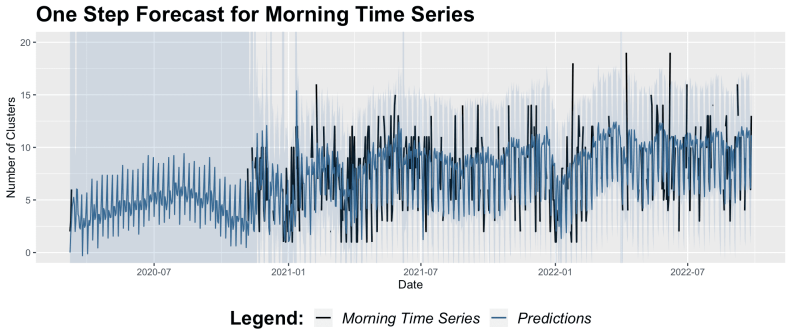

FIGURE .13: Predictions for Morning time series.

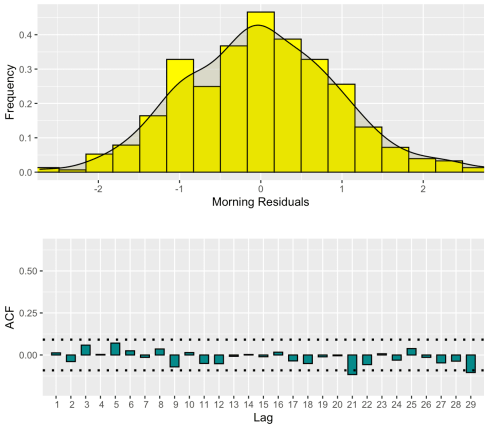

FIGURE .14: Residuals for Noon time series.

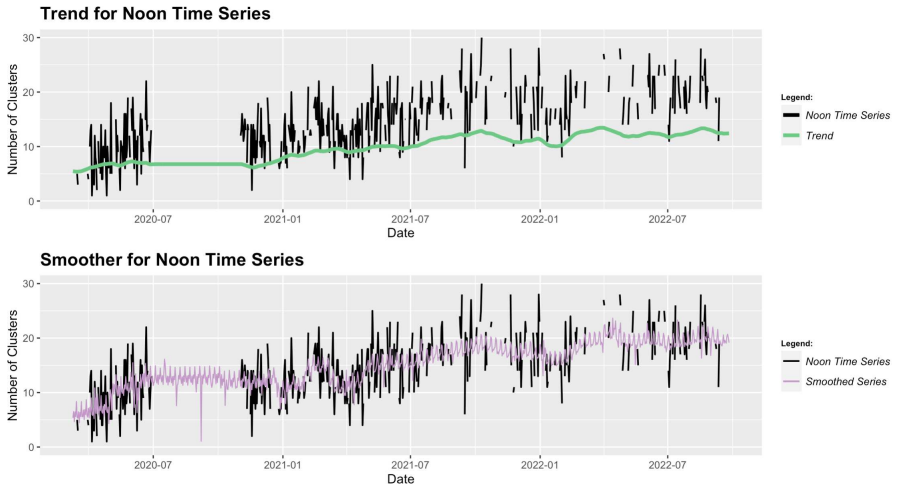

FIGURE .15: Estimated trend for Morning time series.

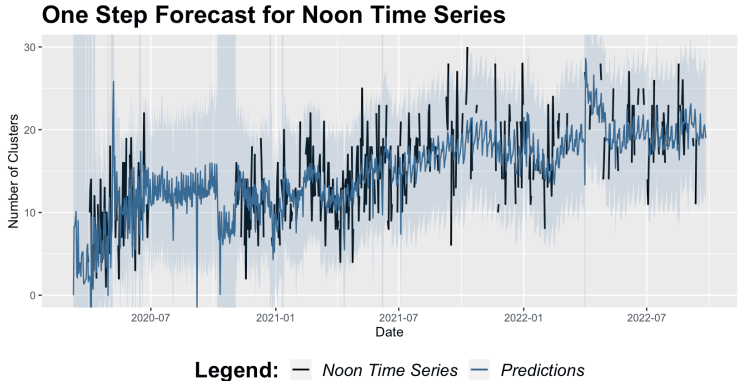

FIGURE .16: Predictions for Noon time series.

