MASTER'S THESIS

Interuniversity Master in Statistics and Operations Research UPC-UB

Title: **Implementing statistical methods to solve DDMRP main issues**

Author: **Núria Xifré Martín**

Advisor: **Lesly Acosta**

Department: **Statistics and Operations Research**

University: **Polytechnic University of Catalonia**

Academic year: 2021-2022

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

UNIVERSITAT DE BARCELONA

Facultat de Matemàtiques i Estadística

Màster en Estadística i Investigació Operativa

Implementing statistical methods to solve DDMRP main issues

Author:
Núria Xifré Martín

Advisor:
Lesly Acosta Argueta

Barcelona, June 1, 2022

Contents

List of Figures

List of Tables

Acronyms

ADU Average Daily Usage. 5–7, 13, 15, 24, 25, 27–29, 43, 50, 51, 54

DDMRP Demand Driven Material Requirements Planning. 6–8, 11, 13, 14, 23, 24, 29, 42, 50, 51, 54, 55

DLT Decoupled Lead Time. 13, 24, 25, 27, 29, 44, 45, 54

DRP Distribution Requirements Planning. 11

ERP Enterprise Resource Planning. 6, 51

GAM Generalized Additive Model. 18, 19

KPI Key Performance Indicator. 23

LTF Lead Time Factor. 50, 51

MAPE Mean Absolute Percentage Error. 23, 31, 40, 43

MRP Material Requirements Planning. 8–11, 50, 55

MSE Mean Square Error. 23

NFP Net Flow Position. 54, 55

OSH Order Spike Horizon. 54

OST Order Spike Threshold. 54, 55

TOG Top of Green. 55

VF Variability Factor. 50, 51

Abstract

This master thesis analyses the supply chain methodology called DDMRP and its known issues. This system is based on Stock Buffers: product storages used order to reduce the variability of demand and supply. However, DDMRP presents two main matters for the stock planners of the companies. The first one is the computation of the ADU, which is estimated by a moving average and gives inaccurate forecasts. The second one is the sizing of the Red Zone of the Stock Buffers, which depends on two factors that are fixed by the planner and can distort the safety stock units. The goal of the study is to find alternatives to these two issues. To compute the ADU more accurately, three time series methodologies are proposed: ARIMA, Holt-Winters and Prophet. In the results, it can be seen that the proposed techniques perform much better than the moving average. Holt-Winters filter is the method chosen as the best one to estimate the ADU, as it gives good results and it is feasible to implement in a production environment. On the other hand, the Red Zone size is computed minimizing a formulated cost function. The cost function takes into account the Average Daily Usage and the cost/profit ratio of the product. Using a Montecarlo simulation which take ADU errors as the random variable , the number of stock units that minimizes the cost function can be found. With this procedure the planner would not need to establish any parameter, only known values of the product.

Chapter 1

Introduction

1.1 Justification

In July 2019 I started working on my current company, ForgeFlow. ForgeFlow is a consultant firm specialized in Supply Chain Management and Enterprise Resource Planning(ERP) implementation. The projects we are involved in include the Demand Driven Material Requirements Planning (DDMRP) methodology for Supply Chain and Odoo as an ERP.

An ERP is a software that a company uses to organize all its tasks: accounting, human resources, purchases, sales, etc. Every big or medium company needs an ERP to have all its information centralized and connected. There are multiple of ERP systems, but Odoo's particularity is that it is open source, so everybody can develop its code. ERPs and Supply Chain methodologies are strongly related, as in practice, the Supply Chain methodology is introduced in the ERP so company stock planners can have an easy tool to visualize their stock needs.

In June 2020, I obtained the certification of Demand Driven Planner Professional by the Demand Driven Institute (https://www.demanddriveninstitute.com), which gave me the knowledge to successfully implement Demand Driven Material Requirements Planning to a company.

As I am very familiar with DDMRP, I am able to identify its advantages and its drawbacks. Among its drawbacks there is the calculation of the Average Daily Usage (ADU)[1]. The ADU is one of the cornerstones of DDMRP, but it is usually computed using a simple rolling

[1]The ADU is the daily rate of use of a specific product

mean. The goal is to forecast the ADU with a more accurate model which takes into account more components.

Another controversial step in DDMRP is the sizing of the Buffers. According to this methodology, the Stock Buffers are divided in three zones: the red, the yellow and the green [2]. To establish the limit of the Red Zone, which is the most important one, it is necessary to have previously computed the ADU and 2 factors: Lead Time Factor and Variability Factor of the product. These two factors can go from 0 to 1 and are determined by the planner of the company. The second goal of the thesis is to properly define a sizing for the Red Zone which do not depend on these factors.

1.2 Aim

As before mentioned, the aim of the thesis is to improve Demand Driven Material Requirements Planning methodology, solving two main problematic issues, such as the calculation of the average daily usage Average Daily Usage and the estimation of Buffer Red Zone.

1.3 Scope

To achieve the goals mentioned in the previous section, the following points are developed along the thesis:

- Research of possible methodologies to perform an accurate computation of the ADU.
- Implement an ARIMA model to compute the ADU
- Implement a Facebook Prophet model to compute the ADU.
- Implement a Holt-Winters method to calculate the ADU.
- Compare all the methods implemented and select the best one to estimate the ADU of a product.
- Research of methodologies to approximate the dimensions of the Red Zone of the Stock Buffer.

[2]The green zone represents the replenishment stock, the yellow zone is the normal usage of stock and the red zone consists of the satefy stock.

Chapter 2

Development

This chapter is divided in three sections: the state of art, the methodology and the statistical analysis. In the first section, the supply chain techniques of Material Requirements Planning and Demand Driven Material Requirements Planning are explained, to understand the context of the thesis. In the second part, the purposed methodologies to solve DDMRP issues are explained. Finally, in the last section, the implementation results of these methodologies on a real database are shown.

2.1 State of the Art

In this section, the Demand Driven Material Requirements Planning are explained, as well as the issues which are controversial in the companies that implement this methodology.

2.1.1 Material Requirements Planning

Demand Driven Material Requirements Planning is itself an innovation from Material Requirements Planning, which has been the most used supply chain methodology since its release in 1975. Material Requirements Planning is devoted to calculate which materials have to be re-supplied and the needed quantity of each one, in order to not run out of stock in any circumstance.

To illustrate better Material Requirements Planning and Demand Driven Material Requirements Planning concepts, a made up example is explained. This example is based on Wood Corner, a company which manufactures furniture. It buys pieces such as wood pan-

els or plastic laminate and produces wardrobes, tables or chairs, that are sold to furniture stores. There are some materials or sub-assemblies that are used to manufacture different final products. To simplify the company materials structure, only two products are taken into account in the scheme below:

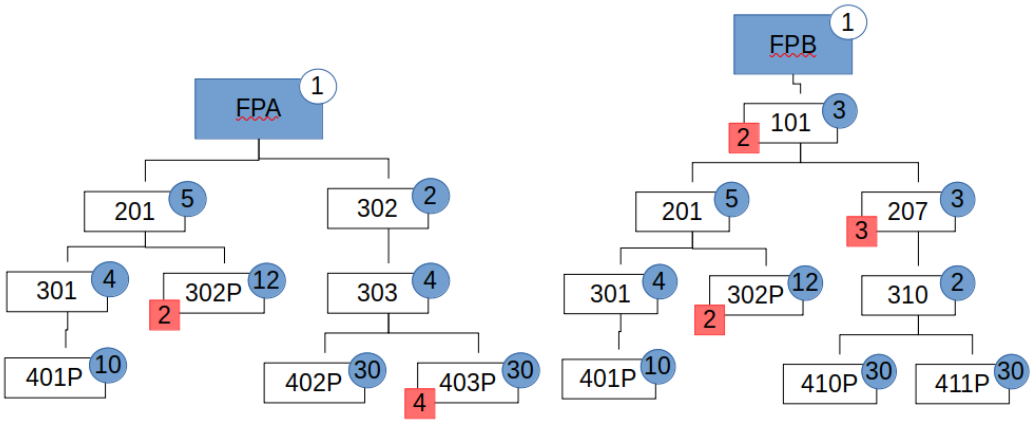


Figure 2.1: Bill of materials of products FPA and FPB. Reference: [1]

The diagram above represents the bill of materials of 2 final products: FPA and FPB. As it can be seen, the sub-assembly 201 is used to manufacture both products. The number in the circle next to each product represents its lead time [1]. The number in the red box, that some products have, is the number of units needed to build the next assembly.

Following the example, Wood Corner receives two orders: 3 units of FPA for the 28 th October and 5 units of FPB for 2 nd November. Up to this point, the Material Requirements Planning program would display a table similar to the following one:

[1]Time between the start and the end of a manufacturing or purchasing process of a product in a supply chain.

Product	Qty. in Stock	Qty. on Demand	Qty. on Supply	Qty. to procure	Release Order Date	Action
FPA	1	3	0	2	27/10	Manufacture
FPB	2	5	0	3	01/11	Manufacture
101	3	10	1	6	29/10	Manufacture
201	5	2	2	0	-	Manufacture
201	3	12	2	7	24/10	Manufacture
301	4	7	0	3	20/10	Manufacture
401P	0	3	2	1	10/10	Purchase
302P	2	14	5	7	12/10	Purchase
302	0	2	0	2	25/10	Manufacture
303	0	2	0	2	21/10	Manufacture
402P	9	2	0	0	-	Purchase
403P	0	8	4	4	21/09	Purchase
207	2	18	6	10	26/10	Manufacture
310	2	10	0	8	24/10	Manufacture
410P	15	8	5	0	-	Purchase
411P	6	8	0	2	24/09	Purchase

Table 2.1: Material Requirements Planning procurement table.

In the table above, it is important to pay attention to the columns of *Quantity to procure* and *Order Release Date*. The planner of Wood Corner follows these data to make the manufacturing and purchasing orders. In the column *Qty. of Stock* there appears the product quantity that Wood Corner has in its warehouse. *Qty. on Demand* represents the current demand of each product. One of the main advantages of Material Requirements Planning is that does not reserve a component for a specific final product, it assigns each sub-assembly to the most urgent requirement. In this case, FPA has to be delivered first, so the current quantities in stock of product 201 is used to build FPA. The column *Quantity on Supply* represents the quantity of the product which is pending to be manufactured or be to received. Finally, *Qty. to procure* corresponds to *Qty. on Demand* minus *Qty. in Stock* and minus *Qty. on Supply*.

From this point of view, it seems that Material Requirements Planning is quite a good system, as it gives the information to not run out of stock or accumulate too much products in the warehouse. However, it has been proven that Material Requirements Planning is not a perfect system, as it is quite limiting and the most part of companies end up customizing their Material Requirements Planning to avoid the *nervousness* [2].

Actually, companies do not input on the MRP system their actual demand, but their forecasts. As it can be seen in table 2.1, both products need to be planned with nearly 2 months in advanced, and maybe the customer is not willing to wait. For this reason, companies use forecasts and then demand is adjusted when actual orders arrive at the company. Constantly changing the demand source is what causes the *nervousness*. This

[2]According to the APICS Dictionary [5], nervousness represents the significant timing or quantity changes that affect on lower levels (raw materials) due to minor changes in higher levels (final products)

is mainly caused by the extremely dependent structure of the MRP.

Another issue of Material Requirements Planning is the distortion of relevant materials. As it has been seen in the table 2.1, Material Requirements Planning assumes full allocation, no order should be started unless all components are available. This means that if a component is delayed, the planned order is also delayed, and this delay is accumulated along all the bill of materials of a final product. Again, this a direct consequence of the dependence in the MRP.

2.1.2 Demand Driven Material Requirements Planning

Demand Driven Material Requirements Planning was born around this idea: give flexibility to Material Requirements Planning and break its dependant structure. Nevertheless, DDMRP is not only based on MRP, but it takes ideas from other supply chain management methodologies such as Distribution Requirements Planning, Lean, Theory of Constraints and Six Sigma. The idea of independence among the flow is taken from Lean, for example. Before moving forward with the ideas of DDMRP, it is important to have its principles in mind: position, protect, pull.

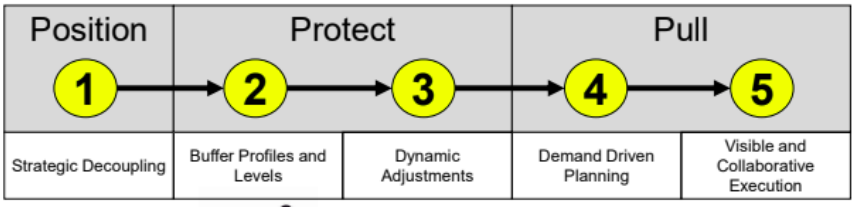


Figure 2.2: The five components of DDMRP.

In Figure 2.2, the main scheme of the DDMRP can be observed. It is important to focus on the first 2 points: strategic decoupling and Buffer Profiles and Levels. These are the main innovations of Demand Driven Material Requirements Planning.

The first step of DDMRP is to position decoupling points of inventory along the components of the bill of materials. They are the solution to create independence in a supply chain. In these decoupling points is where Stock Buffers are located, which is widely explained later. To summarize, decoupling points provide safety stock of some components, which cause a point of rupture of the typical material explosion of a MRP. It can be said that decoupling protects the flow of materials from the possible variability of the system in both directions of the supply chain (demand and supply variability).

To position these decoupling points allow to shorten the planning horizon and lead times are compressed. The most complex part is where to allocate these decoupling points. If the previous example is recovered, the decoupling points would be positioned as the following:

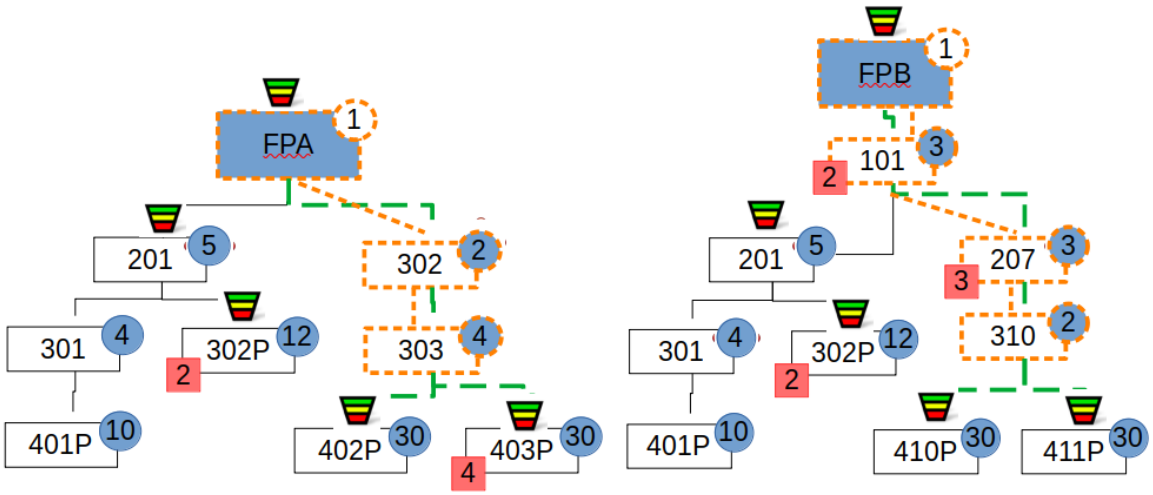


Figure 2.3: Bill of materials of products FPA and FPB with decoupling points and cumulative lead time path (green line) and decoupled lead time path(orange line).Reference: [1]

Before decoupling, the lead times of products FPA and FPB are 37 and 39 days respectively. In Figure 2.3, some decoupling points have been allocated, and a new lead time can be calculated: decoupled lead time [3]. In this case, the decoupling lead time of both final products are 8 days for FPA and 9 days for FPB.

The reasons why the decoupling points have been positioned in each product are still missing. Products 302P, 402P, 403P, 410P and 411P are buffered because they have a really long purchase lead times, so decoupling these 5 products involves a really big compression of the lead time and will prevent the supplier variability from affecting the rest of the components. The decoupling point of 201 has been positioned because is a common product between FPA and FPA, so it could be a critical sub-component and decoupling it would give inventory flexibility to the manufacturing of both products. Finally, 2 buffers have been located in the final products, this is because the customer tolerance time[4] is 1 day for both products, so a buffer is needed in both FPA and FPB, in order to accomplish customer expectations.

[3]The longest cumulative coupled lead time chain in a manufactured item's product structure. It is a form of lead time but is limited and defined by the placement of decoupling points within a product structure

[4]The time the typical customer is willing to wait before seeking an alternative source.

The second step in DDMRP is the Buffer Profiles and Levels definition and in this point is where the issues of Demand Driven Material Requirements Planning arise. Buffers imply building the level of shock absorption at the decoupling point so that it can mitigate variability in both directions. They are formed by three areas: green, yellow and red zone. The green zone represents the frequency and quantity of the orders, the yellow zone represents the primary coverage of the product and the red zone is the safety stock.

With the definition of the buffer zones, lots of new concepts have to be introduced. It is important to have clear the definition of each of them, specially ADU and lead time and variability factors.

- **Average Daily Usage (ADU)**. Average Daily Usage is one of the cornerstones of the buffer sizes, as it appears in the most part of the calculations. ADU is a calculated rate of use of each specific product, and it is computed with a rolling mean with a length-of-period defined by the planner. Obviously, the period chosen to be considered for the rolling mean are determinant for the ADU value of the product. DDMRP introduces forecasting, but it is not the ordering mechanism such in MRP. However, this is a pretty simple method to compute such an important concept inside the DDMRP.
- **Minimum Order Cycle (MOC)**. The order cycle is the number of expected days between orders. It can be imposed through the use of product scheduling wheel or be a desired average number of days between orders.
- **Minimum Order Quantity (MOQ)**. It is possible that the supplier imposes a certain minimum quantity of product to make an order.
- **Lead Time Factor (LTF)**. Lead Time can be divided in three categories: short, medium an long. These designations are relative to the company's environment and product type (purchased or manufactured). For the manufactured parts, the designation of the group are determined by the Decoupled Lead Time. The designation of the lead time factor is quite arbitrary, as it can be a value between 0 and 1, where shorter lead times have smaller lead time factors.
- **Variability Factor (VF)**. Variability can also be classified in three groups: high, medium and low. It can come from the demand side and from the supply side. Nevertheless, for manufactured parts the demand variability is the most important one, while for purchased parts the supply variability is the one to take into account. On summary, DDMRP says that a variability factor from 0 to 1 needs to be assigned to each part.

Once the important concepts are defined, the sizing of the buffer zones can be properly explained:

- **Green Zone**. It is defined by the biggest number of these three options:
  1. Minimum Order Cycle x Average Daily Usage(ADU)
  2. Average Daily Usage(ADU) x Decoupled Lead Time(DLT) x lead time factor
  3. Minimum order quantity of the product.
- **Yellow Zone**. It is defined by the result of Average Daily Usage(ADU) x Decoupled Lead Time(DLT)
- **Red Zone**. It is the sum of two sub-zones:
  1. **Red Base**. It s computed by Average Daily Usage(ADU) x Decoupled Lead Time(DLT) x lead time factor.
  2. **Red Safety**. It is calculated by Red Base x variability factor

An example of a Buffer Zone Calculation would be the following:

Example Part Buffer For Simulation			
Average Daily Usage	10	Green Zone	35
Buffer Profile	M, M (.5), M (.5)		LT Factor: 35 (DLT (7)x ADU (10) x Lead Time Factor (.5))
MOQ	20		Minimum Order Quantity: 20
Imposed or Desired Order Cycle (DOC)	None		Order Cycle: N/A
Decoupled Lead Time (DLT)	7 days	Yellow Zone	70 (7(DLT) x 10(ADU))
Order spike threshold = 26 Order Spike Horizon = 8 days		Red Zone	52 (Red Base (35) + Red Safety (17))
			Red Base: 35 (DLT (7)x ADU (10) x Lead Time Factor (.5))
			Red Safety: 17 (Red Base (35) x Variability Factor (.5))

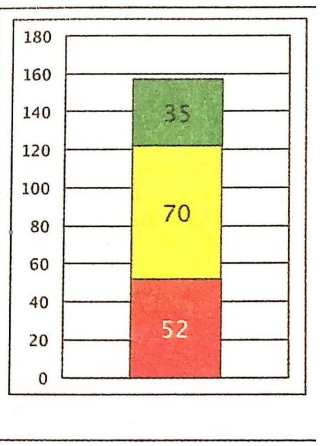


Figure 2.4: Example part buffer calculation summary. Reference: [1]

Once the Stock Buffers are sized for each product, DDMRP jumps to the third step of its principles: pull. The explanation of this concept is not fundamental for the development of this study, so it is allocated in the appendix A.

2.2 Methodology

This section is organized in two parts: the methods used to estimate the Average Daily Usage and for resizing the Red Zone of the Stock Buffer.

As its name indicates, the Average Daily Usage is the mean sales or usage of a product each day (as a product could be used for a sale or for manufacture another product). The goal of this part is to compute an accurate forecast of the daily needs of each product. To do so, various statistical models are fitted for each product and some metrics are obtained. Based on these metrics, one model is chosen to estimate the ADU and subsequently, the Buffer part is fed by this product usage forecast.

2.2.1 ADU computation method

As mentioned in the previous section, the ADU is computed using a rolling mean, which window has to be chosen by the planner. This model seems pretty simple for computing such an important concept in material requirements planning. For this reason, one of the goals of this project is to consider other possible models to compute this measure more accurately. The considered models are: ARIMA, Facebook Prophet and Holt-Winters. Regarding the data, the sales of each product are grouped monthly and the forecast of new values are done per month.

Current method: Simple Moving average

Performing a moving average is the current methodology used by the company of study, Kencove, to calculate the Average Daily Usage. The responsible of planning the stock in this company has fixed a period of 3 months for the rolling window, so the sales forecast of the current month is equal to the average of the number of sales of the past three months. As the ADU is a daily measure, it is necessary to divide the forecasted sales for the month between the number of days of that month. The equation defining this model is the following one:

$$SMA_k = \frac{1}{k} \sum_{i=n-k+1}^{n} p_i$$

The most usual case is that $k = 3$ and p_i are the historic of sales done per month.

Alternative method 1: ARIMA

The ARIMA (Auto Regressive, Integrated and Moving Average) methodology is based on the idea of taking non-stationary time series, making it stationary and modeling the resulting series by a causal and invertible ARMA process[6].The general expression of an $ARIMA(p,d,q)(P,D,Q)_s$ is the following:

$$\phi_p(B)\Phi_P(B^s)(1-B)^d(1-B^s)^D Y_t = \theta_q(B)\Theta_Q(B^s)Z_t \tag{2.1}$$

where B is the backshift operator and (1-B) is the difference operator:

$$BY_t = Y_{t-1}$$

ϕ_p, Φ_P, θ_q and Θ_Q are the characteristic polynomials of ARIMA, which are defined as:

- $\phi_p(B) = 1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p$
- $\theta_q(B) = 1 + \theta_1 B + \theta_2 B^2 + ... + \theta_q B^q$
- $\Phi_P(B) = 1 - \Phi_1 B^s - \Phi_2 B^s P - ... - \Phi_P B^s P$
- $\Theta_Q(B) = 1 + \Theta_1 B^s + \Theta_2 B^s 2 + ... + \Theta_Q B^s Q$

When fitting an ARIMA model ts is needed to check if the time series is stationary or not. In the real world, the most part of data is non-stationary and thus, it is very likely that some transformations to the original time series has to be performed to get a stationary series as required by ARIMA modeling.

