Time series Project (Fabio Costantino)

My role in the Company

As a Content Analyst in the Localization Team, I am responsible for migrating data in and out of our Content Management System called STEP. These are mostly text data - a.k.a. content, which are then published daily in our website for all customers to consume.

As a Localization team, we mainly deal with translated text in 12 different languages, therefore also managing the flow of translation files in and out of the company. In order to translate our content, I have daily contact with external translation agencies and manage the use of a Translation Management System called XTM, which helps organize all files and workflows in a single tool.

Besides, we also provide our manager(s) and the rest of the team with reports of many kind: from tailored collection of data coming from different sources, to reports on products (or families of products) performance.

Some of the tools I use on a daily/weekly basis to perform my job are: **STEP Content Management System**, **XTM Translation Management System**, **Microsoft SQL Server Management Studio**, **Report Builder** from **Adobe Analytics**, **DOMO** as a BI dashboard tool, **Office package**, **Jupyter Notebook** (with **Python**).

Brief

Explore the seasonality for a family of products to establish changes in demand and workload for the Content team.

Project explanation

Business Problem

When chosing the product pages to be optimized in the website, our company often adopts a general and *one-size-fits-all* approach to this task. Often an entire section of the website is revised. This results in an almost constant wave of changes and amendments, that proves challenging not only for the Content and Localization team members but also on the computational capacity of our Content Management System - called **STEP**.

Current Process

At the moment of writing, the team takes on-board every suggestion, internal or external, coming its way at every point in time during the year. The amendment process is therefore very much responding only to external pushes, without a scheduled strategy, leaving very important corrections competing with unimportant, cosmetic changes for attention.

Current issues

It is not uncommon that while a large section of the team's effort is already committed to deal with this indiscriminate wave of changes, the demand for other families of products might be peaking without any attention given to it, simply because nobody is aware of it. However, it wouldn't be feasible to check every week - let alone everyday! - what are the products more in demand presently.

Proposed solution

I propose that, by analyzing the peak and dips of demand by product families, the Content and Localization team can better identify and prioritize those categories of products that need immediate attention - because soon their demand will peak. Having already optimized the pages for a peak in demand will certainly prove beneficial.

This research only looks at one sector, i.e. that of electronics and computers, but if the results prove interesting the scope should be broadened. In order to gain a broader insight, instead of using company data, I will use data from the European Union. This will allow me to be as impartial as possible, since I will be analyzing this problem from the point of view of the public and not of RS as a private company.

Pros and Cons

If I can derive a clear answer from the data, I believe my findings could help the swtich from a passively responsive approach to optimization to a proactive and more informed one. Also, it could help planning workload and even headcount in the long run, once it is coupled with e.g. forecasting on the ingestion of new products (for

example, knowing that the offer of electronic products will grow by 10% in the next 12 months thanks to a contract with a new vendor, the Head of the department might choose to hire an additional employee in time for a peak).

On the other hand, the analysis of seasonality clearly cannot take into account shocks in the systems, both internal or external, therefore it should be repeated overtime in order to identify significant shifts in market demand.

Assumptions

- Although my company operates in several countries, I will limit the scope of this project to 4 main European markets - France, Germany, Italy and Spain.

Hypothesis

I want to understand whether there is a significant difference in seasonality among countries or if I can find a common solution for all of them. I will consider a *significant difference in seasonality* a clear and repeated

0. Import of libraries

First of all I need to import all relevant libraries to this analysis.

In [1]:
```
# To use with basic data wrangling and manipulation
import pandas as pd
import numpy as np

# Used to display screenshots in Jupyter notebook.
from IPython.display import Image

# To merge multiple dataframes together at the same time
from functools import reduce

# To visualize data
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import seaborn as sns

# To visualize decomposition of time series data
from statsmodels.tsa.seasonal import seasonal_decompose

# To visualize partial correlation function
from statsmodels.graphics.tsaplots import plot_pacf

# To normalize data
from sklearn import preprocessing

# To suppress warnings
import warnings; warnings.simplefilter('ignore')

%matplotlib inline
```

