Fourth Assignament

Sergi Carol and Balbina Virgili

May 16, 2018

Introduction

The aim of this fourth assignment is to successfully take our first steps towards time series, by developing a small analysis of two different time series techniques from a chosen dataset. This way, we expect to acquire theoretical and practical knowledge of time series. To be able to do this aim, first we have chosen an appropriate dataset, then we have analyzed several concrete properties of the given data and, finally, according to the analyzed properties, specific time series methods have been applied. We have chosen the *R* language to develop the whole code for this fourth assignment, as it provides many useful libraries to perform time series analysis.

This document begins with an small explanation of our chosen dataset, follows with a short introduction of basic properties of time series and, then, it continues with the explanation and our developed implementation of two different algorithms that we have used to predict a new seasonal period of the data. Finally, this document ends with some conclusions of our experience and the results obtained during the development of the assignment.

Dataset

The dataset that we have chosen for this assignment is titled *Air Quality UCI* (Dheeru and Karra Taniskidou 2017). This dataset contains the responses of a gas multisensor device deployed on the field in an Italian city. Hourly responses averages for over a year are recorded along with gas concentrations references from a certified analyzer. More concretely, it has the following attributes for each response recorded:

- *Date*: (DD/MM/YYYY).
- *Time*: (HH.MM.SS).
- *CO(GT)*: true hourly averaged concentration CO.
- *PT08.S1(CO)*: tin oxide hourly averaged sensor response.
- *NMHC(GT)*: true hourly averaged overall Non Metanic HydroCarbons concentration.
- *C6H6(GT)*: true hourly averaged Benzene concentration.
- *PT08.S2(NMHC)*: titania hourly averaged sensor response.
- *NOx(GT)*: true hourly averaged NOx concentration in ppb.
- *PT08.S3(NOx)*: tungsten oxide hourly averaged sensor response.
- *NO2(GT)*: true hourly averaged NO2 concentration.
- *PT08.S4(NO2)*: tungsten oxide hourly averaged sensor response.
- *PT08.S5(O3)*: indium oxide hourly averaged sensor response.
- *T*: temperature in ºC.
- *RH*: relative Humidity in
- *AH*: Absolute Humidity.

```r
# Load chosen data
AirQualityUCI <- read_delim("AirQualityUCI.csv", ";",
                            escape_double = FALSE,
                            col_types = cols(`CO(GT)` = col_character(),
                            X16 = col_skip(), X17 = col_skip(),
                            T = col_character()),
                            trim_ws = TRUE, na="-200")
```

Time Series

A time series (TS) is a collection of data recorded over a period of time, which can be time—weekly, monthly, quarterly, or yearly, that is used to derive hidden insights to make informed decision making. In other words, TS is a collection of data points collected at constant time intervals that are a variable of interest because it helps to analyze data that contains information from the past to determine the long term trend to try to forecast the future or perform some other form of analysis. For example, most of business houses work on time series data to analyze sales numbers for the next year, website traffic, competition position and much more. Time series models are very useful models when data is serially correlated.

During this assignment, we will just consider **univariate time series** which is a sequence of measurements of the same variable collected over time at regular time intervals. The main characteristics of them are the following ones:

- The order of the observations matters. Changing the order could change the meaning of the data.
- The observations does not need to be time dependent.

- The observations does not need to be identically distributed.

The main objective of analyzing time series usually is to determine a model that describes the pattern of the time series, by describing the important features of the time series pattern to forecast future values of the series. Our objective for this assignment is try to predict the temperature for the next day of the specific Italian city. This procedure is developed during the next sections with our chosen dataset.

Preprocessing needed

The first step that need to be done for being able to analyze the dataset is making the right preprocessing to our chosen data.

After analyzing our already loaded data of 9355 records, we have joined the data of columns *Date* and *Time* into the already existent Data column and we have deleted the *Time* column afterwards. Then, we realized that *Date* column was defined as a character variable, so we have changed it to time variable. To do so, we have set all new values joined to an specific format, which will be needed later on to further analyze the TS. Then, the samples with this *Date* value missing have been removed.

```
# Join date and time to the same column
date_time <- paste(AirQualityUCI$Date, AirQualityUCI$Time)

# Remove previous columns
AirQualityUCI$Date = date_time
AirQualityUCI$Time = NULL

#Change form character to POSIXlt variable
AirQualityUCI$Date = strptime(AirQualityUCI$Date, format="%d/%m/%Y %H.%M.%S")
#Delete missing values
AirQualityUCI = AirQualityUCI[!is.na(AirQualityUCI$Date),]
```

Furthermore, as our final objective is to predict the temperature for the next day, the *T* column of the dataset is the one that provides the temperature values that will be needed for our analysis. As this column has also been loaded as a character value, we have converted it to numeric but to successfully achieve it, we first have needed to change the character , to ., otherwise, the transformed values that we obtained were not the correct ones. We also have decided to substitute the missing values of *T* for the average temperature of the whole samples, instead of removing them as they are aprox. 360 samples.