The general procedure to construct a classic ARIMA model can be summarized as follows:

To transform a given series into stationarity, first it has to be checked that it has constant variance. If that is not the case, the scale of the data should to be changed by applying a proper Box-Cox transformation, commonly a logarithmic one. Next, it is necessary to verify if there exists a seasonal behaviour on the data, a similar pattern that repeats itself for a constant period s. If a seasonal pattern is present, the series would be non-stationary and a seasonal difference of order s should be applied; $s = 12$ in case of monthly data. Finally, ensure that the series is stationary by checking if the mean is constant. If that is not the case, regular differences should be applied until the mean is considered constant. Up to this point, it is clear how many (d) regular differences and seasonal differences (D of order s) would be needed to achieve stationarity; see equation 2.1.

Based on the configurations observed in the ACF and PACF plots, the order of the characteristic polynomials in expression 2.1 can be determined.

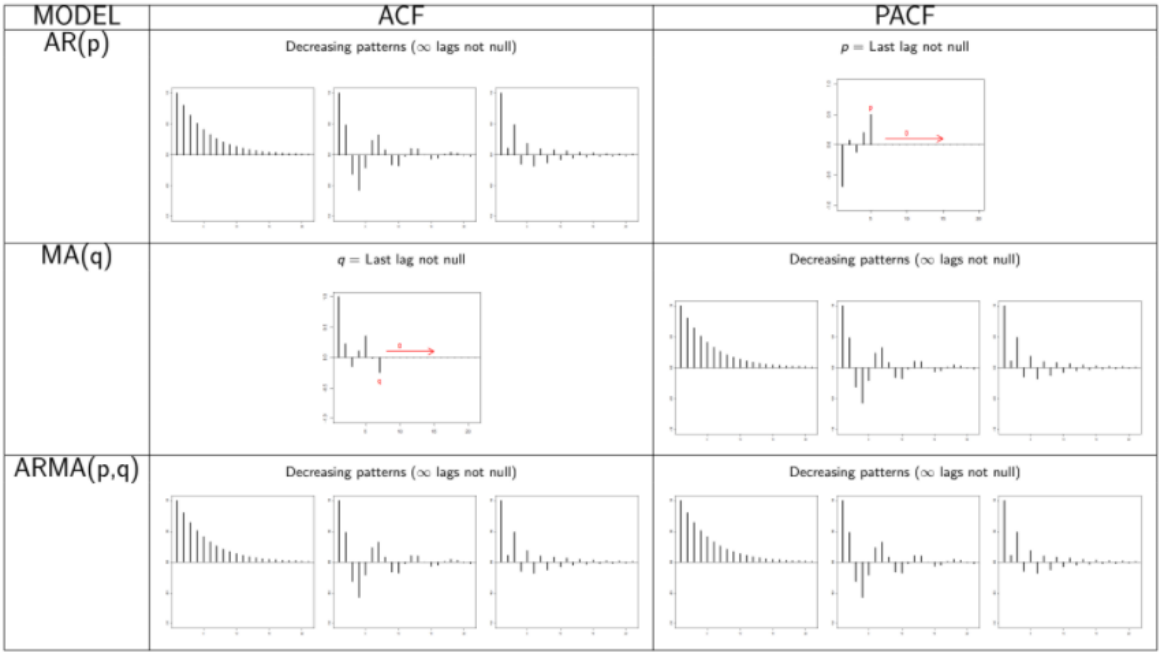


Figure 2.5: Summary table of the identification of the parameters p and q based on ACF and PACF plots.

To identify an MA(q) model, it is necessary to see a decreasing pattern of the lags in the PACF plot, as well as observing that in the ACF plot the lags bigger than q are zero. On the other hand, the AR(p) model shows a decreasing pattern in the ACF plot, while in the PACF all the lags bigger tan p are zero. In the case of having an ARMA(p,q) model, both ACF and PACF have decreasing patterns. To determine the values of p and q in an ARMA(p,q), it is necessary to check different combinations, starting from $p = 1$ and $q = 1$ and progressively increasing the order, until find a suitable model according to classic ARIMA hypothesis. In practice, at least two models are proposed for further analysis. [7]

The estimation of the model parameters is carried out using maximum likelihood. Once the ARIMA model has been specified and estimated, the validation process takes place. Usually, at this step, at least two models are proposed for further analysis according to validation results and adequacy measures like the AIC or BIC.

A model is considered valid if it does not contradicts the model assumptions. In this case, if the residuals or random part of the model behave as a white Gaussian noise, say $Z_t \sim N(0, \sigma^2)$. In summary, it should be ensured that the residuals are independent, normal and have an homogeneous variance. It is also important to ensure that the model is causal and invertible before using the fitted model for prediction.

Finally, it is important to evaluate the results obtained from the group of models fitted and validated. When fitting a model, some measures can be obtained: related to the goodness-of-fit of the model to the series. However, it is also important to take into account those measures which indicate the percentage of error of the model prediction.

Extensions of Classic Arima

Sometimes the variable measured is affected by an exogenous fact that changes its natural dynamics [6]. It is in this point, when the outliers treatment can be useful, as a model with outliers can give imprecise forecasts. The idea behind the outliers treatment is to identify those unexpected events in the past history, measure its impact and make the corresponding corrections. There exist three types of outliers:

- Additive Outlier. This kind of event affects only one period of the series.
- Transitory Change. It consists of an event which affects severely on a specific period and in the following periods its effect decreases. In the end the series returns to the original level.
- Level Shift. It influences one period and its impact remains in the next periods of the time series.

After identifying them, the time series is rebuild ignoring those outliers and obtaining a linearized series. Then the whole process is repeated, possibly obtaining a new ARIMA model [8].

Another extension of the classic ARIMA model consists on correcting the calendar effects. Calendar effects can only be applied to monthly time series and try to explain the differences measured by the time series depending on some configurations in the months. Those configurations considered are the Easter and the number of trading days per month. As it is widely known, the dates of Easter are different for each year. In the case that the series is affected by the Easter week, predictions would fail to reflect those changes. Taking into account the month of the Easter week would prevent this error. Same thing is done with the number of trading days per month.

Alternative method 2: Prophet model

The Prophet model was developed in 2017 by Sean J. Taylor and Benjamin Letham in Facebook. It is a model based on Generalized Additive Model, a generalized linear model with a linear predictor involving a sum of smooth functions of covariates. The model

decomposes the time series in the following elements:

$$y(t) = g(t) + s(t) + h(t) + e_t \tag{2.2}$$

where g(t) is the trend function, s(t) the seasonality and h(t) represents the effects of holidays or events that are irregularly distributed along the time series. The error term e_t is supposed to be normally distributed.

Generalized Additive Model are a middle step between Generalized Linear Model (GLM) and Generalized Nonparametric Multiple Regression Model (GNMRM). On the one hand, it avoids having the linearity restrictions of the GLM. On the other hand, it solves the course of the dimensionality present in the GNMRM. The main advantage of GAM is that it decomposes easily and accommodates new components if necessary, for example a custom seasonality or additional regressors.

The equations of GAM are the following:

$$E(Y) = \mu$$

$$\mu = \alpha + \sum_{j=1}^{p} g_j(x_j)$$

The functions g_j are non-parametric functions of each explanatory variable j. In the case of the Prophet model, there is only one independent variable: time. In this case, the function g is build on the basis done in the decomposition of an additive time series shown in equation 2.2. Each component of the equation can take different expressions, depending on the parameters configured by the programmer.

The trend can be linear or logistic. In the case of forecasting a products sales, the linear trend is chosen, because there is no upper limit in the number of sales per product. The trend is represented by a piecewise linear growth model:

$$g(t) = (k + a(t)^T\delta)t + (m + a(t)^T\gamma)$$

where k is the growth rate, $a(t)$ is the changepoint indicator, δ represents the rate adjustments, m an offset parameter and γ_j are an adjustment to the offset parameter.

Changepoints are trend changes which take place in the growth model, as the rate is not constant. $a(t)$ is a vector which can only contains 0 and 1. If there are S changepoints at time steps s_j being $j = 1, ..., S$, the vector $a(t)$ is defined as following:

$$a_t \begin{cases} 1, & \text{if } t \geq s_j \\ 0, & \text{otherwise} \end{cases}$$

The vector δ is composed of the variation in rates that occurs at time s_j (at each changepoint). Finally, the parameter γ is an adjustment to the offset parameter m in order to connect the endpoints of the segments. To make the function continuous, γ is set to $-s_j\delta_j$ [9]

As the data is monthly, the only seasonality taken into account is the yearly one. Prophet uses Fourier series to build a malleable model for the periodic series. The seasonal effects are approximated with:

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

which is a standard Fourier Series with P as the regular period, in the case of yearly data, P=365.25. It is required to estimate $2N$ parameters $\beta = [a_1, b_1, ..., a_N, b_N]^T$. In order to do this, a matrix of periodic vectors for every time step of past and future data is defined. Taking:

$$X(t) = \left[cos\left(\frac{2\pi(1)t}{P}\right), ..., sin\left(\frac{2\pi(N)t}{P}\right)\right]$$

then,

$$s(t) = X(t)\beta$$

The value of N impacts the flexibility of the model. A higher N allows fitting seasonal patterns more quickly, thus increasing flexibility and the risk of overfitting.

Moving forward to holidays and events, the user is able to give a listing of historical and future events that are eligible to affect the studied time series. Holidays and events are just dummy variables added to the model. It is assumed that the effect of holidays is independent. The equation used to modelize the holidays effect is the following:

$$h(t) = Z(t)\kappa$$

taking into account that

$$Z(t) = [1(t \in D_1), ..., 1(t \in D_L)]$$

Each holiday i has a set of past and future days D_i. The equations above, indicate if time t belongs to the holiday i and the parameter κ_i represents the change in the forecast.[10]

The way to estimate the parameters of the additive model is a Bayesan approach that enables the automatic selection of the change points and other model parameters.[11]

Alternative method 3: Holt-Winters method

Time series is a union of different components such as the trend, seasonal and error. The three components can be combined in multiple ways. It can be a purely additive model (as in prophet model):

$$y = T + S + E$$

Another possibility would be a multiplicative model:

$$y = T \cdot S \cdot E$$

Multiple possibilities can be made combining the components by addition or multiplication. The nature of the error component is ignored (additive or multiplicative) because the decision does not take effect on the result of the forecast. Ignoring the error component, fifteen exponential smoothing methods can be made:

Trend component	Seasonal component		
	N (None)	A (Additive)	M (Multiplicative)
N (None)	N,N	N,A	N,M
A (Additive)	A,N	A,A	A,M
A_d (Additive damped)	A_d,N	A_d,A	A_d,M
M (Multiplicative)	M,N	M,A	M,M
M_d (Multiplicative damped)	M_d,N	M_d,A	M_d,M

Figure 2.6: Exponential smoothing methods coming from possible combinations between trend and seasonal components. Reference:[2]

The trend is the combination between a level term and a growth term. The way this two terms are combined is what determines the pattern of the trend component: none, additive, additive damped, multiplicative or multiplicative damped. A damped trend is used when the growth rate as the end of the time series is not likely to continue in the future. On the other hand, seasonal component only has three patterns:none, additive and multiplicative.

Each of this method is also know by other name. For example, method (N,N) is also called simple exponential smoothing, method (A,N) is the Holt's linear method and methods (A,A) and (A,M) represent Holt-Winters' additive and multiplicative methods. These latter methods are the ones chosen to perform the last ADU forecast. It is important to remember to distinguish exponential smoothing methods from the underlying state space

models. In this case, the forecasting is done with exponential smoothing method, which only produces point forecasts: there is not any confidence interval.

Holt-Winters exponential smoothing method is a forecasting technique which tries to model three aspects of a time series: level, trend and seasonality. Starting with the Holt Winter's method with multiplicative seasonality, the equation of the forecast is the following:

$$\hat{y}_{t+h|t} = (\ell_t + b_t h) s_{t-m+h_m^+}$$

where $\hat{y}_{t+h|t}$ is the forecast for h periods ahead, l_t and b_t are the level and the growth of the series at time t, $s_{t-m+h_m^+}$ is the seasonal component, m is the length of seasonality and $h_m^+ = [(h-1) \text{mod m}] + 1$. The level, growth and seasonal components can be defined by the next equations:

- Trend:
$$\ell_t = \alpha \frac{y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1})$$
- Growth:
$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*) b_{t-1}$$
- Seasonal:
$$s_t = \gamma \frac{y_t}{\ell_t} + (1-\gamma) s_{t-m}$$

On the other hand, the Holt-Winter's method with additive seasonality presents the following equation:

$$\hat{y}_{t+h|t} = \ell_t + b_t h + s_{t-m+h_m^+} \tag{2.3}$$

As it can be seen, the equation is very similar to equation 2.2.1, with the difference that the seasonal component is added and not multiplied to the other terms. The rest of the components can be defined as the following:

- Trend:
$$\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1})$$
- Growth:
$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*) b_{t-1}$$
- Seasonal:
$$s_t = \gamma(y_t - \ell_t) + (1-\gamma) s_{t-m}$$

In both methods, there are parameters which value is constant along time: α,β^*, γ. The value of these coefficients is usually restricted to be between 0 and 1.

Once all the components of the main equation are modeled, the only unknown parameters are β^*, α, γ and the initial conditions s_0, b_0, ℓ_0. The parameters are estimated with an iterative optimization method which tries to minimize the MSE.

When initial conditions and parameters are set, the component equations at each time step l_i, b_i, s_i can be computed. Finally the equation 2.2.1 or 2.3 can be calculated by the time steps from 0 to $n + h$, being n the time step with the last known value and h the farthest forecasted time step.[2]

Chosen Performance Metric for model predictability

To evaluate the accuracy of each model, a Key Performance Indicator(KPI) should be computed to assess the quality of the model. If two or more models are compared for the same product, the metrics indicate which model is better for that product. As the goal is to choose which model gives a better performance for the overall product portfolio, a relative metric should be used, such as as the Mean Absolute Percentage Error.

MAPE is one of the most commonly used KPIs. The formula of this metric is the following one:

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} \frac{|A_t - F_t|}{A_t}$$

being A_t the actual demand, F_t the forecasted demand and n the number of observations.

Using MAPE has some advantages and disadvantages. Among the advantages is that as it is a relative metric, it allows the comparison between products that have different scale. However, the main disadvantage is that it penalizes more the over-stocking than the under-stocking.

2.2.2 Stock Buffers Computation method

As stated in the section 1.1, one of the most controversial parts of the Demand Driven Material Requirements Planning is the setting of the lead time and variability factors. The planner of the company is who takes the decision of which specific value (between 0 and 1) these factors have for each product.

The only clue given is that products with a short lead time should have a lead time factor closer to 0 and the products with a long lead time should have a factor near to 1. Same happens with variability factor: products with little variability should have a small factor and products with a large variability should have a variability factor closer to 1.

It is important to remember that these two factors are fundamental to compute the most critic buffer: the red zone. The red zone of a buffer represents what in other methodologies is called the safety stock. The safety stock is additional stock kept in the warehouse which is maintained in order to prevent from stockouts caused by the variability in supply and demand.

Going back to the previously introduced example of DDMRP shown in image 2.4, it can be seen how introducing different lead time and variability factors could vary the size of the red zone. In the example, the ADU had a value of 10, DLT was 7 days and Lead Time and Variability factors were 0.5. In that case, the resulting Red Zone had a value of 52 units of stock.

However, if the Lead Time Factor is changed to 0.3 and the Variability Factor is changed to 0.6, the computation is quite different:

$$\text{Red Zone} = \text{Red Base} + \text{Red Safety} = \text{Red Base}(1 + VF) = (DLT \cdot ADU \cdot LTF)(1 + VF)$$

$$\text{Red Zone} = (7 \cdot 10 \cdot 0.3)(1 + 0.6) = 33.6 \approx 34$$

As it can be seen, slightly changing the Lead Time and Variability factors, the safety stock changes from 52 to 34, which represents a change of the 34.6%. With this example, is shown that this method is not appropriate to compute the buffer size of he Red Zone. For this reason, another methodology based on Monte Carlo method is proposed.

Cost function

To explain the proposed calculation method, it is important to keep in mind the purpose of having a Red Zone in the Stock Buffer: do not have stock outs. However, it is also remarkable the fact that products have a cost, and accumulating to much stock implies having money of the company stuck in the warehouse. Considering that, the stock kept in the warehouse has to be a middle ground between not having stock ruptures and having too much stock.

At this point is important to define a cost function. A cost function is the function to minimize in this optimization problem. Actually, the companies which have safety stock, what they want to avoid is losing money when managing stock. For this reason, it makes sense to introduce the cost and the profit of a final product into the cost function. The first approach to this cost function could be:

$$f(x, y) = \alpha x^{+} + \beta y^{+}$$

where α is the cost of the product and β it is the profit, defined as the revenue minus the cost of a product, x represents the stock not sold and y represent the number of units which could have been sold if the stock did not ran out. On summary, the optimization problem consists on minimizing the lost money. Notice that x and y can only be positive, so it is the maximum between its number and 0.

The way to determine x and y is the following:

- x **or units of product which are not sold**. The number of units not sold are the difference between the stock and the accumulated demand until receiving the next supply.

$$x = stock - \sum_{i=1}^{DLT} ADU_i$$

  Note that *stock* is the current level of safety stock kept in the warehouse and $\sum_{i=1}^{DLT} ADU_i$ is the number of sold units until the next supply is received. Remember that ADU is the daily demand and DLT is the number of days needed to manufacture the product.

- y **or units of product which could have been sold if there has not been a stock out**.

$$y = \sum_{i=1}^{DLT} ADU_i - stock$$

  In this case, the number of sales missed is computed by the difference between accumulated demand and safety stock.

Going back to the cost function, it has the following expression:

$$f(ADU, stock) = cost(stock - \sum_{i=1}^{DLT} ADU_i)^+ + profit(\sum_{i=1}^{DLT} ADU_i - stock)^+$$

Dividing the whole equation by the cost of the product:

$$f(ADU, stock) = (stock - \sum_{i=1}^{DLT} ADU_i)^+ + \frac{profit}{cost}(\sum_{i=1}^{DLT} ADU_i - stock)^+ \tag{2.4}$$

Up to this point, the cost function is already defined. The goal of the optimization problem is to find which level of stock minimizes the cost function of equation 2.4. Next section introduces the Montecarlo method, used to find this level of stock.

Montecarlo method

Montecarlo simulation or Montecarlo method is a statistical approach used to solve complex mathematical problems via the generation of random variables. It was developed by Stan Ulam and John von Neumann during the the 40s.

Below, a diagram to understand better the Montecarlo method:

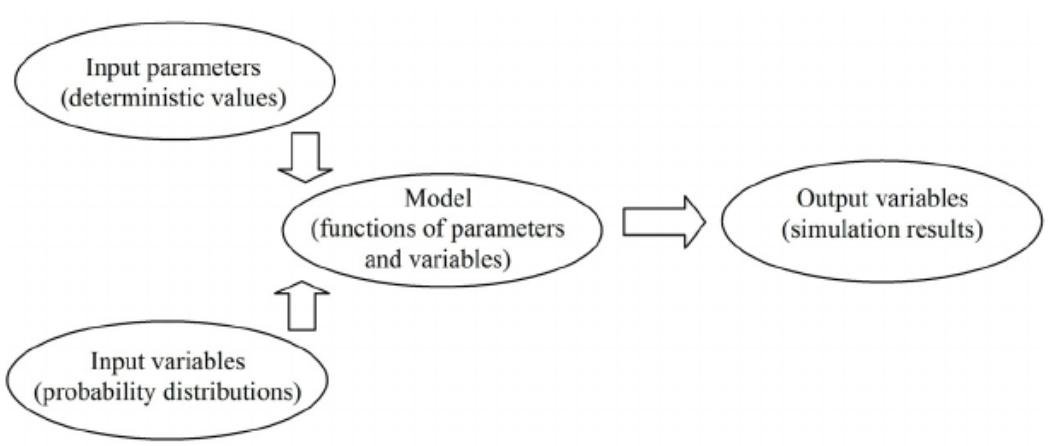


Figure 2.7: Explanatory diagram of the Montecarlo method. Reference: [3]

The scheme above shows the 3 main elements of Montecarlo simulation: input data, model and output variables. The input data has to be introduced by the user: it can be deterministic or a random number generated by a probability distribution identified. Next step is to compute the output variable by the model, which is a function or set of functions that relate the input with the output. Finally, the output variable or the target of the simulation is obtained. [3]

The process described above must be repeated several times, a minimum of 10,000 to obtain reliable results. In each iteration, the input variable are different, as they are a random number generated from a probabilistic distribution.

Going back to the buffer resizing problem, Montecarlo method can be used to compute the cost function for a specific stock safety level. For a number of stock units, the steps presented in the following diagram are carried out:

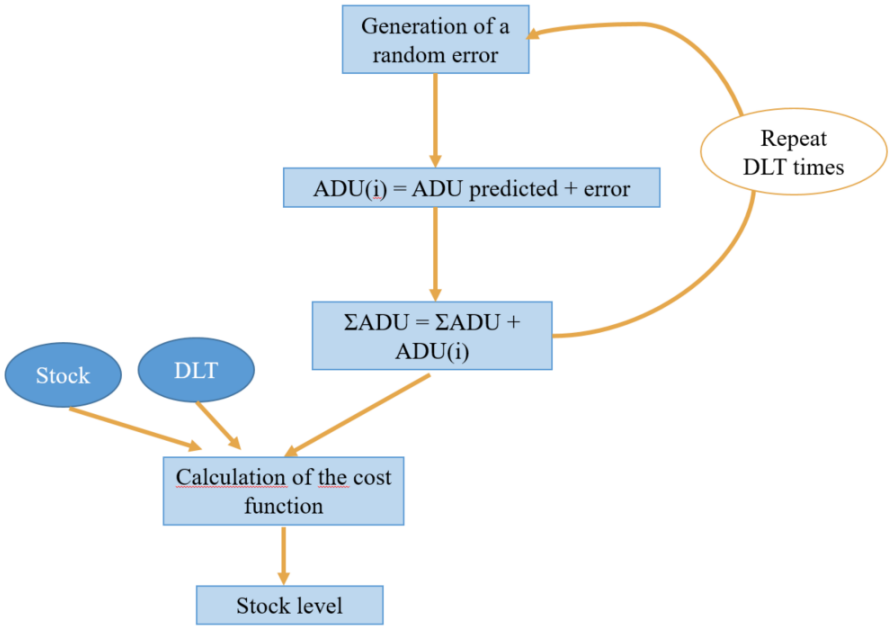


Figure 2.8: Diagram of the process followed to compute the cost function and its stock level.