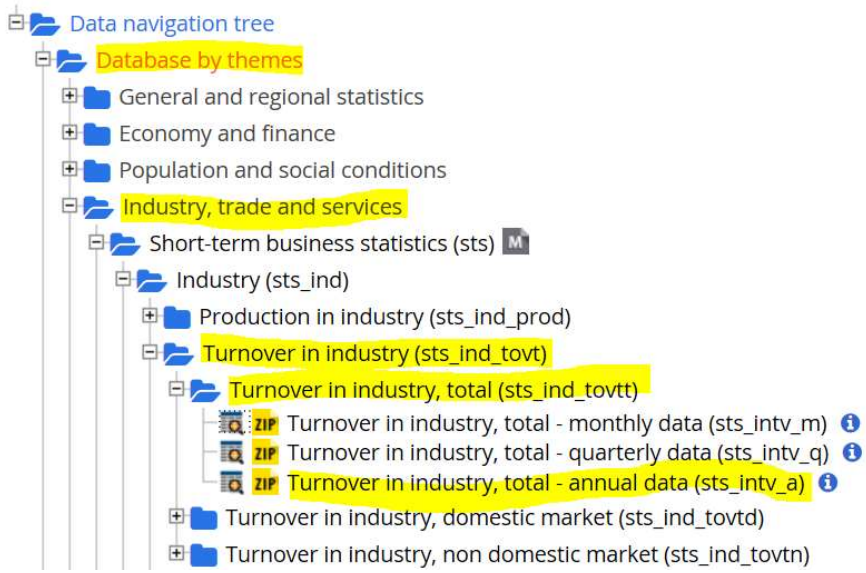
```
C:\Users\E0658269\AppData\Local\Continuum\anaconda3\lib\site-packages\statsmo
dels\tools\_testing.py:19: FutureWarning: pandas.util.testing is deprecated.
Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
C:\Users\E0658269\AppData\Local\Continuum\anaconda3\lib\site-packages\statsmo
dels\compat\pandas.py:23: FutureWarning: The Panel class is removed from pand
as. Accessing it from the top-level namespace will also be removed in the nex
t version
  data_klasses = (pandas.Series, pandas.DataFrame, pandas.Panel)
```

1. Sourcing and cleaning data

Data from the European Union have been gathered, showing the turnover for **manufacture of computer, electronics and optical products**. The same data can be found here: https://ec.europa.eu/eurostat/data/database (https://ec.europa.eu/eurostat/data/database), then following the highlighted sections in the picture.

In [2]:
```
# Access to the database
Image('path_to_data.PNG', width=500, height=500)
```

Out[2]:



To obtain exactly the same dataset, the following categories must be chosen:

- **Classification of economic activities**: Manufacture of computers, electronics and optical products (C26)
- **Seasonal adjustment**: Calendar adjusted data, not seasonally adjusted data
- **Business trend indicator**: Index of turnover - Total
- **Unit of measure**: 2015=100

Now I will load the data obtained.

In [3]:
```
# Sourcing the data by reading the csv file
data = pd.read_csv('sts_intv_m_1_Data.csv')

# Visualizing the last few rows to check correct implementation
data.tail()
```

Out[3]:

	TIME	GEO	INDIC_BT	NACE_R2	S_ADJ	UNIT	Value	Flag and Footnotes
625	2020-03	France	Index of turnover - Total	Manufacture of computer, electronic and optica...	Calendar adjusted data, not seasonally adjuste...	Index, 2015=100	:	NaN
626	2020-03	Italy	Index of turnover - Total	Mining and quarrying; manufacturing	Calendar adjusted data, not seasonally adjuste...	Index, 2015=100	:	NaN
627	2020-03	Italy	Index of turnover - Total	Manufacture of computer, electronic and optica...	Calendar adjusted data, not seasonally adjuste...	Index, 2015=100	:	NaN
628	2020-03	United Kingdom	Index of turnover - Total	Mining and quarrying; manufacturing	Calendar adjusted data, not seasonally adjuste...	Index, 2015=100	:	NaN
629	2020-03	United Kingdom	Index of turnover - Total	Manufacture of computer, electronic and optica...	Calendar adjusted data, not seasonally adjuste...	Index, 2015=100	:	NaN

It is immediately clear that some rows have null values (:), which must be addressed. I chose to create a separate dataframe called `na` to simplify the tasks of exploring these values.

In [4]:
```
# Creating a separate dataframe for NA values
na = data[data['Value'] == ':']

# Exploring the dataframe
na.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8 entries, 622 to 629
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   TIME                8 non-null      object
 1   GEO                 8 non-null      object
 2   INDIC_BT            8 non-null      object
 3   NACE_R2             8 non-null      object
 4   S_ADJ               8 non-null      object
 5   UNIT                8 non-null      object
 6   Value               8 non-null      object
 7   Flag and Footnotes  0 non-null      object
dtypes: object(8)
memory usage: 576.0+ bytes
```

The problem seems to be minimal, and I decided to simply delete these rows. However, before doing so, I want to be sure that this wouldn't create some imbalance between the countries represented in the dataset.

In [5]:
```
# Counting the null values per country
na.groupby('GEO')['Value'].value_counts()
```

Out[5]:
```
GEO             Value
France          :        2
Italy           :        2
Spain           :        2
United Kingdom  :        2
Name: Value, dtype: int64
```

Germany seems to have 2 valid datapoints more than the other countries. In order to understand further, I will see whether there is any connection with the column `TIME`.

