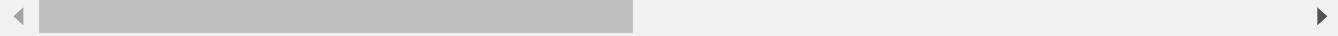


▼ Importação das bibliotecas para uso

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller, kpss
from statsmodels.tsa.seasonal import seasonal_decompose
import statsmodels.graphics.tsaplots as sgt
from statsmodels.tsa.statespace.sarimax import SARIMAX
import statsmodels.tsa.stattools as sts
import seaborn as sns
sns.set()
import matplotlib.pyplot as plt
%matplotlib notebook
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")

/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning:
    import pandas.util.testing as tm
```



▼ Base de Dados extraída e tratada previamente

Base de dados Despesas da União (2014 a 2020)

Tratamento do arquivo para abordagem

fonte: <http://www.portaldatransparencia.gov.br/despesas>

```
despesa_uniao = pd.read_excel('despesas_2014a20.xls', parse_dates=True)
```

```
despesa_uniao.head()
```

periodo	orgao	entidade_vinculada_orgao	vl_pago
despesa_uniao.info()			
<class 'pandas.core.frame.DataFrame'>			
RangeIndex: 20551 entries, 0 to 20550			
Data columns (total 4 columns):			
# Column	Non-Null Count	Dtype	
0 periodo	20551 non-null	datetime64[ns]	
1 orgao	20551 non-null	object	
2 entidade_vinculada_orgao	20551 non-null	object	
3 vl_pago	20551 non-null	float64	
dtypes: datetime64[ns](1), float64(1), object(2)			
memory usage: 642.3+ KB			
despesa_uniao.dtypes			
periodo	datetime64[ns]		
orgao	object		
entidade_vinculada_orgao	object		
vl_pago	float64		
dtype: object			
df_desp = despesa_uniao			
df_desp.info()			
<class 'pandas.core.frame.DataFrame'>			
RangeIndex: 20551 entries, 0 to 20550			
Data columns (total 4 columns):			
# Column	Non-Null Count	Dtype	
0 periodo	20551 non-null	datetime64[ns]	
1 orgao	20551 non-null	object	
2 entidade_vinculada_orgao	20551 non-null	object	
3 vl_pago	20551 non-null	float64	
dtypes: datetime64[ns](1), float64(1), object(2)			
memory usage: 642.3+ KB			

Agrupando as "Entidades" por "orgao"/Ministério

```
df_result = df_desp.groupby(['periodo', 'orgao'])['vl_pago'].sum()
```

```
df_result
```

periodo	orgao	
2014-01-01	20000 - Presidência da República	6.51

20113	- Ministério do Planejamento, Desenvolvimento e Gestão	6.11
22000	- Ministério da Agricultura, Pecuária e Abastecimento	6.27
24000	- Ministério da Ciência, Tecnologia, Inovações e Comunicações	7.88
25000	- Ministério da Economia	2.30
2020-12-01	53000 - Ministério do Desenvolvimento Regional	4.03
	54000 - Ministério do Turismo	1.80
	55000 - Ministério da Cidadania	3.45
	63000 - Advocacia-Geral da União	3.85
	81000 - Ministério da Mulher, Família e Direitos Humanos	2.59

Name: vl_pago, Length: 1927, dtype: float64

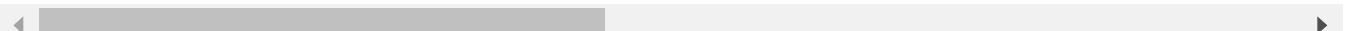


```
type(df_result)  
pandas.core.series.Series
```

```
df_result2 = pd.DataFrame(df_result)
```

```
df_result2.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
MultiIndex: 1927 entries, (Timestamp('2014-01-01 00:00:00'), '20000 - Presidência da Re  
Data columns (total 1 columns):  
 #   Column   Non-Null Count  Dtype    
---  --    
 0   vl_pago  1927 non-null    float64  
dtypes: float64(1)  
memory usage: 19.8+ KB
```



```
df_result2.reset_index()
```

	periodo	orgao	vl_pago
0	2014-01-01	20000 - Presidência da República	6.519071e+07

```
df_result2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 1927 entries, (Timestamp('2014-01-01 00:00:00'), '20000 - Presidência da Re
Data columns (total 1 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   vl_pago   1927 non-null   float64
dtypes: float64(1)
memory usage: 19.8+ KB
```

Transformando os Ministérios em colunas para melhor tratamento

```
df_result3 = df_result2.pivot_table('vl_pago', 'periodo', 'orgao')
```

1927 rows × 3 columns

```
df_result3.head()
```

orgao	20000 - Presidência da República	20113 - Ministério do Planejamento, Desenvolvimento e Gestão	22000 - Ministério da Agricultura, Pecuária e Abastecimento	24000 - Ministério da Ciência, Tecnologia, Inovações e Comunicações	25000 - Ministério da Economia	Mir
-------	----------------------------------	--	---	---	--------------------------------	-----

periodo

2014-01-01	6.519071e+07	6115671.40	6.272410e+08	7.881719e+08	2.307079e+11	4.828
2014-02-01	8.479716e+07	7413.41	7.469514e+08	6.870223e+08	8.926030e+10	7.076
2014-03-01	8.807581e+07	539261.48	7.540673e+08	6.506453e+08	1.554504e+11	6.493
2014-04-01	1.130643e+08	449034.76	7.810936e+08	7.230355e+08	1.923474e+11	9.155
2014-05-01	1.147648e+08	131330.31	8.134798e+08	9.396719e+08	6.895201e+10	9.063