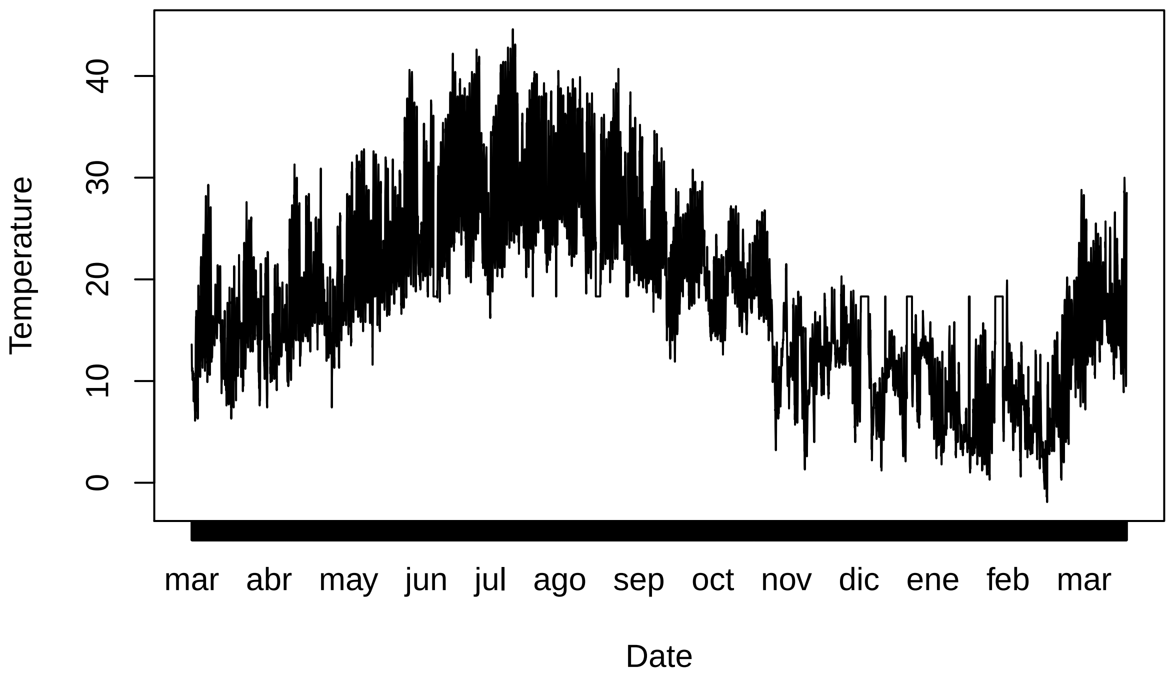
```
AirQualityUCI$T =  as.numeric(sub(",", ".", AirQualityUCI$T, fixed = TRUE))
#which(is.na(AirQualityUCI$T))
AirQualityUCI$T[is.na(AirQualityUCI$T)] <- mean(AirQualityUCI$T, na.rm = TRUE)
```

Time Series Analysis

After preparing the data as needed, we have just keep *Date* and *T* columns for our further analysis. A plot with the results obtained after finishing our data preprocessing is showed below.

```
df <- subset(AirQualityUCI, select = -c(2,3,4,5,6,7,8,9,10,11,13,14))
```

```
plot(df,xaxt="n",ylab="Temperature", type='l')
axis.POSIXct(1, at=df$Date, format="%b")
```



We can see our time series with the plot showed. On it, the different records seem to have an inverse tendency each half a year and it seems to have similar values for final March and April months. So we can imagine that exists a periodicity of period 1 year but we cannot see it as our data just last one year and a month. But we can also see that exists a seasonality as there is regularly repeating pattern of highs and lows related to calendar time such as days. Also, we cannot see clear outliers, but it is clear that noise exists on the data. And, finally, it seems that this time series could probably be described using an additive model, as days fluctuations are roughly constant in size over time and do not seem to depend on the level of the time series, and the random fluctuations also seem to be roughly constant in size over time.

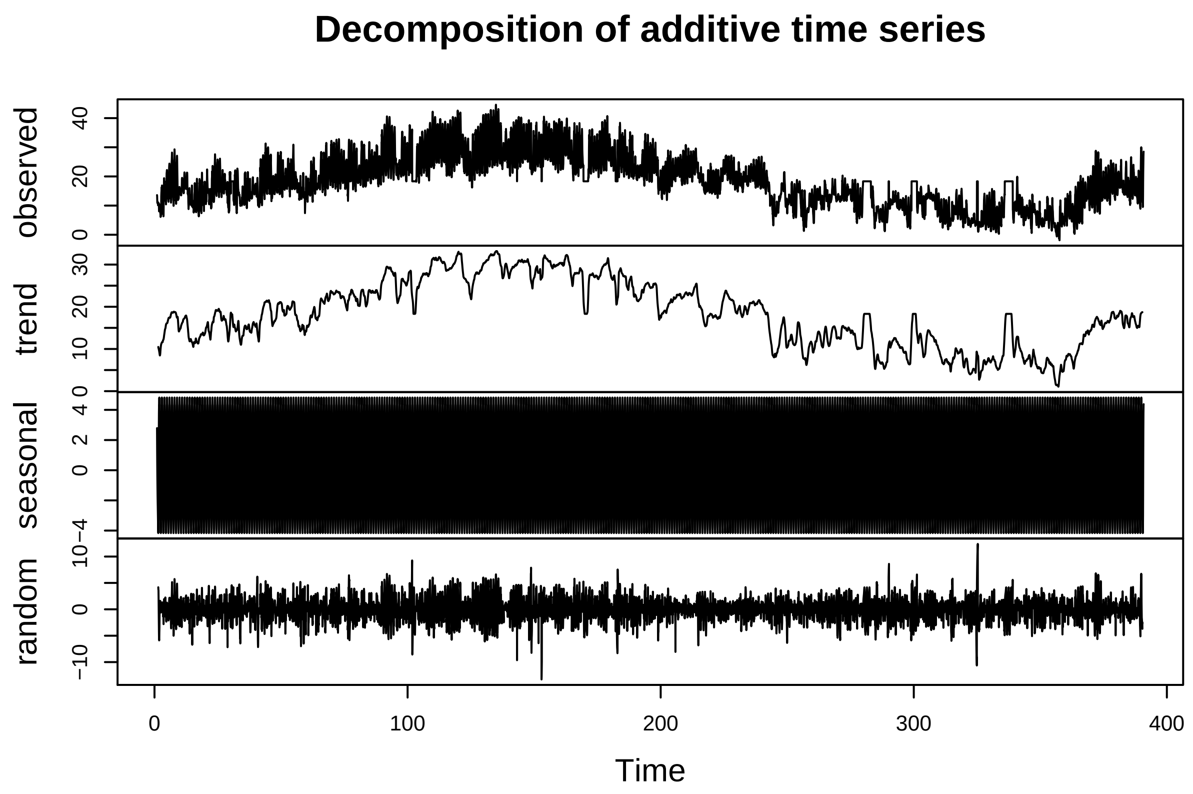
To make things easier, the preprocessed dataframe has been converted to a time series variable with frequency 24 as there is a record for each hour which makes 24 observations per day.

```r
# Frequency = 24 due to 24 observations per day (1 per hour)
time_series <- ts(df$T, frequency=24)
```

As we have decided that our time series can be described using an additive model and it is seasonal, to be able to deeply analyze it, we have decomposed it. Decomposing a time series means separating it into its basic constituent components, which are a trend component, an irregular component and a seasonal component, if it is a seasonal time series.