In [6]:
```
# Extracting all the months that are affected by the null values
na['TIME'].unique()
```

Out[6]:
```
array(['2020-03'], dtype=object)
```

Now I know that only March 2020 seems to be affected, therefore I can get rid of that single month for all countries, to maintain balance in the dataset.

At the same time, since there are many columns that I am not going to use, I will drop them, leaving only information about the date, the country and the total value for the turnover.

In [7]:
```
# Getting rid of all rows with values equalling ":"
data = data[data['TIME'] != '2020-03']

# Creating a smaller dataset containing TIME, GEO, and Value only.
data_filtered = data[['TIME', 'GEO', 'Value']]

# Checking the types of data contained in the dataset
data_filtered.dtypes
```

Out[7]:
```
TIME     object
GEO      object
Value    object
dtype: object
```

Clearly, as a last piece of data cleaning, I need to change the data types:

- Column `TIME` should be transformed into dates
- Column `Value` should be changed to floats

In [8]:
```
# Assigning the correct data type to 'TIME' and 'Value' columns
data_filtered['TIME'] = pd.to_datetime(data_filtered['TIME'], format='%Y-%m')
data_filtered['Value'] = data_filtered['Value'].astype(float)

# Checking proper implementation
data_filtered.dtypes
```

Out[8]:
```
TIME     datetime64[ns]
GEO              object
Value           float64
dtype: object
```

Because of the hypothesis I want to test, I will need to create one dataset per country. Once the division is done, each new dataset will not need the `GEO` column any longer, therefore it will be dropped. The observations will be grouped by date and the sum will be displayed in case more than one entry was present for that particular time stamp.

In [9]:
```
# Creating distinct datasets
data_Germany = data_filtered[data_filtered['GEO'] == 'Germany'].drop('GEO', ax
is=1).groupby(['TIME']).sum()

data_Spain = data_filtered[data_filtered['GEO'] == 'Spain'].drop('GEO', axis=1
).groupby(['TIME']).sum()

data_France = data_filtered[data_filtered['GEO'] == 'France'].drop('GEO', axis
=1).groupby(['TIME']).sum()

data_Italy = data_filtered[data_filtered['GEO'] == 'Italy'].drop('GEO', axis=1
).groupby(['TIME']).sum()

data_UK = data_filtered[data_filtered['GEO'] == 'United Kingdom'].drop('GEO',
axis=1).groupby(['TIME']).sum()
```

In [10]:
```
# Visualizing one dataset as an example
data_Italy.head()
```

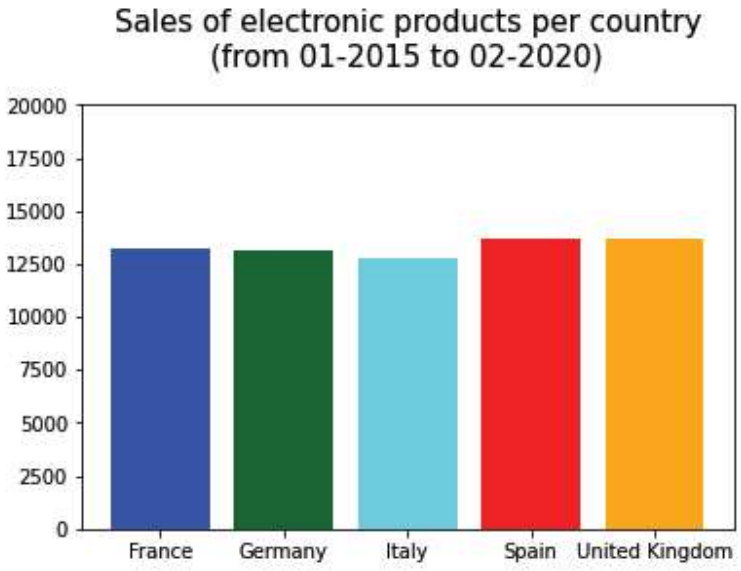
Out[10]:

TIME	Value
2015-01-01	158.9
2015-02-01	178.7
2015-03-01	214.6
2015-04-01	191.2
2015-05-01	204.3

2. Data visualization

Three basic visualization will be presented in this section. The first shows how the turnover is distributed among the five countries.

