Time Series Forecast using the STL model.

Jorge Najera.

Introduction.

Derived from the widespread use of the computer in most areas of work, research, business, etc. As well as the creation of programs and parcels to make efficient and possible tasks that otherwise would be impossible or very laborious, it becomes a reality the possibility of handling large volumes of information for analysis and even the inference about their. From this tendency to make productive tasks more efficient with the use of computational techniques, the use of Machine Learning was born.

On the other hand, the use of statistical models that allow the prediction of variables is not something new; However, they have been enhanced on a large scale thanks to the collection of data, the generation of databases, and the advancement in technologies that make more powerful and robust models proposed.The use of these predictive models have allowed companies to make decisions based primarily on data; the calculation of the demand for goods and services, the forecast of sales and expenses, the relationship between the variables of interest with the macroeconomic environment, etc.

The purpose of the following text is to create a small guide that helps to forecast the time series using the STL model in R. This forecasting method was selected due to the robustness and the scope it has to efficiently forecast a wide range of series; and in turn, it facilitates decision-making based on the fact that it allows a series to be broken down into its components; tendency, seasonality and errors.The first step to start series is to first define what is a time series and what we mean when we talk about a forecast. A time series is a list of dates, each of which is associated with a value (a number). Time series are a structured way of representing data. Visually, it is a curve that evolves over time.

On the other hand, the forecast of the time series means that we extend the historical values to the future, where there are still no measurements available. The forecast is usually made to optimize areas such as inventory levels, production capacity or personnel levels.There are two main structural variables that define a time series forecast:

- The period, which represents the level of aggregation. The most common periods are months, weeks and days.

- The horizon, which represents the number of advance periods that must be predicted.

Data of a series of time can be broken down into individual components to facilitate their study which are explained below: the Trend and the Seasonality. The trend of a series of time is the long-term component that represents the growth or decrease in the series over a broad period. As you can see the trend is the propensity to increase or decrease in the values of the data of a series of time, which remains over a very extended period of time, that is, it will not change in the distant future until they have significant or radical changes in the environment in which it is immersed and that determines the behavior of the series of time under study, changes that could be originated as, for example, by scientific discoveries, technological advances, cultural, geopolitical, demographic, religious changes , etc.

The seasonal component is a pattern of change that repeats itself year after year.The pattern of change is usually an increase or a quantitative decrease in the observed values of a specific time series. It is worth mentioning that although in most cases the seasonal pattern is a phenomenon that occurs in periods of time of approximately one year; This phenomenon can also be manifested in periods of time, whether they are less than or greater than one year.

To best illustrate these phenomena, the wine sales database was selected, which is found as a test database of the forecast package. The series has a monthly periodicity, which facilitates the calculation of both a trend and the seasonal component (which in this case is monthly). First of all, we load the necessary packages to carry out the analysis and the forecast. It is necessary that these and the other used packages are installed before running the code. The code to install and load these packages is:

```
install.packages("dplyr")
install.packages("ggplot2")
install.packages("tidyquant")
install.packages("forecast")
install.packages("fANCOVA")
library(dplyr)
library(ggplot2)
library(tidyquant)
library(forecast)
library(fANCOVA)
```

Once installed and loaded the necessary packages, we proceed to load the series of the forecast package, created the dummy variable of period, which we will occupy later. In turn, as is customary in the analysis of data science, it is necessary to separate a training base and another test base, in order to measure the efficiency of the forecast. The code to perform all these steps is as follows:

```
Sales<-data.frame(Sales=forecast::wineind,
Period=1:nrow(as.matrix(forecast::wineind)))

Train<- Sales %>% dplyr::filter(Period<165)
Test<- Sales %>% dplyr::filter(Period<=176 & Period>=165)
```

An important difference that has the selection of the training set and test with the classification models is that in time series the order of the observations does matter, since it seeks to predict the behavior of a series in a period after the last one that has the Serie. To divide the dataset into training and testing we will use only the last 12 observations of the series as a test, since most of the models used in Time Series usually perform efficient forecasting in short periods of time, that is, not more than 12 observations.

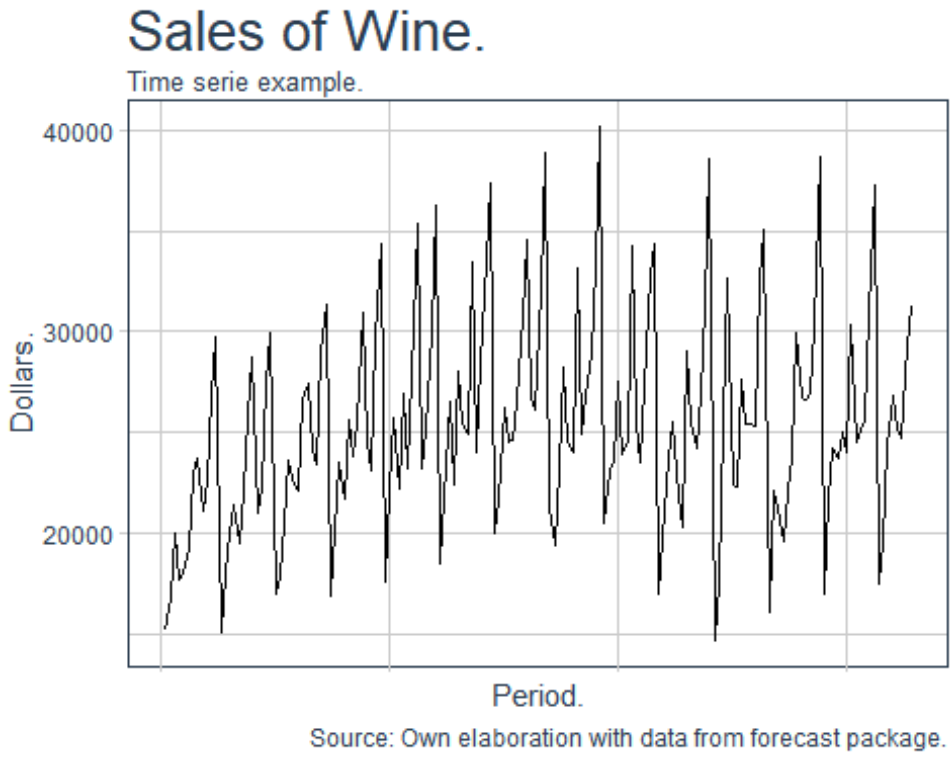
Data Explore.