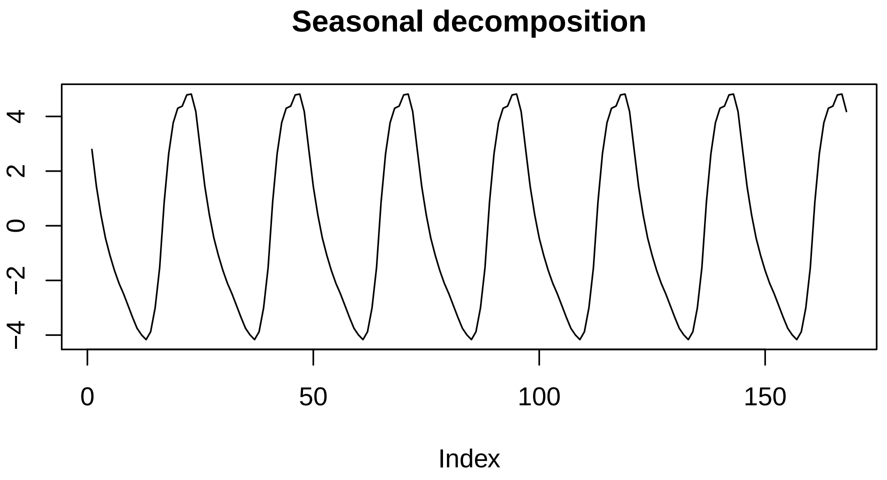
```r
decomposed <- decompose(time_series)
```

```r
plot(decomposed)
```



With the plot obtained, trend component show the whole tendency of the time series so we are able to see that the warmest temperatures are found between July and August months with a temperature around 30ºC. On the other side, the coldest temperatures are found during January with values around 5ºC. The pics shown on this trend can be produced for the substitution of the NAs values for the mean one. Moreover, random component is considerable for the whole observations and seem to be quite constant.

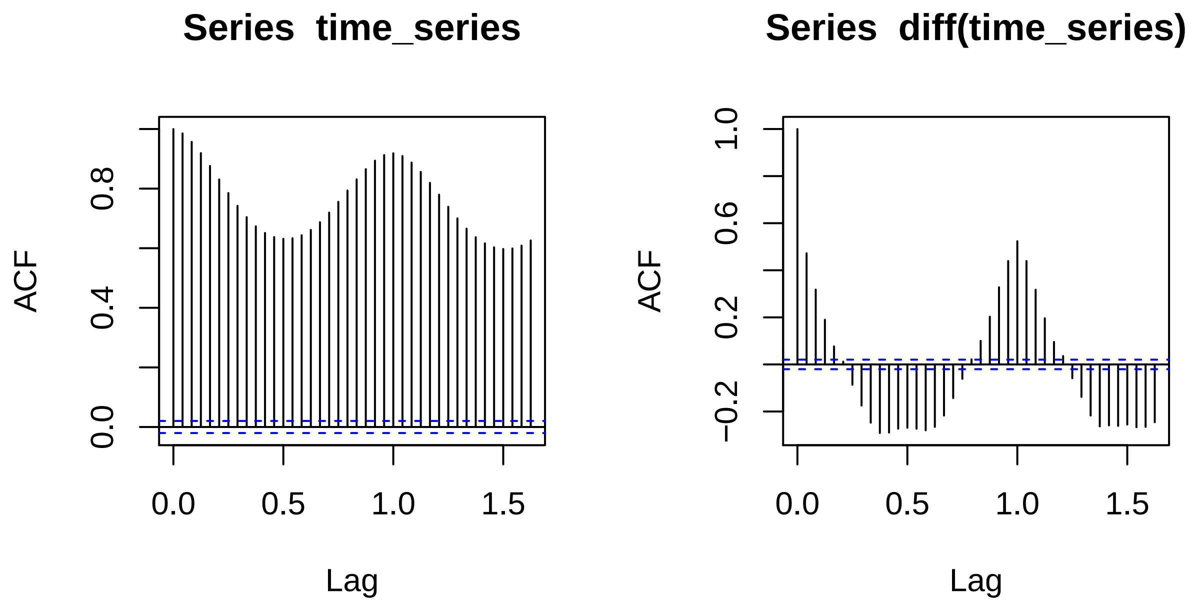
```
plot(decomposed$seasonal[seq(1,24*7)], type='l', ylab="", main ="Seasonal decomposition")
```



As the seasonal component is difficult to see with the first plot retrieved, we have plot it for just a week (7 days). With this new plot, we can see that exists the same tendency that is repeated each day, which makes our time series clearly seasonal.

We have also calculated the *acf* function which calculates the autocovariance or autocorrelation of our TS, in order to know up to what extent current values are related to the past values and other useful characteristics of the plot. Thus, we can know which past values will be most helpful in predicting future values and it will also help us to check the stationability of it.

```
par(mfrow=c(1,2))
acf(time_series)
acf(diff(time_series))
```



In fact, the final characteristic that need to be studied before starting our prediction from the created time series is stationary series.