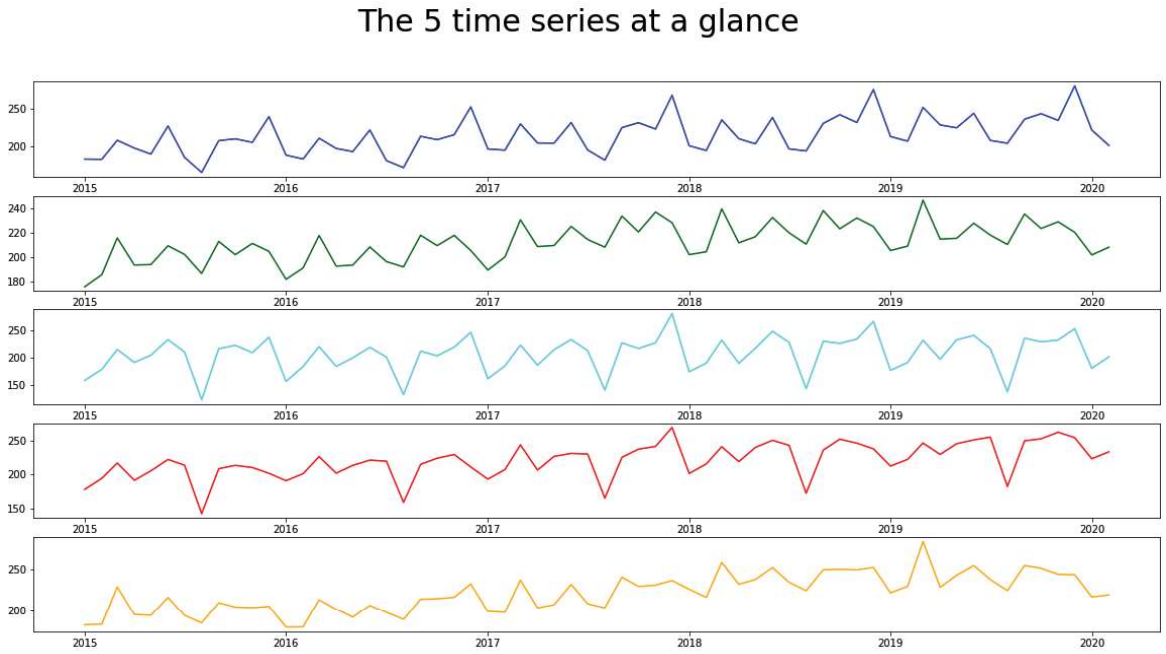
In [11]:
```
# First I create a subset with sales grouped by countries
sum_of_products = data_filtered.groupby(['GEO']).sum().reset_index()

# Then I create a barplot with the values summed up by country
my_colors = ['blue', 'darkgreen', 'cyan', 'red', 'orange']
plt.bar(sum_of_products['GEO'], sum_of_products['Value'], color = my_colors)

# Giving a title and labels to each axis
plt.title('Sales of electronic products per country\n(from 01-2015 to 02-202
0)', fontsize = 15, pad = 20)
plt.xticks(sum_of_products['GEO'])

# Establishing a limit on the y axis to squeeze the image
plt.ylim(0, 20000)

# Showing the plot
plt.show()
```



This visualization clearly shows that the dataset is well balanced, with Spain and UK being ever so slightly above the rest.

The second visualization shows the five time series, again divided by country. This is only to have a general idea of the series, which I will be exploring in more details later.

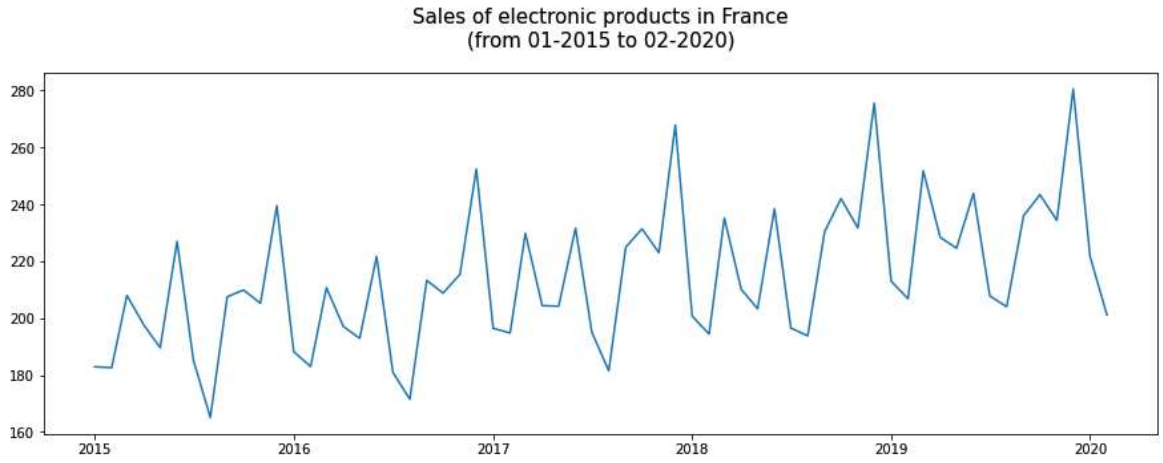
In [12]:
```
# Creating a figure with 5 subplots
fig, axs = plt.subplots(nrows=5, ncols=1)

# Giving a title to the graph
fig.suptitle('The 5 time series at a glance', fontsize=30)

# Creating the 5 plots
axs[0].plot(data_France, color='blue', label='France')
axs[1].plot(data_Germany, color='darkgreen', label='Germany')
axs[2].plot(data_Italy, color='cyan', label='Italy')
axs[3].plot(data_Spain, color='red', label='Spain')
axs[4].plot(data_UK, color='orange', label='UK')

# Assigning a size to the figure
plt.gcf().set_size_inches(20, 10)

# Showing the plot
plt.show()
```