The exploration of our data is an essential step for any type of analysis that we wish to perform. If we do not know the structure of our data, its properties and peculiarities, then we can find problems to analyze, model and interpret results.No matter how sophisticated a statistical model or machine learning technique is, if we do not know what we are giving it to work, we will hardly know what we will get from them.

As a first approach to the series of interest, we will make a timeless graphic about the wine sales series. In a temporary graph, in the Y axis the variable of interest is selected (in our case, the wine sales series), and in the x-axis, the period variable that we create (which we will occupy as the variable of the weather). To make this graph, we will occupy the ggplot2 package, using the ggplot function, in the following way:

```
Train %>% ggplot2::ggplot(aes(y=Sales,x=Period)) +
  geom_line() +
  labs(title='Sales of Wine.',
       subtitle = "Time serie example.",
       x = "Period.", y = "Dollars."
       ,caption="Source: Own elaboration with data from forecast
package.", color="Serie:")+
  theme_tq()+
  theme(plot.title = element_text(hjust = 0, size = 20),axis.text.x =
element_blank())
```

The graphic result of the code is as follows:

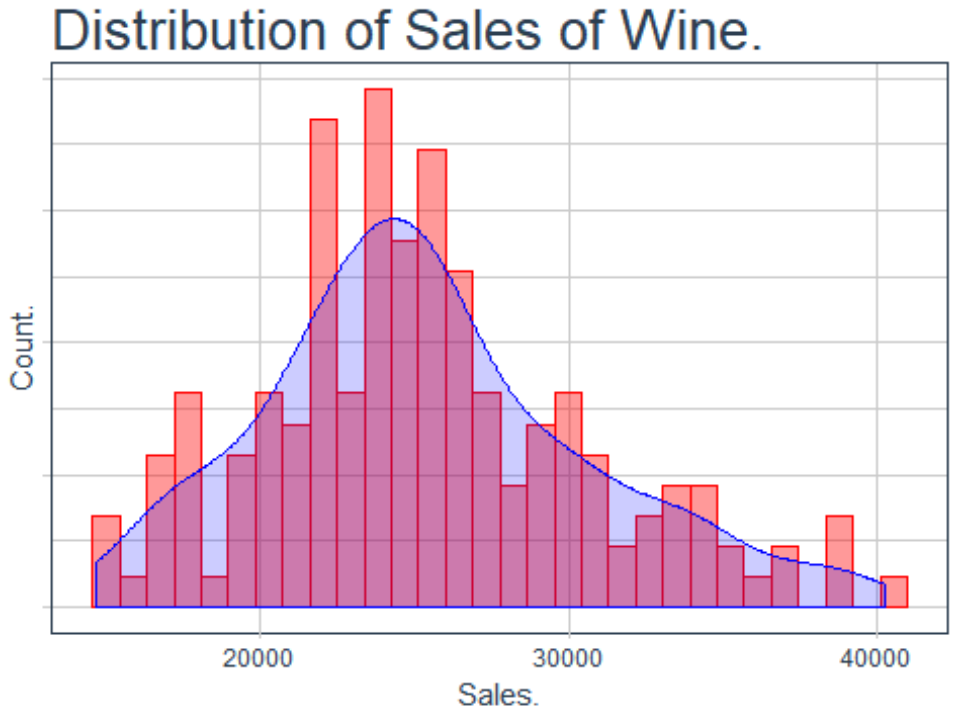


At first glance we can observe several characteristics of the series: first, that it maintains a slightly similar behavior, without radical changes in the behavior of the trend; Secondly, it presents a strong seasonal behavior, which explains these ups and downs, far from the average of the series.

To better visualize both the average of the series in general and its deviations, the analysis of the histogram is usually used. Its function is to graphically display numbers, variables and figures so that the results are displayed more clearly and orderly. For our case in general, help us see how far is the average of the complete series with respect to other values. In other words, the possibility that the average can be an efficient indicator of the phenomenon in question.The code to generate the histogram of the series is:

```
Train %>% dplyr::select(Sales) %>%
 ggplot(aes(Sales)) +
  geom_histogram(aes(y =..density..),
                 col="red",
                 fill="red",
                 alpha = .4) +
  geom_density(color="blue", fill="blue", alpha = .2) +
  labs(title='Distribution of Sales of Wine.', x = "Sales.", y = "Count."
       ,caption="Source: Own elaboration with data from forecast
package.")+
```

```
  theme_tq()+
  theme(plot.title = element_text(hjust = 0, size = 20),axis.text.y =
element_blank())
```



Source: Own elaboration with data from forecast package.