Stationary Series

It is important to know if our data is stationary, this is due to the fact that we cannot build a prediction model with data unless the observation are stationary. Being stationary means that we are looking for a constant variance over time and a constant autocorrelation structure with no periodic fluctuations.("6.4.4.2. Stationarity," n.d.)

In order to test for the stationarity of our data we use the *Augmented Dickey-Fuller **(ADF)** Test* which test the *null hypostesis* that a unit root is present in a time series sample. By *unit root* we mean a feature of some time series in which in the case of a change on the trend line of the data, a time series with the unit root feature will not converge again with the trend line. (Nielsen 2015)

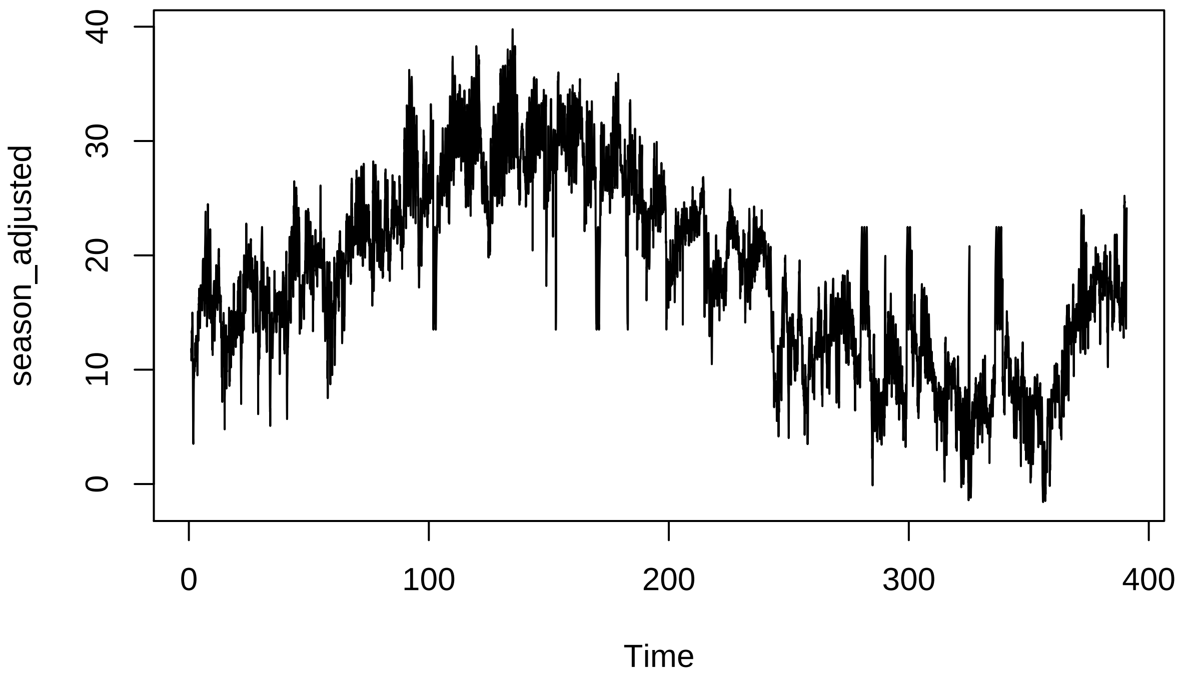
The following method test the data for stationarity.

```
adf.test(time_series, alternative="stationary") # Not stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  time_series
## Dickey-Fuller = -3.0468, Lag order = 21, p-value = 0.1349
## alternative hypothesis: stationary
```

We can see how our p-value is bigger than 0.05, thus our data is non-stationary. Then, in order to work with the TS, first we must covert the data into stationary. To do so, we de-seasonalize our TS by removing the already calculated seasonal component from our timeseries. (Business, n.d.)

```
season_adjusted = time_series - decomposed$seasonal
plot(season_adjusted)
```



We can see how quite a bit of "noise" has been removed from our data by substracting the seasonal component. If we test again for the stationary component of the data we can see how now the p-value being lower than 0.05 indicates that the data is now stationary.

```
adf.test(season_adjusted, alternative="stationary") # Stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  season_adjusted
## Dickey-Fuller = -3.6021, Lag order = 21, p-value = 0.03234
## alternative hypothesis: stationary
```

From now on, we will use the seasonal adjusted stationary data instead of the pure time series data.

Forecast Prediction

Our main goal is to try to predict the temperature of the Italian city for an small period of time. Once data have been captured for the time series to be forecasted, we need to select a model for forecasting. In order to do so, we will use two useful methods for predictions according to the characteristics observed of our timeseries. These are **Arima** and **HoltWinters**. We want to see how this two methods work and behave and to check if they produce any significant differences in their results for forecasting the temperature.

Arima

Arima (*Autoregressive integrated moving average*) models are used for forecasting a time series which is stationary (such in our case). An Arima model can be viewed as a filter that tries to separate the signal from the noise, and the signal is then extrapolated into the future to obtain forecasts. The Arima equation is linear, in which the predictor if formed of lags (prior values) of the dependent variables and/or lag errors. (Business, n.d.) That is:

Predicted value of $Y = a$ constant and/or a weighted sum of one or more recent values of Y and/or a weighted sum of one or more recent values of the errors.