As an example, let us zoom in one of them, for example the one for France.

In [13]:
```
# Setting the size of the figure
plt.figure(figsize=(15,5))

# Giving a title
plt.title('Sales of electronic products in France\n(from 01-2015 to 02-2020)',
fontsize = 15, pad = 20)

# Plotting the data for France, to check that everything worked properly
plt.plot(data_France)
```

Out[13]:
```
[<matplotlib.lines.Line2D at 0x2477424a668>]
```



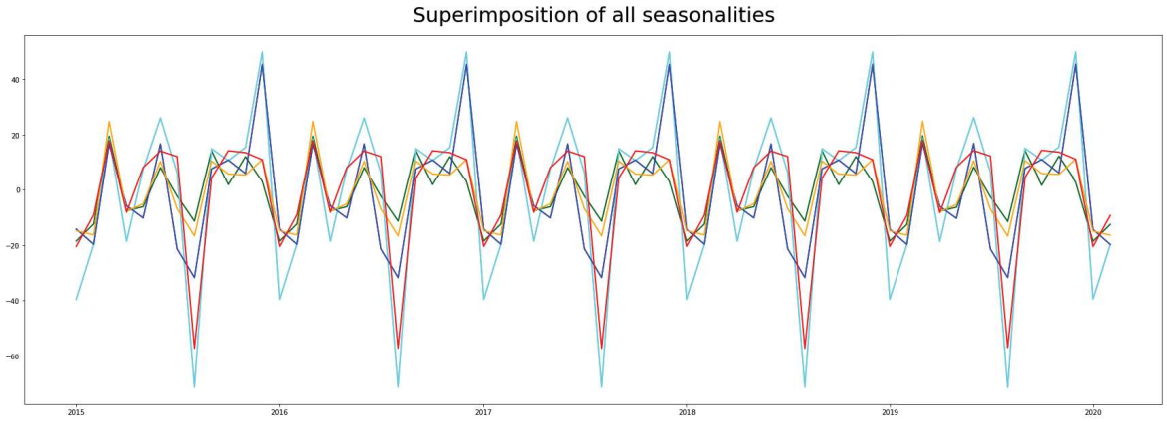
3. Modelling

3.1 Merging seasonal trends

In this section, I will analyze the seasonal trends in all countries using the `seasonal_decomposition()` function. From the visualizations in the previous section it seems to me that the time series don't vary widely over time, therefore I will choose a additive model for my decompositions.

Next, I will superimpose the seasonal trends, to validate or reject my initial hypothesis.

In [14]:
```
# Create objects that store the decomposition of each dataset
decomposition_Germany = seasonal_decompose(data_Germany, model = 'additive')
decomposition_Spain = seasonal_decompose(data_Spain, model = 'additive')
decomposition_France = seasonal_decompose(data_France, model = 'additive')
decomposition_Italy = seasonal_decompose(data_Italy, model = 'additive')
decomposition_UK = seasonal_decompose(data_UK, model = 'additive')

# Creating a plot
plt.plot(decomposition_Germany.seasonal, color='darkgreen', linewidth=2, label
='Germany')
plt.plot(decomposition_Italy.seasonal, color='cyan', linewidth=2, label='Ital
y')
plt.plot(decomposition_France.seasonal, color='blue', linewidth=2, label='Fran
ce')
plt.plot(decomposition_UK.seasonal, color='orange', linewidth=2, label='UK')
plt.plot(decomposition_Spain.seasonal, color='red', linewidth=2, label='Spain'
)

# Giving a title
plt.title('Superimposition of all seasonalities', fontsize=30, pad=20)

# Assigning a size to the figure
plt.gcf().set_size_inches(30, 10)

# Showing the plot
plt.show()
```



From this graph we can appreciate that local variations seem to follow a pretty similar pattern throughout all the countries involved, which allows me to treat the seasonality information as a whole new dataset. Therefore, I can accept my initial hypothesis.

At this point, I can explore the seasonality for the whole family of products as a single entity. In order to do so, I will create a single time series with the lowest and highest observations among all countries. This will give me an overall picture that would allow me to inform the decision makers better.