```
df_result3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 84 entries, 2014-01-01 to 2020-12-01
Data columns (total 28 columns):
 #   Column
---  -----
 0   20000 - Presidência da República
 1   20113 - Ministério do Planejamento, Desenvolvimento e Gestão
 2   22000 - Ministério da Agricultura, Pecuária e Abastecimento
 3   24000 - Ministério da Ciência, Tecnologia, Inovações e Comunicações
 4   25000 - Ministério da Economia
 5   26000 - Ministério da Educação
 6   30000 - Ministério da Justiça e Segurança Pública
 7   32000 - Ministério de Minas e Energia
 8   33000 - Ministério da Previdência Social
 9   35000 - Ministério das Relações Exteriores
10   36000 - Ministério da Saúde
11   37000 - Controladoria-Geral da União
12   38000 - Ministério do Trabalho e Emprego
13   39000 - Ministério da Infraestrutura
14   40000 - Ministério do Trabalho
15   41000 - Ministério das Comunicações
16   42000 - Ministério da Cultura
17   44000 - Ministério do Meio Ambiente
18   49000 - Ministério do Desenvolvimento Agrário
19   51000 - Ministério do Esporte
20   52000 - Ministério da Defesa
21   53000 - Ministério do Desenvolvimento Regional
22   54000 - Ministério do Turismo
23   55000 - Ministério da Cidadania
24   57000 - Ministério das Mulheres, Igualdade Racial, da Juventude e dos Direitos Hum
25   58000 - Ministério da Pesca e Aquicultura
26   63000 - Advocacia-Geral da União
27   81000 - Ministério da Mulher, Família e Direitos Humanos
dtypes: float64(28)
memory usage: 19.0 KB
```

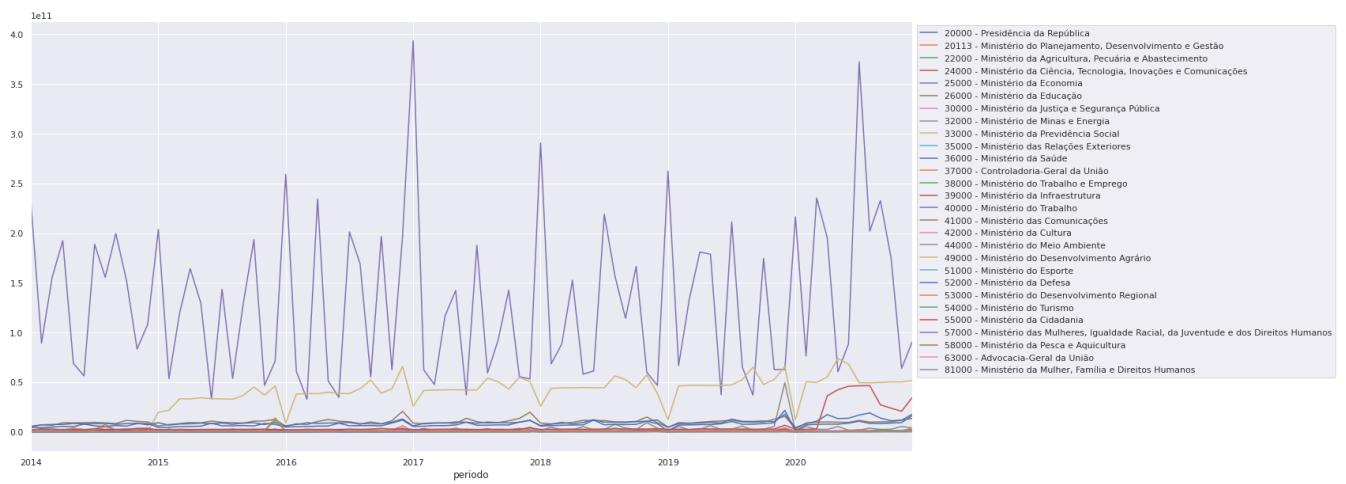


Cria-se lista com todos os Ministérios (coluna "orgao"), facilitando o histórico de Despesas desde 2014

```
gastos = [x for x in df_result3]
```

A plotagem é inicial, apenas para compreender o cenário de gastos por Ministério da União

```
ax = df_result3.plot(figsize = (20,10))
ax.legend(loc=0, bbox_to_anchor=(1.0,1.0));
```



É latente observar que o Ministério da Economia é o órgão que mais "gasta" dentro do orçamento da União. A partir disso, vamos listar os Ministérios que mais geram despesas para a União por ranking, facilitando algumas observações futuras

```
total = {}

for i in df_result3:
    total[i] = df_result3[i].sum()

total

{'20000 - Presidência da República': 12443979478.76,
 '20113 - Ministério do Planejamento, Desenvolvimento e Gestão': 61424735.379999995,
 '22000 - Ministério da Agricultura, Pecuária e Abastecimento': 103959278832.75,
 '24000 - Ministério da Ciência, Tecnologia, Inovações e Comunicações': 60327380333.450
 '25000 - Ministério da Economia': 10929667012389.3,
 '26000 - Ministério da Educação': 831551580398.22,
 '30000 - Ministério da Justiça e Segurança Pública': 83861908140.51,
 '32000 - Ministério de Minas e Energia': 280308751114.8,
 '33000 - Ministério da Previdência Social': 3149112994569.6396,
 '35000 - Ministério das Relações Exteriores': 23694826964.620003,
 '36000 - Ministério da Saúde': 803468112167.6101,
 '37000 - Controladoria-Geral da União': 6387952507.239999,
 '38000 - Ministério do Trabalho e Emprego': 21256005350.36,
 '39000 - Ministério da Infraestrutura': 100227376618.17003,
 '40000 - Ministério do Trabalho': 4732231.68,
 '41000 - Ministério das Comunicações': 5795206337.89,
 '42000 - Ministério da Cultura': 2504113.5,
 '44000 - Ministério do Meio Ambiente': 15876208506.359999,
 '49000 - Ministério do Desenvolvimento Agrário': 95125633.77000001,
 '51000 - Ministério do Esporte': 83784547.25,
 '52000 - Ministério da Defesa': 612883489898.2599,
 '53000 - Ministério do Desenvolvimento Regional': 162800263580.89996,
```

```
'54000 - Ministério do Turismo': 13816169656.53,  
'55000 - Ministério da Cidadania': 525008701345.55994,  
'57000 - Ministério das Mulheres, Igualdade Racial, da Juventude e dos Direitos Humanos': 219986359.62,  
'58000 - Ministério da Pesca e Aquicultura': 23359783516.51,  
'63000 - Advocacia-Geral da União': 23359783516.51,  
'81000 - Ministério da Mulher, Família e Direitos Humanos': 1028048939.0100001}
```

◀ ▶

```
total_pd = pd.DataFrame.from_dict(total, orient='index', columns=[ 'TOTAL' ])
```