When we perform the histogram and the graph of the series of the series, we can see the effects that, in effect, there is a medium that is between 20,000 and 30,000 dollars, with a certain bias towards the left, that is, that the value of wine Sales of wine have more values at 20,000 and 30,000, and less at values higher than those said. Another important characteristic that the series has a bias towards the left is that, therefore, the mean of the series is different from the median, so the series does not present a normal distribution.This is common, since there are effects that allow to determine the behavior of the series, and therefore its value is influenced by other variables that prevent it from being distributed in a normal way.

The exploratory analysis allowed us to determine that the series needs to be modeled using a model that allows to consider the effects of the determinants that prevent the series from being distributed in a normal way. As mentioned at the beginning of the writing, it is proposed to develop an STL model due to the robustness and the effectiveness it has with the handling of time series that have a tendency and a seasonal effect.

STL Model.

As already mentioned at the beginning of the writing, a time series can be divided into 3 components: the trend, the seasonality and the error or residuals of the model.

In such a way that to elaborate a forecast it is necessary to make the estimation of each component, either individually or jointly. On the other hand, there are two different approaches: the classical approach, which considers that each component should be calculated in a deterministic way, and the modern approach, which assumes that they exist but that they are stochastic (in other words, they are not calculable), so that only transformations can be made to eliminate the effect until reaching a series of white noise.

The STL model is a deterministic model that allows the components to be calculated separately using different methods. It estimates the behavior of the trend using a LOESS regression, and in turn, calculates the seasonal component by selecting one of more models, but it is usually selected only between 2: the seasonal ARIMA model, or the ETS model. The main difference that the STL model has with the others is that, when considering the trend as a LOESS estimation, it is extremely flexible to the changes in the trend of the series, unlike the linear regression, which assumes that the series maintains the same constant.

Trend.

As mentioned previously, the way to calculate the trend using the STL model is to calculate it from a LOESS regression. LOESS combines the simplicity of linear least squares regression with the flexibility of non-linear regression by fitting simple models on local subsets of data to create a function that describes the deterministic part of the variation in point-to-point data. In fact, one of the main attractions of this method is that it is not necessary to specify a global function to fit a model to the data. In return, a greater calculation power is necessary.

Because it is so computationally intensive, LOESS would have been practically impossible to use at the time when the least squares regression was developed. Most of the other modern methods for process modeling are similar to those of LOESS in this regard. These methods have been consciously designed to use our current calculation capacity to achieve objectives not easily achieved by traditional methods.

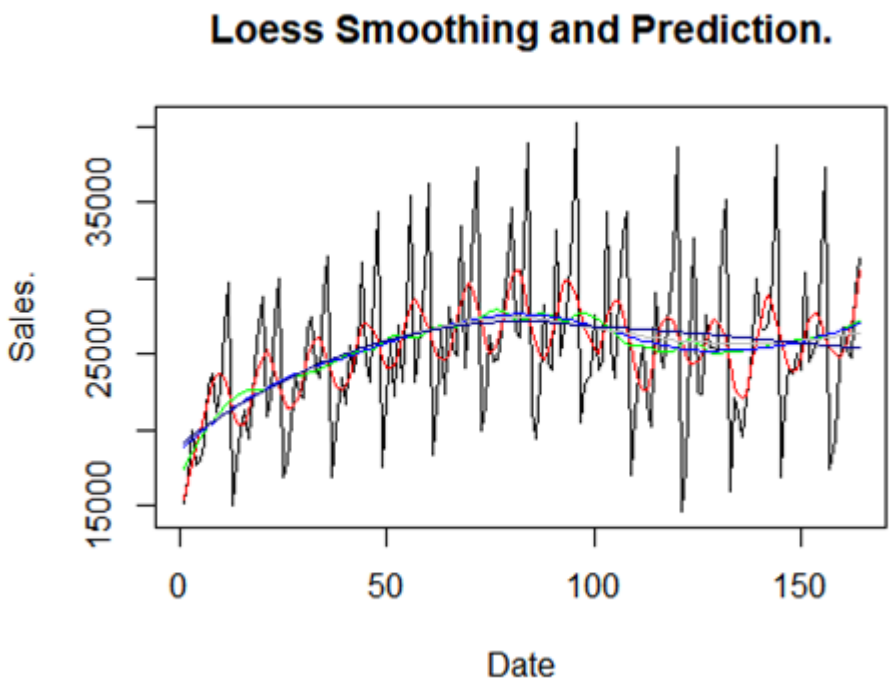
The key parameter for the estimation of the regression LOESS is the span. The span is the degree of smoothing of the series. Higher smoothing values (h) produce softer functions that move less in response to fluctuations in the data. The smaller the h, the closer the adjustment of the regression function to the data will be. Using too small a value of the smoothing parameter is not desirable because the regression function will begin to capture the random error in the data. The useful values of the smoothing parameter are generally in the range of 0.25 to 0.5 for most LOESS applications. As an example to this smoothing difference we will occupy different values of span for the same regression, in order to compare the results, using the following code:

```
#Estimation:
loessMod10 <- loess(Sales ~ Period, data=Train, span=0.10) # 10%
smoothing span
loessMod25 <- loess(Sales ~ Period, data=Train, span=0.25) # 25%
smoothing span
loessMod50 <- loess(Sales ~ Period, data=Train, span=0.50) # 50%
smoothing span
loessMod75 <- loess(Sales ~ Period, data=Train, span=0.75) # 75%
smoothing span
loessMod100 <- loess(Sales ~ Period, data=Train, span=1) # 100% smoothing
span
```