The scheme above represents one of the iterations performed in Montecarlo simulation. In this case, the target output is the results obtained by computing the cost function stated in the previous section for a certain number of stock. The input parameters are the number of stock units, the DLT and the forecasted ADU of a product. On the other hand, the input variable is $\sum_{i=1}^{DLT} ADU_i$ as it is calculated adding a deterministic parameter (the forecasted ADU) and a random variable. This random variable represents the error performed in the ADU forecasting. To choose how this error is represented, the residuals of the predictions must be analysed and fitted in the appropriate distribution. The choice of this distribution is discussed below. Nevertheless, regardless of the distribution, a different error is obtained in each iteration.

Every time that the process above is computed, a cost for a specific number of stock units is obtained. Repeating this procedure N times, N values of the cost function are achieved. At this point, the distribution of the cost function could be plotted. However, the value of interest is the median value of the cost function (the median is chosen instead of the mean because it is less sensible to outliers).

Up to this point, the program has computed the median cost function for a certain level of stock. Nevertheless, the interest of this study is to determine which stock level minimizes the cost function. For this reason, the process explained above is repeated for different

stock levels. Finally, a table with the median cost function for many different stock levels is obtained. Searching for the minimum of the cost function, the corresponding level of stock is known. This number of units is the most appropriate one to minimize the money lost in stock for a company.

Residuals distribution

The choice of the residual distribution is a key step for the Montecarlo simulation. Depending of the chosen distribution, the simulated ADUs would take a shape or another. As in the most part of the analysis, there was not a single attempt, but an iterative process.

In the first place, it is necessary to select the residuals which are analysed to be fitted in a known distribution. A small iterative process is repeated in order to obtain the residuals of a yearly prediction of the ADU during the last six years by cross-validation.

Cross-validation is a statistical method that can help with that in a more robust way that just splitting the time series in test and train dataset once.This technique consists in splitting the dataset into some parts. Then, the model is trained using all folds except one, which is the test set. This operation is repeated changing each time the fold which represents the test set. Finally, the metric of the model is the average of all the metrics obtained in each operation.

In Machine Learning, the process of splitting the data into folds is done randomly, but in time series it can not be done like this, as future values could be used to predict past values. For this reason, it has to be done in temporal order. The way to do that is to take the first training set as the first data of the temporal series and predicting the next data points. These forecasted data points are included in the next training set and the following values are predicted. This operation is repeated until the training data set includes nearly the whole dataset. [4] An illustrative image of this process is showed below:

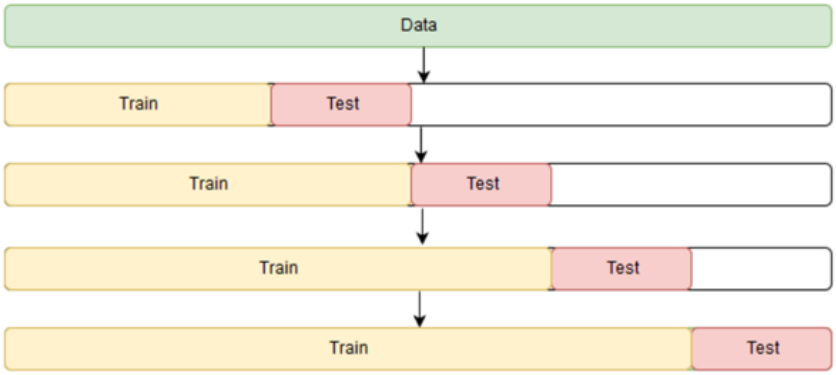


Figure 2.9: Diagram of a cross-validation process with time series. Reference: [4]

At this point of the analysis, it is important to remember that the forecast was performed monthly, but daily data is needed for the Montecarlo simulation. For this reason, the total monthly ADU is divided by the number of days in that month to obtain the daily data. This process is performed both for the observed as for the predicted ADU and making the difference between them, a daily residual is obtained. Obviously, the residual is the same along the month, but it is a way to start expressing the results in a daily format, as the measure of DLT is commonly expressed in days.

Once the residuals are obtained, the goal is to find a known distribution that matches the residuals results. This is a complex task as each product could have a different distribution. These possible differences among products could be a problem when trying to optimize the process, as each product should be computed separately.

2.3 Statistical Analysis

In this section, the methodologies explained in the previous chapter applied with data extracted from the Kencove database.

2.3.1 Database explanation

The database obtained to perform all the computations related to this project is obtained from Kencove Farm Fences Supplier. It is an American company based in Pensylvania, which manufactures and sells agricultural fences. Kencove sells around 6,000 products, but are currently using DDMRP only in some test products.

The three products chosen to perform this study are the products with IDs 76, 79 and 173. These products were chosen because they are test products and are representative of the type of products Kencove is most interested on predicting. Product 76 is a crimp tool, that belong to the tools section, product 79 is is a Zinc wire roll (belongs on the wire's category) and product 173 is a pair of crimp sleeves which are connectors.

The sales data which is available goes from January 2012 until December 2021. However, due to COVID-19 pandemic, the data used in the prediction of the ADU will comprehend from January 2012 until December 2019. The reason for doing this exclusion is to find the most accurate forecasting methodology excluding the possible effect of the pandemic. The sales evolution of these three products can be found in the annex B.

2.3.2 ADU computation results

In this section the different methodologies to compute the ADU are analysed: moving mean, ARIMA, Prophet and Holt-Winters. To illustrate the results obtained, only product 173 are analysed and explained in detail. The results of other products are also included in the comparative analysis section.

Moving Mean

Firstly, the current ADU calculation method is studied. As it is said in the methodology, DDMRP proposes a rolling mean to a have a forecast of the ADU. In this case, Kencove, chose a window of 3 months to compyte this rolling mean. Computing the mean of the previous 3 months, the next value can be obtained. Doing this operation until the last month of 2019, a series of predicted values can be obtained. Below, both series are plotted (the observed and the predicted one).

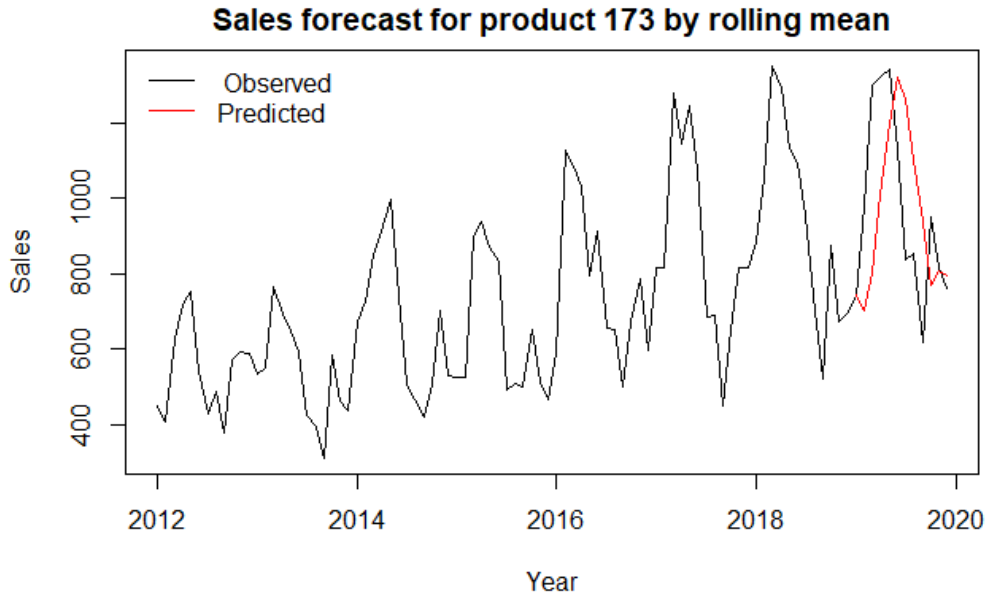


Figure 2.10: Rolling mean prediction plot for product 173.

As it can be seen in Figure 2.10, the predicted series starts at April, as 3 historical values are needed to compute the first value. Observe that the peaks and troughs are moved regarding the current series. This causes a significant error in the prediction of the sales, specifically, the MAPE of the forecast of the last year is **0.229**. This error is quite big, as it is nearly a 25% of error in the prediction of the future sales.

ARIMA

The study of ARIMA is a little different from Holt-Winters or Prophet. In the case of ARIMA, fitting a model to data is not an automatic process, because there are steps where interpretation is necessary before choosing the right model.

For the studied product, the period that comprises January 2012 until December 2018 represents the training data and the interval from January 2019 until December 2019 is the test data.

The first step of ARIMA is to ensure that data is stationary. In order to do so, the mean-variance plot is shown in Figure 2.11.

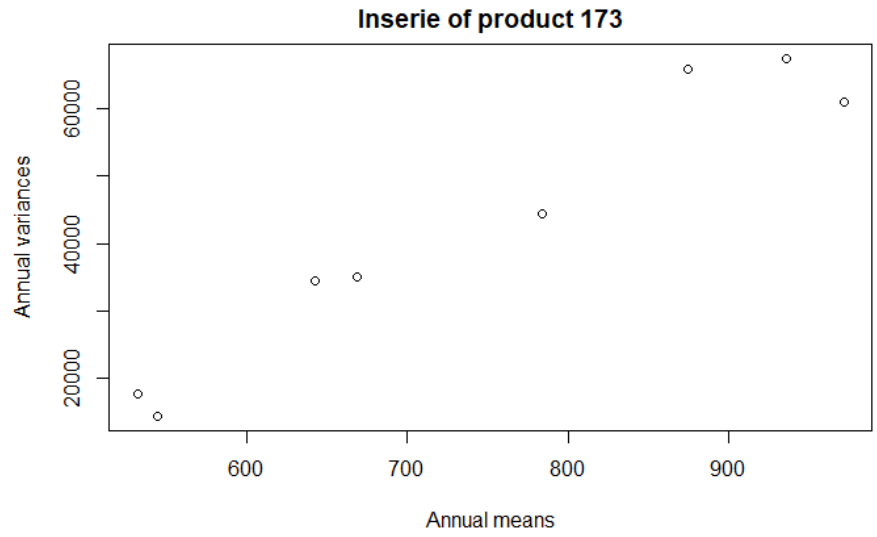


Figure 2.11: Mean-variance plot for product 173

The mean-variance plot clearly portrays an increase of the variance when the mean grows. For this reason, the variance can not be considered constant and a logarithmic transformation has to be taken, $log(Y_t)$.

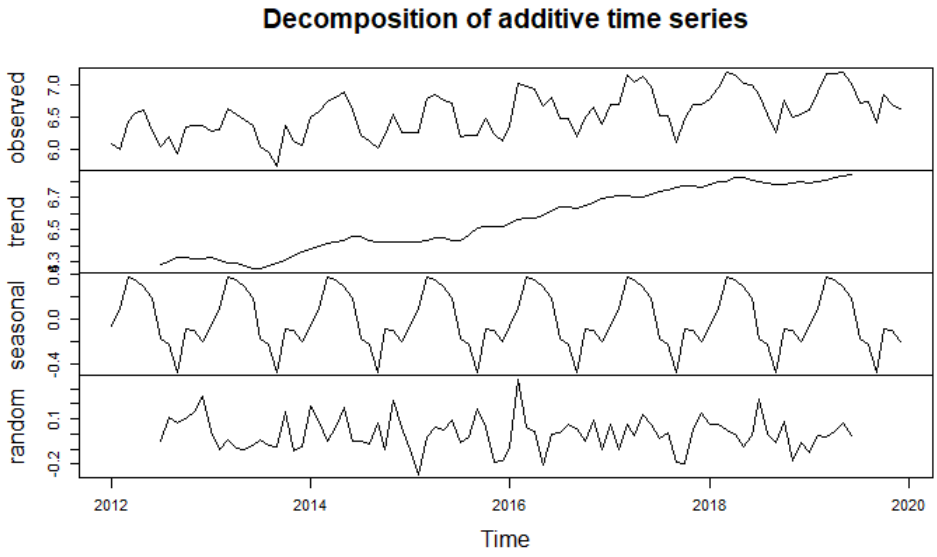


Figure 2.12: Decomposition of the time series for product 173

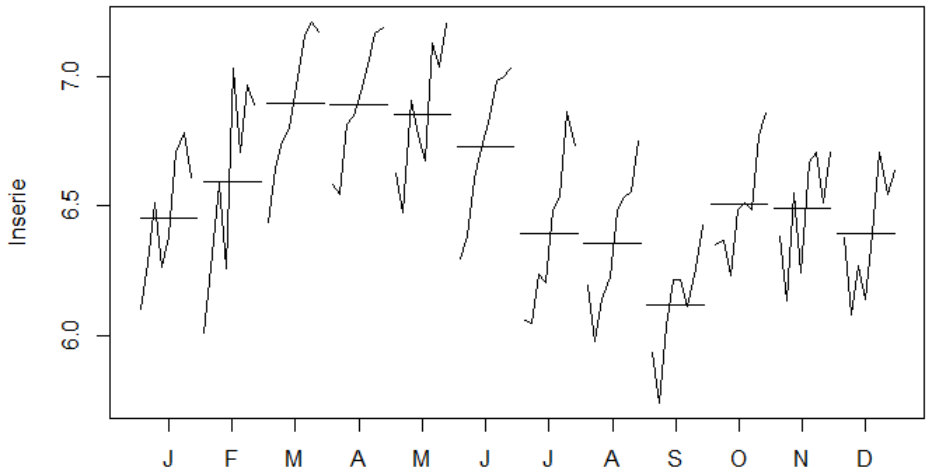


Figure 2.13: Monthplot of product 173

The decomposition of the time series, Figure 2.12, shows that there is a global upward trend and that there is presence of seasonality. A monthplot, Figure 2.13, is helpful to identify better the seasonality. The sales of product 173 are lower in summer months, while in March, April or May it reaches the higher values. To eliminate the seasonality a seasonal difference of order $s = 12$ $(1 - B^{12})log(Y_t)$ is applied.

Applying one regular difference to the seasonally differentiated series, brings the mean to 0 but the variance increases with respect to the just seasonally differentiated series, for this reason the study is done with $(1 - B^{12})log(Y_t)$. The next step is to perform the ACF and PACF plots:

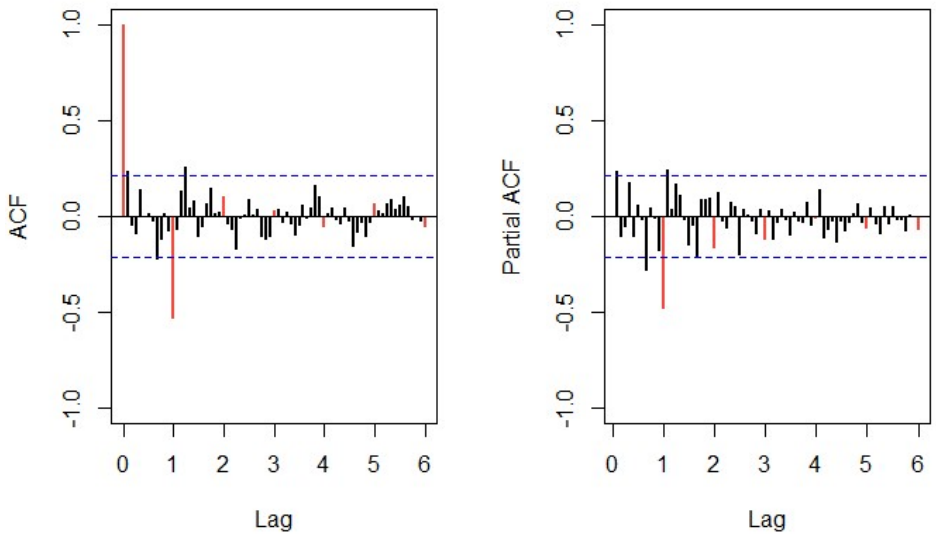


Figure 2.14: ACF and PACF for $(1 - B^{12})log(Y_t)$.

Looking at the Figure 2.14, both plots have a decreasing pattern for the regular part, with just the first lag significant in both ACF and PACF, thus models MA(1) and AR(1) can be identified. Regarding the seasonal part, it can be observed a decreasing pattern in the PACF plot and just the first significant lag in the ACF plot, so an initial SMA(1) is considered.

Initially, models such as $ARIMA(1,0,0)(0,0,1)_s$ or $ARIMA(0,0,1)(0,0,1)_s$ are considered. However, when performing the validations, all these models do not accomplish the hypothesis of invertibility as the characteristic polynomial of the AR part lie very close to the unit circle. For this reason, other models with SARMA(1,1) and SAR(1) in the seasonal part are proposed, such as $ARIMA(0,0,1)(1,0,1)_{12}$, $ARIMA(1,0,1)(1,0,1)_{12}$, $ARIMA(0,0,1)(1,0,0)_{12}$, $ARIMA(1,0,1)(1,0,0)_{12}$, $ARIMA(1,0,0)(1,0,0)_{12}$ and $ARIMA(1,0,0)(1,0,1)_{12}$.

In the case of these models, it can be seen that they are invertible as well as causal and that they accomplish the three hypothesis of the residuals: normality, independence and homoscedasticity.

As an example of the validation process, the analysis of the model $ARIMA(1,0,0)(1,0,1)_{12}$ is presented. Some of the outputs will be placed in the appendix in order to make this section easy to follow. In the first place, the normality of the residuals, with constant variance, has to be assured.

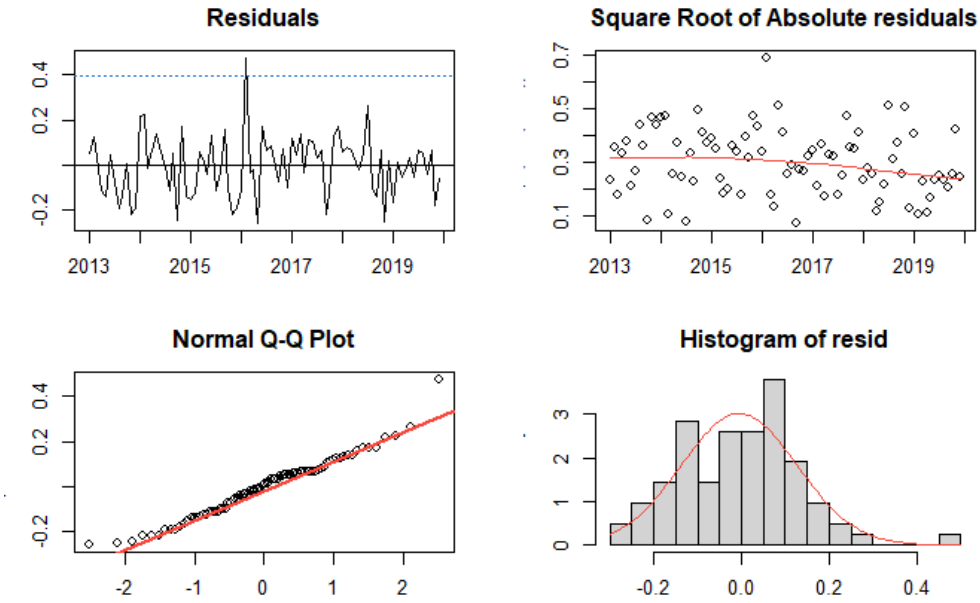


Figure 2.15: Validation plots of model $ARIMA(1,0,0)(1,0,1)_{12}$

In the figure above, two graphics stand out: the square root of absolute residuals and the Normal QQ Plot. In both plots it can be seen how the homoscedasticity and the normality of residuals are accomplished. In the Normal QQ Plot, however, an outlier stands out in the upper-right part of the plot. In the extension of the ARIMA, outliers will be detected and this outlier will be tried to be pushed aside.

The outputs commented below appear in the appendix C.

Looking the normality tests, the Anderson-Darling, Jarque-Bera and Shapiro-Wilks tests do not reject the null hypothesis of normality of the residuals. The homoscedasticity test of studentized Breusch-Pagan give a p-value of 0.2892 so the homogeneity of residuals variance hypothesis can not be rejected. Finally, the independence test of Durbin-Watson is also passed.

The estimated $ARIMA(1,0,0)(1,0,1)_{12}$ model is invertible, meaning that all the roots of the regular MA-characteristic polynomial lie outside the unit circle, i.e the roots are grater than one in absolute value. With that, the model can be expressed as a convergent π-weight AR(∞) model.

The proposed model is also causal: it can be expressed as a convergent MA(∞) due to the fact that all the complex roots of the characteristic polynomial of the AR part lie outside the unit circle.

Finally, looking at the Ljung-Box statistic test, it can be seen that all p-values are higher than 0.05, which means that the residuals independence hypothesis is accomplished. Thus, the proposed model for product 173 is considered valid being causal and invertible with white noise Gaussian residuals.

After validating each of the models and performing the prediction for the last year of series (2019), some indicators of the goodness-of-fit and prediction accuracy can be obtained:

Model	par	σ^2	AIC	BIC	MAPE(%)	meanLength
$ARIMA(0,0,1)(1,0,1)_{12}$	4	0.02	-75.67	-63.51	8.7	568.96
$ARIMA(1,0,1)(1,0,1)_{12}$	5	0.02	-74.17	-59.58	8.6	567.45
$ARIMA(0,0,1)(1,0,0)_{12}$	3	0.02	-63.26	-53.54	11.7	660.53
$ARIMA(1,0,1)(1,0,0)_{12}$	4	0.02	-61.81	-49.66	11.6	659.34
$ARIMA(1,0,0)(1,0,0)_{12}$	3	0.02	-63.74	-54.02	11.4	659.11
$ARIMA(1,0,0)(1,0,1)_{12}$	4	0.02	-76.10	-63.95	8.5	569.11

Table 2.2: Results of the goodness-of-fit and prediction accuracy of the proposed models for classic ARIMA for product 173.

In the Table 2.2 it can be observed that models $ARIMA(1,0,0)(1,0,1)_{12}$, $ARIMA(1,0,1)(1,0,1)_{12}$ and $ARIMA(1,0,0)(1,0,1)_{12}$ are the best in terms of goodness-of-fit and with a similar prediction error to the other models. For these reasons, they are selected to apply the extensions of classic ARIMA: calendar effects correction and outliers treatment.

The two Calendar Effects considered are Weekdays effect and the Easter effect. None of these effects has a T-ratio bigger than 2 in any of the three models, so they are not considered significant.

After that, the outliers detection is applied. With a critical threshold of 2.8, just one atypical observation was detected, the same for the three models, an additional observation in January 2015. Linearizing the series, it can be seen again that taking a regular difference also increases the variance of the series, so the study is performed with the seasonally differentiated series.

Plotting the ACF and the PACF, some models can be identified. Again, implementing a model with a MA(1) in the seasonal part implies to not accomplishing the hypothesis of invertibility of ARIMA. For this reason, other models are tried for the seasonal part, such as AR(1) or ARMA(1,1). For the regular part, MA(1),AR(1) and ARMA(1,1) are tried. After trying some models, the chosen model with significant coefficients and that accomplishes

all the ARIMA hypothesis is $ARIMA(1,0,0)(1,0,1)_{12}$ for the linearized series, which is one of the models identified for the non-linearized series.

To perform the analysis of the residuals and to check the hypothesis of the estimated models, the validation process is performed.