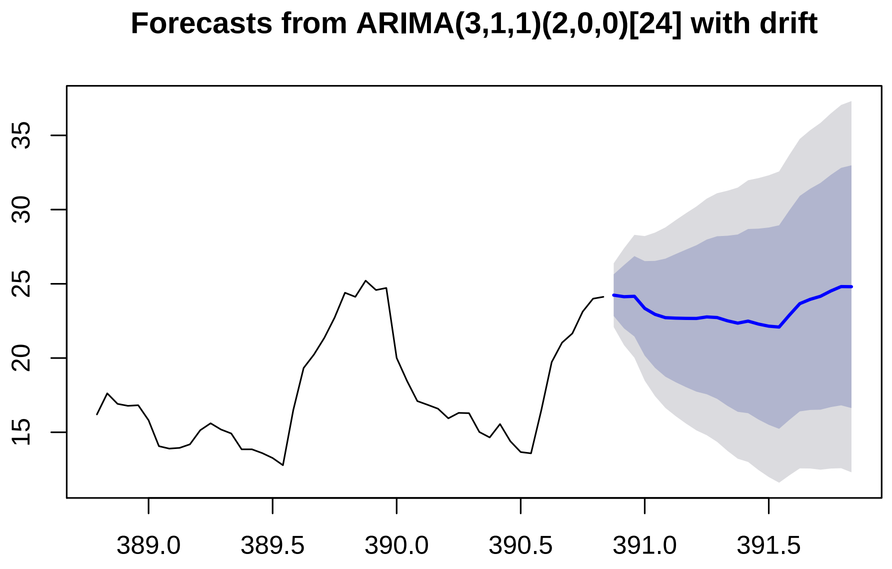
The arima model is composed of three different parameters.

- **p**: number of autoregressive terms.
- **d**: number of nonseasonal differences.
- **q**: number of lagged forecast errors.

In our case we will use the *auto.arima* method which automatically calculates the values needed for the arima model, in our case we put a seasonal model in the arima so it calculates the different temperatures for the following days taking into account the seasonal nature of our data.

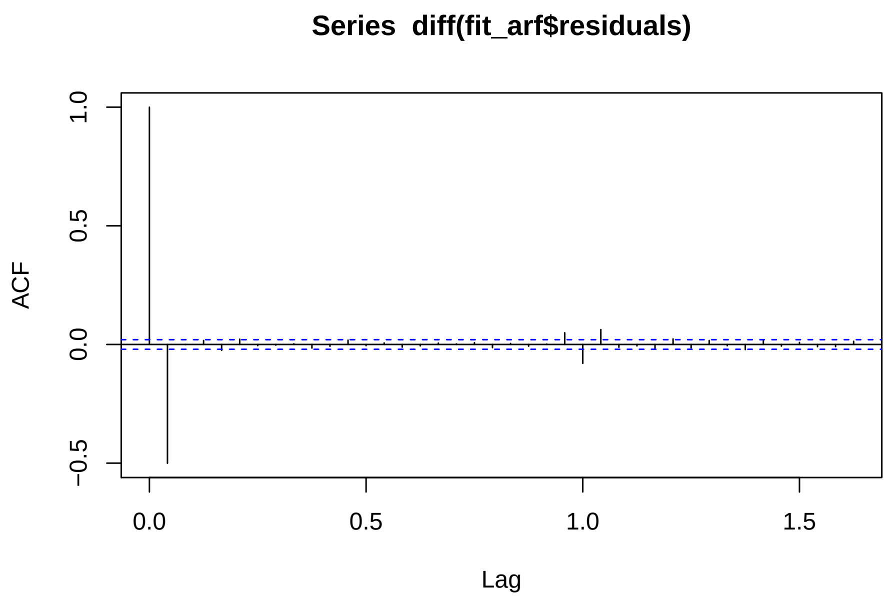
```
fit_ar <- auto.arima(season_adjusted, seasonal=TRUE)
fit_arf <- forecast(fit_ar, h=24)
```

```
plot(forecast(fit_arf,h=24), include = 50)
```