```
total_pd.sort_values(by='TOTAL', ascending=False).head(6)
```

TOTAL	
25000 - Ministério da Economia	1.092967e+13
33000 - Ministério da Previdência Social	3.149113e+12
26000 - Ministério da Educação	8.315516e+11
36000 - Ministério da Saúde	8.034681e+11
52000 - Ministério da Defesa	6.128835e+11
55000 - Ministério da Cidadania	5.250087e+11

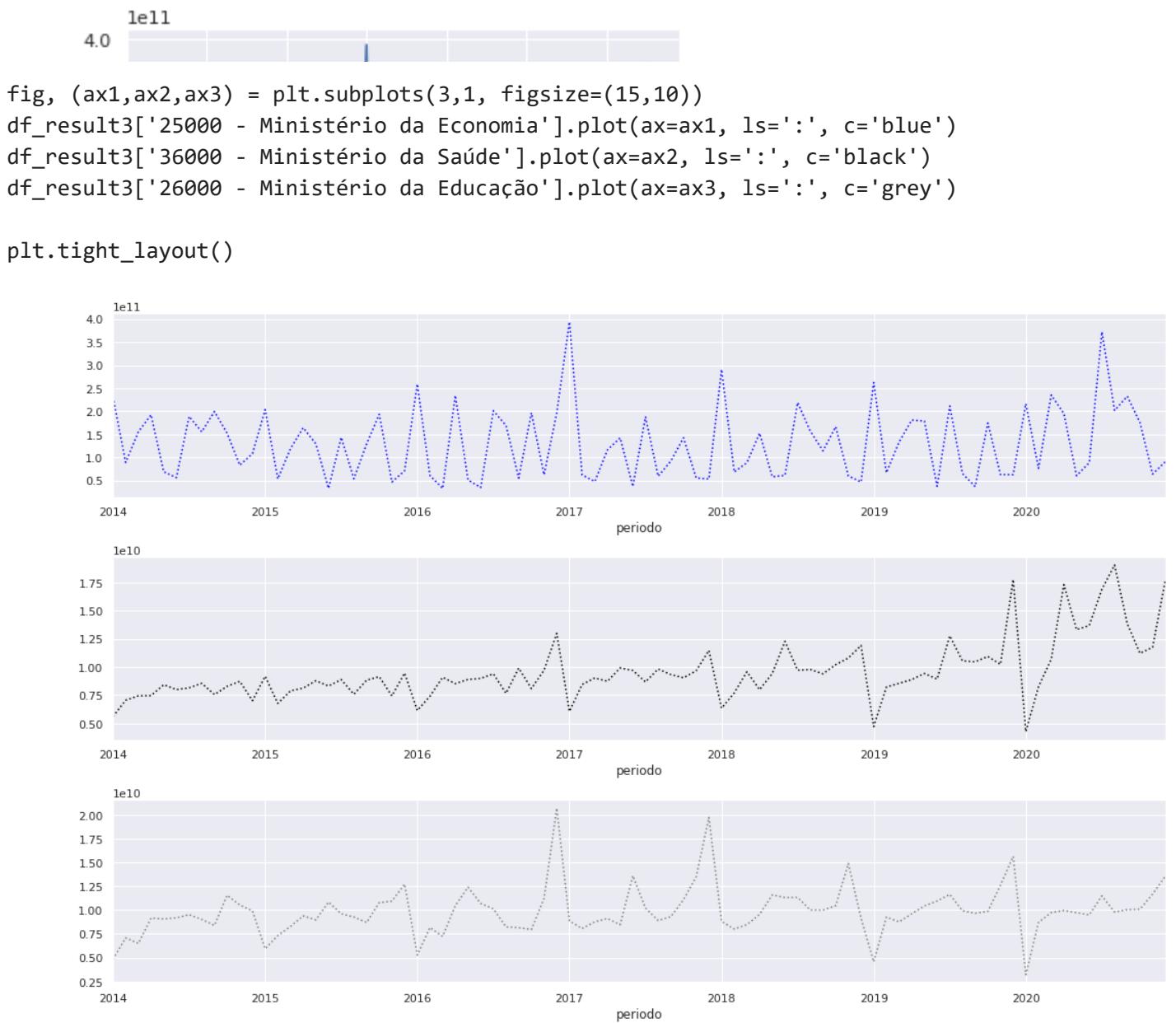
Para fins de estudo nesse Trabalho de Conclusão, optamos por analisar 3 Ministérios de suma importância para o Brasil e que, pelo ranking demonstrado, estão no top 5 de despesas para a União:

Ministério da Economia

Ministério da Saúde

Ministério da Educação

```
df_result3['25000 - Ministério da Economia'].plot();
```



Foi possível observar uma ligeira tendência de aumento/crescimento nas despesas desses Ministérios. Além disso, há sazonalidade nos gastos. Essas informações ficarão mais claras

adiante. Faz-se necessário observar um descolamento das despesas no ano de 2020, em decorrência da situação pandemica do Coronavírus

```
bkp_gastos = df_result3.copy()
```

```
bkp_gastos = bkp_gastos/1000000          # transforma em milhões de R$  
bkp_gastos.index.freq = 'MS'             # month start frequency - frequência mensal
```

#fonte https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#offset-aliases

```
bkp_gastos.head()
```

orgao	20000 - Presidência da República	20113 - Ministério do Planejamento, Desenvolvimento e Gestão	22000 - Ministério da Agricultura, Pecuária e Abastecimento	24000 - Ministério da Ciência, Tecnologia, Inovações e Comunicações	25000 - Ministério da Economia	Min:	E:
-------	----------------------------------	--	---	---	--------------------------------	------	----

periodo

2014-01-01	65.190711	6.115671	627.240972	788.171858	230707.875070	4828	
2014-02-01	84.797157	0.007413	746.951356	687.022323	89260.298717	7076	
2014-03-01	88.075813	0.539261	754.067281	650.645312	155450.420693	6493	
2014-04-01	113.064284	0.449035	781.093551	723.035499	192347.415307	9155	
2014-05-01	114.764818	0.131330	813.479756	939.671941	68952.008458	9063	

```
bkp_gastos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 84 entries, 2014-01-01 to 2020-12-01
Freq: MS
Data columns (total 28 columns):
 #   Column
--- 
 0   20000 - Presidência da República
 1   20113 - Ministério do Planejamento, Desenvolvimento e Gestão
 2   22000 - Ministério da Agricultura, Pecuária e Abastecimento
```

```
3 24000 - Ministério da Ciência, Tecnologia, Inovações e Comunicações
4 25000 - Ministério da Economia
5 26000 - Ministério da Educação
6 30000 - Ministério da Justiça e Segurança Pública
7 32000 - Ministério de Minas e Energia
8 33000 - Ministério da Previdência Social
9 35000 - Ministério das Relações Exteriores
10 36000 - Ministério da Saúde
11 37000 - Controladoria-Geral da União
12 38000 - Ministério do Trabalho e Emprego
13 39000 - Ministério da Infraestrutura
14 40000 - Ministério do Trabalho
15 41000 - Ministério das Comunicações
16 42000 - Ministério da Cultura
17 44000 - Ministério do Meio Ambiente
18 49000 - Ministério do Desenvolvimento Agrário
19 51000 - Ministério do Esporte
20 52000 - Ministério da Defesa
21 53000 - Ministério do Desenvolvimento Regional
22 54000 - Ministério do Turismo
23 55000 - Ministério da Cidadania
24 57000 - Ministério das Mulheres, Igualdade Racial, da Juventude e dos Direitos Hum
25 58000 - Ministério da Pesca e Aquicultura
26 63000 - Advocacia-Geral da União
27 81000 - Ministério da Mulher, Família e Direitos Humanos
dtypes: float64(28)
memory usage: 19.0 KB
```

bkp_gastos.index

```
DatetimeIndex(['2014-01-01', '2014-02-01', '2014-03-01', '2014-04-01',
                 '2014-05-01', '2014-06-01', '2014-07-01', '2014-08-01',
                 '2014-09-01', '2014-10-01', '2014-11-01', '2014-12-01',
                 '2015-01-01', '2015-02-01', '2015-03-01', '2015-04-01',
                 '2015-05-01', '2015-06-01', '2015-07-01', '2015-08-01',
                 '2015-09-01', '2015-10-01', '2015-11-01', '2015-12-01',
                 '2016-01-01', '2016-02-01', '2016-03-01', '2016-04-01',
                 '2016-05-01', '2016-06-01', '2016-07-01', '2016-08-01',
                 '2016-09-01', '2016-10-01', '2016-11-01', '2016-12-01',
                 '2017-01-01', '2017-02-01', '2017-03-01', '2017-04-01',
                 '2017-05-01', '2017-06-01', '2017-07-01', '2017-08-01',
                 '2017-09-01', '2017-10-01', '2017-11-01', '2017-12-01',
                 '2018-01-01', '2018-02-01', '2018-03-01', '2018-04-01',
                 '2018-05-01', '2018-06-01', '2018-07-01', '2018-08-01',
                 '2018-09-01', '2018-10-01', '2018-11-01', '2018-12-01',
                 '2019-01-01', '2019-02-01', '2019-03-01', '2019-04-01',
                 '2019-05-01', '2019-06-01', '2019-07-01', '2019-08-01',
                 '2019-09-01', '2019-10-01', '2019-11-01', '2019-12-01',
                 '2020-01-01', '2020-02-01', '2020-03-01', '2020-04-01',
                 '2020-05-01', '2020-06-01', '2020-07-01', '2020-08-01',
                 '2020-09-01', '2020-10-01', '2020-11-01', '2020-12-01'],
                dtype='datetime64[ns]', name='periodo', freq='MS')
```

bkp_gastos.tail()

orgao	20000 - Presidência da República	20113 - Ministério do Planejamento, Desenvolvimento e Gestão	22000 - Ministério da Agricultura, Pecuária e Abastecimento	24000 - Ministério da Ciência, Tecnologia, Inovações e Comunicações	25000 - Ministério da Economia	Mi
-------	----------------------------------	--	---	---	--------------------------------	----

periodo

2020-08-01	122.175583	NaN	1248.361933	711.402219	201871.600921	978
2020-09-01	130.013960	NaN	2097.552637	863.959599	232582.689376	1004
2020-10-01	125.132673	NaN	1352.127887	738.107874	175522.286653	1009
2020-11-01	160.358941	NaN	1490.222018	1094.729006	63872.607726	1167
2020-12-01	248.431446	NaN	2362.797897	1897.914302	90715.459922	1356