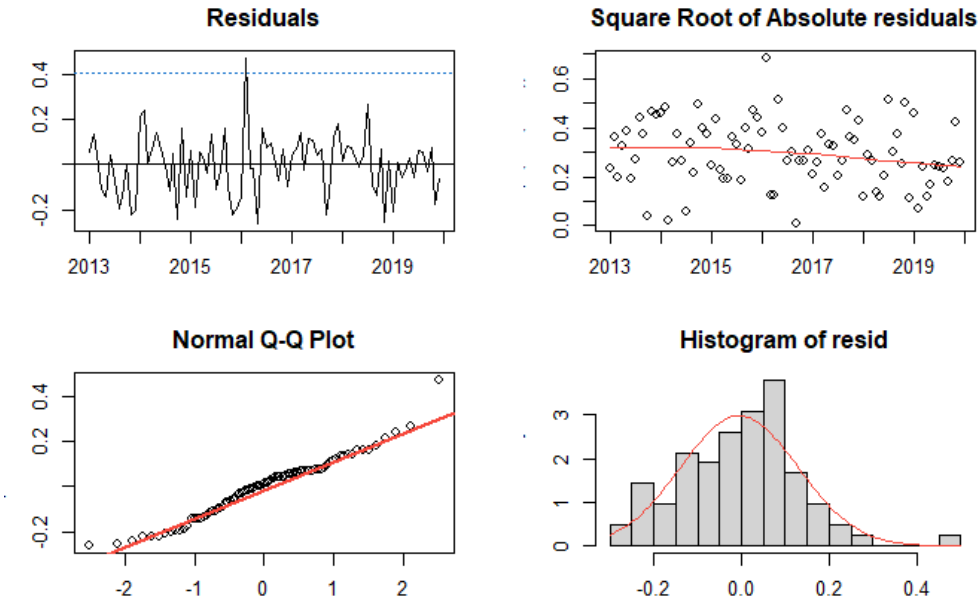


Figure 2.16: Validation plots of model $ARIMA(1,0,0)(1,0,1)_{12} + Atip$.

In the plot above, it can be seen that the homoscedasticity of the residuals is accomplished. In the normality plots, it can be seen that the residuals is normally distributed. However, the outlier seen in plot 2.15 could not be detected by the outlier treatment with the critical threshold assigned.

It could be checked that the model accomplished all the suppositions of the residuals: normality, independence and homoscedasticity. Apart from that the model was also invertible and causal.

Next, the prediction of the last year of series is performed for model $ARIMA(1,0,0)(1,0,1)_{12}+$ $Atip$. Comparing the results obtained with the three best models of Table 2.2, the following KPIs can be obtained:

Model	par	σ^2	AIC	BIC	MAPE (%)	mean Length
$ARIMA(0,0,1)(1,0,1)_{12}$	4	0.02	-75.67	-63.51	8.7	568.96
$ARIMA(1,0,1)(1,0,1)_{12}$	5	0.02	-74.17	-59.58	8.6	567.45
$ARIMA(1,0,0)(1,0,1)_{12}$	4	0.02	-76.10	-63.95	8.5	569.11
$ARIMA(1,0,0)(1,0,1)_{12} + At.$	4	0.02	-75.65	-63.49	9.1	570.39

Table 2.3: Summary of the goodness-of-fit and forecast results of the best models of classic ARIMA and the model treating outliers.

It can be seen that the four models have similar results. In the case of the model $ARIMA(1,0,0)(1,0,1)_{12} + At.$ it performs very similar to the same model without outliers treatment, but with a slightly better performance on the forecasting. However, the four models are good candidates to be chosen.

Finally, the prediction of models $ARIMA(1,0,0)(1,0,1)_{12}$+$Atip.$ and $ARIMA(1,0,0)(1,0,1)_{12}$ are compared:

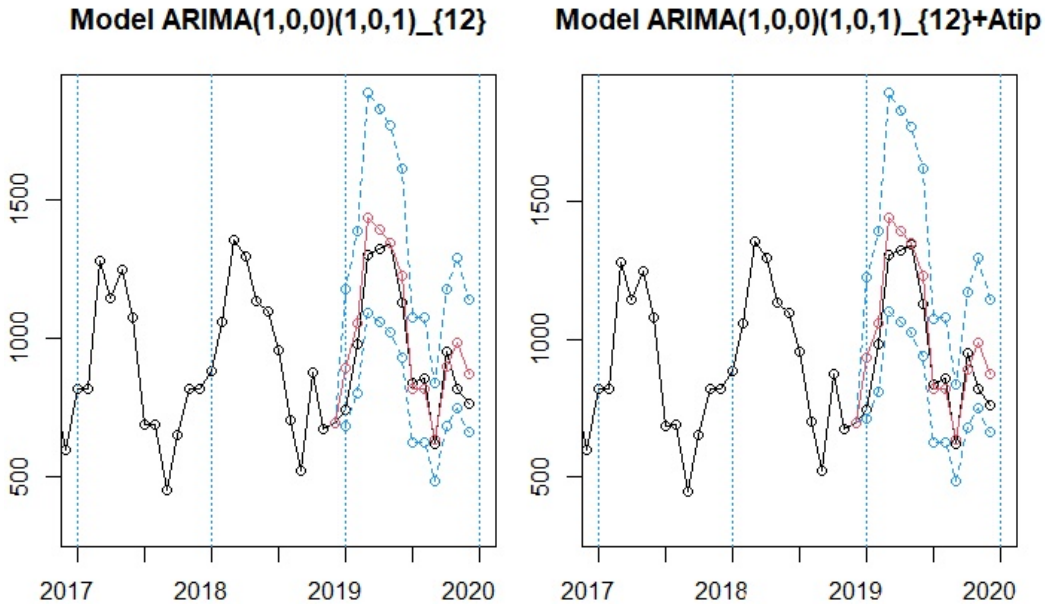


Figure 2.17: Comparison of the forecasts performed by the model $ARIMA(1,0,0)(1,0,1)_{12}$ with and without outlier treatment.

Comparing the results of the two forecasts, it can be seen that the prediction of both models is nearly the same. It can be seen that both models over estimate the sales of

March and April. Additionally, none of the models is able to predict the descent in the sales of November and December. Despite of this, it can be seen that both models do a very good forecast of the sales of 2019. In order to keep the criteria of simplicity, the model $ARIMA(1,0,0)(1,0,1)_{12}$ is chosen to be the most appropriate to fit the sales of product 173, as it has the best goodness-of-fit and forecasting results.

Holt-Winters

Moving forward to the Holt-Winters method, below the process of computing the monthly forecast of the sales of product 173 is shown.

The function applied to the series is the one named HoltWinters of the library *stats*. The criteria to find the initial parameters is to minimize the square prediction error. As it is explained in the methodology section 2.2.1, Holt Winters method can have a multiplicative or a additive seasonality. In this section, both methods are studied.

Holt-Winters with multiplicative seasonality

For the Holt-Winters method with additive seasonality, a MAPE of **8.44%** is obtained. Below, the plot with the actual and fitted values:

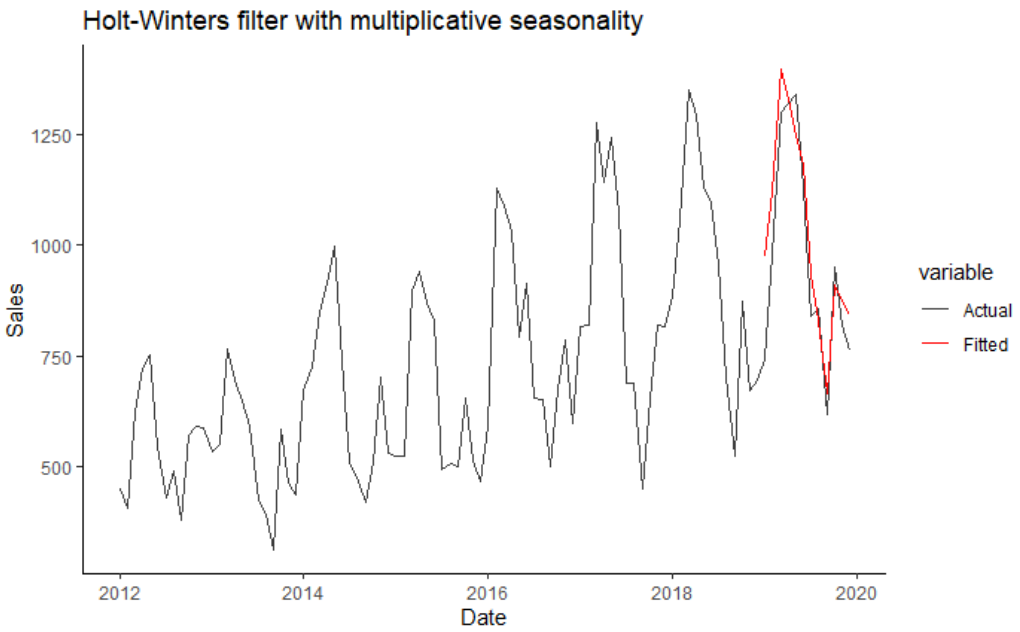


Figure 2.18: Forecast of sales of product 173 using the Holt-Winters method with multiplicative seasonality.

Holt-Winters with additive seasonality

On the other hand, applying Holt-Winters filter with additive seasonality to the data and computing the MAPE of the last year forecast, an error of **9.16%** is obtained. As it can be seen, the method with multiplicative seasonality performs slightly better than the one with the additive one.

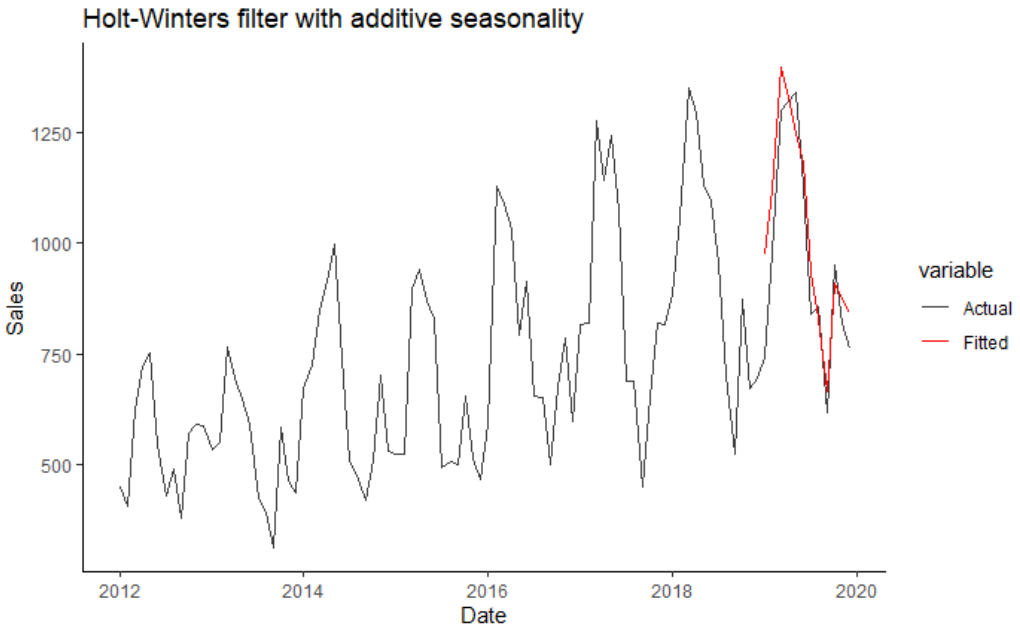


Figure 2.19: Forecast of sales of product 173 using the Holt-Winters method with additive seasonality.

Comparing both plots, there are not significant differences in the fitted values. However, there is a difference in the value of the MAPEs, where the Holt-Winters' with multiplicative seasonality gives a lower error than the additive one. This is expected because of the shape of the time series. The range of the series increases with the level, which indicates that the series has a multiplicative seasonality. Taking this into account and the lower MAPE, the Holt-Winters method chosen to predict sales of product 173 should be the one with multiplicative seasonality.

Prophet

The last forecasting methodology is Prophet model. As explained in methodology, Prophet model is very flexible and it can be customized by several parameters. In this case, no

additional regressors are added, as Kencove did not provide any extra information about what other variables could affect on their sales.

The *prophet* function of library *fbprophet* is used to apply the model. Once the forecast is obtained, the MAPE is computed having into account the real data. Finally, a plot of the forecast of the interval between January 2019 and December 2019 can be obtained.

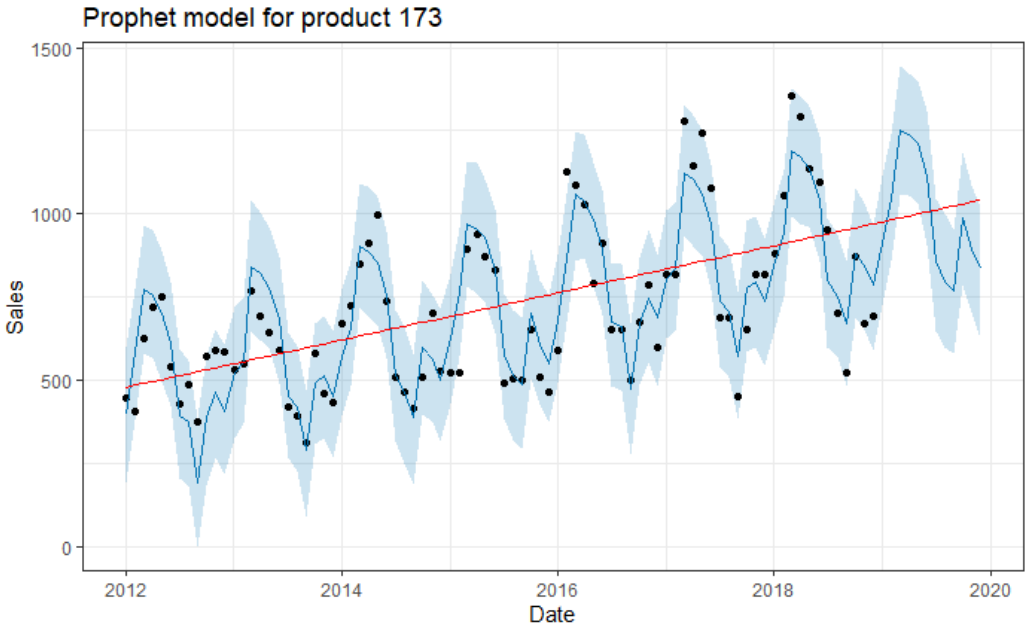


Figure 2.20: Forecast of sales of product 173 using Prophet model.

In the plot above, the black points represent the actual sales, the blue line means the forecasted sales by the model with its corresponding 95% confidence interval. On the other hand, the red line symbolizes the trend of the data. As it can be seen, the sales are growing each year, without any change point.

In this case, a MAPE of **8.97%** is obtained for the last period forecast. The model obtained has only trend and seasonal components, as no holiday or extra regressor has been added to the model. The trend is choosen to be linear, as there did not exist any maximum number of sales. The resulting trend equation is the following:

$$g(t) = 0.3627915t + 0.3542836$$

The vector δ has values of the order of magnitude of -10, so it is considered negligible. That is the reason why the prophet has not detect any changepoint in the time series. On the other hand, only the yearly seasonality is taken into account, as the data was monthly.

The seasonality component has the following expression:

$$s(t) = \left[cos\left(\frac{2\pi(1)t}{365.25}\right), sin\left(\frac{2\pi t}{365.25}\right), cos\left(\frac{4\pi(1)t}{365.25}\right), ..., sin\left(\frac{20\pi t}{365.25}\right) \right] \beta$$

being β:

$$\beta = [0.18, 0.02, 0.16, 0.11, -0.11, ..., 0.15, -0.15]^T$$

In this case, the value N is chosen to be 10 as it is the value set by default in the prophet algorithm in the case of yearly seasonality. This is because according to [10], the value of 10 works well with the majority of series. However, a small hyperparameter tuning was tried in order to see the MAPEs obtained with a different number of N. The result get did not improve the one with $N = 10$, so the default value was left.

Comparative analysis

Finally, a comparative analysis of the five forecasting methods is performed for all the products analysed in this study: products with IDs 76, 79 and 173:

Product	**MAPE (%)**				
	Mov. Mean	ARIMA	HW (mult.)	HW (add.)	Prophet
76	70.7	18.8	29.1	26.7	32.3
79	51.9	33.3	29.3	31.7	41.1
173	22.9	8.47	8.44	9.16	8.97

Table 2.4: Summary table of the MAPE obtained in the prediction of the 2019 sales of products 76,79 and 173 using the 5 explained methodologies.

As it can be seen in Table 2.4, the current methodology used by DDMRP and Kencove (rolling mean) brings pretty bad results in the ADU calculation, reaching a 70% of error in product 76. It can be seen that there are relative big differences in the errors between products. These differences are due that product 173 has a more clear seasonal behaviour, while products 76 and 79 have inexplicable atypical values, that none of the ADU computation methods is able to predict.

The performance of each methodology varies depending on the product. In product 76, ARIMA has better performance than Holt-Winters and Prophet. In this case, the Holt-Winters with additive seasonality has a better performance than the multiplicative one. This is expected as the range of the series do not vary much with the level. The opposite

case happens with products 79 and 173, that is why in these products the Holt-Winters filter with multiplicative seasonality has a lower MAPE than the method with additive seasonality.

Regarding product 79, Holt-Winters with has the best performance, specially the method with multiplicative seasonality. In this case Prophet method gives a 41.1% of error which is significantly worse than the other methods proposed. However, it does not behave as bad as the moving mean, which has a MAPE of 51.9%.

In the case of product 173, ARIMA, Holt-Winters and Prophet have a very similar performance, having an approximate error of 9%. On the other hand, the moving mean still presents a notable MAPE of 22.9%.

2.3.3 Stock Buffer computation results

The first step of the estimation of the Stock Buffer Red Zone is to choose a methodology to estimate the ADU. The most important characteristic of the selected method is that it should be simple to automatize, as the ADU estimation process will be repeated several times in order to obtain the residuals. Looking into the code, it can be seen that ARIMA needs several steps of analysis and interpretation, so it can not be easily automatized. On the other hand, Holt-Winters filter and Prophet model are straightforward to implement inside a production environment. Among these two methods, in Table 2.4, it can be seen that Holt-Winters has a slightly better performance than Prophet.For these reasons, the chosen model to compute the ADU of a product is the Holt-Winters filter (with additive or multiplicative seasonality depending on the product), as it has pretty good results and it can be easily implemented in DDMRP.

Once a the methodology to obtain the ADU of each product is chosen, next step is to size the Stock Buffer of every item. In this case, only the process followed for product 79 is explained, but in table 2.5 the result for the three other products is shown. The product chosen is 79 because is the one which has a poorer forecast, so sizing the buffers will have a bigger impact on the stock flow. Stock buffers take more importance in products like 79, as the forecast is not enough accurate to rely completely on it. This is the reason why the sizing of the Red Zone is explained for this product.

Remember that for this analysis the whole available database will be used (from January 2012 until December 2021). To obtain the residuals distribution all the possible data is needed, in order to find a known distribution that can fit the data. In the estimation of the ADU, the pandemic years were left in order to find the most accurate ADU computation method ignoring COVID effects. However, buffers have to be sized to stand variability

situations such as the one lived in 2020 and 2021, so including these years of data brings the opportunity to estimate the Red Zone with these conditions.

First step is to analyse the residuals of the ADU forecasting of product 79. In order to do so, the cross-validation method needs to be done. In the case of the products presented in the Kencove database, the first training set will take values from January 2012 until December 2014, forecasting the following year. The operation will be repeated until forecasting the last year of data. For each forecast, the residuals are kept, obtaining 84 values which will form a distribution.

Once the prediction is obtained, the residuals are computed by subtracting the forecast from the current sales. This way, the monthly residuals are gotten. However, it is necessary to obtain the daily residuals, as the units of DLT are in days. In order to do so, the monthly residuals are divided by the number of days of each month.

The first group of daily residuals is kept and the next iteration is done amplifying one year the training set. This process is repeated successively until 7 predictions are computed and its residuals saved.

Unlike ARIMA that its residuals have to fit in a Normal distribution to accomplish the hypothesis, Holt-Winters filter is not a model, so its residuals do not have to follow any particular distribution. At first sight, the residuals does not seem to fit any known function, mainly because of its big tails.

However, the function *fitdistr* from library *MASS* can be used to obtain the most fitting parameters of some well-known distributions. Taking into account that the values are located in the real line, data is not uniform and it is more or less symmetric, three possible distributions are selected: Normal, Cauchy and Logistic. In the Figure 2.21 , the residuals distribution compared to Normal and Cauchy distribution can be observed:

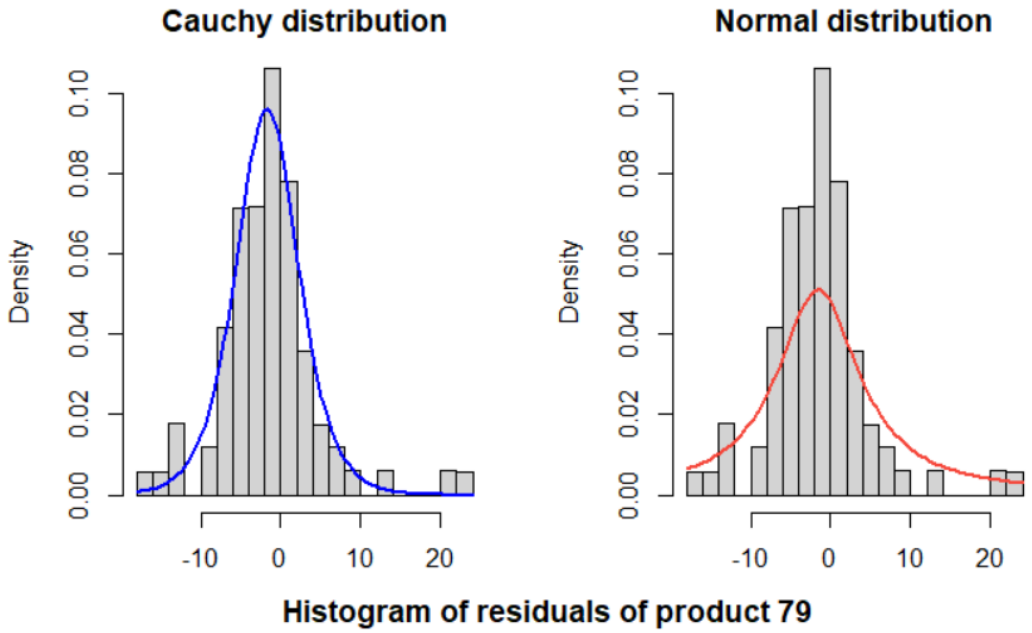


Figure 2.21: Histogram of the residuals of the prediction vs. Cauchy and Normal distributions.

As it can be seen in the plot above, the Cauchy distribution fits quite well the middle values, but does fit the heavy tails. The Normal distribution, on the other hand, seems to fits better the tails but does not fit the central values.

Unfortunately, none of these distributions passed the Kolmogorov-Smirnoff or the Chi-Square test, so they could not be assigned to the residuals distribution. However, both Normal and Cauchy distribution will be tried for product 79 in order to see possible differences when changing the distribution.

Once the function to simulate has been chosen, a daily prediction of the ADU has to be obtained. To get this prediction, the Holt-Winters with multiplicative seasonality method will be used to obtain monthly forecast in the first place. After that, the monthly sales will be divided by the days of the month to obtain the ADU daily data.