We save the results of the predictions in Data frames that allow us to plot as a comparison each prediction along with the actual training base. It was necessary, to perform the LOESS regression estimation, to select an explanatory variable and an explained variable. As it is a series of time, we use as an explanatory variable the fictitious variable that we create with the name Period, and the variable to explain is the level of wine sales. A brief explanation of why these variables were selected in this order is due to the fact that we seek to find the relationship (or the effect, in this case) that the time has on the wine sales level.

```
#Predictions:
smoothed10 <- predict(loessMod10)
smoothed25 <- predict(loessMod25)
smoothed50 <- predict(loessMod50)
smoothed75 <- predict(loessMod75)
smoothed100 <- predict(loessMod100)

plot(Train$Sales, x=Train$Period, type="l", main="Loess Smoothing and
Prediction.", xlab="Date", ylab="Sales.")
lines(smoothed10, x=Train$Period, col="red")
lines(smoothed25, x=Train$Period, col="green")
lines(smoothed50, x=Train$Period, col="blue")
lines(smoothed75, x=Train$Period, col="grey")
lines(smoothed100, x=Train$Period, col="darkblue")
```

The graph resulting from the previous written code is:



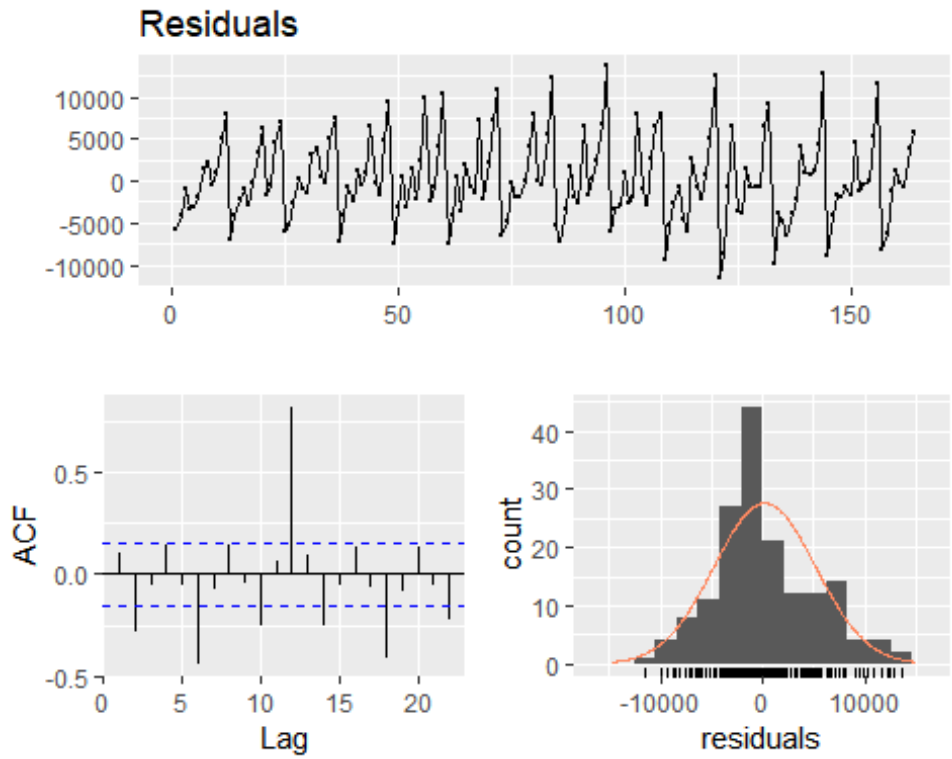
We can observe the comparison between different span values separately. Part of the work of the data scientist is to find the value that helps to maximize the estimation of the different models, and in this way avoid problems of overfitting or underfitting. In such a way that we will seek to minimize the estimation error from different span values for the series. To achieve this result, we will use the loess.as function, from the fANCOVA package.

The loess.as function aims to select the optimal smoothing value from two methods:: bias-corrected Akaike information criterion (aicc); and generalized cross-validation (gcv).The code to calculate the optimal span value is:

```
LoessOptim<-fANCOVA::loess.as(Train$Period, Train$Sales, user.span =
NULL,
                              plot = FALSE)

LoessOptim[["pars"]][["span"]]

## [1] 0.7906048
```

We obtain that the value that minimizes the estimation error of the model is 0.79. On the other hand, using the checkresiduals () function of the forecast package we can carry out a brief analysis of the residuals of the estimate, in order to contrast that the waste is distributed in a normal way, with a constant variance and an average equal to 0.

```
forecast::checkresiduals(LoessOptim$residuals)
```



The analysis of the residuals helps us to contrast interesting and useful results for the general analysis of the series. In the first place, that the series presents a seasonal behavior, in such a way that the waste has fallen and lowered in specific periods of time; This is not surprising, since we are assuming that the series is only composed of the component of the trend.