```
bkp_gastos.describe()
```

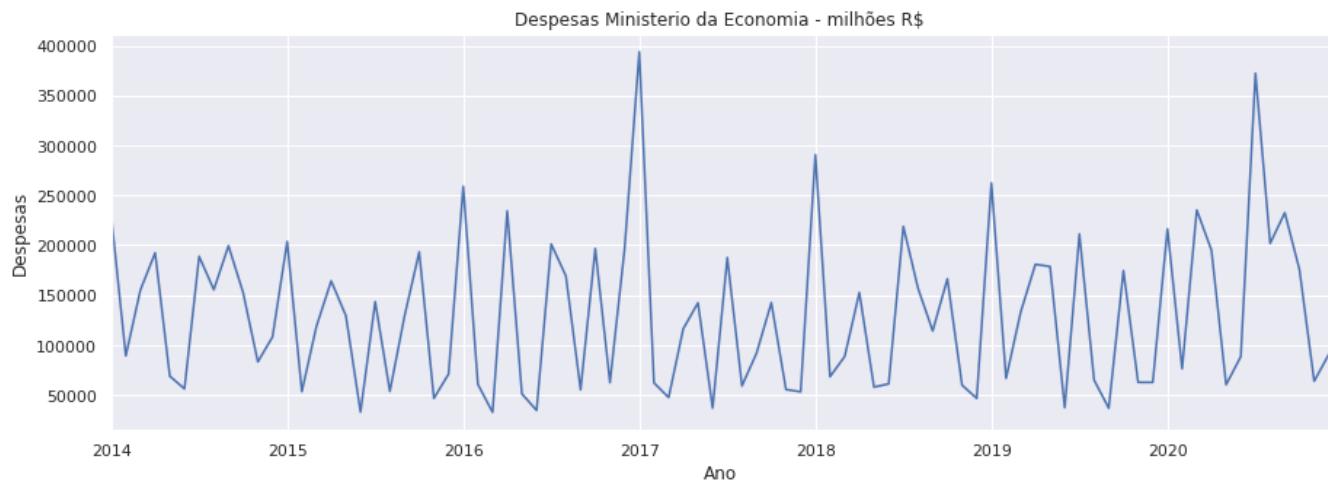
```
bkp_gastos.isna().sum()
```

```
orgao
20000 - Presidência da República
20113 - Ministério do Planejamento, Desenvolvimento e Gestão
22000 - Ministério da Agricultura, Pecuária e Abastecimento
24000 - Ministério da Ciência, Tecnologia, Inovações e Comunicações
25000 - Ministério da Economia
26000 - Ministério da Educação
30000 - Ministério da Justiça e Segurança Pública
32000 - Ministério de Minas e Energia
33000 - Ministério da Previdência Social
35000 - Ministério das Relações Exteriores
36000 - Ministério da Saúde
37000 - Controladoria-Geral da União
38000 - Ministério do Trabalho e Emprego
39000 - Ministério da Infraestrutura
40000 - Ministério do Trabalho
41000 - Ministério das Comunicações
42000 - Ministério da Cultura
44000 - Ministério do Meio Ambiente
49000 - Ministério do Desenvolvimento Agrário
51000 - Ministério do Esporte
52000 - Ministério da Defesa
53000 - Ministério do Desenvolvimento Regional
54000 - Ministério do Turismo
55000 - Ministério da Cidadania
57000 - Ministério das Mulheres, Igualdade Racial, da Juventude e dos Direitos Humanos
58000 - Ministério da Pesca e Aquicultura
63000 - Advocacia-Geral da União
81000 - Ministério da Mulher, Família e Direitos Humanos
dtype: int64
```

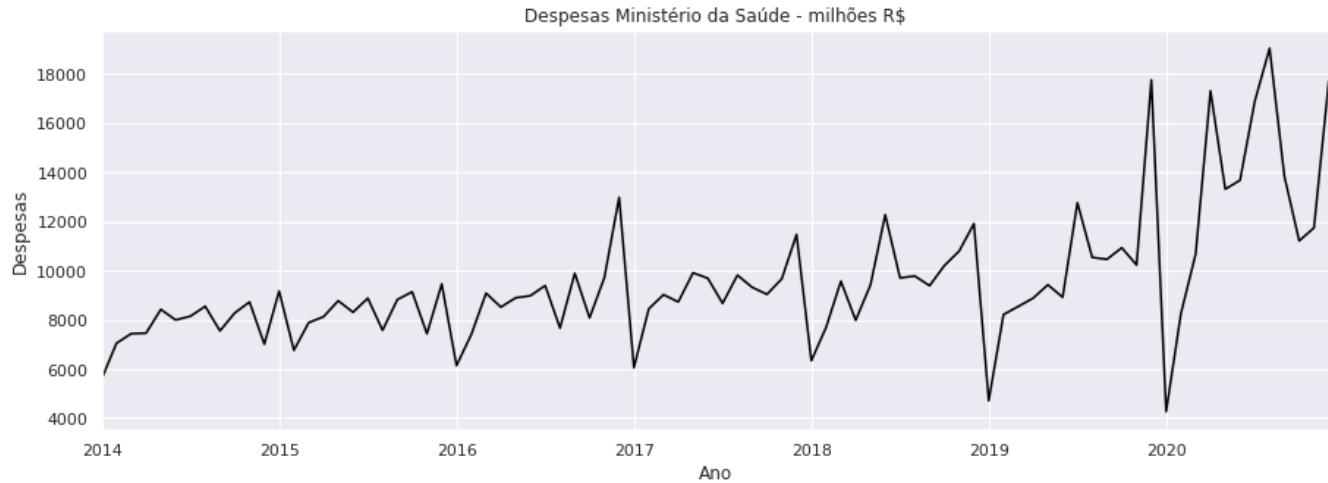
Nota-se que os Ministérios da Economia, Saúde e Educação não possuem valores NaN em sua base

▼ Plot dos dados

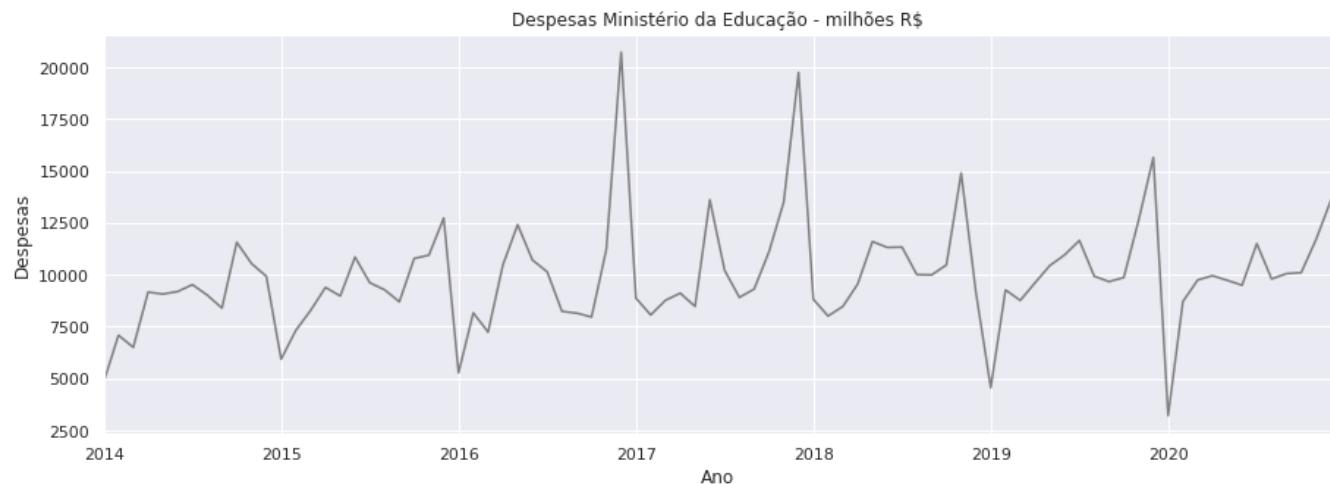
```
title='Despesas Ministerio da Economia - milhões R$'
ylabel='Despesas'
xlabel='Ano'
ax = bkp_gastos['25000 - Ministério da Economia'].plot(figsize=(15,5),title=title);
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel);
```