In the case of product 79, the known data is a cost of 1.17$, a profit of 3.58$ and a DLT of 37 days. The forecast of the ADU starts at January 2022, so the ADUs used in the simulation will be the ones corresponding to the first 37 days of the year.

It is worth to say that the simulation of the Red Zone should be done with 37 days in advance, in order to be able to react to the results. In this case, the simulation performed

should have been done in November 2021, in order to have time to buy more products to the supplier if necessary.

With all this information, the computing of the cost function with Montecarlo method can be performed. The equations of the cost function are:

$$f(ADU, stock) = \left(stock - \sum_{i=1}^{DLT} ADU.sim_i \right)^+ + \frac{profit}{cost} \left(\sum_{i=1}^{DLT} ADU.sim_i - stock \right)^+$$

being $ADU.sim_i$:

$$ADU.sim_i = ADU_i + Normal(\mu, \sigma)$$

This is in the case of choosing the Normal distribution to explain residuals. In the case of choosing the Cauchy distribution for the residuals, the $ADU.sim_i$ would be different:

$$ADU.sim_i = ADU_i + Cauchy(x_0, \gamma)$$

Finally, a table with the result of the cost function depending on the stock is gotten for each iteration. As it will be seen in the cost function vs. stock plot, there is quite variance among the cost function results. For this reason,it is necessary to obtain the median cost function for each stock level, in order to obtain a single measure.

Below, the plots which represent the results obtained with Normal and Cauchy distributions to represent the residuals:

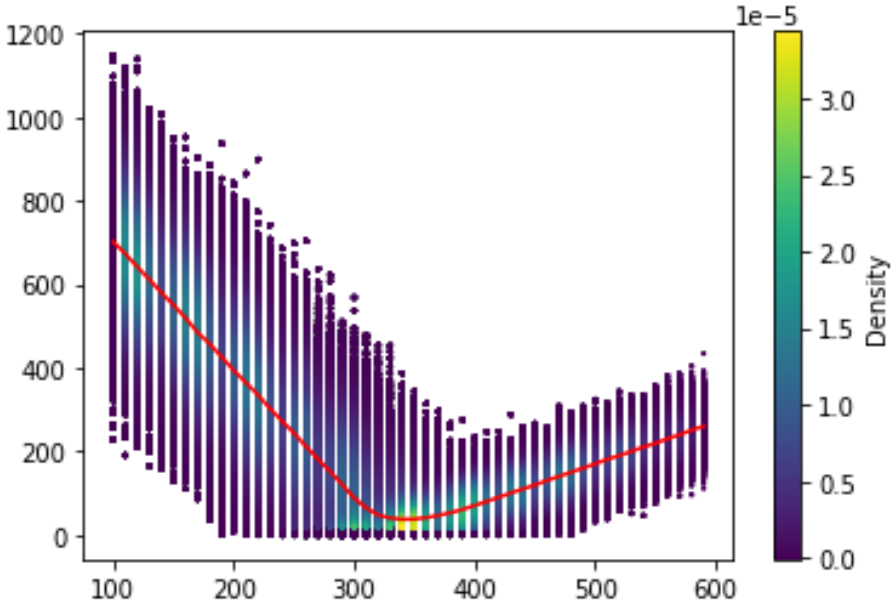


Figure 2.22: Value of the cost function depending on the stock level, if residuals follow a Normal distribution for product 79.

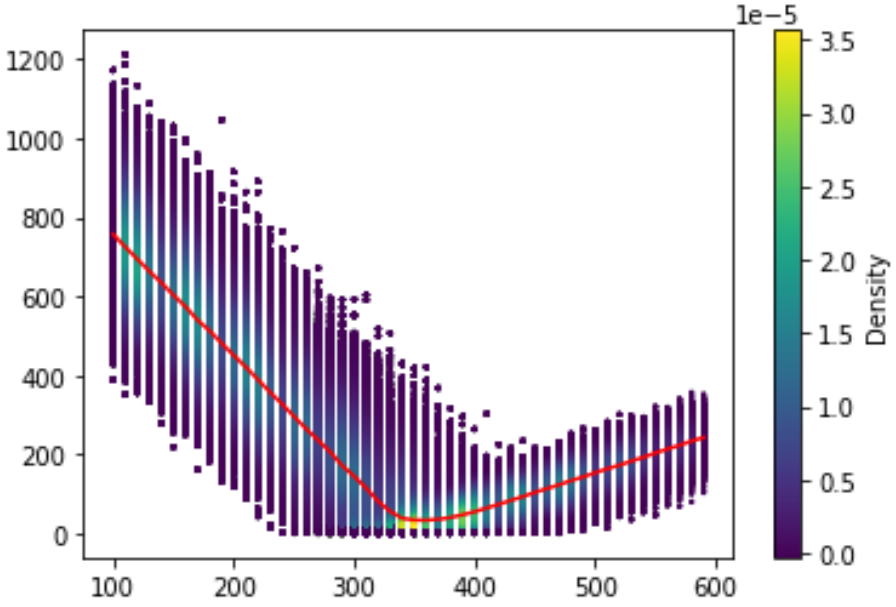


Figure 2.23: Value of the cost function depending on the stock level, if residuals follow a Cauchy distribution for product 79.

In the figures above, the density plot of the results of the cost function versus the stock is shown. As it can be seen the results do not vary much between Normal and Cauchy distributions: the cost function has big variance until 300 units of stock, approximately. After that, the results seem much more scattered because the positive limit imposed in the cost function. The red line represents the median cost function for each level of stock. The diagram has a clear global minimum, between 300 and 380 units of stock.

Another point of the plots is the shape of the median line. Before reaching the minimum the slope of of the median cost function line is more steep than after reaching the minimum. This is related to the ratio between the profit and cost. As this ratio is bigger than 1, it is more profitable to exceed stock rather than run out of it. For this reason, having more than 400 units is less penalized by the cost function than having less than 300 units.

Computing the exact minimum, by the *min* function, it is obtained that **340 units of stock** will imply a minimum cost function in the case of using the **Normal distribution**. On the other hand, the level of stock necessary to minimize the cost function is of **360 units if the Cauchy distribution** is used. As the Cauchy distribution seems to fit better in the residuals histogram, the 360 units of stock are chosen to size the Red Zone of the buffer.

In the case of products 76 and 173, the distribution that fits better the residuals of the forecasts is the Cauchy distribution. Below, the simulation of product 173 with Cauchy distribution:

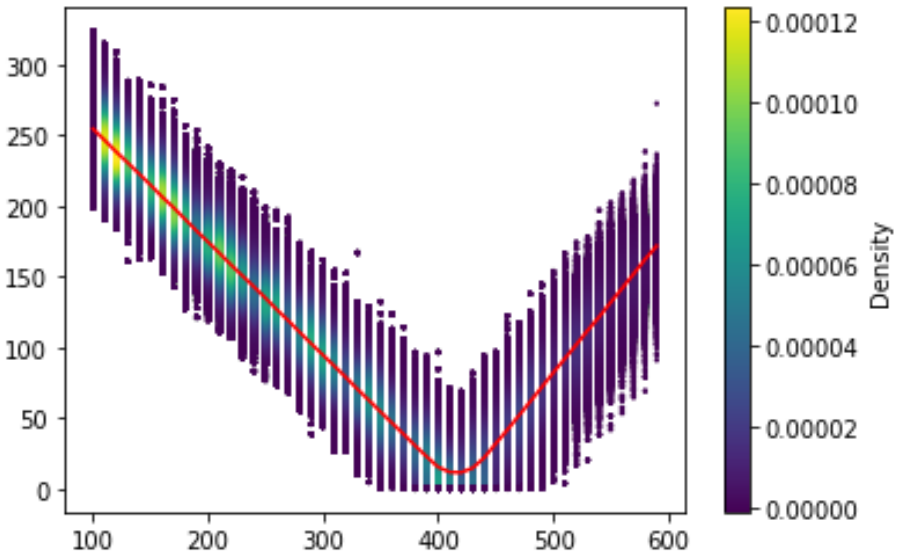


Figure 2.24: Value of the cost function depending on the stock level, if residuals follow a Cauchy distribution for product 173.

Regarding product 173, it can be seen that the simulation has a lot less variance than the one of product 79. This is caused because the forecast of this product showed a considerable fewer error than the one of product 79. This way, the residuals are gathered in a tighter range of values and the simulation has less variance. In this case, the ratio between profit and cost is 0.8, so having too much stock is more penalised than having too little. This is reflected in Figure 2.24, where the slope after the minimum is less sharp than before it.

Finally, the table that shows the results of the three simulations performed for products 76,79 and 173. All three simulations are done considering that the residuals of the forecast follow a Cauchy distribution.

Product	Ratio profit/cost	DLT	ADU_{avg}	Stock level(Red Zone)
79	3.06	37	10	360
76	4.23	43	9	410
173	0.80	12	33	420

Table 2.5: Results of the Montecarlo simulation for the computation of the Red Zone for products 79,76 and 173.

As it can be seen in the table above, the resulting Stock level is pretty similar to the value resulting from $ADU_{avg}DLT$. This is expected, as the Red Zone original computation method also depends on this value. Nevertheless, in this method the ratio of profit/cost of the product is taken into account to compute the buffer zone, as it is the main concern of the company.

Chapter 3

Conclusions

DDMRP is a supply chain methodology developed with the goal of giving more flexibility and break the dependent structure of the classic Material Requirements Planning. However, DDMRP presents some issues when estimating the Average Daily Usage of a product and the safety stock (or Red Zone) of Stock Buffers. Stock Buffers are an innovation introduced by DDMRP as a tool for decoupling the flow of stock.

On one hand, the calculation of the daily sales of the product or ADU is said to be done with a moving average. However, this methodology is quite disconcerting for company planners, as they have to establish the window of this rolling mean and it does not give accurate forecasts. For this reason, other methodologies to predict the daily sales of the product are proposed. The three methodologies suggested to compute the ADU are: ARIMA, Holt-Winters and Prophet.

On the other hand, to compute the dimension of the Red Zone, planners have to establish two factors: LTF and VF. This can be confusing in the most part of the cases, so an alternative solution is proposed: a Montecarlo simulation. The aim of the process is to find the units of stock that minimize a stated cost function. The cost function formulated has as the main goal to have the maximum possible revenue, as the cost and profit of the product are taken into account.

The database used to explore these hypothesis was extracted from a company called Kencove, which had been testing DDMRP on some products. Three representative products of the company were selected in order to obtain results of the above proposals: products 76,79 and 173.

Taking these experimental products, different behaviours were detected when computing the ADU. There were products which had little error when computing the ADU, of the

order of the 10%. However, there were other products that using the same methodology give an error of nearly the 30%. This was expected, as the DDMRP expects having non-accurate forecasts as it is something very common in supply chain management. In this kind of products is where DDMRP Stock Buffers are more useful.

Applying ARIMA, Holt-Winters and Prophet to forecast one year of sales, the results of these three methodologies were considerably better than the ones obtained by a rolling mean, which is the method proposed by DDMRP. However, ARIMA would be less feasible to implement it in a production environment such as Kencove company, as they do not have any time series expert who could choose the appropriate model for each product along time.

Holt-Winters and Prophet do not present this problematic and comparing the results obtained with the three products, Holt-Winters methodology had a slightly better performance than Prophet. For this reason, Holt-Winters filter was selected to be the most appropriate method to compute the ADU, choosing additive or multiplicative seasonality depending on the product series.

In the case of Montecarlo simulation, the first step is to identify which distribution do the residuals of the forecast follow. When analysing the three products, it can be seen that the Cauchy distribution is the one that fits better the data. Performing the Montecarlo simulation, the number of stock units that minimize the stated cost function can be obtained. These stock levels will be the Red Zones of the Stock Buffers of each product. The main advantages of this solution regarding the one proposed by DDMRP, is that the simulation takes into account the cost/profit ratio of the product and it does not depend on LTF and VF.

Next step of this study would be to implement these changes in the DDMRP production environment of Kencove. It would be interesting to see how the forecasting methodologies and Montecarlo simulation can be implemented in the Enterprise Resource Planning of the company.

On conclusion, it has been found that there are better methodologies than a moving mean to compute the ADU. Using Holt-Winters the planner would not need to establish the wide of the rolling window and the error obtained is significantly smaller. Regarding the Montecarlo simulation proposed to estimate the Red Zone, the planner has no longer to decide the Variability Factor and Lead Time Factor and the cost/profit of the product in the calculation of the safety zone. These two changes reach the aim of the master thesis: estimate the Average Daily Usage more accurately and define a sizing for the Red Zone which do not depend on choosing factors in a subjectively way.

Bibliography

[1] Ptak, Carol and Smith, Chad, *Demand Driven Materials Requirements Planning (DDMPR)*. 2019.

[2] R. J. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder, *Springer Series in Statistics Forecasting with Exponential Smoothing*. 2008.

[3] R. Pellegrino and N. Costantino, "A Monte Carlo Simulation and Fuzzy Delphi-Based Approach to Valuing Real Options in Engineering Fields," *Risk Management for the Future - Theory and Cases*, no. April 2012, 2012.

[4] Soumya Shrivastava, "Cross Validation in Time Series.." https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4, 2020.

[5] Blackstone, John H., and James F. Cox, *APICS Dictionary | ASCM*.

[6] M. Marti-Recober, P. Munoz Gracia, J. A. Sanchez-Espigares, and L. M. Acosta Argueta, *Forecasting Time Series*. 2020.

[7] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications*. Springer, 2000.

[8] J. Sanchez Espigares and L. Acosta, "Forecasting Time Series Course Material," 2020.

[9] "Facebook Prophet and the Stock Market — Advancing Analytics." https://www.advancinganalytics.co.uk/blog/2021/7/26/facebook-prophet-and-the-stock-market-part-2.

[10] S. J. Taylor and B. Letham, "Forecasting at Scale," *American Statistician*, vol. 72, no. 1, pp. 37–45, 2018. https://peerj.com/preprints/3190/.

[11] R. J. Hyndman and G. Athanasopoulos, "Forecasting: Principles and Practice." https://otexts.com/fpp3/.

[12] D. Peña, *Análisis de series temporales*. Alianza Editorial, 2005.

[13] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *Journal of Experimental Social Psychology*, vol. 49, no. 4, pp. 764–766, 2013.

[14] W. Robson, "The Math of Prophet." https://medium.com/future-vision/the-math-of-prophet-46864fa9c55a, 2019.

[15] R. Miclo, M. Lauras, F. Fontanili, J. Lamothe, and S. Melnyk, "Demand driven mrp: assessment of a new approach to materials management," *International Journal of Production Research*, vol. 57, pp. 1–16, 04 2018.

[16] N. Vanderput, *Data Science for Supply Chain Forecasting*. De Gruyter, 2nd ed., 2021.

[17] N. Vanderput, *Inventory Optimization: Models and Simulations*. De Gruyter, 1st ed., 2020.

[18] R. Miclo, *Challenging the "Demand Driven MRP" Promises : a Discrete Event Simulation Approach*. PhD thesis, 12 2016.

[19] M. Ihme and R. Stratton, "Evaluating demand driven mrp: a case based simulated study," 2015.

[20] M. Gilliland, L. Tashman, and U. Sglavo, *Business Forecasting: Practical Problems and Solutions*. Wiley, 1st ed.

Appendix A

Demand Driven Planning

It is important to know the Demand Driven Material Requirements Planning day-to-day mechanism. The planner is the person in charge to launch manufacturing and purchasing orders in a company and the responsible of the supply chain. Each day the net flow equation has to be computed. The net flow equation is:

$$\text{on-hand} + \text{on-order} - \text{qualified sales order demand} = \text{net flow position} \tag{A.1}$$

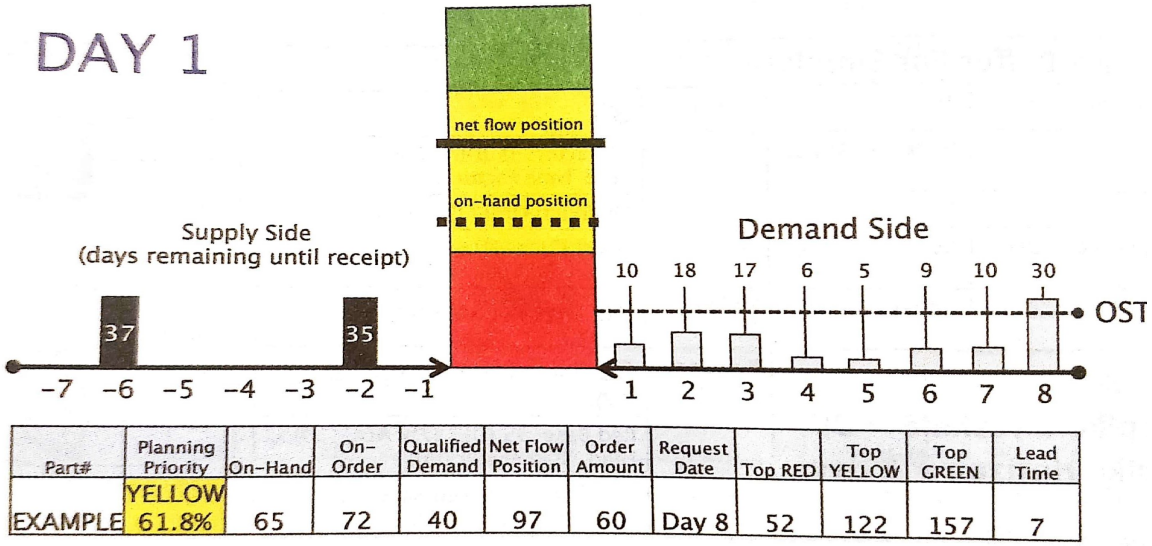
where *on-hand* represents the quantity of stock physically available, *on-order* is the quantity of stock that has been ordered but not received and *qualified sales order demand* is the sum of sales past due, sales orders due and quantity ordered in qualified spikes. A qualified spike is an order where the quantity of product requested is above the Order Spike Threshold and the order date is inside the Order Spike Horizon. Usually the Order Spike Threshold is computed as 3 times the ADU or 50% of the red zone and the Order Spike Horizon is the DLT plus one.

The supply order generation is based on the calculation of net flow position for every decoupled component each day. Depending on the result of the net flow equation, the planner will launch an order or not. If the Net Flow Position falls into the green zone, the planner does not need to launch any order, if the NFP is inside the yellow or red zone, the product needs to be refilled. The quantity the planner has to order is the difference between the Net Flow Position and the top of green (TOG), which is the sum of the green, yellow and red zones. This means that a typical order will be at least the full amount of the green zone.

In summary, the planner will view a table with the buffer colour each product is in that day. Apart from the color, the planner will also have access to the planning priority, a percentage which indicates how urgent is the refilling of that product. Is computed dividing

the NFP by the top of green. That way, if the planner has to choose between to order two products in yellow zone it can choose the one with a lower percentage of TOG. This relative priority distinction is a crucial differentiator between conventional MRP and DDMRP.

To end this introduction to DDMRP, an example of supply order Generation is shown.



Part#	Planning Priority	On-Hand	On-Order	Qualified Demand	Net Flow Position	Order Amount	Request Date	Top RED	Top YELLOW	Top GREEN	Lead Time
EXAMPLE	YELLOW 61.8%	65	72	40	97	60	Day 8	52	122	157	7

Figure A.1: Simulation Day 1. Reference: [1]

In the picture above, a simulation of the stock situation of a product is presented. In this case, there are 65 units in the warehouse, and there are 72 units pending to arrive: 35 in 2 days and 37 in 6 days. On the other hand, today there is a demand of 10 units of the product and no units pending to deliver. As it can be seen, there only one order which exceeds the Order Spike Threshold of 26, so no qualified spikes are considered in a time window of 8 days (DLT+1) If we compute the net flow equation:

$$65 + 72 - (10 + 0 + 30) = 97 \tag{A.2}$$

As the yellow zone goes from 52 to 122, the Net Flow Position of the product is inside it. Computing the planning priority:

$$\frac{97}{157} = 61.8 \tag{A.3}$$

In this case, the planning priority is Yellow, 61.8%. The planner needs to refill the product, and the quantity ordered will be 60 as it is the difference between 157 and 97. The planner launches the order and the next day all these parameters are recomputed.

When supply order is generated at a higher level, decoupling stops the explosion of the

bill of material at decoupling points placed at lower levels. The explosion will be stopped because the products in decoupling points are buffered. The difference between material explosion in MRP and DDMRP can be seen below:

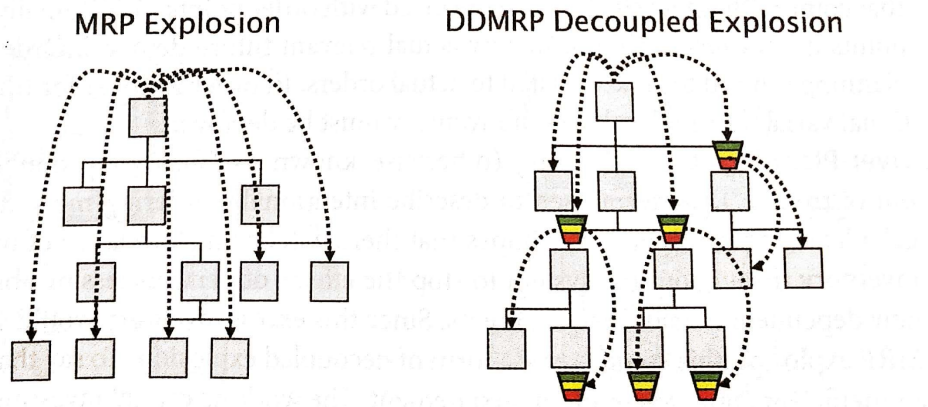


Figure A.2: Explosion of two Bill of Materials: one following MRP and the other one DDMRP. Reference: [1]

Appendix B

Sales history for products 76,79 and 173

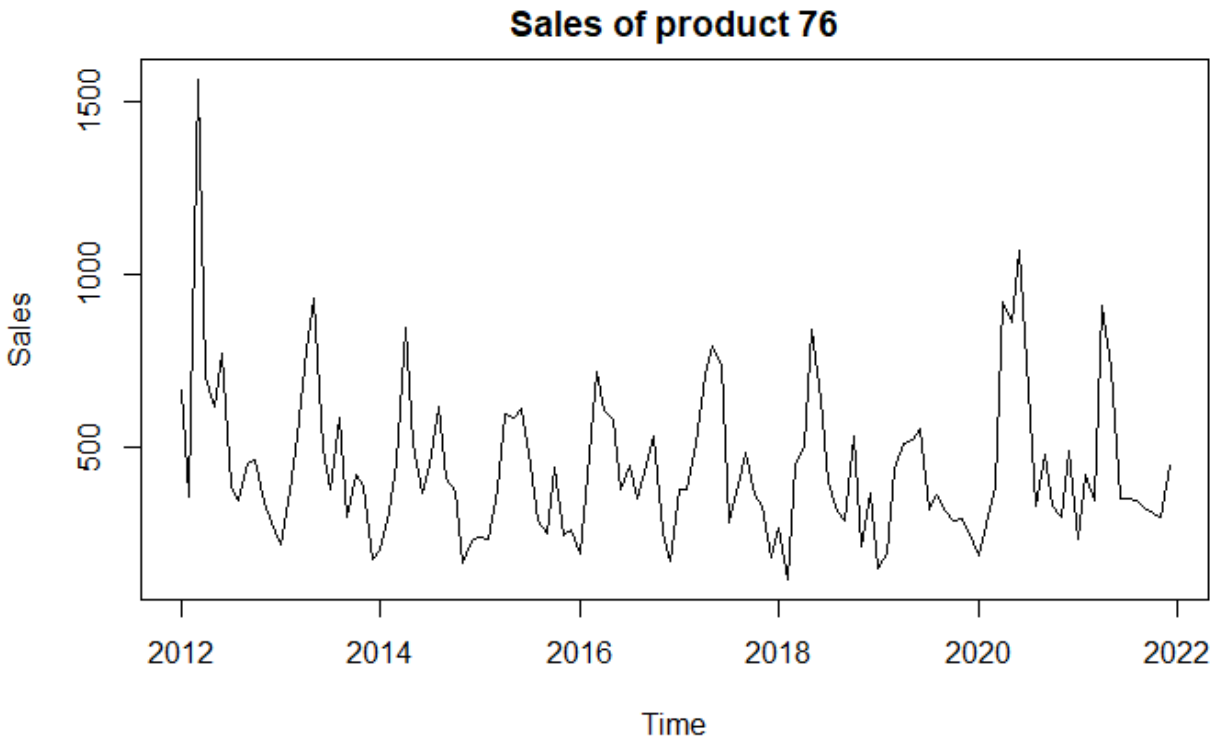


Figure B.1: Sales history of product 76.