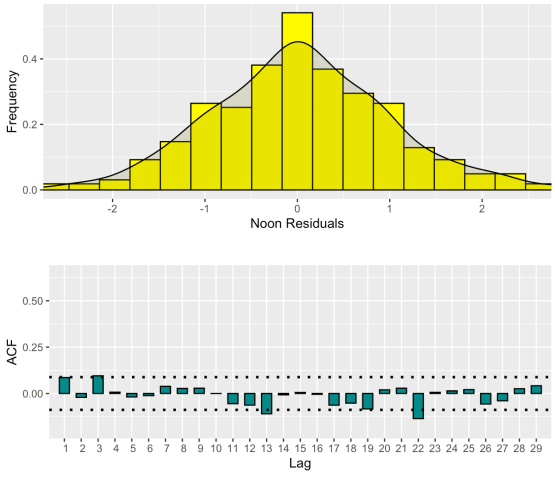

FIGURE .17: Residuals for Noon time series.

FIGURE .18: Estimated trend for Evening time series.

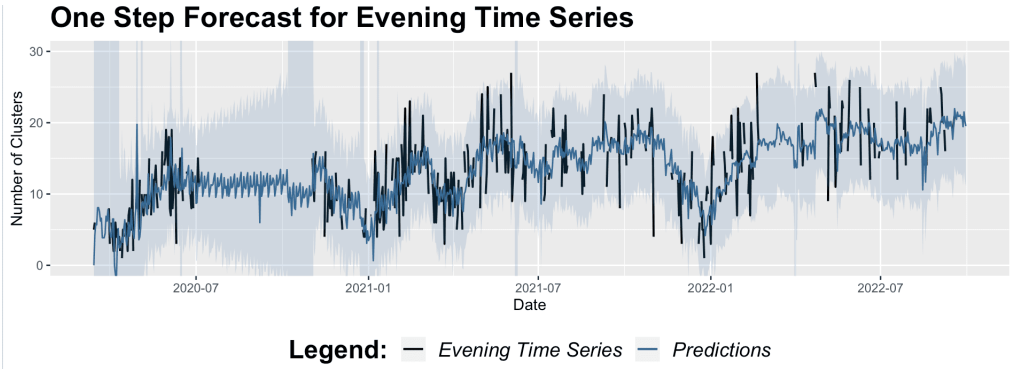

FIGURE .19: Predictions for Evening time series.

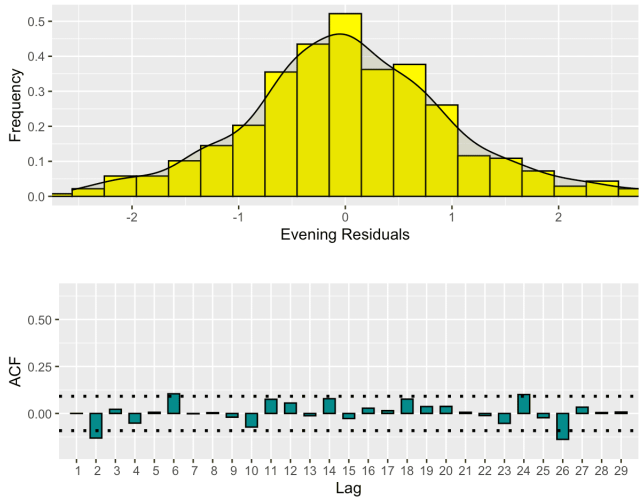

FIGURE .20: Residuals for Evening time series.

Prophet results for Average Variance Within series

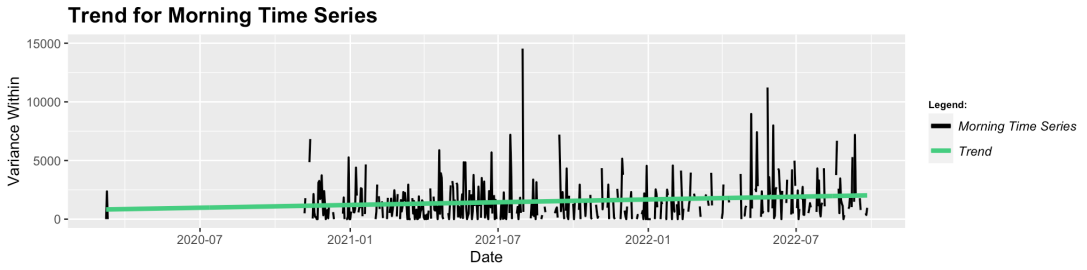

FIGURE .21: Estimated trend for Morning time series.