In [15]:
```
# First I create a list with all the seasonality dataframes
list_of_seasonalities = [decomposition_Germany.seasonal,
                         decomposition_Italy.seasonal,
                         decomposition_France.seasonal,
                         decomposition_UK.seasonal,
                         decomposition_Spain.seasonal]

# I merge all dataframes together with a lambda function on the command reduce
seasonality_merged = reduce(lambda  left,right: pd.merge(left,right,on=['TIME'
],
                                                 how='outer'), list_of_seasonalitie
s)

# I create a lambda function to extrapolate the absolute min and max value fro
m each row
maxCol=lambda x: max(x.min(), x.max(), key=abs)

# Then a new column that shows these values is added to the dataframe
seasonality_merged['Min_max'] = seasonality_merged.apply(maxCol,axis=1)

# Only the Min_max column is of our interest, therefore I will only leave that
seasonality_merged = pd.DataFrame(seasonality_merged['Min_max'])

# Now I can visualize the new dataframe
seasonality_merged.head()
```

Out[15]:

TIME	Min_max
2015-01-01	-39.561736
2015-02-01	-19.732431
2015-03-01	24.782326
2015-04-01	-18.613819
2015-05-01	-10.233472

To make the visualization and the analysis easier and more clear, I will normalize the series obtained, so that I can establish a clear cut-line to draw conclusions. To normalize, I choose to substract from each observation the **mean** of all observations (this will bring all data closer to zero), then I divide each observation by its **standard deviation**.

In [16]:
```
# Normalizing the dataset
seasonality_merged_normalized = (seasonality_merged-seasonality_merged.mean())
/seasonality_merged.std()

# Checking implementation
seasonality_merged_normalized.head()
```

Out[16]:

TIME	Min_max
2015-01-01	-1.111704
2015-02-01	-0.494808
2015-03-01	0.890060
2015-04-01	-0.460008
2015-05-01	-0.199293

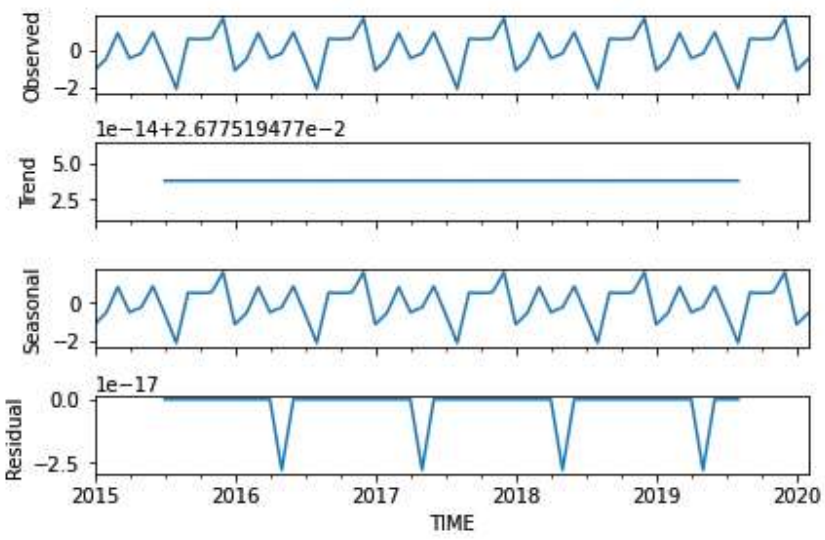
3.2 Checking if the series is white noise

Before going any farther into my analysis, I want to be sure that this series is not white noise, meaning that it is actually helpful to analyze it and it is not just random numbers being thrown at me. A white noise time series would have 3 characteristics:

- A **mean** equal to 0
- A **standard deviation** constant with time
- A **correlation** between lags equalling 0

I can check the first 2 assumptions decomposing `seasonality_merged_normalized` and observing the residuals.

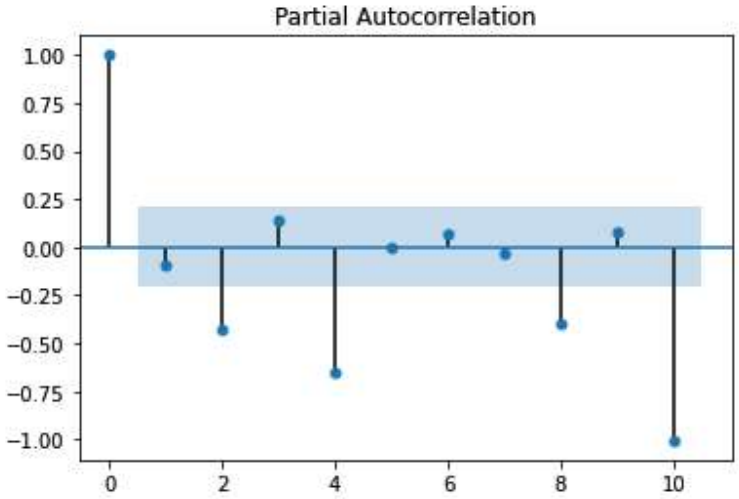
In [17]:
```
# Creating a decomposition
seasonality_merged_normalized_decomposition = seasonal_decompose(seasonality_merged_normalized, model='additive')

# Visualizing it
seasonality_merged_normalized_decomposition.plot();
```