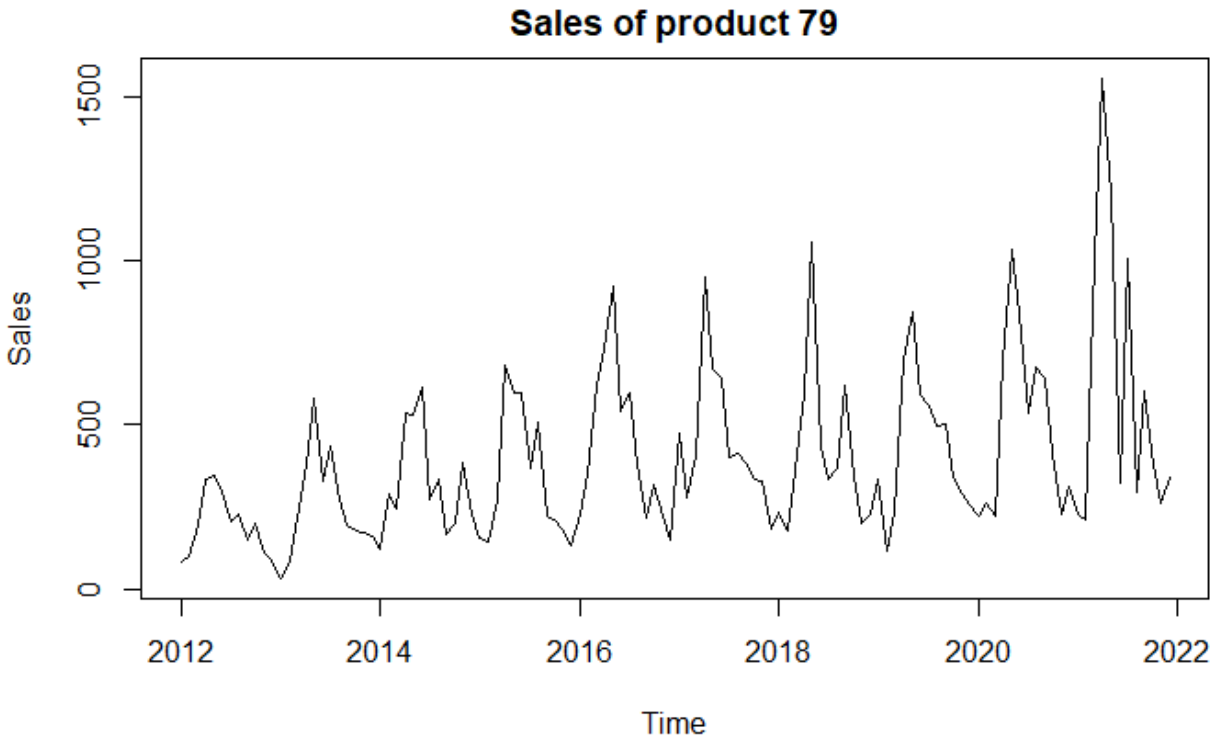


Figure B.2: Sales history of product 79.

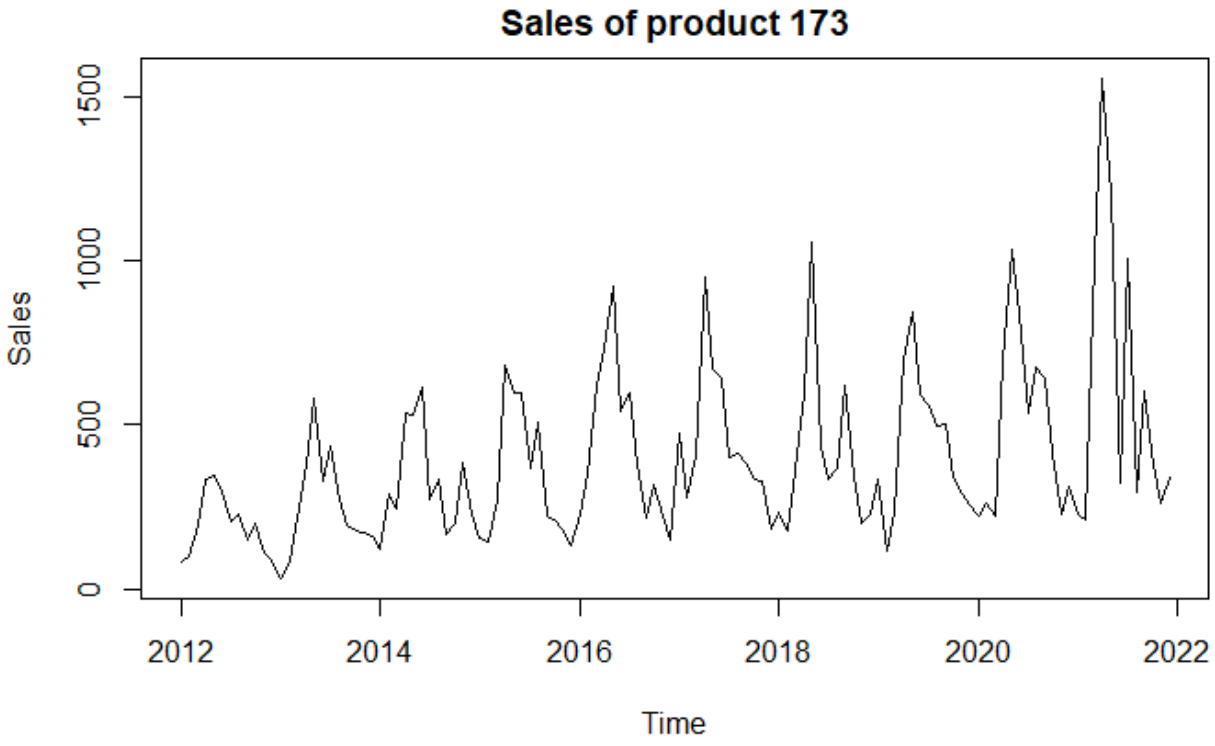


Figure B.3: Sales history of product 173.

Appendix C

$ARIMA(1,0,0)(1,0,1)_{12}$ results

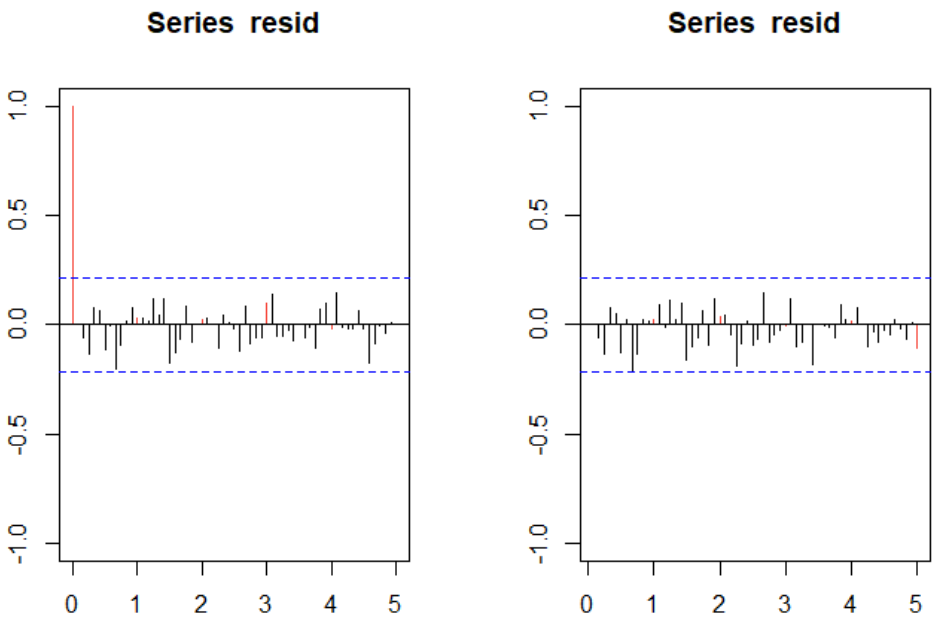


Figure C.1: ACF and PACF plots for residuals of model $ARIMA(1,0,0)(1,0,1)_{12}$

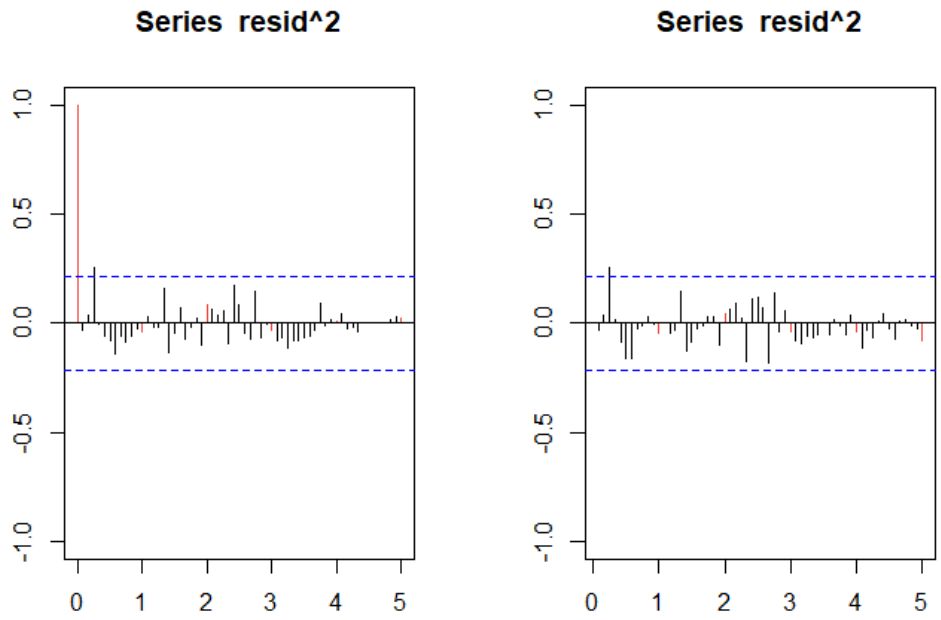


Figure C.2: ACF and PACF plots for square residuals of model $ARIMA(1,0,0)(1,0,1)_{12}$

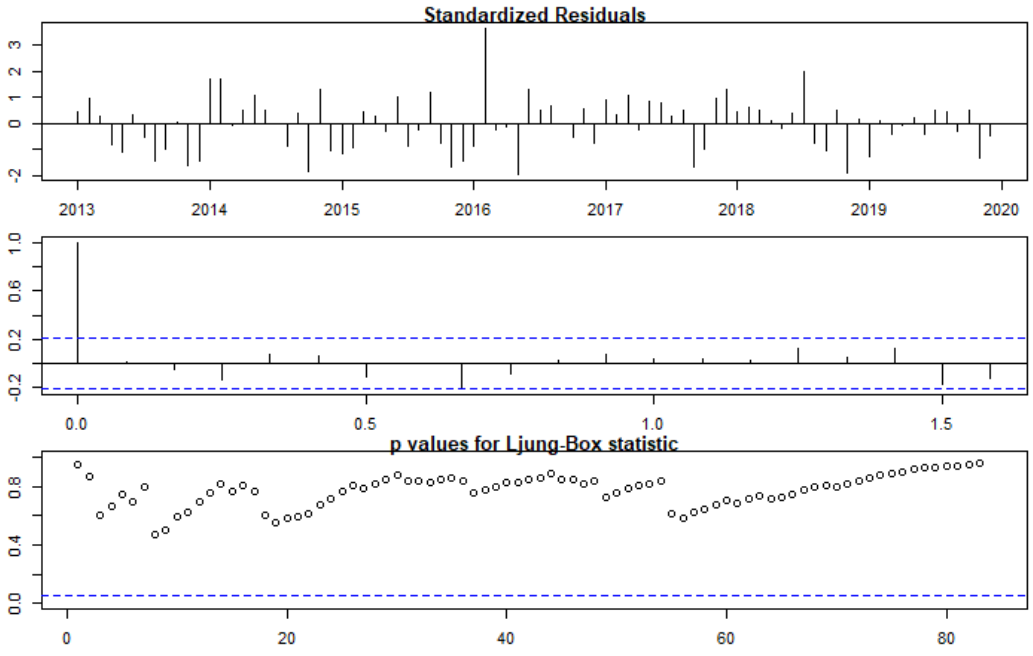


Figure C.3: Standardized residuals and plot of the Ljung-Box statistic $ARIMA(1,0,0)(1,0,1)_{12}$

Listing C.1: Pi-weights and Normality tests

```
-----------------------------------------------------------------

Call:
arima(x = d12lnserie.lin1, order = c(1, 0, 0), seasonal = list(order = c(1,
    0, 0), period = 12))

Coefficients:
         ar1     sar1  intercept
      0.6078  -0.5785     0.0941
s.e.  0.0869   0.0848     0.0324

sigma^2 estimated as 0.03138:  log likelihood = 23.53,  aic = -39.05

Modul of AR Characteristic polynomial Roots:  1.046668 1.046668 1.046668 1.046668 1.046668 1.046668 1.0

Modul of MA Characteristic polynomial Roots:

Psi-weights (MA(inf))

--------------------
       psi 1        psi 2        psi 3        psi 4        psi 5        psi 6        psi 7
 0.607762931  0.369375780  0.224492907  0.136438467  0.082922243  0.050397065  0.030629468
       psi 8        psi 9       psi 10       psi 11       psi 12       psi 13       psi 14
 0.018615455  0.011313784  0.006876098  0.004179038 -0.575945769 -0.350038489 -0.212740418
      psi 15       psi 16       psi 17       psi 18       psi 19       psi 20
-0.129295740 -0.078581158 -0.047758715 -0.029025976 -0.017640913 -0.010721493

Pi-weights (AR(inf))

--------------------
      pi 1       pi 2       pi 3       pi 4       pi 5       pi 6       pi 7       pi 8
 0.6077629  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
      pi 9      pi 10      pi 11      pi 12      pi 13      pi 14      pi 15      pi 16
 0.0000000  0.0000000  0.0000000 -0.5784856  0.3515821  0.0000000  0.0000000  0.0000000
     pi 17      pi 18      pi 19      pi 20
 0.0000000  0.0000000  0.0000000  0.0000000

Normality Tests

--------------------

        Shapiro-Wilk normality test

data:  resid(model)
W = 0.97295, p-value = 0.07387


        Anderson-Darling normality test

data:  resid(model)
A = 0.4243, p-value = 0.3111


        Jarque Bera Test

data:  resid(model)
X-squared = 14.264, df = 2, p-value = 0.0007992
```

Listing C.2: Homoscedasticity and independence tests

```
Homoscedasticity Test

--------------------

        studentized Breusch-Pagan test

data:  resid(model) ~ I(obs - resid(model))
BP = 0.84326, df = 1, p-value = 0.3585


Independence Tests

--------------------

        Durbin-Watson test

data:  resid(model) ~ I(1:length(resid(model)))
DW = 2.2002, p-value = 0.7911
alternative hypothesis: true autocorrelation is greater than 0


Ljung-Box test
     lag.df  statistic   p.value
[1,]      1  0.8652961 0.3522609
[2,]      2  1.4872648 0.4753840
[3,]      3  1.6112025 0.6568524
[4,]      4  1.7707746 0.7778247
[5,]     12 13.1011588 0.3617311
[6,]     24 30.8178435 0.1591046
[7,]     36 43.4188094 0.1845961
[8,]     48 51.3597876 0.3434964
```

Appendix D

Code used for the estimation of the ADU

The development of the code to estimate the Average Daily Usage (for the moving mean and the three alternative methods) is done in R language.

Estimation of ADU of product 173 using a moving mean

```
library(knitr)
library(zoo)
library(readr)
library(ggplot2)


df<-read_csv("final_data/df173_or.csv")
(serie<-ts(df$y,start=2012,freq=12))
pred<-head(rollmean(serie,k=3),-1)
pred_serie <- ts(pred,freq=12,start=c(2012,4))
last_year_pred<-head(tail(rollmean(serie,k=3),13),12)
last_year_pred

obs<-tail(df$y,12)
pr<-round(as.numeric(last_year_pred),0)

obs
pr
MAPE_rolling_mean=sum(abs(obs-pr)/obs)/12
MAPE_rolling_mean#MAPE: mean absolute prediction error
#plotting
ts.plot(serie, pred_serie, gpars = list(col = c("black", "red"),xlab="Year",
ylab="Sales",main = "Sales␣forecast␣for␣product␣173␣by␣rolling␣mean"))
legend("topleft", bty="n", lty=c(1,1), col=c("black","red"),
       legend=c("␣Observed␣", "Predicted␣"))
data_zoo <- as.zoo(pred_serie)
last_zoo <- data_zoo[(length(data_zoo)-11):length(data_zoo)]
p2<-as.ts(last_zoo)
#plotting
ts.plot(serie, p2
```

```
        , gpars = list(col = c("black", "red"),xlab="Year",ylab="Sales",
        main = "Sales␣forecast␣for␣product␣173␣by␣rolling␣mean"))
legend("topleft", bty="n", lty=c(1,1), col=c("black","red"),
       legend=c("␣Observed␣", "Predicted␣"))
```