```
title='Despesas Ministério da Saúde - milhões R$'
ylabel='Despesas'
xlabel='Ano'
ax = bkp_gastos['36000 - Ministério da Saúde'].plot(figsize=(15,5),title=title, c='black');
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel);
```



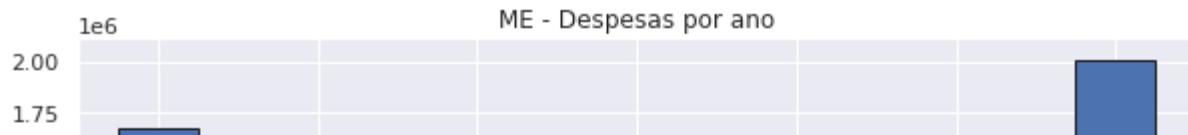
```
title='Despesas Ministério da Educação - milhões R$'
ylabel='Despesas'
xlabel='Ano'
ax = bkp_gastos['26000 - Ministério da Educação'].plot(figsize=(15,5),title=title, c='grey');
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel);
```



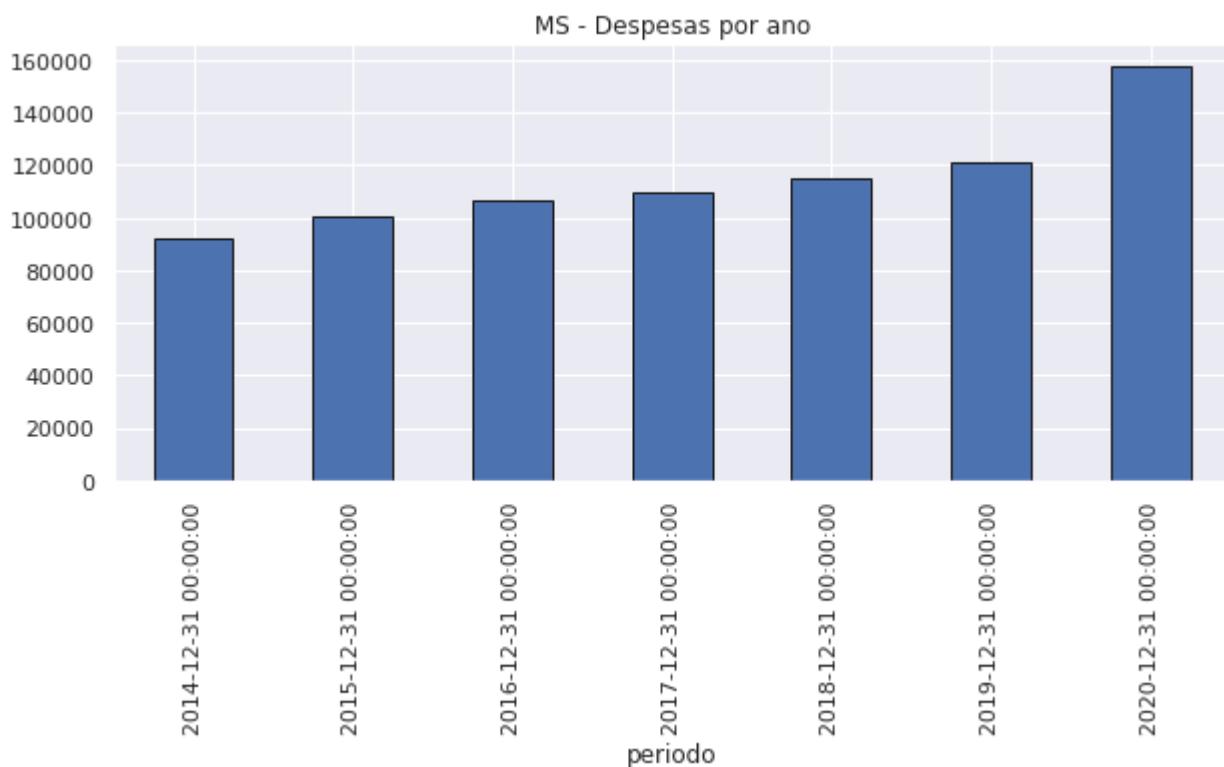
O gráfico das despesas do Ministério da Economia é bem diferente dos outros listados. Por ser o Ministério que gerencia e dita as regras de gastos da União, talvez esse seja um dos motivos de suas despesas ser tão peculiar e de grande volume. Os Ministérios da Saúde e Educação possuem tendência e sazonalidade semelhantes.

Interessante notar nesse gráfico o aumento de despesas para o ano de 2020 no Ministério da Saúde, com redução da mesma no Ministério da Educação, em decorrência da pandemia do Coronavírus

```
# Variação por ano
bkp_gastos['25000 - Ministério da Economia'].resample('A').sum().plot.bar(figsize = (10,4),
```



```
bkp_gastos['36000 - Ministério da Saúde'].resample('A').sum().plot.bar(figsize = (10,4), grid=True)
```