Secondly, the series presents a different distribution to the normal, since there is an important peak in the Gauss campaign plotted. In such a way that, according to the results, it is necessary to estimate in turn the

Trend + Seasonal.

The STLF function allows the calculation of the seasonal component from selecting a model that meets this specific task. The most common options are usually the method by model ARIMA and mor model ETS. Both models have an important effect that facilitates the calculation of seasonality once the trend is already conceived (which was already calculated from LOESS). To be sure that the appropriate model was selected to model the behavior of the seasonal component, both the Akaike criterion and the RMSE of both models will be compared, and we will select the one that best suits us according to our purposes.

As a first step, it is necessary to define the training series as a time series, with a periodicity of 12 (since we are considering a monthly seasonality that is repeated year after year). For the forecast (12 months) of the series, we will need to select the s.window = 12, because we will look for the behavior of the seasonal component with a periodicity of 12 months. In turn, as the calculation of the optimal value of the span for the estimation of the trend was made, it will be added to the model from the criterion t.window. We will start by making the forecast with the ETS model.

```
Ts<-ts(Train$Sales, freq=12)
ForecastEts<-forecast::stlf(Ts, h = 12, s.window = 12, t.window =
0.7906048,method = c("ets"))
ForecastEts[["model"]][["aic"]]

## [1] 3333.966
```

We can check that the Akaike information selection criterion tells us that the value is 3333. Now, we perform the same process that was done, but changing to an ARIMA model with the following code:

```
ForecastArima<-forecast::stlf(Ts, h = 12, s.window = 12, t.window =
0.7906048,method = c("arima"))
ForecastArima[["model"]][["aic"]]

## [1] 2944.866
```

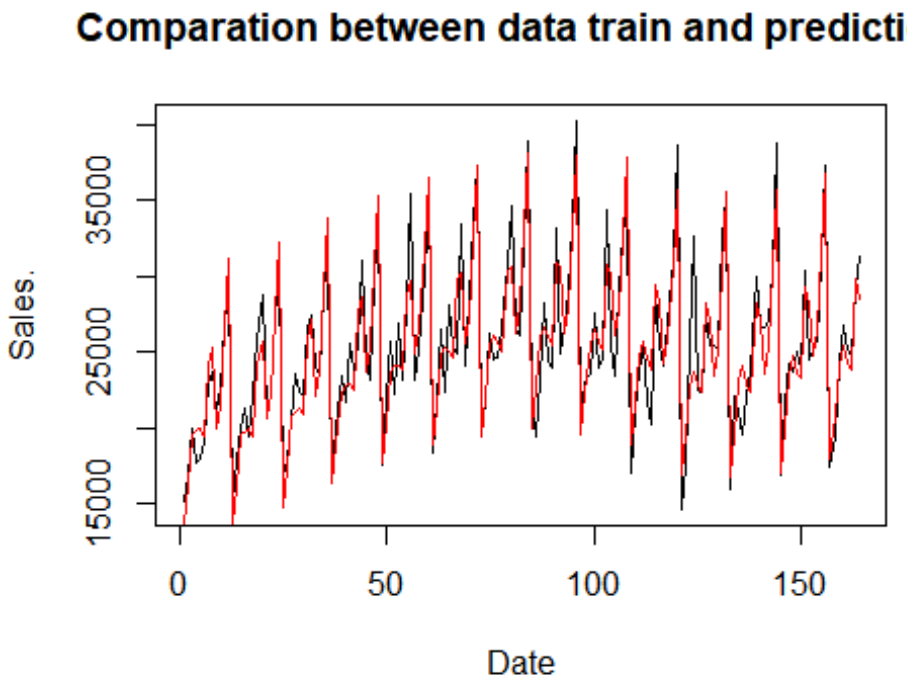
Using the calculation of the selection criteria on the proposed ARIMA model, we contrasted that, according to the selection criteria, the ARIMA model is better to perform the forecast of the series than the model with ETS. Now, we will proceed to contrast with the RMSE and verify if the ARIMA model has a greater predictive power than the ETS model. For this we will use the following code:

```
ForecastArima<-as.numeric(ForecastArima$mean)
TestSales<-as.numeric(Test$Sales)
A<-data.frame(forecast::accuracy(ForecastArima,TestSales))

ForecastEts<-as.numeric(ForecastEts$mean)
B<-data.frame(forecast::accuracy(ForecastEts,TestSales))
Accuracy<- rbind(ARIMA=A,ETS=B)
Accuracy

##               ME     RMSE      MAE        MPE     MAPE
## ARIMA -1156.386 2783.632 2035.860 -6.477117 9.782419
## ETS    -923.317 2633.770 1973.445 -5.430143 9.304786
```

When calculating the Accuracy criteria for both forecasts, using the test database, we contrasted that, unlike the Akaike information criterion, the model with the best results was the model with the seasonal component calculated from the ETS. ., with an RMSE lower than that of the ARIMA model. In this way, we proceed to make the forecast of the series using only the ETS model as a seasonal component.