Estimation of ADU of product 173 using a ARIMA

```
## ------------------------------------------------------------------------
library(forecast)
library(xtable)
knitr::purl("ARIMA_2.Rmd", "ARIMA2.R", documentation = 3)


## ------------------------------------------------------------------------
#################Validation#################################
validation=function(model,dades){
  s=frequency(get(model$series))
  resid=model$residuals
  par(mfrow=c(2,2),mar=c(3,3,3,3))
  #Residuals plot
  plot(resid,main="Residuals")
  abline(h=0)
  abline(h=c(-3*sd(resid),3*sd(resid)),lty=3,col=4)
  #Square Root of absolute values of residuals (Homocedasticity)
  scatter.smooth(sqrt(abs(resid)),main="Square␣Root␣of␣Absolute␣residuals",
                 lpars=list(col=2))

  #Normal plot of residuals
  qqnorm(resid)
  qqline(resid,col=2,lwd=2)

  ##Histogram of residuals with normal curve
  hist(resid,breaks=20,freq=FALSE)
  curve(dnorm(x,mean=mean(resid),sd=sd(resid)),col=2,add=T)


  #ACF & PACF of residuals
  par(mfrow=c(1,2))
  acf(resid,ylim=c(-1,1),lag.max=60,col=c(2,rep(1,s-1)),lwd=1)
  pacf(resid,ylim=c(-1,1),lag.max=60,col=c(rep(1,s-1),2),lwd=1)
  par(mfrow=c(1,1))

  #ACF & PACF of square residuals
  par(mfrow=c(1,2))
  acf(resid^2,ylim=c(-1,1),lag.max=60,col=c(2,rep(1,s-1)),lwd=1)
  pacf(resid^2,ylim=c(-1,1),lag.max=60,col=c(rep(1,s-1),2),lwd=1)
  par(mfrow=c(1,1))

  #Ljung-Box p-values
  par(mar=c(2,2,1,1))
  tsdiag(model,gof.lag=7*s)
  cat("\n----------------------------------------------------------------------\n")
  print(model)

  #Stationary and Invertible
  cat("\nModul␣of␣AR␣Characteristic␣polynomial␣Roots:␣",
      Mod(polyroot(c(1,-model$model$phi))),"\n")
  cat("\nModul␣of␣MA␣Characteristic␣polynomial␣Roots:␣",
      Mod(polyroot(c(1,model$model$theta))),"\n")

  #Model expressed as an MA infinity (psi-weights)
```

```
psis=ARMAtoMA(ar=model$model$phi,ma=model$model$theta,lag.max=36)
names(psis)=paste("psi",1:36)
cat("\nPsi-weights␣(MA(inf))\n")
cat("\n--------------------\n")
print(psis[1:20])
plot(psis,type="h",main="Pesos␣Psis␣-␣MA␣infinito")

#Model expressed as an AR infinity (pi-weights)
pis=-ARMAtoMA(ar=-model$model$theta,ma=-model$model$phi,lag.max=36)
names(pis)=paste("pi",1:36)
cat("\nPi-weights␣(AR(inf))\n")
cat("\n--------------------\n")
print(pis[1:20])

plot(pis,type="h",main="Pesos␣Pis␣-␣AR␣infinito")

## Add here complementary tests (use with caution!)
##-----------------------------------------------------------
cat("\nNormality␣Tests\n")
cat("\n--------------------\n")

##Shapiro-Wilks Normality test
print(shapiro.test(resid(model)))

suppressMessages(require(nortest,quietly=TRUE,warn.conflicts=FALSE))
##Anderson-Darling test
print(ad.test(resid(model)))

suppressMessages(require(tseries,quietly=TRUE,warn.conflicts=FALSE))
##Jarque-Bera test
print(jarque.bera.test(resid(model)))

cat("\nHomoscedasticity␣Test\n")
cat("\n--------------------\n")
suppressMessages(require(lmtest,quietly=TRUE,warn.conflicts=FALSE))
##Breusch-Pagan test
obs=get(model$series)
print(bptest(resid(model)~I(obs-resid(model))))

cat("\nIndependence␣Tests\n")
cat("\n--------------------\n")

##Durbin-Watson test
print(dwtest(resid(model)~I(1:length(resid(model)))))

##Ljung-Box test
cat("\nLjung-Box␣test\n")
print(t(apply(matrix(c(1:4,(1:4)*s)),1,function(el) {
  te=Box.test(resid(model),type="Ljung-Box",lag=el)
  c(lag=(te$parameter),statistic=te$statistic[[1]],p.value=te$p.value)})))
##*************End of complementary tests*****************************************

#Sample ACF vs. Teoric ACF
par(mfrow=c(2,2),mar=c(3,3,3,3))
acf(dades, ylim=c(-1,1) ,lag.max=36,main="Sample␣ACF")

plot(ARMAacf(model$model$phi,model$model$theta,lag.max=36),ylim=c(-1,1),
     type="h",xlab="Lag",  ylab="", main="ACF␣Teoric")
abline(h=0)

#Sample PACF vs. Teoric PACF
pacf(dades, ylim=c(-1,1) ,lag.max=36,main="Sample␣PACF")
```

```
  plot(ARMAacf(model$model$phi,model$model$theta,lag.max=36, pacf=T),ylim=c(-1,1),
       type="h", xlab="Lag", ylab="", main="PACF␣Teoric")
  abline(h=0)
  par(mfrow=c(1,1))
}
################## ('Validation') ################################

## ------------------------------------------------------------------------------
df<-read.csv("final_data/df173_or.csv")
(serie<-ts(df$y,start=2012,freq=12))
length(serie)
summary(serie)
start(serie)
end(serie)
frequency(serie)

## ------------------------------------------------------------------------------
plot(serie, main="Sales␣of␣product␣173",ylab="Sold␣units",xlab="Year")
abline(v=2012:2019,lty=3,col=4)

## ------------------------------------------------------------------------------
## Mean - variance plot
m=apply(matrix(serie,nr=12),2,mean)
v=apply(matrix(serie,nr=12),2,var)
plot(m,v,xlab="Annual␣means",ylab="Annual␣variances",main="lnserie␣of␣product␣173")

## ------------------------------------------------------------------------------
boxplot(serie~floor(time(serie)),main="Yearly␣boxplot")
lnserie<-log(serie)

## ------------------------------------------------------------------------------
##DescomposItion of time series
plot(decompose(lnserie))

## ------------------------------------------------------------------------------
monthplot(lnserie)

## ------------------------------------------------------------------------------
d12lnserie=diff(lnserie,12)
plot(d12lnserie,ylab="d12lnserie")
abline(h=0)
abline(h=mean(d12lnserie), col=2)

## ------------------------------------------------------------------------------
d1d12lnserie=diff(d12lnserie)
d1d1d12lnserie=diff(d1d12lnserie)
plot(d1d12lnserie,ylab="d1d12lnserie")
abline(h=mean(d1d12lnserie), col=2)

## ------------------------------------------------------------------------------
var_table<-matrix(c(var(lnserie),var(d12lnserie),var(d1d12lnserie),var(d1d1d12lnserie)),nrow=4)
colnames(var_table)<-c("Variance")
rownames(var_table)<-c("lnserie","d12lnserie","d1d12lnserie","d1d1d12lnserie")
var_table
mean_table<-matrix(c(var(lnserie),mean(d12lnserie),mean(d1d12lnserie), mean(d1d1d12lnserie)),nrow=4)
colnames(mean_table)<-c("Means")
rownames(mean_table)<-c("lnserie","d12lnserie","d1d12lnserie","d1d1d12lnserie")
mean_table

## ------------------------------------------------------------------------------
par(mfrow=c(1,2))
```

```
acf(d12lnserie,ylim=c(-1,1),col=c(2,rep(1,11)),lwd=2,lag.max=72)
pacf(d12lnserie,ylim=c(-1,1),col=c(rep(1,11),2),lwd=2,lag.max=72)

## ----------------------------------------------------------------------------
#1 MA(1)SARMA(1)
(mod1=arima(d12lnserie,order=c(0,0,1),seasonal=list(order=c(1,0,1),period=12),include.mean = TRUE))

## ----------------------------------------------------------------------------
#2 ARMA(1,1)SARMA(1.1)
(mod2=arima(d12lnserie,order=c(1,0,1),seasonal=list(order=c(1,0,1),period=12),include.mean = TRUE))

## ----------------------------------------------------------------------------
#3 MA(1)SAR(1)
(mod3=arima(d12lnserie,order=c(0,0,1),seasonal=list(order=c(1,0,0),period=12),include.mean = TRUE))

## ----------------------------------------------------------------------------
#*4 ARMA(1,1)SAR(1)
(mod4=arima(d12lnserie,order=c(1,0,1),seasonal=list(order=c(1,0,0),period=12),include.mean = TRUE))

## ----------------------------------------------------------------------------
#*5 AR(1)SAR(1)
(mod5=arima(d12lnserie,order=c(1,0,0),seasonal=list(order=c(1,0,0),period=12),include.mean = TRUE))

## ----------------------------------------------------------------------------
#*6 AR(1)SARMA(1)
(mod6=arima(d12lnserie,order=c(1,0,0),seasonal=list(order=c(1,0,1),period=12),include.mean = TRUE))

## ----------------------------------------------------------------------------
#Adequacy measures
adeq_table<-matrix(c(AIC(mod1),AIC(mod2),AIC(mod3),AIC(mod4),AIC(mod5),AIC(mod6),
BIC(mod1),BIC(mod2),BIC(mod3),BIC(mod4),BIC(mod5),BIC(mod6),mod1$sigma2,mod2$sigma2,
mod3$sigma2,mod4$sigma2,mod5$sigma2,mod6$sigma2),nrow=6)
rownames(adeq_table)<-c("mod1:MA(1)ARMA(1,1)","mod2:ARMA(1,1)ARMA(1,1)",
"mod3:MA(1)AR(1)","mod4:ARMA(1,1)AR(1)","mod5:AR(1)AR(1)","mod6:AR(1)ARMA(1,1)")
colnames(adeq_table)<-c("AIC","BIC","sigma2")
adeq_table
xtable(adeq_table)


## ----------------------------------------------------------------------------
#Validation model 1
validation(mod1,d12lnserie)
#Validation model 2
validation(mod2,d12lnserie)
#Validation model 3
validation(mod3,d12lnserie)
#Validation model 4
validation(mod4,d12lnserie)
#Validation model 5
validation(mod5,d12lnserie)
#Validation model 6
validation(mod6,d12lnserie)


## ----------------------------------------------------------------------------
ar1<-c(1,0,0)
ma1<-c(0,0,1)
arma<-c(1,0,1)
ar3<-c(3,0,0)
ma4<-c(0,0,4)
```

```
## ------------------------------------------------------------------------------
ultim=c(2018,12)                        #Dic 2018

lnserie1=window(lnserie,end=ultim+c(1,0))
lnserie2=window(lnserie,end=ultim)
serie1=window(lnserie,end=ultim+c(1,0))
serie2=window(lnserie,end=ultim)
d12serie1=diff(serie1,12)
d12serie2=diff(serie2,12)


## ------------------------------------------------------------------------------
#Model 1
#Fit the model to the complete series: Model ARIMA(0,0,1)(1,0,1)12
(mod1.p=arima(d12serie1,order=ma1,seasonal=list(order=arma,period=12))) #LA
(mod2.p=arima(d12serie2,order=ma1,seasonal=list(order=arma,period=12))) #LA

## ------------------------------------------------------------------------------
pred=predict(mod2.p,n.ahead=12)
pr1<-window(diffinv(pred$pred,12,xi=window(lnserie,start=ultim+c(-1,1),end=ultim+c(0,0))),
start=ultim)
model<-mod2.p$model
varZ<-mod2.p$sigma
ma<-ARMAtoMA(ar=mod2.p$phi,ma=mod2.p$theta,lag.max=11)
se1<-c(0,sqrt((cumsum(c(1,ma))^2)*varZ))
tl1<-ts(exp(pr1-1.96*se1),start=ultim,freq=12)
tu1<-ts(exp(pr1+1.96*se1),start=ultim,freq=12)
pr1<-ts(exp(pr1),start=ultim,freq=12)   #predictions in original scale

## ------------------------------------------------------------------------------
ts.plot(serie,tl1,tu1,pr1,lty=c(1,2,2,1),col=c(1,4,4,2),xlim=ultim[1]+c(-3,+2),type="o",
main="Model␣ARIMA(0,0,1)(1,0,1)_{12}")
abline(v=(ultim[1]-3):(ultim[1]+2),lty=3,col=4)

## ------------------------------------------------------------------------------
##### Also, compute and report predictive ability measures: RMSPE and MAPE
obs=window(serie,start=ultim)
RMSPE1I=sqrt(sum(((obs-pr1)/obs)^2)/12) #RMSPE: root mean squared prediction error
cat("RMSPE=",RMSPE1I,"\n")
MAPE1I=sum(abs(obs-pr1)/obs)/12         #MAPE: mean absolute prediction error
cat("MAPE=",MAPE1I,"\n")
##### Tabulize values of: point and interval predictions, observations and prediction-errors.
(previs1I=window(cbind(tl1,pr1,tu1,serie,error=round(serie-pr1,3)),start=ultim))

## ------------------------------------------------------------------------------
#Model 2: d12lnserie  mod2:ARMA(1,1)ARMA(1,1)
(mod1.p=arima(d12serie1,order=arma,seasonal=list(order=arma,period=12)))
(mod2.p=arima(d12serie2,order=arma,seasonal=list(order=arma,period=12)))
pred=predict(mod2.p,n.ahead=12)
pr2<-window(diffinv(pred$pred,12,xi=window(lnserie,start=ultim+c(-1,1),end=ultim+c(0,0))),start=ultim)
model<-mod2.p$model
varZ<-mod2.p$sigma
ma<-ARMAtoMA(ar=mod2.p$phi,ma=mod2.p$theta,lag.max=11)
se1<-c(0,sqrt((cumsum(c(1,ma))^2)*varZ))
tl2<-ts(exp(pr2-1.96*se1),start=ultim,freq=12)
tu2<-ts(exp(pr2+1.96*se1),start=ultim,freq=12)
pr2<-ts(exp(pr2),start=ultim,freq=12)   #predictions in original scale

## ------------------------------------------------------------------------------
ts.plot(serie,tl2,tu2,pr2,lty=c(1,2,2,1),col=c(1,4,4,2),xlim=ultim[1]+c(-3,+2),type="o"
,main="Model␣ARIMA(1,0,1)(1,0,1)_{12}")
abline(v=(ultim[1]-3):(ultim[1]+2),lty=3,col=4)
```

```
## ------------------------------------------------------------------------------
##### Tabulize values of: point and interval predictions, observations and prediction-errors.
(previs2I=window(cbind(tl2,pr2,tu2,serie,error=round(serie-pr2,3)),start=ultim))

## ------------------------------------------------------------------------------
##### Also, compute and report predictive ability measures: RMSPE and MAPE
obs=window(serie,start=ultim)
RMSPE2I=sqrt(sum(((obs-pr2)/obs)^2)/12)
#RMSPE1: root mean squared prediction error; for Mod1 # Error = obs - pred
cat("RMSPEo=",RMSPE2I,"\n")
MAPE2I=sum(abs(obs-pr2)/obs)/12       #MAPE: mean absolute prediction error
cat("MAPEo=",MAPE2I,"\n")

## ------------------------------------------------------------------------------
##### Model 3: d12lnserie  MA(1)AR(1)######
(mod1.p=arima(d12serie1,order=ma1,seasonal=list(order=ar1,period=12)))
(mod2.p=arima(d12serie2,order=ma1,seasonal=list(order=ar1,period=12)))
pred=predict(mod2.p,n.ahead=12)
pr3<-window(diffinv(pred$pred,12,xi=window(lnserie,start=ultim+c(-1,1),
end=ultim+c(0,0))),start=ultim)
model<-mod2.p$model
varZ<-mod2.p$sigma
ma<-ARMAtoMA(ar=mod2.p$phi,ma=mod2.p$theta,lag.max=11)
se1<-c(0,sqrt((cumsum(c(1,ma))^2)*varZ))
tl3<-ts(exp(pr3-1.96*se1),start=ultim,freq=12)
tu3<-ts(exp(pr3+1.96*se1),start=ultim,freq=12)
pr3<-ts(exp(pr3),start=ultim,freq=12)   #predictions in original scale

## ------------------------------------------------------------------------------
ts.plot(serie,tl3,tu3,pr3,lty=c(1,2,2,1),col=c(1,4,4,2),xlim=ultim[1]+c(-3,+2),type="o"
,main="Model␣ARIMA(1,1,1)(1,1,1)_{12}")
abline(v=(ultim[1]-3):(ultim[1]+2),lty=3,col=4)

## ------------------------------------------------------------------------------
##### Tabulize values of: point and interval predictions, observations and prediction-errors.
(previs3I=window(cbind(tl3,pr3,tu3,serie,error=round(serie-pr3,3)),start=ultim))


## ------------------------------------------------------------------------------
##### Also, compute and report predictive ability measures: RMSPE and MAPE
obs=window(serie,start=ultim)
RMSPE3I=sqrt(sum(((obs-pr3)/obs)^2)/12)
#RMSPE1: root mean squared prediction error; for Mod1 # Error = obs - pred
cat("RMSPEo=",RMSPE3I,"\n")
MAPE3I=sum(abs(obs-pr3)/obs)/12       #MAPE: mean absolute prediction error
cat("MAPEo=",MAPE3I,"\n")


## ------------------------------------------------------------------------------
##### Model 4:  ARMA(1,1)AR(1)#######
(mod1.p=arima(d12serie1,order=arma,seasonal=list(order=ar1,period=12)))
(mod2.p=arima(d12serie2,order=arma,seasonal=list(order=ar1,period=12)))
pred=predict(mod2.p,n.ahead=12)
pr2<-window(diffinv(pred$pred,12,xi=window(lnserie,start=ultim+c(-1,1),end=ultim+c(0,0))),
start=ultim)
model<-mod2.p$model
varZ<-mod2.p$sigma
ma<-ARMAtoMA(ar=mod2.p$phi,ma=mod2.p$theta,lag.max=11)
se1<-c(0,sqrt((cumsum(c(1,ma))^2)*varZ))
tlI<-ts(exp(pr2-1.96*se1),start=ultim,freq=12)
tuI<-ts(exp(pr2+1.96*se1),start=ultim,freq=12)
```

```
prI<-ts(exp(pr2),start=ultim,freq=12)   #predictions in original scale

## ------------------------------------------------------------------------------
ts.plot(serie,tlI,tuI,prI,lty=c(1,2,2,1),col=c(1,4,4,2),xlim=ultim[1]+c(-3,+2),type="o",
main="Model ARIMA(1,1,1)(1,1,1)_{12}")
abline(v=(ultim[1]-3):(ultim[1]+2),lty=3,col=4)

## ------------------------------------------------------------------------------
##### Tabulize values of: point and interval predictions, observations and prediction-errors.
(previs4I=window(cbind(tlI,prI,tuI,serie,error=round(serie-prI,3)),start=ultim))

## ------------------------------------------------------------------------------
##### Also, compute and report predictive ability measures: RMSPE and MAPE
obs=window(serie,start=ultim)
RMSPE4I=sqrt(sum(((obs-prI)/obs)^2)/12)
#RMSPE1: root mean squared prediction error; for Mod1 # Error = obs - pred
cat("RMSPEo=",RMSPE4I,"\n")
MAPE4I=sum(abs(obs-prI)/obs)/12       #MAPE: mean absolute prediction error
cat("MAPEo=",MAPE4I,"\n")

## ------------------------------------------------------------------------------
#### Model 5: d12lnserie  ARMA(1,1)AR(1)
(mod1.p=arima(d12serie1,order=ar1,seasonal=list(order=ar1,period=12)))
(mod2.p=arima(d12serie2,order=ar1,seasonal=list(order=ar1,period=12)))

pred=predict(mod2.p,n.ahead=12)
pr2<-window(diffinv(pred$pred,12,xi=window(lnserie,start=ultim+c(-1,1),end=ultim+c(0,0))),start=ultim)
model<-mod2.p$model
varZ<-mod2.p$sigma
ma<-ARMAtoMA(ar=mod2.p$phi,ma=mod2.p$theta,lag.max=11)
se1<-c(0,sqrt((cumsum(c(1,ma))^2)*varZ))

tlI<-ts(exp(pr2-1.96*se1),start=ultim,freq=12)
tuI<-ts(exp(pr2+1.96*se1),start=ultim,freq=12)
prI<-ts(exp(pr2),start=ultim,freq=12)   #predictions in original scale

## ------------------------------------------------------------------------------
ts.plot(serie,tlI,tuI,prI,lty=c(1,2,2,1),col=c(1,4,4,2),xlim=ultim[1]+c(-3,+2),type="o",
main="Model ARIMA(1,1,1)(1,1,1)_{12}")
abline(v=(ultim[1]-3):(ultim[1]+2),lty=3,col=4)

## ------------------------------------------------------------------------------
##### Tabulize values of: point and interval predictions, observations and prediction-errors.
(previs5I=window(cbind(tlI,prI,tuI,serie,error=round(serie-prI,3)),start=ultim))

## ------------------------------------------------------------------------------
##### Also, compute and report predictive ability measures: RMSPE and MAPE

obs=window(serie,start=ultim)
RMSPE5I=sqrt(sum(((obs-prI)/obs)^2)/12)
#RMSPE1: root mean squared prediction error; for Mod1 # Error = obs - pred
cat("RMSPEo=",RMSPE5I,"\n")
MAPE5I=sum(abs(obs-prI)/obs)/12       #MAPE: mean absolute prediction error
cat("MAPEo=",MAPE5I,"\n")

## ------------------------------------------------------------------------------
#### Model 6: d12lnserie  AR(1)ARMA(1,1)
(mod1.p=arima(d12serie1,order=ar1,seasonal=list(order=arma,period=12)))
(mod2.p=arima(d12serie2,order=ar1,seasonal=list(order=arma,period=12)))

pred=predict(mod2.p,n.ahead=12)
pr2<-window(diffinv(pred$pred,12,xi=window(lnserie,start=ultim+c(-1,1),end=ultim+c(0,0))),start=ultim)
```

```
model<-mod2.p$model
varZ<-mod2.p$sigma
ma<-ARMAtoMA(ar=mod2.p$phi,ma=mod2.p$theta,lag.max=11)
se1<-c(0,sqrt((cumsum(c(1,ma))^2)*varZ))

tlI6<-ts(exp(pr2-1.96*se1),start=ultim,freq=12)
tuI6<-ts(exp(pr2+1.96*se1),start=ultim,freq=12)
prI6<-ts(exp(pr2),start=ultim,freq=12)   #predictions in original scale
## ------------------------------------------------------------------------------------
ts.plot(serie,tlI6,tuI6,prI6,lty=c(1,2,2,1),col=c(1,4,4,2),xlim=ultim[1]+c(-3,+2),
type="o",main="Model␣ARIMA(1,0,0)(1,0,1)_{12}")
abline(v=(ultim[1]-3):(ultim[1]+2),lty=3,col=4)
## ------------------------------------------------------------------------------------
##### Tabulize values of: point and interval predictions, observations and prediction-errors.
(previs6I=window(cbind(tlI6,prI6,tuI6,serie,error=round(serie-prI6,3)),start=ultim))

## ------------------------------------------------------------------------------------
##### Also, compute and report predictive ability measures: RMSPE and MAPE
obs=window(serie,start=ultim)
RMSPE6I=sqrt(sum(((obs-prI6)/obs)^2)/12)
cat("RMSPEo=",RMSPE6I,"\n")
MAPE6I=sum(abs(obs-prI6)/obs)/12
cat("MAPEo=",MAPE6I,"\n")


## ------------------------------------------------------------------------------------
options(width = 200)
par=c(length(coef(mod1)),length(coef(mod2)),length(coef(mod3)),length(coef(mod4)),
length(coef(mod5)),length(coef(mod6)))
Sigma2Z=c(mod1$sigma2,mod2$sigma2,mod3$sigma2,mod4$sigma2,mod5$sigma2,mod6$sigma2)
AIC=c(AIC(mod1),AIC(mod2),AIC(mod3),AIC(mod4),AIC(mod5),AIC(mod6))
BIC=c(BIC(mod1),BIC(mod2),BIC(mod3),BIC(mod4),BIC(mod5),BIC(mod6))
RMSPE=c(RMSPE1I,RMSPE2I,RMSPE3I,RMSPE4I,RMSPE5I,RMSPE6I)
MAPE=c(MAPE1I,MAPE2I,MAPE3I,MAPE4I,MAPE5I,MAPE6I)
meanLength=c(sum(previs1I[,3]-previs1I[,1]),sum(previs2I[,3]-previs2I[,1]),
sum(previs3I[,3]-previs3I[,1]),sum(previs4I[,3]-previs4I[,1]),
sum(previs5I[,3]-previs5I[,1]),sum(previs6I[,3]-previs6I[,1]))/12
result=data.frame(par,Sigma2Z,AIC,BIC,RMSPE,MAPE,meanLength)
row.names(result)=rownames(adeq_table)
result

## ------------------------------------------------------------------------------------
xtable(result)

## ------------------------------------------------------------------------------------
source("CalendarEffects.r")
data=c(start(d12lnserie)[1],start(d12lnserie)[2], length(d12lnserie))
data.complete=c(start(lnserie)[1],start(lnserie)[2], length(lnserie))
wTradDays=Wtrad(data)
wEast=Weaster(data)
wTradDays1=tail(wTradDays,-1)
wEast1=tail(wEast,-1)
wTradDays2=Wtrad(data.complete)
wEast2=Weaster(data.complete)

## ------------------------------------------------------------------------------------
(mod1TD=arima(d12lnserie,order=ma1,seasonal=list(order=arma,period=12),xreg=wTradDays))
(mod1Ea=arima(d12lnserie,order=ma1,seasonal=list(order=arma,period=12),xreg=wEast))
(mo1CE=arima(d12lnserie,order=ma1,seasonal=list(order=arma,period=12),
xreg=data.frame(wTradDays,wEast)))

## ------------------------------------------------------------------------------------
```

```
(mod2TD=arima(d12lnserie,order=arma,seasonal=list(order=arma,period=12),xreg=wTradDays))
(mod2Ea=arima(d12lnserie,order=arma,seasonal=list(order=arma,period=12),xreg=wEast))
(mo2CE=arima(d12lnserie,order=arma,seasonal=list(order=arma,period=12),
xreg=data.frame(wTradDays,wEast)))

## ---------------------------------------------------------------------------------------
(mod6TD=arima(d12lnserie,order=ar1,seasonal=list(order=arma,period=12),xreg=wTradDays))
(mod6Ea=arima(d12lnserie,order=ar1,seasonal=list(order=arma,period=12),xreg=wEast))
(mod6CE=arima(d12lnserie,order=ar1,seasonal=list(order=arma,period=12),
xreg=data.frame(wTradDays,wEast)))

## ---------------------------------------------------------------------------------------
########## Atipics (Outliers) ################################################
source("atipics2.R")


## ---------------------------------------------------------------------------------------
mod.atip1=outdetec(mod1,dif=c(1,12),crit=3,LS=FALSE)
# automatic detection of outliers with crit=2.8 and LS =FALSE
mod.atip1$sigma2
mod1$sigma2


## ---------------------------------------------------------------------------------------
atipics=mod.atip1$atip[order(mod.atip1$atip[,1]),] #order outliers by date of ocurrence
meses=c("Ene","Feb","Mar","Abr","May","Jun","Jul","Ago","Sep","Oct","Nov","Dic")
data.frame(atipics,Fecha=paste(meses[(atipics[,1]-1)%%12+1],start(lnserie)[1]+
((atipics[,1]-1)%/%12)),perc.Obs=exp(atipics[,3])*100)



## ---------------------------------------------------------------------------------------

##Compare observed series with the linearized (without outliers) one.
lnserie.lin1=lineal(lnserie,mod.atip1$atip) #returns the linearized series in log-scale
serie.lin1=exp(lnserie.lin1)                #get linearized series in original scale



## ---------------------------------------------------------------------------------------
#Plot together (in original scale) the observed and the linearized series (without outliers)
plot(serie.lin1,col=2, ylim=c(min(serie),max(serie)))
#In red: linearized series (without outliers) but in original scale
lines(serie)           #In black: original series



## ---------------------------------------------------------------------------------------
##Efecto de los at  picos en la serie de logaritmos
plot(lnserie-lnserie.lin1)


## ---------------------------------------------------------------------------------------
mod.atip2=outdetec(mod2,dif=c(1,12),crit=2.8,LS=F)
# automatic detection of outliers with crit=2.8 and LS =FALSE
mod.atip2$sigma2
mod2$sigma2


## ---------------------------------------------------------------------------------------
atipics=mod.atip2$atip[order(mod.atip2$atip[,1]),] #order outliers by date of ocurrence
meses=c("Ene","Feb","Mar","Abr","May","Jun","Jul","Ago","Sep","Oct","Nov","Dic")
```

```
data.frame(atipics,Fecha=paste(meses[(atipics[,1]-1)%%12+1],start(lnserie)[1]+
((atipics[,1]-1)%/%12)),perc.Obs=exp(atipics[,3])*100)



## ---------------------------------------------------------------------------------------
mod.atip6=outdetec(mod6,dif=c(1,12),crit=2.8,LS=F)
# automatic detection of outliers with crit=2.8 and LS =FALSE
mod.atip6$sigma2
mod6$sigma2


## ---------------------------------------------------------------------------------------
atipics=mod.atip6$atip[order(mod.atip6$atip[,1]),] #order outliers by date of ocurrence
meses=c("Ene","Feb","Mar","Abr","May","Jun","Jul","Ago","Sep","Oct","Nov","Dic")
data.frame(atipics,Fecha=paste(meses[(atipics[,1]-1)%%12+1],start(lnserie)[1]+
((atipics[,1]-1)%/%12)),perc.Obs=exp(atipics[,3])*100)

################Identification for the linearized series lnserie.lin#########################

d12lnserie.lin1<-diff(lnserie.lin1,12)
d1d12lnserie.lin1=diff(d12lnserie.lin1)
d1d1d12lnserie.lin1<-diff(d1d12lnserie.lin1,1)
## ---------------------------------------------------------------------------------------
var_table.lin1<-matrix(c(var(lnserie.lin1),var(d12lnserie.lin1),var(d1d12lnserie.lin1),
var(d1d1d12lnserie.lin1)),nrow=4)
colnames(var_table.lin1)<-c("Variance")
rownames(var_table.lin1)<-c("lnserie.lin1","d12lnserie.lin1","d1d12lnserie.lin1",
"d1d1d12lnserie.lin1")
var_table.lin1
mean_table.lin1<-matrix(c(var(lnserie.lin1),mean(d12lnserie.lin1),
mean(d1d12lnserie.lin1), mean(d1d1d12lnserie.lin1)),nrow=4)
colnames(mean_table.lin1)<-c("Means")
rownames(mean_table.lin1)<-c("lnserie.lin1","d12lnserie.lin1","d1d12lnserie.lin1",
"d1d1d12lnserie.lin1")
mean_table

## ---------------------------------------------------------------------------------------
par(mfrow=c(1,2))
acf(d12lnserie.lin1,ylim=c(-1,1),lag.max=72,col=c(2,rep(1,11)),lwd=2)
pacf(d12lnserie.lin1,ylim=c(-1,1),lag.max=72,col=c(rep(1,11),2),lwd=2)


## ---------------------------------------------------------------------------------------
(mod.lin1=arima(d12lnserie.lin1,order=c(1,0,1),seasonal=list(order=c(0,0,1),period=12)))


## ---------------------------------------------------------------------------------------
(mod.lin2=arima(d12lnserie.lin1,order=c(1,0,0),seasonal=list(order=c(0,0,1),period=12)))


## ---------------------------------------------------------------------------------------
(mod.lin3=arima(d12lnserie.lin1,order=c(1,0,0),seasonal=list(order=c(1,0,1),period=12)))


## --------------------Validation-----------------------------------------------------------
validation(mod.lin3,d12lnserie.lin1)
## --------------------Prediction-----------------------------------------------------------
ultim=c(2018,12)

serie.lin_1=window(serie.lin1,end=ultim+c(1,0))
lnserie.lin_1=log(serie.lin_1) #log transformed
```

```
d12lnserie.lin_1=diff(lnserie.lin_1,12)
d1d12lnserie.lin_1=diff(d12lnserie.lin_1,1)
serie.lin_2=window(serie.linear1,end=ultim)
lnserie.lin_2=log(serie.lin_2)
d12lnserie.lin_2=diff(lnserie.lin_2,12)
d1d12lnserie.lin_2=diff(d12lnserie.lin_2,1)
#Fit the model to the complete series: lnserie1
(mod1.p=arima(d12lnserie.lin_1,order=ar1,seasonal=list(order=arma,period=12)))
(mod2.p=arima(d12lnserie.lin_2,order=ar1,seasonal=list(order=arma,period=12)))

pred=predict(mod2.p,n.ahead=12)
pr2<-window(diffinv(pred$pred,12,xi=window(lnserie,start=ultim+c(-1,1),end=ultim+c(0,0))),start=ultim)
model<-mod2.p$model
varZ<-mod2.p$sigma
ma<-ARMAtoMA(ar=mod2.p$phi,ma=mod2.p$theta,lag.max=11)
se1<-c(0,sqrt((cumsum(c(1,ma))^2)*varZ))

tlIlin<-ts(exp(pr2-1.96*se1),start=ultim,freq=12)
tuIlin<-ts(exp(pr2+1.96*se1),start=ultim,freq=12)
prIlin<-ts(exp(pr2),start=ultim,freq=12)   #predictions in original scale
## ---------------------------------------------------------------------------
ts.plot(serie,tlIlin,tuIlin,prIlin,lty=c(1,2,2,1),col=c(1,4,4,2),xlim=ultim[1]+c(-3,+2),
type="o",main="Model␣ARIMA(1,0,0)(1,0,1)_{12}")
abline(v=(ultim[1]-3):(ultim[1]+2),lty=3,col=4)
## ---------------------------------------------------------------------------
(previs6I=window(cbind(tlIlin,prIlin,tuIlin,serie,error=round(serie-prIlin,3)),start=ultim))
## ---------------------------------------------------------------------------
obs=window(serie,start=ultim)
RMSPE6I=sqrt(sum(((obs-prIlin)/obs)^2)/12) #RMSPE: root mean squared prediction error
cat("RMSPEo=",RMSPE6I,"\n")
MAPE6I=sum(abs(obs-prIlin)/obs)/12       #MAPE: mean absolute prediction error
cat("MAPEo=",MAPE6I,"\n")

## ---------------------------------------------------------------------------
sum(previs6I[,3]-previs6I[,1])/12


## ---------Zoom in last years-----------------------------------------------------------------------------
par(mfrow=c(1,2), mar =c(1,2,4,1))
ts.plot(serie,tlI6,tuI6,prI6,lty=c(1,2,2,1),col=c(1,4,4,2),xlim=ultim[1]+c(-3,+2),
type="o",main="Model␣ARIMA(1,0,0)(1,0,1)_{12}")
abline(v=(ultim[1]-3):(ultim[1]+2),lty=3,col=4)
ts.plot(serie,tlIlin,tuIlin,prIlin,lty=c(1,2,2,1),col=c(1,4,4,2),xlim=ultim[1]+c(-3,+2),
type="o",main="Model␣ARIMA(1,0,0)(1,0,1)_{12}+Atip")
abline(v=(ultim[1]-3):(ultim[1]+2),lty=3,col=4)

## ---------------------------------------------------------------------------
serie<-window(serie, 2017, c(2017, 36))
## ---------------------------------------------------------------------------
par(mfrow=c(1,2), mar =c(2,2,4,1))
ts.plot(serie,tlI6,tuI6,prI6,lty=c(1,2,2,1),col=c(1,4,4,2),xlim=ultim[1]+c(-1,+2),
type="o",main="Model␣ARIMA(1,0,0)(1,0,1)_{12}")
abline(v=(ultim[1]-3):(ultim[1]+2),lty=3,col=4)
ts.plot(serie,tlIlin,tuIlin,prIlin,lty=c(1,2,2,1),col=c(1,4,4,2),xlim=ultim[1]+c(-1,+2),
type="o",main="Model␣ARIMA(1,0,0)(1,0,1)_{12}+Atip")
abline(v=(ultim[1]-3):(ultim[1]+2),lty=3,col=4)
```