```
bkp_gastos['26000 - Ministério da Educação'].resample('A').sum().plot.bar(figsize = (10,4), grid=True)
```

MEd - Despesas por ano

Foi possível observar de forma clara o aumento de despesas para o ano de 2020 nos Ministérios da Economia e Saúde, e a consequente redução de gastos no Ministério da Educação



▼ STATSMODEL para obter tendência



Utilizamos o filtro Hodrick-Prescott

$$yt = \tau t + ct$$

onde:

yt = série temporal

τt = componente de tendência

ct = componente cíclica

lambda (lamb) = 129600 indicando mês

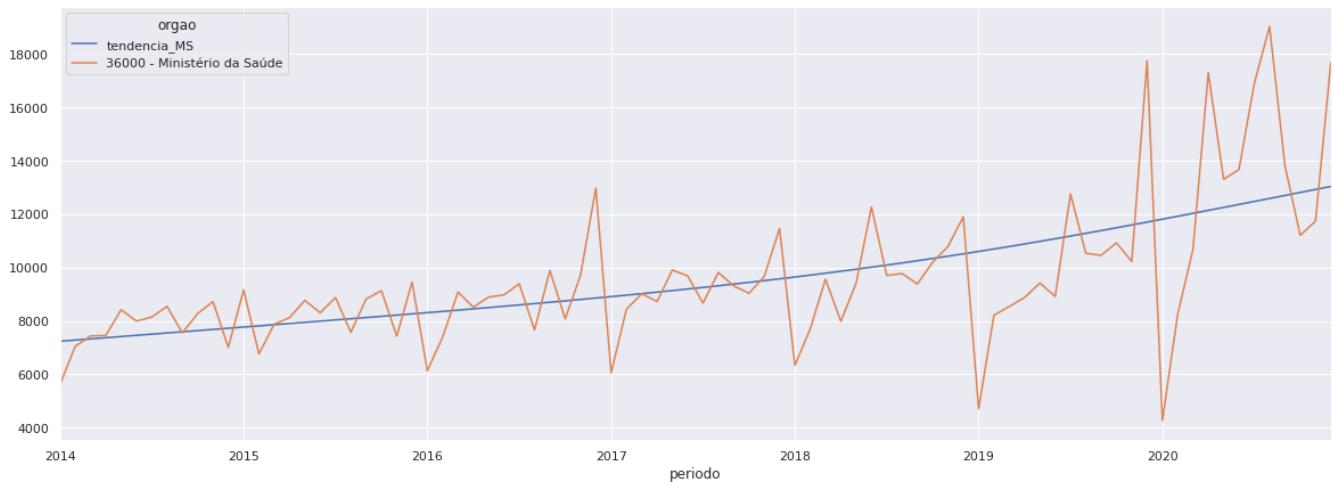
fonte: https://www.statsmodels.org/stable/generated/statsmodels.tsa.filters.hp_filter.html

```
from statsmodels.tsa.filters.hp_filter import hpfilter
# Separando as variáveis
gastos_cycle, gastos_trend = hpfilter(bkp_gastos['25000 - Ministério da Economia'], lamb=1296)

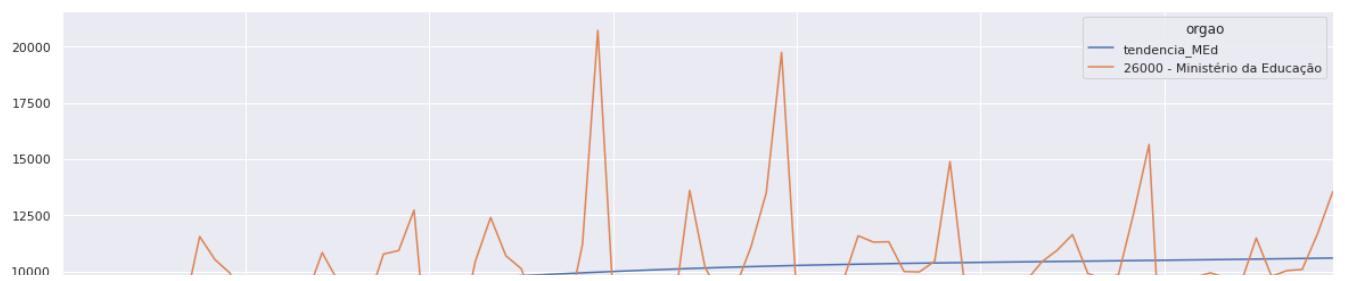
bkp_gastos['tendencia_ME'] = gastos_trend
bkp_gastos[['tendencia_ME','25000 - Ministério da Economia']].plot(figsize = (20,7)).autoscale()
```



```
gastos_cycle, gastos_trend = hpfilter(bkp_gastos['36000 - Ministério da Saúde'], lamb=129600)
bkp_gastos['tendencia_MS'] = gastos_trend
bkp_gastos[['tendencia_MS','36000 - Ministério da Saúde']].plot(figsize = (20,7)).autoscale(zero=True)
```



```
gastos_cycle, gastos_trend = hpfilter(bkp_gastos['26000 - Ministério da Educação'], lamb=129600)
bkp_gastos['tendencia_MEd'] = gastos_trend
bkp_gastos[['tendencia_MEd','26000 - Ministério da Educação']].plot(figsize = (20,7)).autoscale(zero=True)
```



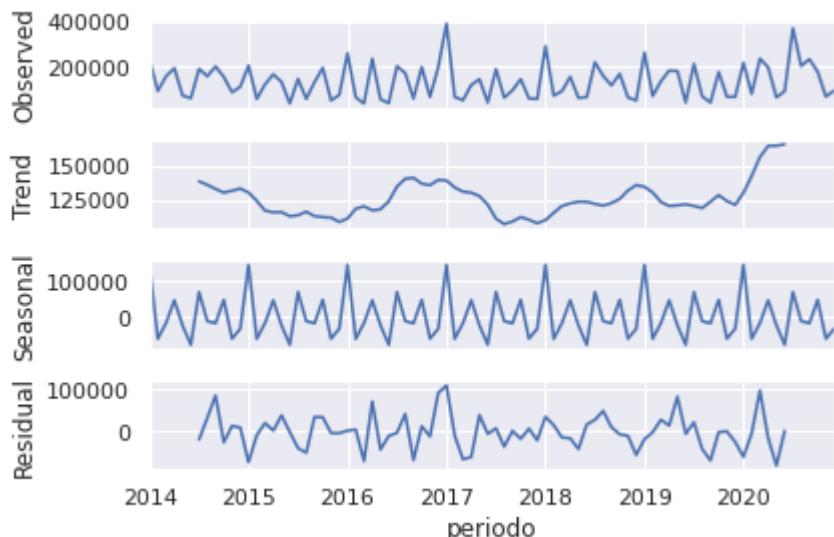
▼ ERROR / TREND / SEASONALITY

ETS

O decompose é uma tarefa estatística que descontroi a série temporal em diversos componentes, cada um representando uma determinada categoria

fonte: https://en.wikipedia.org/wiki/Decomposition_of_time_series

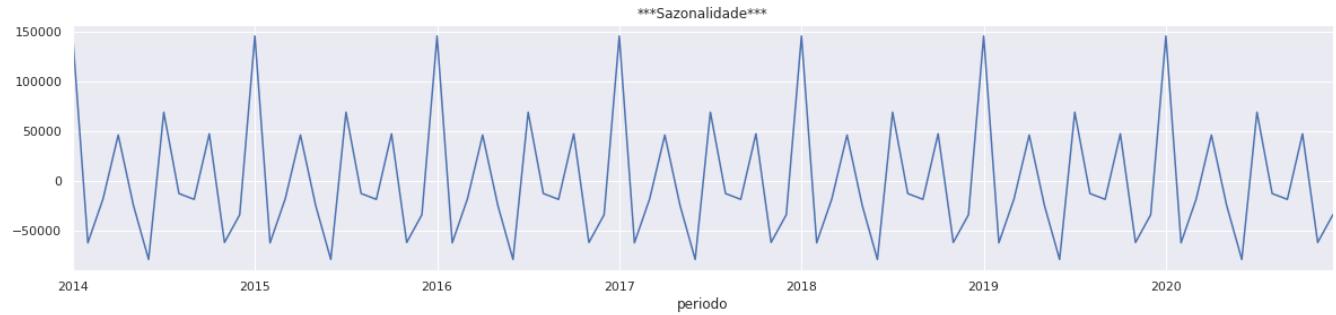
```
from statsmodels.tsa.seasonal import seasonal_decompose  
visao_geral = seasonal_decompose(bkp_gastos['25000 - Ministério da Economia'], model='add')  
visao_geral.plot();
```



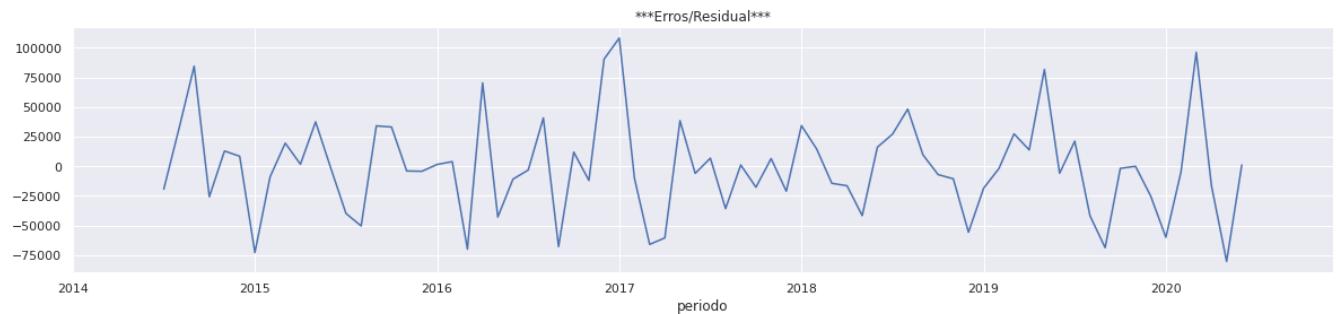
```
visao_geral.trend.plot(figsize=(20,4), title='***Tendência***');
```