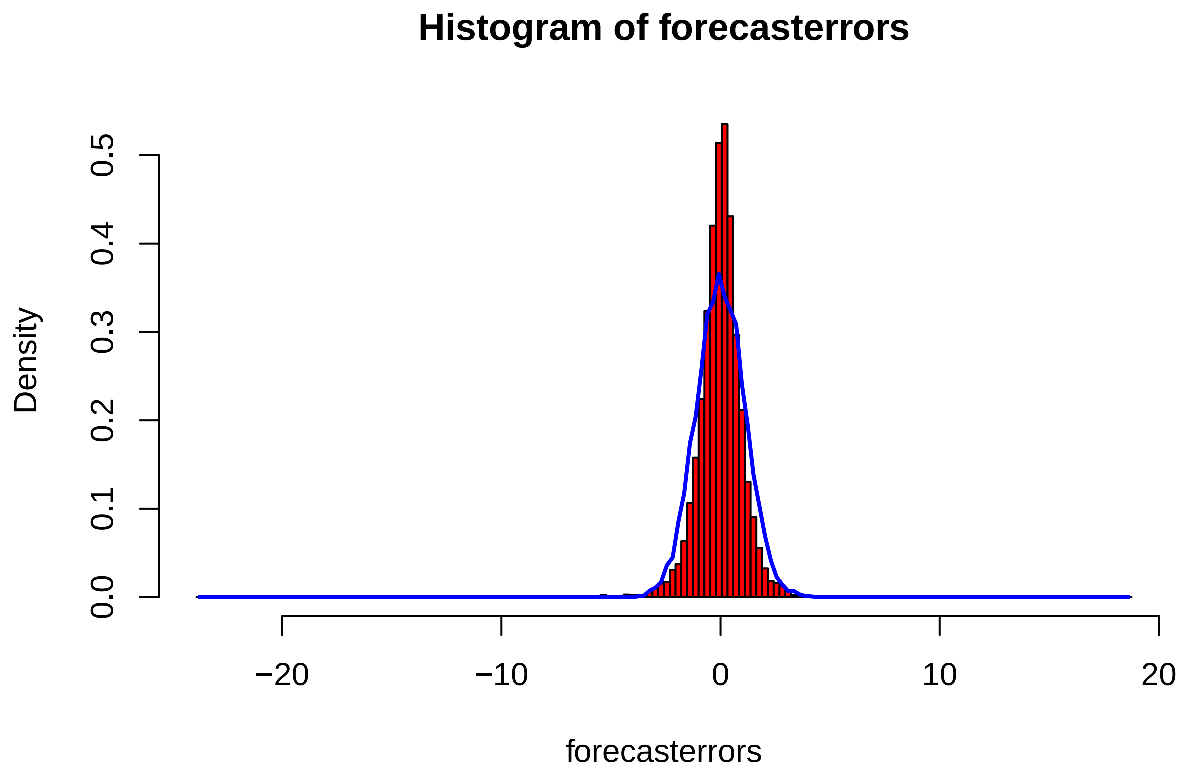


We can see how the prediction of the arima shows that the mean fit in the following days will have the same temperature as the current day, yet if we look at the whole fit (dark blue) we can see how the forecast predicts that there can be a lot of variety in the temperature, meaning that the average blue line of the fitted data can not be trusted as a perfect predictor, yet it seems like the temperature will first go down and then rise again, similarly to the previous days in our data.

```
fit_arf$residuals[is.na(fit_arf$residuals)] = 0 # If no difference set to 0
acf(diff(fit_arf$residuals))
```



```r
plotForecastErrors(fit_arf$residuals)
```



From the acf plot, it seems that there is little evidence of auto-correlation for the forecast errors, and from the forecast errors, we can observe how the errors are normally distributed with mean zero and constant variance over time, yet we can see how the errors appear to be a bit to high. This suggests that the implemented Arima method provides an adequate forecasting model although not perfect.

HoltWinters

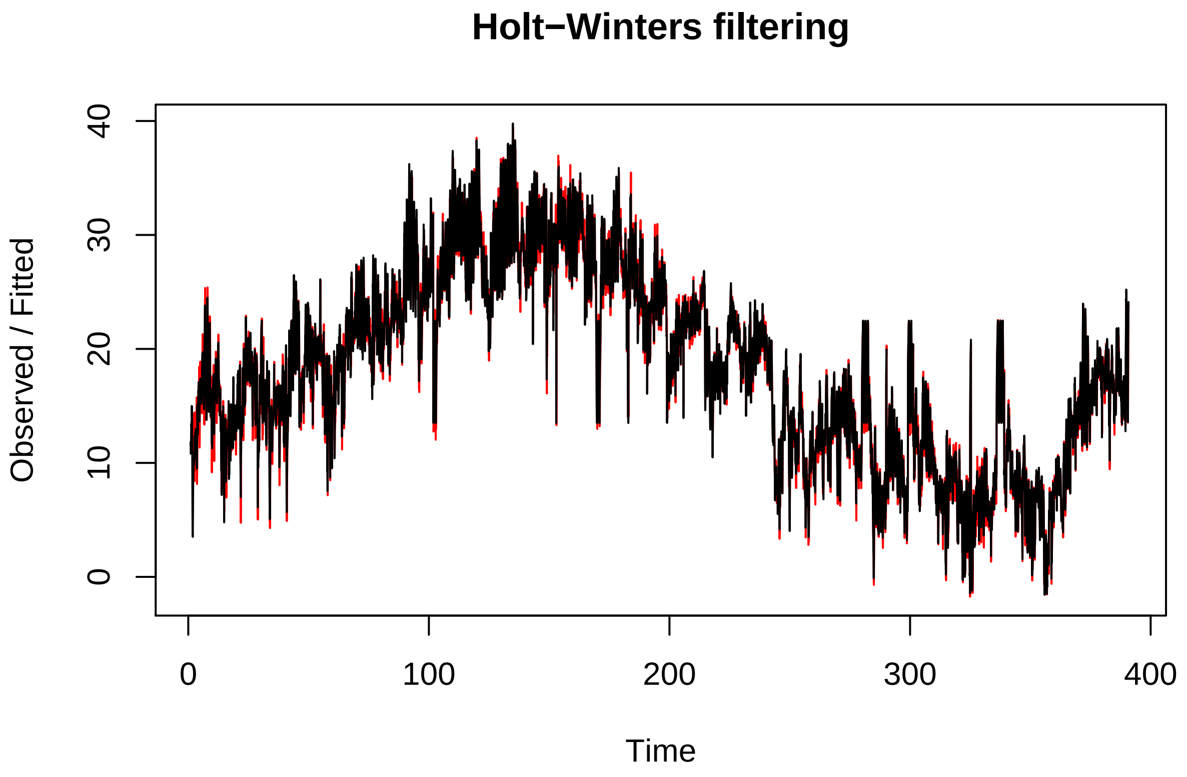
Holt Winters, also known as Triple Exponential Smoothing, is a method used for forecasting a time series which is *stationary* and shows *trend* and *seasonality*. Exponential smoothing is a procedure that continually revises a forecast in the light of more recent experience. Exponential Smoothing assigns exponentially decreasing weights as the observation gets older. In other words, recent observations are given relatively more weight in forecasting than the older observations (Kalekar 2004). It is controlled by three parameters, so the main objective of this method is to estimate the level (alpha), slope (beta) and seasonal component (gamma) at the current time point.

It is usually applied recursively to each successive observation in the series as the new smoothed value (forecast) is computed as the weighted average of the current observation and the previous smoothed observation. The alpha, beta and gamma parameters all have values between 0 and 1 and the values that are close to 0 mean that relatively little weight is placed on the most recent observations when making forecasts of future values.