The comparison between the predictions of the series with the real data of the training set is made using the following code:

```
ForecastEts<-forecast::stlf(Ts, h = 12, s.window = 12, t.window =
0.7906048,method = c("ets"))
plot(Train$Sales, x=Train$Period, type="l", main="Comparation between
data train and prediction.", xlab="Date", ylab="Sales.")
lines(ForecastEts$fitted, x=Train$Period, col="red")
```



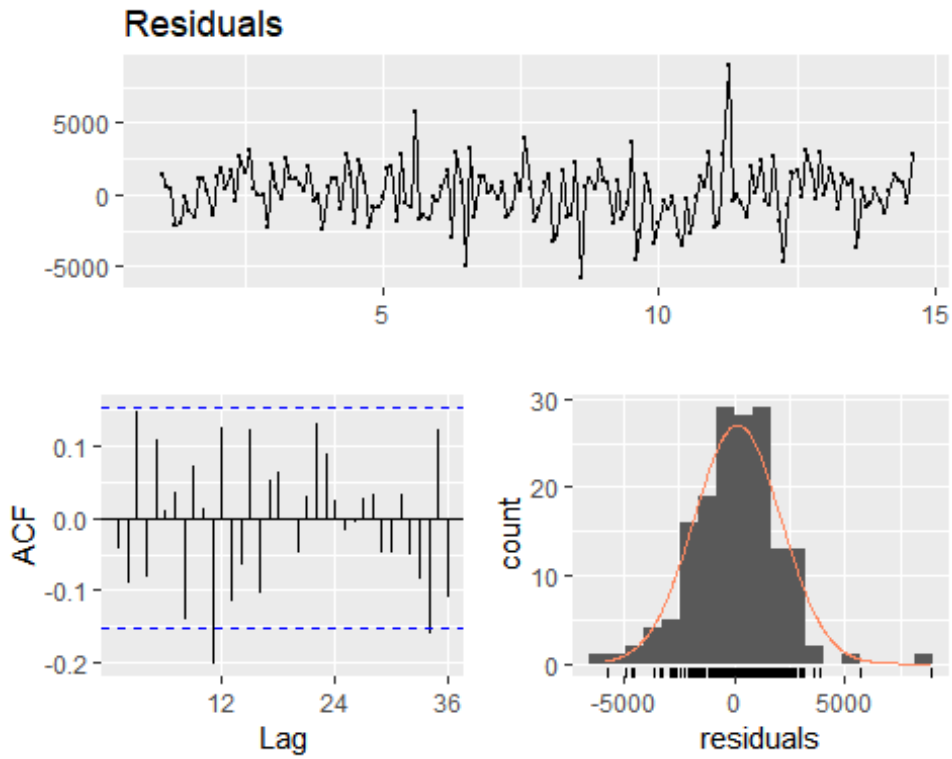
Residual Analysis for the final model.

As mentioned previously, the analysis of waste is of the utmost importance because it allows us to observe what kind of behavior the series needs to model. In the previous analysis, for example, it was discovered that there was a seasonal pattern that the only estimate by LOESS could not capture. On the other hand, using the STLF function, we were able to estimate both the trend and the seasonal component estimate. We then perform again the analysis of the residuals of the new model, in

order to know if the distribution of them follows a normal behavior, with constant and average variance equal to 0.

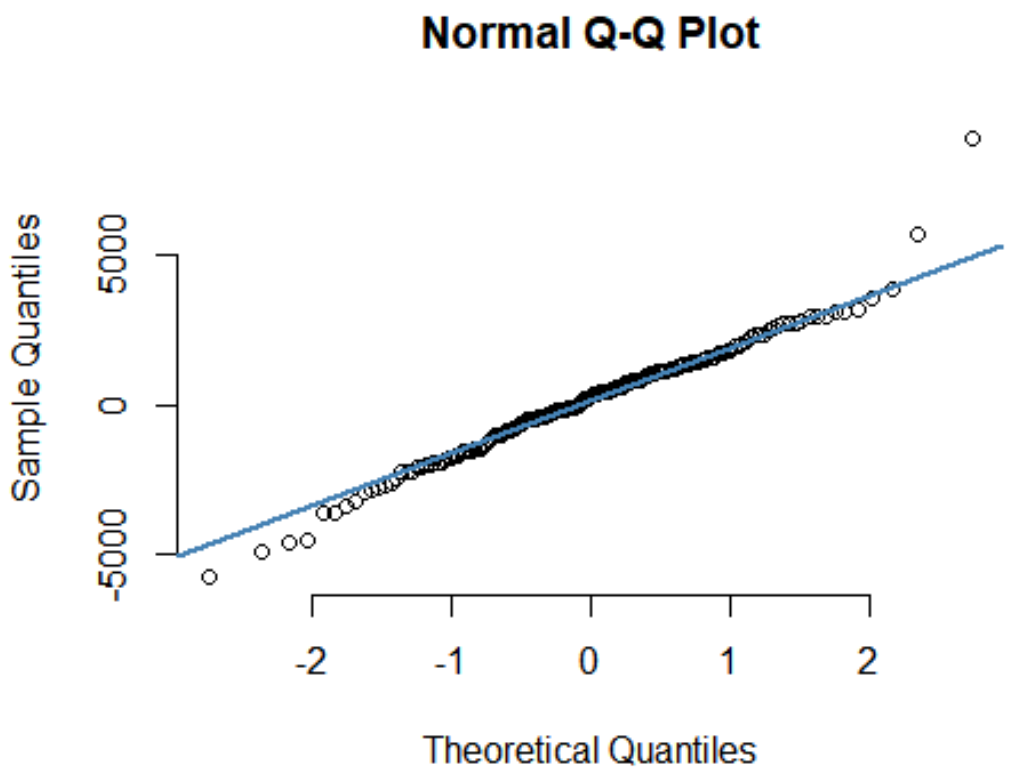
To perform the analysis of the waste we use, in the same way, the next code:

```
forecast::checkresiduals(ForecastEts$residuals)
```