```
visao_geral.seasonal.plot(figsize=(20,4), title='***Sazonalidade***');
```



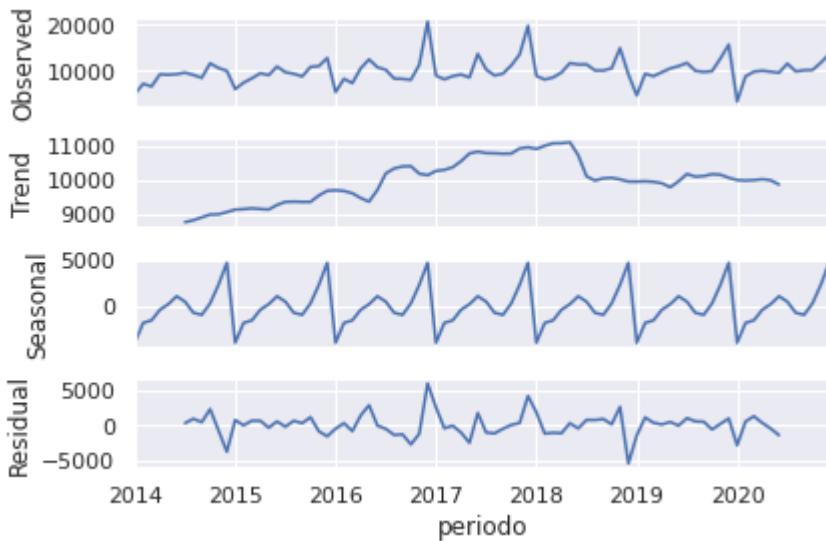
```
visao_geral.resid.plot(figsize=(20,4), title='***Erros/Residual***');
```



```
visao_geral = seasonal_decompose(bkp_gastos['36000 - Ministério da Saúde'], model='add')
visao_geral.plot();
```



```
visao_geral = seasonal_decompose(bkp_gastos['26000 - Ministério da Educação'], model='add')
visao_geral.plot();
```



▼ Teste Estatísticos de Dickey-Fuller e KPSS

O Teste de Dickey-Fuller busca, entre outras coisas, identificar se os dados trabalhados são classificados como estacionários ou não estacionários.

Usamos o valor P base de 5%, ou seja, caso o valor P esteja abaixo desses 5%, significa que a série é estatisticamente estacionária

```
#Teste de Dickey Fuller - ME
from statsmodels.tsa.stattools import adfuller
adfinput = adfuller(bkp_gastos['25000 - Ministério da Economia'])
adftest = pd.Series(adfinput[0:4], index=['Teste Estatistico Dickey Fuller','Valor-P',
                                             'Lags Usados','Número de observações usadas'])
adftest = round(adftest,4)

for key, value in adfinput[4].items():
    adftest["Valores Críticos (%s)"%key] = value.round(4)
if adfinput[1] <= 0.05:
    print("É estacionário")
```

```
else:
    print("É não-estacionária")
```

```
adftest
```

```
É não-estacionária
Teste Estatistico Dickey Fuller      -2.6843
Valor-P                               0.0768
Lags Usados                          12.0000
Número de observações usadas        71.0000
Valores Críticos (1%)                -3.5260
Valores Críticos (5%)                -2.9032
Valores Críticos (10%)               -2.5890
dtype: float64
```

O teste KPSS já pressupõe que a série é estacionária e só não será se o valor P for inferior a 5%, ou o teste estatístico for menor que algum valor crítico escolhido

```
from statsmodels.tsa.stattools import kpss
kpss_input = kpss(bkp_gastos['25000 - Ministério da Economia'])
kpss_test = pd.Series(kpss_input[0:3], index=['Teste Estatistico KPSS', 'Valor-P', 'Lags Usados'])
kpss_test = round(kpss_test,4)

for key, value in kpss_input[3].items():
    kpss_test["Valores Críticos (%s)"%key] = value
kpss_test

Teste Estatistico KPSS      0.1495
Valor-P                     0.1000
Lags Usados                 12.0000
Valores Críticos (10%)     0.3470
Valores Críticos (5%)      0.4630
Valores Críticos (2.5%)    0.5740
Valores Críticos (1%)      0.7390
dtype: float64
```

```
#Teste de Dickey Fuller - MS
```

```
adfinput = adfuller(bkp_gastos['36000 - Ministério da Saúde'])
adftest = pd.Series(adfinput[0:4], index=['Teste Estatistico Dickey Fuller', 'Valor-P',
                                             'Lags Usados', 'Número de observações usadas'])
adftest = round(adftest,4)

for key, value in adfinput[4].items():
    adftest["Valores Críticos (%s)"%key] = value.round(4)
if adfinput[1] <= 0.05:
    print("É estacionário")
else:
    print("É não-estacionária")
```