As our time series has been already adjusted, meaning that we have subtracted the seasonal component from it to make it stationary, the gamma parameter has been set to false for our implementation of Holt Winters method.

```
#### Predict using HoltWinters ####
df.prediction <- HoltWinters(season_adjusted)
```

```
plot(df.prediction)
```



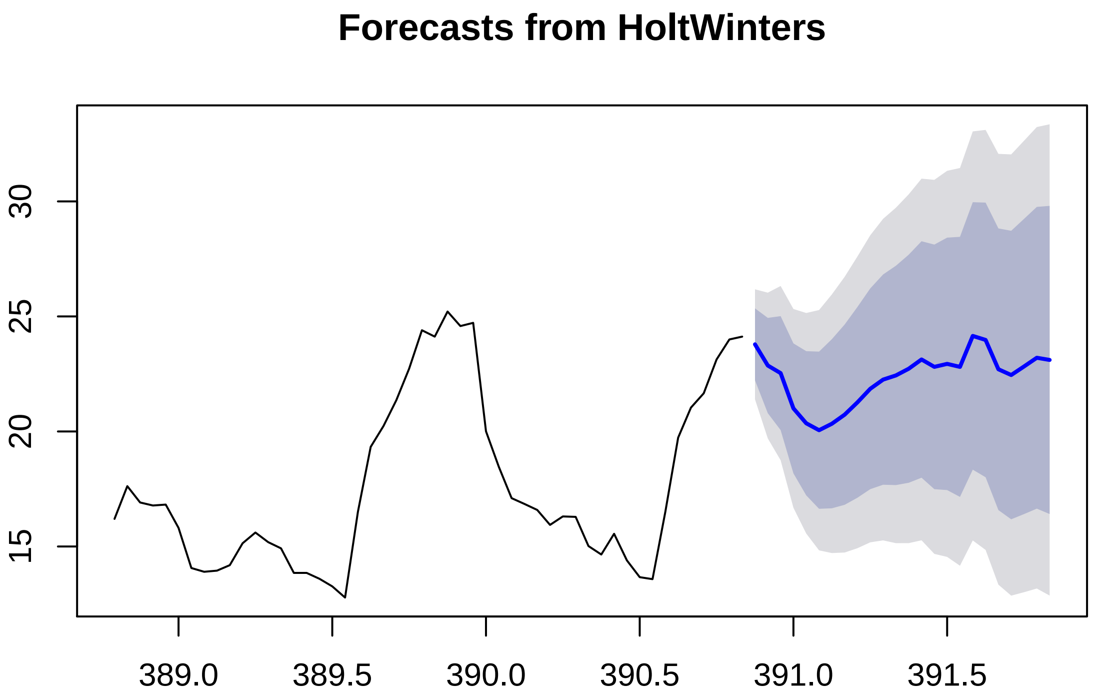
The estimated values of alpha, beta and gamma are 0.864, 0 and 1, respectively. The value of alpha (0.864) and gamma (1) indicate that the estimate of the level at the current time point is based upon most recent observations. And the value of beta (0), indicates that the estimate of the slope b of the trend component is not updated over the time series and it is set equal to its initial value.

With the plot retrieved, we can see that the Holt Winters time series created (red line) seems to be very successful in predicting the original time series (black line), as they obtain the similar time series.

After the model has been created using Holt Winters, we predict the values for the next day (24 values, one for each hour).

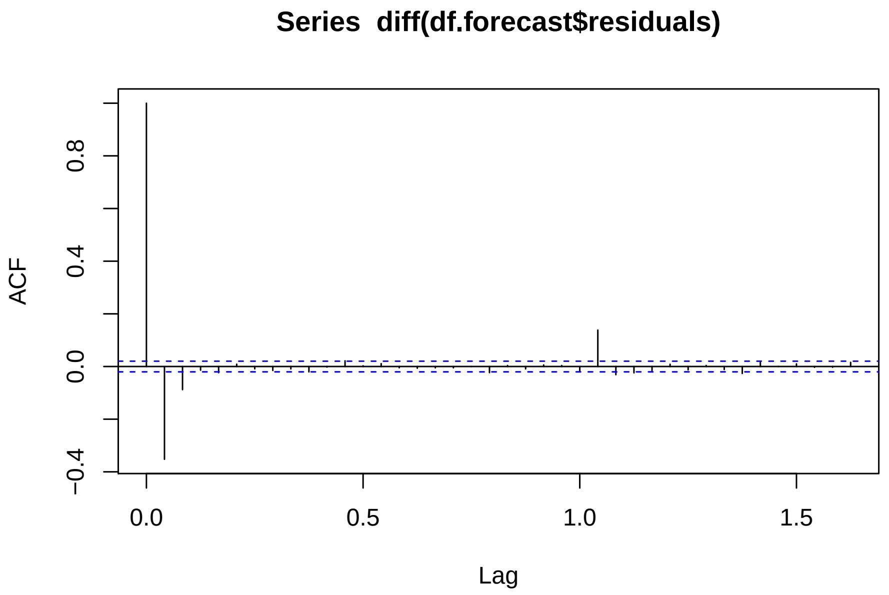
```
df.forecast <- forecast(df.prediction, h=24)
```

```r
plot(df.forecast, include=50)
```