We can now observe that the residuals have a behavior closer to the normal than the previous proposed model, since there are no higher values than the Gaussian distribution in the graph. On the other hand, there are no significant problems of autocorrelation between the waste. As another important analysis, we proceed to calculate the Q-Q curve that helps us compare the normality of waste. A Cuantil-Cuantil chart allows you to see how close the distribution of a data set to some ideal distribution or compare the distribution of two data sets. If it is interesting to compare with the Gaussian distribution, it is called normal probability graph. The data is sorted and graph the i-th data against the corresponding quantile Gaussian. The code to elaborate this graph is the following:

```
qqnorm(ForecastEts$residuals, pch = 1, frame = FALSE);
qqline(ForecastEts$residuals, col = "steelblue", lwd = 2)
```



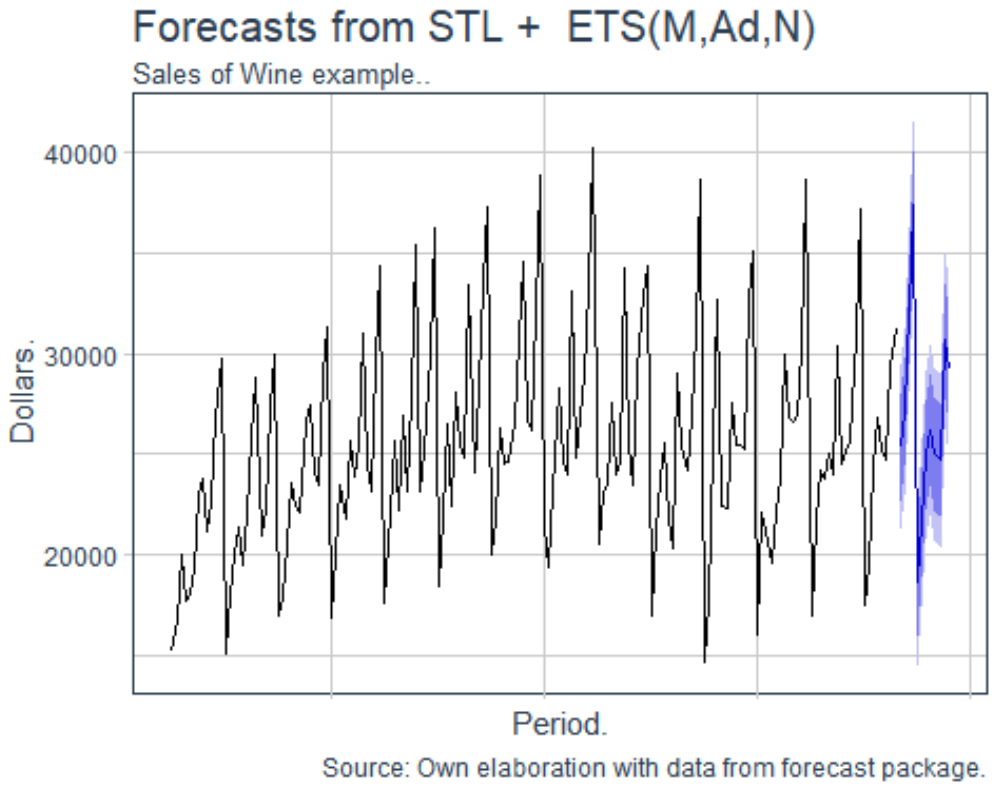
As we can see in the Q-Q plot, there is a normal behavior of the waste series. The objective of the Temporary Series is to decompose the series observed in two parts: one is the dependent part of the past and the other the unpredictable part. The use of the ETS model allowed to capture, in an effective way, the behavior of the component of the trend, in such a way that the only thing that "remains" of the series is white noise, that is, random variations that can not be predicted.

Conclusions.

Once the almost normal behavior of the residuals is contrasted, and obtaining a model with a fairly acceptable RMSE, we can proceed to carry out the forecast of the series and make the comparison with the original test series. First, we can make the model forecast chart using the autoplot function, which is inside the ggplot2 package. Note: it must be remembered that the autoplot function only works with time series, so that when we want to occupy the function the series must be established as a series of time using the ts function.