Even without more exploration, it is clearly visible that the **mean is not equal to 0** and that the **standard deviation is not constant** with time - there are periods of no volatility and some with volatility.

Finally, there seems to be correlation between data points at different lags, but in order to check this more in details, I will also plot the **partial autocorrelation function**.

In [18]:
```
# Plotting the autocorrelation function for the first 10 lags
plot_pacf(seasonality_merged_normalized, lags=10, alpha=0.10);
```



In a white noise time series, we would have expected all (or most) lags to be close to zero, whereas this plot clearly shows several correlated data points.

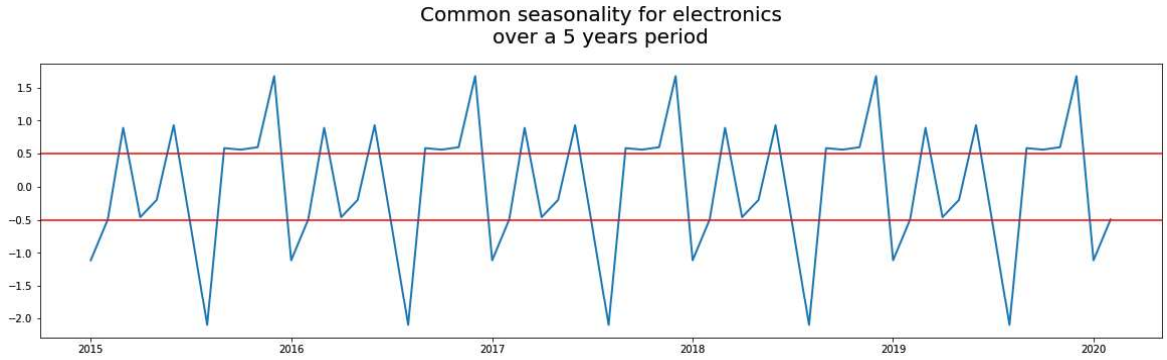
I can conclude that this time series is not white noise and it therefore bears meaning for my investigation.

3.3. Investigating common seasonality

With the data obtained and checked, I can plot a new seasonality graph for the whole family of products.

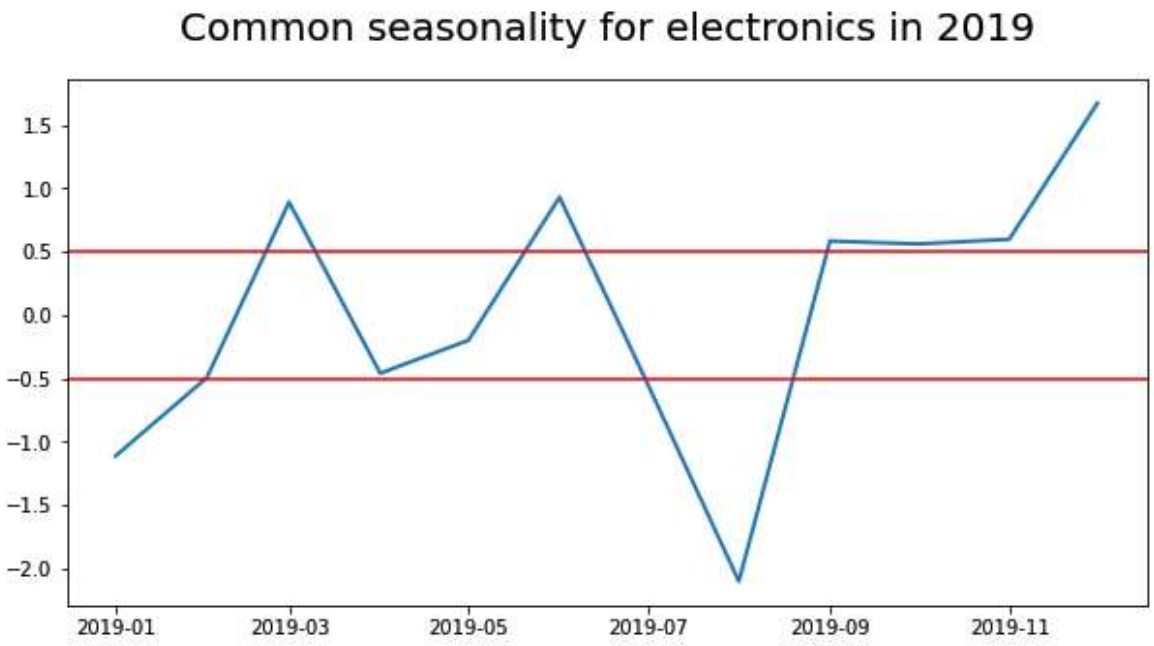
This graph will also contain two red lines, each marking a +/-0.5 line in the graph. This means that I will consider as a positive/negative spikes in demand anything that is above or below 50% of the average.