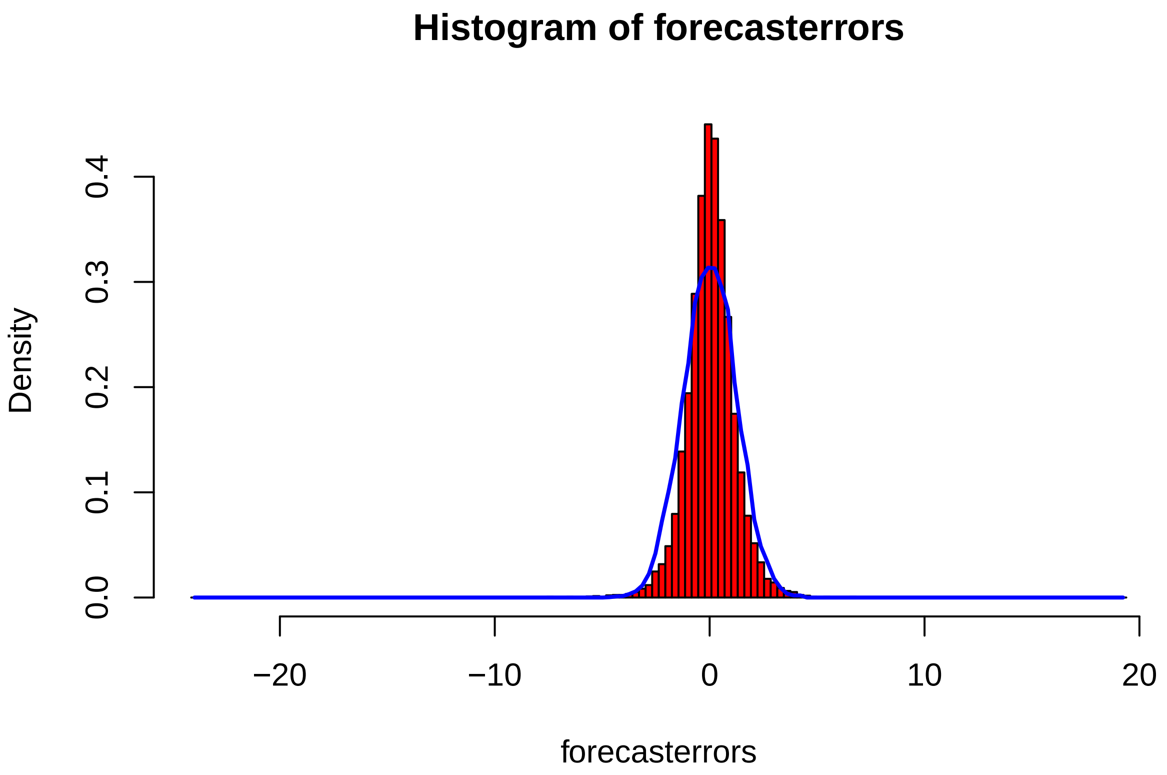
As we can see Holt Winters says that the in the best fit scenario the temperature will be the average temperature of the days before. yet if we look at the whole fit (dark blue) we can see how the forecast predicts more less the limits on the maximum and minimum temperatures of the days before. In order to be sure of our predictions we also check the errors in the forecast as we did with ARIMA method.

```r
df.forecast$residuals[is.na(df.forecast$residuals)] = 0 # If no difference set to 0
acf(diff(df.forecast$residuals))
```



We can investigate whether the predictive model can be improved upon by checking whether the in-sample forecast errors show non-zero auto-correlations at lags 1-20, by making a correlogram and carrying out the Ljung-Box test.

```r
plotForecastErrors(df.forecast$residuals)
```



From the acf plot, it seems that there is little evidence of auto-correlation for the forecast errors, and from the forecast errors, we can observe how the errors are normally distributed with mean zero and constant variance over time. This suggests that the implemented Holt Winters method provides an adequate predictive model.

Conclusions

With this paper we hope to have shown our learning procedure regarding the time series topic as well as, have shown how it is possible to use time series in order to forecast future data using two different techniques, *arima* and *HoltWinters*. Our understanding of time series models has grown over the period we have been working on this project, since we have done many iterations over our different models, the pre processing of the data and the adjustments over the data that we have done.

When working with time series there are some important steps to take into account that we have learned during the time we have worked in this project:

- Checking if our data is stationary or not.
- Checking if our data is additive or multiplicative.
- Checking if our data has outliers or not.
- Checking if our data has seasonability or not.
- Decomposing our data to better understand it and check for the trend.

Knowing if our data is stationary or not or all the other characteristics, is the first step to do when working with time series since it affects the possible models that we can be used in our forecasting. Yet, it is possible to make data that is non-stationary into stationary, as demonstrated in our paper. This has been done as explained, by removing the seasonal component of the data, but there are many other ways to achieve it.

Furthermore, the decomposition of the different components helps up get a better understanding of our data because it helps to see the other characteristics needed for the TS. For example, by knowing the trend we can see if our dataset has and additive or a multiplicative component, meaning that there are trends that repeat over time or if the trend line just keeps increasing.

Regarding the forecasting, both models procedure different outcomes. *HoltWinter* seems to have a better prediction than *arima*, as it follows the trend line of the previous days better, and has a lower error model, while *arima* has more of an straight line with a big *sd* from the mean which does not seem to follow the trend from the previous days, this also makes the errors for the *arima* model to be bigger.

To sum up, we want to stress the importance of spending time on analyzing the above mentioned characteristics of our time series before starting any other process, since these greatly affects which models can be used in order to predict the data, as well as choosing the correct parameters for the models.

The time series world is a big topic which we have only done an small exploration into, and there is still a lot of knowledge to be explored in this topic which we did not have the time to do well into, yet we found it to be a really interesting subject.

References

"6.4.4.2. Stationarity." n.d. *1.3.3.14.6. Histogram Interpretation: Skewed (Non-Normal) Right.* U.S. Department of Commerce. https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc442.htm.

Business, Fuqua School of. n.d. "Introduction to Arima Models." *Seasonal Differencing in ARIMA Models.* https://people.duke.edu/~rnau/411arim.htm.

———. n.d. "Stationarity and Differencing of Time Series Data." *Seasonal Differencing in ARIMA Models.* https://people.duke.edu/~rnau/411diff.htm.

Dheeru, Dua, and Efi Karra Taniskidou. 2017. "UCI Machine Learning Repository - Air Quality Data Set." University of California, Irvine, School of Information; Computer Sciences. https://archive.ics.uci.edu/ml/

datasets/Air+quality.

Kalekar, Prajakta S. 2004. “Time Series Forecasting Using Holt-Winters Exponential Smoothing.” *Kanwal Rekhi School of Information Technology* 4329008: 1–13.

Nielsen, Heino Bohn. 2015. “Non-Stationary Time Series and Unit Root Tests,” October, 1–25. http://www.econ.ku.dk/metrics/Econometrics2_05_II/Slides/08_unitroottests_2pp.pdf.