We make the forecast and graphing it using the following :

```
autoplot(ForecastEts)+
  labs(subtitle = "Sales of Wine example..",
       x = "Period.", y = "Dollars."
       ,caption="Source: Own elaboration with data from forecast
package.", color="Serie:")+
  theme_tq()+
  theme(plot.title = element_text(hjust = 0, size = 15),axis.text.x =
element_blank())
```



The resulting graph allows us to confirm that the behavior of the forecast is quite similar to the behavior of the series in general, this is a good indication since it means that the achievement model captures all the main effects that influence the behavior

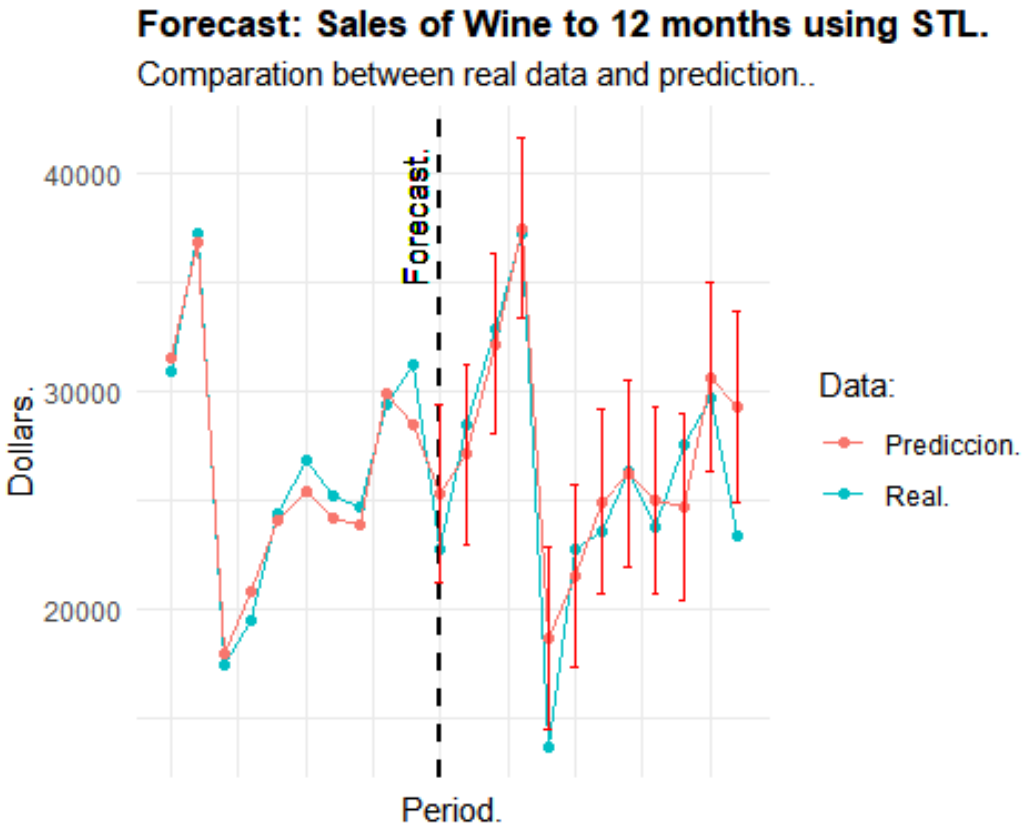
of the series in general. Note: within the predicted graph, we can observe the confidence intervals at a 90 and 95% level of significance, in which are the values within which the real values of the series will fall.

This is very important, since, as we have seen, the forecast will not always be exactly the same value as the real data, so the intervals help to have a greater margin of error that makes it easier for the decision maker to be aware of the possible deviations of a forecast.

In order to better demonstrate this characteristic, we will proceed to make a graph that serves as a comparison between the predicted value, the real value, and the confidence intervals of the same to which the forecast should fall. For that, we will use the ggplot function in the same way, and with the geom_errorbar command we will create the confidence intervals, in order to see how many of the real values of the training set fall within.

```
Intervals<-data.frame(ForecastEts$upper,ForecastEts$lower)


FinalData<-data.frame(Real=Sales$Sales,
Prediction=c(as.numeric(ForecastEts$fitted),as.numeric(ForecastEts$mean))
,Upper=c(rep(NA,nrow(Train)),as.numeric(Intervals$X95.)),
Lower=c(rep(NA,nrow(Train)),as.numeric(Intervals$X95..1)))

FinalData$Date<- 1:nrow(FinalData)


FinalData<-FinalData %>% dplyr::filter(Date>=155)



ggplot(data=FinalData,aes(x = Date, y = Real,color="Real.")) +
  geom_line() +
  geom_point() +
  geom_vline(aes(xintercept =165), lty = 2, size=1)+
  geom_line(aes(x = Date, y = Prediction, color="Prediccion.")) +
  geom_point(aes(x = Date, y =Prediction ,color="Prediccion."))+
  geom_text(aes(x=164,y=38000,label="Forecast."),angle = 90,
colour="black")+
  geom_errorbar(aes(ymin=Lower, ymax=Upper), width=0.3,color="red")+
  labs(title="Forecast: Sales of Wine to 12 months using STL.",
       subtitle="Comparation between real data and prediction..",
x='Period.',y='Dollars.', color='Data:')+
  theme_minimal()+
  theme(plot.title = element_text(size = 13, face = "bold"),axis.text.x =
element_blank())
```

The resulting graph of the code elaborates is the following one:



The resulting graph allows us to visualize several interesting things: first, that the model proposed for forecasting was efficient, since the predicted values are very close to the real test values. Second, that at least 10 of the 12 real values of the series are within the confidence interval of all predicted values, so it represents an important percentage of correct guesses in the forecast.

At the moment of elaborating a model to forecast time series, there is no reason to lose sight of the fact that the longer the period of time there is a greater probability that the forecast will not be efficient, due to the very nature of the series.