In [19]:
```
# Creating a plot
plt.plot(seasonality_merged_normalized, linewidth=2)

# Creating two line for +/-0.5
plt.axhline(y=0.5, color='r', linestyle='-')
plt.axhline(y=-0.5, color='r', linestyle='-')

# Assigning a title
plt.title('Common seasonality for electronics\nover a 5 years period', fontsiz
e=20, pad=20)

# Assigning a size to the figure
plt.gcf().set_size_inches(20, 5)

# Showing the plot
plt.show()
```



It is very clear that each year in the past 5 has the same up's and down's in seasonality. Lastly, I will now zoom only in year 2019 as an example and draw some more specific and actionable conclusions.

In [20]:
```
# Creating a plot
plt.plot(seasonality_merged_normalized['2019'], linewidth=2)

# Creating two line for +/-0.5
plt.axhline(y=0.5, color='r', linestyle='-')
plt.axhline(y=-0.5, color='r', linestyle='-')

# Assigning a title
plt.title('Common seasonality for electronics in 2019', fontsize=20, pad=20)

# Assigning a size to the figure
plt.gcf().set_size_inches(10, 5)

# Showing the plot
plt.show()
```



3.4 Conclusions

In the product family of **electronics**, I will suggest to the team the following periods as particularly important for enrichment and amendment. These suggestions take into account a window of 3-5 days on average that it takes for an amendment to be performed in our CMS and to go live in the website (the 5 days mark is for amendments and enrichment that involve a massive number of products, to be pushed for republications to the web).

- **14-28th of February** (in preparation for the peak in March)
- **15-30th of May** (in prepartion for the peak in June)
- **15-30th of August** (in preparation for a peak/plateau in September)
- **15-30th of November** (in preparation for the peak in December)

These data suggest that the dates nicely align with a 8-9 weeks average gap in the demand for electronics that can be used to concentrate on other families of products. Subsequent similar investigation could lay out a schedule for the whole year that can be presented to the Content, Localization and Management team.

4. Communicating result and operationalize

Lastly, the results obtained will be communicated with the Content team and a list of products should be prepared and circulated. If listing all products would be not very productive and would take way too long time and effort, I would also suggest to create a list of families of products, which is exactly what I plan to do here.

This list is by no means exhaustive, but it should be tailored by Business Users that can identify the products with more granularity.

As we said at the beginning, the data explored are for manufactured products in 3 sections: computers, electronics and optical products. Here there are families of products divided in 3 sections.

Computers

- Desktop and laptop computers
- Main frame computers
- Hand-held computers (e.g. PDA)
- Magnetic disk drives, flash drives and other storage devices
- Optical (e.g. CD-RW, CD-ROM, DVD-ROM, DVD-RW) disk drives
- Printers
- Monitors
- Keyboards
- All types of mice, joysticks, and trackball accessories
- Dedicated computer terminals
- Computer servers
- Scanners, including bar code scanners
- Smart card readers
- Virtual reality helmets
- Computer projectors (video beamers)

Electronics

- Capacitors
- Resistors
- Microprocessors
- Electron tubes
- Electronic connectors
- Bare printed circuit boards
- Integrated circuits (analogue, digital or hybrid)
- Diodes, transistors, and related discrete devices
- Inductors (e.g. chokes, coils, transformers)
- Solenoids, switches, and transducers for electronic applications
- Semiconductor, finished or semi-finished
- Display components (plasma, polymer, LCD)
- Light emitting diodes (LED)

Optical products

- Optical instruments and lenses, such as binoculars, microscopes, telescopes, prisms, and lenses (except ophthalmic)
- Products for coating or polishing of lenses (except ophthalmic) and the mounting of lenses (except ophthalmic)
- Optical mirrors
- Optical gun sighting equipment
- Optical positioning equipment
- Optical magnifying instruments
- Optical machinist's precision tools
- Optical comparators
- Optical measuring and checking devices and instruments (e.g. fire control equipment)
- Laser assemblies

This concludes my analysis.