Estimation of ADU of product 173 using a Holt-Winters

```
library(prophet)
library(ggplot2)
```

```
library(forecast)
library(openxlsx)
library(astsa)
library(xts)
library(tsbox)
library(imputeTS)
library(reshape)
library(goft)
library(lubridate)
library(TSstudio)
library(MASS)
library(tidyverse)
library(ggpubr)

df<-read_csv("final_data/df173_or.csv")
(serie_or<-ts(df$y,start=2012,freq=12))
df["ds"]<-seq(ISOdate(2012,1,1), by = "month", length.out = 96)

df1 <- df[seq(1,84),]
obs<-df[seq(85,96),]
df1$ds<-as.Date(df1$ds)
serie <- xts(df1$y, df1$ds)
serie <-ts_ts(serie)
hw <- HoltWinters(serie,seasonal = "mult")
hw_mult<-hw
plot(hw)

forecast <- predict(hw, n.ahead = 12)
pr<-as.numeric(forecast)
obs_y<-obs$y
mape=sum(abs(obs_y-pr)/obs_y)/12 #computing MAPE
print(mape)

for_values<-data.frame(time=round(time(forecast), 3), value_forecast=as.data.frame(forecast)$fit)
actual_values<-data.frame(time=round(time(serie_or), 3), Actual=c(serie_or))
graphset<-merge(actual_values, for_values, all=TRUE, by='time')
graphset$Fitted<-c(rep(NA, NROW(graphset)-(NROW(for_values))), for_values$value_forecast)
graphset.melt<-melt(graphset[, c('time', 'Actual', 'Fitted')], id='time')

pm<-ggplot(graphset.melt,aes(x=time, y=value,col=variable))+geom_line(size=0.7)+
scale_color_manual(values=c("gray24", "red"))+
theme_classic()+
labs(title="Holt-Winters␣filter␣with␣multiplicative␣seasonality",x="Date", y = "Sales")
pm

df1 <- df[seq(1,84),]
obs<-df[seq(85,96),]
df1$ds<-as.Date(df1$ds)
serie <- xts(df1$y, df1$ds)
serie <-ts_ts(serie)
hw <- HoltWinters(serie,seasonal = "add")
plot(hw)

forecast <- predict(hw, n.ahead = 12)
pr<-as.numeric(forecast)
obs_y<-obs$y
mape=sum(abs(obs_y-pr)/obs_y)/12 #computing MAPE
print(mape)


for_values<-data.frame(time=round(time(forecast), 3), value_forecast=as.data.frame(forecast)$fit)
actual_values<-data.frame(time=round(time(serie_or), 3), Actual=c(serie_or))
```

```
graphset<-merge(actual_values, for_values, by='time', all=TRUE)
graphset$Fitted<-c(rep(NA, NROW(graphset)-(NROW(for_values) )), for_values$value_forecast)
graphset.melt<-melt(graphset[, c('time', 'Actual', 'Fitted')], id='time')

pa<-ggplot(graphset.melt,aes(x=time, y=value,col=variable))+geom_line(size=0.7)+
scale_color_manual(values=c("gray24", "red"))+theme_classic()+
labs(title="Holt-Winters␣filter␣with␣additive␣seasonality",x="Date", y = "Sales")
pa

figure <- ggarrange(pm,pa,
                    ncol = 2, nrow = 1)
figure
```

Estimation of ADU of product 173 using a Prophet

```
library(prophet)
library(ggplot2)
library(forecast)
library(openxlsx)
library(MASS)
library(tidyverse)

df<-read_csv("final_data/df173_or.csv") #creation of a dataframe with the variables
df["ds"]<-seq(ISOdate(2012,1,1), by = "month", length.out = 96)
#creation of a column called "ds" with the dates of the variable sum
names(df)[1]<-"y" #prophet function expects that column names are y and ds.
df1 <- df[seq(1,84),]
obs<-df[seq(85,96),]
#df1 <- outlier_treatment(df1)
model = prophet(interval.width = 0.95)
model=fit.prophet(model,df1)

future<-make_future_dataframe(model,periods=12,freq="month")
forecast <- predict(model, future)
pr<-tail(forecast['yhat'],12)
df<-read_csv("final_data/df173_or.csv") #creation of a dataframe with the variables
df["ds"]<-seq(ISOdate(2012,1,1), by = "month", length.out = 96)
#creation of a column called "ds" with the dates of the variable sum
names(df)[1]<-"y" #prophet function expects that column names are y and ds.
df1 <- df[seq(1,84),]
obs<-df[seq(85,96),]
obs<-obs$y
print(obs)
print(pr)
mape.lin=sum(abs(obs-pr)/obs)/12 #computing the MAPE
print(mape.lin)

prophet_plot_components(model,forecast)

plot(model,forecast) +
  add_changepoints_to_plot(model)+
  xlab("Date") +
  ylab("Sales") +
  theme_bw() +
  ggtitle(paste0("Prophet␣model␣for␣product␣", "173"))
model
```

Appendix E

Code used for the estimation of the Red Zone

The development of the code to estimate the Red Zone Size is done in Pyhton language.

```
# -*- coding: utf-8 -*-
"""
Created on Thu Apr 21 12:39:34 2022

@author: Pc
"""

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import cm
from matplotlib.colors import Normalize
from scipy.interpolate import interpn
from datetime import datetime
from calendar import monthrange
from scipy.stats import cauchy
import time

def density_scatter( x , y, ax = None, sort = True, bins = 20, **kwargs )   :
    """
    Scatter plot colored by 2d histogram
    """
    if ax is None :
        fig , ax = plt.subplots()
    data , x_e, y_e = np.histogram2d( x, y, bins = bins, density = True )
    z = interpn( ( 0.5*(x_e[1:] + x_e[:-1]) , 0.5*(y_e[1:]+y_e[:-1]) ) , data ,
    np.vstack([x,y]).T , method = "splinef2d", bounds_error = False)

    #To be sure to plot all data
    z[np.where(np.isnan(z))] = 0.0

    # Sort the points by density, so that the densest points are plotted last
    if sort :
        idx = z.argsort()
        x, y, z = x[idx], y[idx], z[idx]
```

```
     ax.scatter( x, y, c=z, **kwargs )

     norm = Normalize(vmin = np.min(z), vmax = np.max(z))
     cbar = fig.colorbar(cm.ScalarMappable(norm = norm), ax=ax)
     cbar.ax.set_ylabel('Density')

     return ax
def trunc_cauchy( lim_inf,lim_sup,loc,scale,size):
    u_inf = np.arctan((lim_inf - loc)/scale)/np.pi + 1/2
    u_sup = np.arctan((lim_sup - loc)/scale)/np.pi + 1/2
    uniform = np.random.uniform(u_inf, u_sup, size=size)
    rand_cauchy =  loc + scale * np.tan(np.pi*(uniform - 1/2))
    return rand_cauchy

def montecarlo_normal(mu,sigma,stock_inf,stock_sup,profit,cost,dlt,pred_adu,stock_int):

    start = time.time()
    #convert monthly to daily data for predictions
    daily_data=pd.DataFrame()
    for inidex,row in pred_adu.iterrows():
        daily_data_month=pd.DataFrame()
        date=datetime.strptime(row['date'], '%Y-%m-%d')
        month= date.month
        year= date.year
        days=monthrange(year,month)
        last_day=days[1]
        avg_adu=row['fit']/last_day
        x = datetime(year, month, 1)
        y = datetime(year, month, last_day)
        daily=pd.date_range(x,y,freq='D').strftime('%Y-%m-%d').tolist()
        daily_data_month['date']=daily
        daily_data_month['adu']=avg_adu
        daily_data=pd.concat([daily_data, daily_data_month])
    daily_data=daily_data.reset_index(drop=True)
    adu_pred_list=[]
    sum_adu_list=0
    for i in range(dlt):
        adu_pred=daily_data.loc[i,'adu']
        adu_pred_list.append(adu_pred)
        sum_adu_list=sum_adu_list+adu_pred
    adu_avg=sum_adu_list/dlt
    d=[]
    df_list=[]
    rate=profit/cost
    for stock in range(stock_inf,stock_sup,stock_int):
        for j in range(10000):
            sum_adu=0
            for i in range(dlt):
                res_normal=np.random.normal(mu, sigma, 1)
                adu=adu_pred_list[i]+res_normal
                sum_adu=sum_adu+adu
            unit=1
            y=max(((sum_adu-stock)/unit),0)
            x=max(((stock-sum_adu)/unit),0)
            cost_function=rate*y+x
            d.append(
                    {'stock':stock,
                     'cost_f':float(cost_function)
                            })
        df_it=pd.DataFrame(d)
        df_list.append(df_it)
```

```
    data=pd.concat(df_list)
    end = time.time()
    print("Time:",end - start)

    #plotting results
    x=data["stock"].values
    y=data["cost_f"].values
    density_scatter( x, y, bins = [30,30],s=1.5 )

    return data

def montecarlo_cauchy(loc_cauchy,scale_cauchy,stock_inf,
stock_sup,profit,cost,dlt,pred_adu,stock_int):

    start = time.time()
    #convert monthly to daily data for predictions
    daily_data=pd.DataFrame()
    for inidex,row in pred_adu.iterrows():
        daily_data_month=pd.DataFrame()
        date=datetime.strptime(row['date'], '%Y-%m-%d')
        month= date.month
        year= date.year
        days=monthrange(year,month)
        last_day=days[1]
        avg_adu=row['fit']/last_day
        x = datetime(year, month, 1)
        y = datetime(year, month, last_day)
        daily=pd.date_range(x,y,freq='D').strftime('%Y-%m-%d').tolist()
        daily_data_month['date']=daily
        daily_data_month['adu']=avg_adu
        daily_data=pd.concat([daily_data, daily_data_month])
    daily_data=daily_data.reset_index(drop=True)
    adu_pred_list=[]
    sum_adu_list=0
    for i in range(dlt):
        adu_pred=daily_data.loc[i,'adu']
        adu_pred_list.append(adu_pred)
        sum_adu_list=sum_adu_list+adu_pred
    adu_avg=sum_adu_list/dlt
    print(adu_avg)

    d=[]
    df_list=[]
    rate=profit/cost
    for stock in range(stock_inf,stock_sup,stock_int):
        for j in range(10000):
            sum_adu=0
            for i in range(dlt):
                res_cauchy=trunc_cauchy(loc=loc_cauchy, scale=scale_cauchy,
                lim_inf=-15,lim_sup=25, size=1)
                adu=adu_pred_list[i]+res_cauchy
                sum_adu=sum_adu+adu
            unit=1
            y=max(((sum_adu-stock)/unit),0)
            x=max(((stock-sum_adu)/unit),0)
            cost_function=rate*y+x
            d.append(
                    {'stock':stock,
                     'cost_f':float(cost_function)
                            })
        df_it=pd.DataFrame(d)
        df_list.append(df_it)
```

```
    data=pd.concat(df_list)
    end = time.time()
    print("Time:",end - start)

    #plotting results
    x=data["stock"].values
    y=data["cost_f"].values
    density_scatter( x, y, bins = [30,30],s=1.5 )

    return data


def medians_plot(data,stock_int,stock_inf,stock_sup):
    medians=[]
    for stock in range(stock_inf,stock_sup,stock_int):
        med=data[data["stock"]==stock].median()
        medians.append({
                'stock':stock,
                'med_cost_f':med[1]
                })
    medians_df=pd.DataFrame(medians)
    plt.plot(medians_df["stock"],medians_df["med_cost_f"],color='red')
    min_stock=medians_df[medians_df["med_cost_f"]==medians_df["med_cost_f"].min()]
    print(min_stock)

#Product 79

pred_adu_79=pd.read_csv("C:/Users/Pc/Desktop/TFM/tfm/db/new_products/buffers
/data/prediction79_2022.csv")
cost_79=1.17
profit_79=3.58
dlt_79=37
loc_cauchy_79= -1.68924761
scale_cauchy_79=2.60617607
stock_inf=100
stock_sup=600
stock_int=10
mu_79=-1.56662775
sigma_79=6.23792478


data_cauchy_79=montecarlo_cauchy(loc_cauchy_79,scale_cauchy_79,stock_inf,
stock_sup,profit_79,cost_79,dlt_79,pred_adu_79,stock_int)
medians_plot(data_cauchy_79,stock_int,stock_inf,stock_sup)

data_normal_79=montecarlo_normal(mu_79,sigma_79,stock_inf,stock_sup,
profit_79,cost_79,dlt_79,pred_adu_79,stock_int)
medians_plot(data_normal_79,stock_int,stock_inf,stock_sup)


#Product 76

pred_adu_76=pd.read_csv("C:/Users/Pc/Desktop/TFM/tfm/db/new_products/
buffers/data/prediction76_2022.csv")
cost_76=26
profit_76=110
dlt_76=12
loc_cauchy_76= 0.005694305
scale_cauchy_76=3.199268351
stock_inf=300
stock_sup=600
stock_int=10
```

```
mu_76=0.37881765
sigma_76=5.71465095


data_cauchy_76=montecarlo_cauchy(loc_cauchy_76,scale_cauchy_76,stock_inf,
stock_sup,profit_76,cost_76,dlt_76,pred_adu_76,stock_int)
medians_plot(data_cauchy_76,stock_int,stock_inf,stock_sup)

data_normal_76=montecarlo_normal(mu_76,sigma_76,stock_inf,stock_sup,profit_76,
cost_76,dlt_76,pred_adu_76,stock_int)
medians_plot(data_normal_76,stock_int,stock_inf,stock_sup)

#Product 173

pred_adu_173=pd.read_csv("C:/Users/Pc/Desktop/TFM/tfm/db/
new_products/buffers/data/prediction173_2022.csv")
cost_173=0.75
profit_173=1.25
dlt_173=12
loc_cauchy_173= 1.21807716
scale_cauchy_173=2.87343000
stock_inf=100
stock_sup=600
mu_173=1.72556444
sigma_173=5.12702625


data_cauchy_173=montecarlo_cauchy(loc_cauchy_173,scale_cauchy_173,stock_inf,stock_sup,
profit_173,cost_173,dlt_173,pred_adu_173,stock_int)
medians_plot(data_cauchy_173,stock_int,stock_inf,stock_sup)

data_normal_173=montecarlo_cauchy(mu_173,sigma_173,stock_inf,stock_sup,profit_173,
cost_173,dlt_173,pred_adu_173,stock_int)
medians_plot(data_normal_173,stock_int,stock_inf,stock_sup)
```