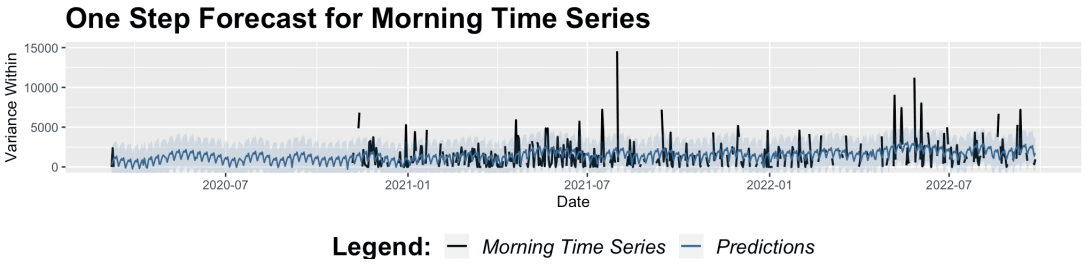

FIGURE .22: Predictions for Morning time series.

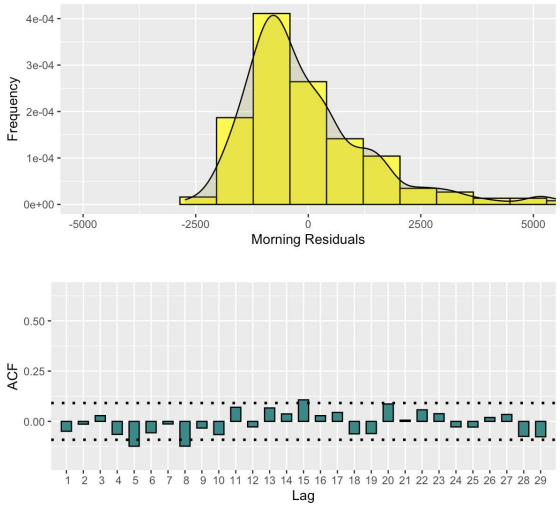

FIGURE .23: Residuals for Morning time series.

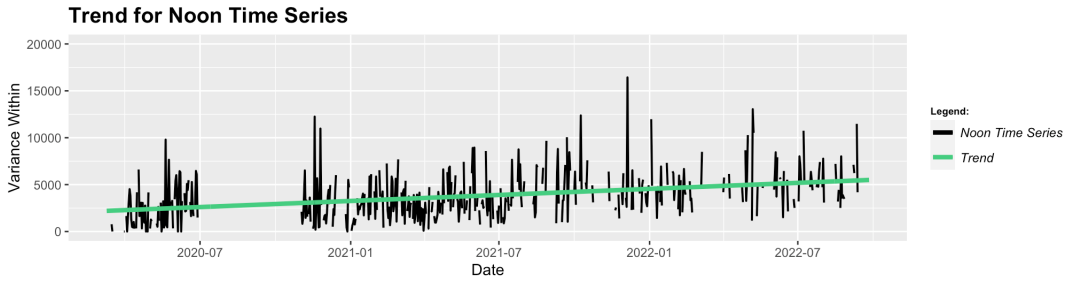

FIGURE .24: Estimated trend for Noon time series.

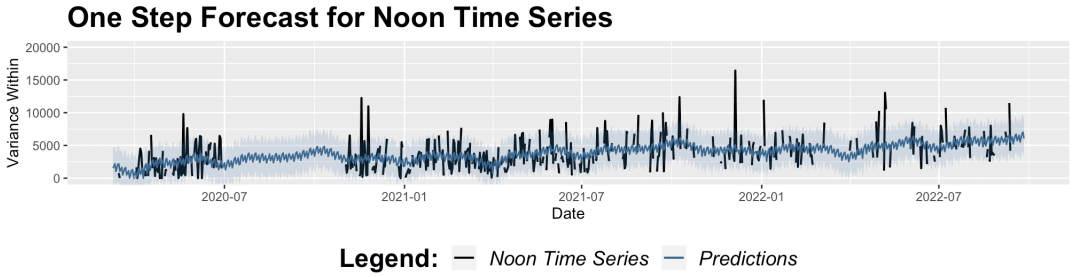

FIGURE .25: Predictions for Noon time series.

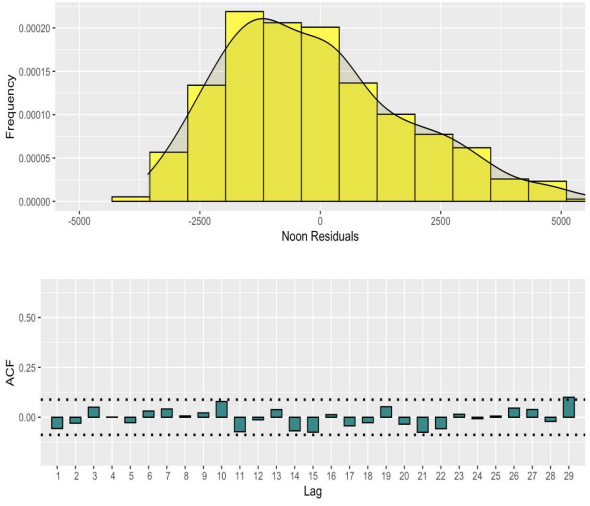

FIGURE .26: Residuals for Noon time series.

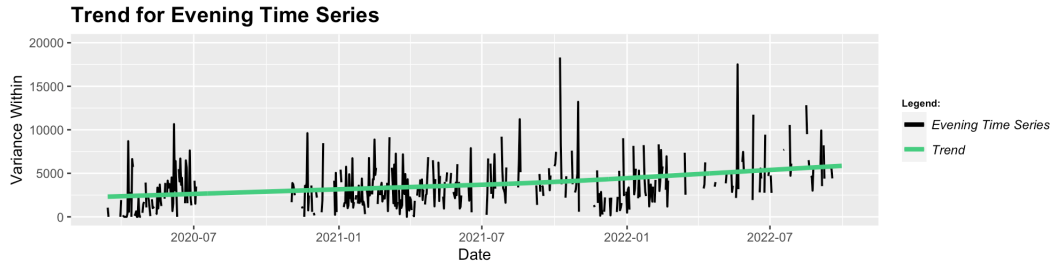

FIGURE .27: Estimated trend for Evening time series.

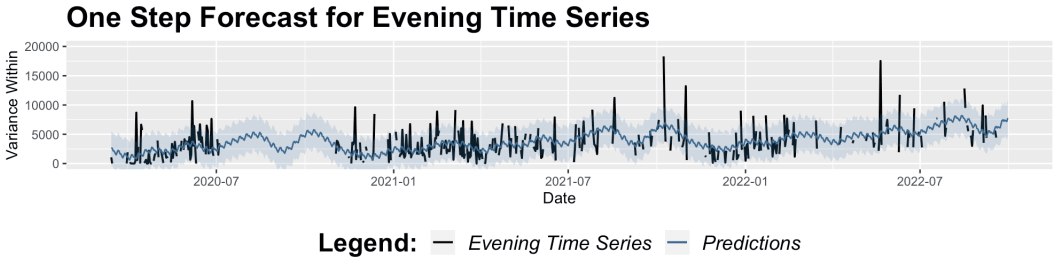

FIGURE .28: Predictions for Evening time series.

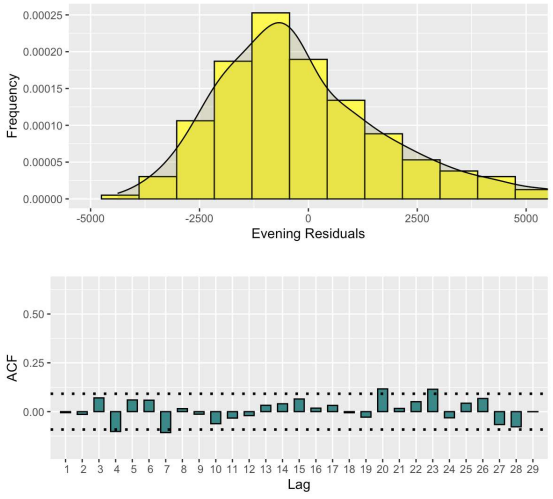

FIGURE .29: Residuals for Evening time series.