```
adftest
```

É não-estacionária

Teste Estatistico Dickey Fuller	2.3060
Valor-P	0.9990
Lags Usados	11.0000
Número de observações usadas	72.0000
Valores Críticos (1%)	-3.5246
Valores Críticos (5%)	-2.9026
Valores Críticos (10%)	-2.5887

dtype: float64

```
kpss_input = kpss(bkp_gastos['36000 - Ministério da Saúde'])
kpss_test = pd.Series(kpss_input[0:3], index=['Teste Statistico KPSS', 'Valor-P', 'Lags Usados'])
kpss_test = round(kpss_test,4)
```

```
for key, value in kpss_input[3].items():
    kpss_test["Valores Críticos (%s)"%key] = value
kpss_test
```

Teste Statistico KPSS	0.6311
Valor-P	0.0198
Lags Usados	12.0000
Valores Críticos (10%)	0.3470
Valores Críticos (5%)	0.4630
Valores Críticos (2.5%)	0.5740
Valores Críticos (1%)	0.7390

dtype: float64

```
#Teste de Dickey Fuller - MED
```

```
adfinput = adfuller(bkp_gastos['26000 - Ministério da Educação'])
adftest = pd.Series(adfinput[0:4], index=['Teste Estatistico Dickey Fuller', 'Valor-P',
                                             'Lags Usados', 'Número de observações usadas'])
adftest = round(adftest,4)
```

```
for key, value in adfinput[4].items():
    adftest["Valores Críticos (%s)"%key] = value.round(4)
if adfinput[1] <= 0.05:
    print("É estacionário")
else:
    print("É não-estacionária")
```

```
adftest
```

É não-estacionária

Teste Estatistico Dickey Fuller	-1.7018
Valor-P	0.4302
Lags Usados	11.0000
Número de observações usadas	72.0000
Valores Críticos (1%)	-3.5246

```
Valores Críticos (5%)           -2.9026
Valores Críticos (10%)          -2.5887
dtype: float64
```

```
kpss_input = kpss(bkp_gastos['26000 - Ministério da Educação'])
kpss_test = pd.Series(kpss_input[0:3], index=['Teste Estatístico KPSS', 'Valor-P', 'Lags Usados']
kpss_test = round(kpss_test,4)

for key, value in kpss_input[3].items():
    kpss_test["Valores Críticos (%s)"%key] = value
kpss_test

Teste Estatístico KPSS      0.4899
Valor-P                      0.0439
Lags Usados                  12.0000
Valores Críticos (10%)     0.3470
Valores Críticos (5%)       0.4630
Valores Críticos (2.5%)     0.5740
Valores Críticos (1%)       0.7390
dtype: float64
```

▼ Transformando dados Não Estacionários em Estacionários

Primeiro método: INFLAÇÃO

Para tentar tornar a série estacionária, colocaremos a série toda com base nos valores atuais pelo IPCA, acumulando do final do período (dez/20) até o início do estudo.

fonte: <https://agenciadenoticias.ibge.gov.br/>

```
infl = pd.read_excel('IPCA_ajustada.xlsx', sheet_name='IPCA', parse_dates=True)
infl['Data'] = pd.to_datetime(infl['Data'])
infl.set_index('Data', inplace=True)
infl.head()
```

```
#Criando colunas de Ano e Mês para mesclar os dados com a base do IPCA
bkp_gastos['ANO'] = bkp_gastos.index.year
bkp_gastos['Mês'] = bkp_gastos.index.month
```

```
#Mesclando
```

```
index = bkp_gastos.index
bkp_gastos = bkp_gastos.merge(infl.loc[:,['ANO','Mês','Acumulado']], how='left', on=['ANO','Mês'])
bkp_gastos['Despesa ME Ajustada'] = bkp_gastos['25000 - Ministério da Economia'] * bkp_gastos['Acumulado']
bkp_gastos['Despesa MS Ajustada'] = bkp_gastos['36000 - Ministério da Saúde'] * bkp_gastos['Acumulado']
bkp_gastos['Despesa MEd Ajustada'] = bkp_gastos['26000 - Ministério da Educação'] * bkp_gastos['Acumulado']
bkp_gastos.set_index(index, inplace=True)
bkp_gastos.head()
```

20000 - Presidência da República	20113 - Ministério do Planejamento, Desenvolvimento e Gestão	22000 - Ministério da Agricultura, Pecuária e Abastecimento	24000 - Ministério da Ciência, Tecnologia, Inovações e Comunicações	25000 - Ministério da Economia	Ministérios
----------------------------------	--	---	---	--------------------------------	-------------

periodo

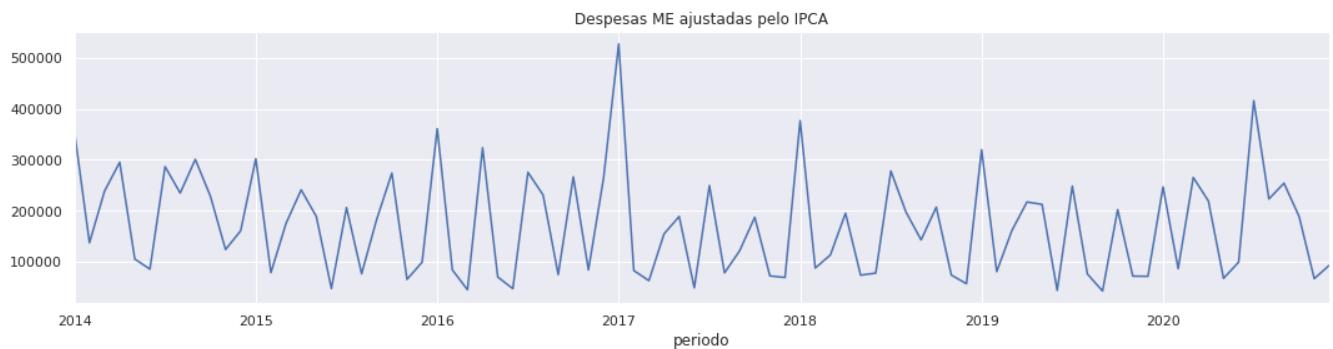
2014-01-01	65.190711	6.115671	627.240972	788.171858	230707.875070	4828
2014-02-01	84.797157	0.007413	746.951356	687.022323	89260.298717	7076
2014-03-01	88.075813	0.539261	754.067281	650.645312	155450.420693	6493
2014-04-01	113.064284	0.449035	781.093551	723.035499	192347.415307	9155
2014-05-01	114.764818	0.131330	813.479756	939.671941	68952.008458	9063

```
#Teste de Dickey Fuller
```

```
def adfuller_test(serie, figsize=(18,4), plot=True, title=""):
    if plot:
        serie.plot(figsize=figsize, title=title)
        plt.show()
#Teste de Dickey Fuller sobre a primeira diferenciação
adf = adfuller(serie)
output = pd.Series(adf[0:4], index=['Teste Estatístico Dickey Fuller','Valor-P',
                                      'Lags Usados','Número de observações usadas'])
output = round(output,4)
```

```
for key, value in adf[4].items():
    output["Valores Críticos (%s)"%key] = value.round(4)
return output
```

```
#adfuller_test(bkp_gastos['Despesa Ajustada'],title='Despesas da União ajustadas pelo IPCA')
adfuller_test(bkp_gastos['Despesa ME Ajustada'],title='Despesas ME ajustadas pelo IPCA')
```

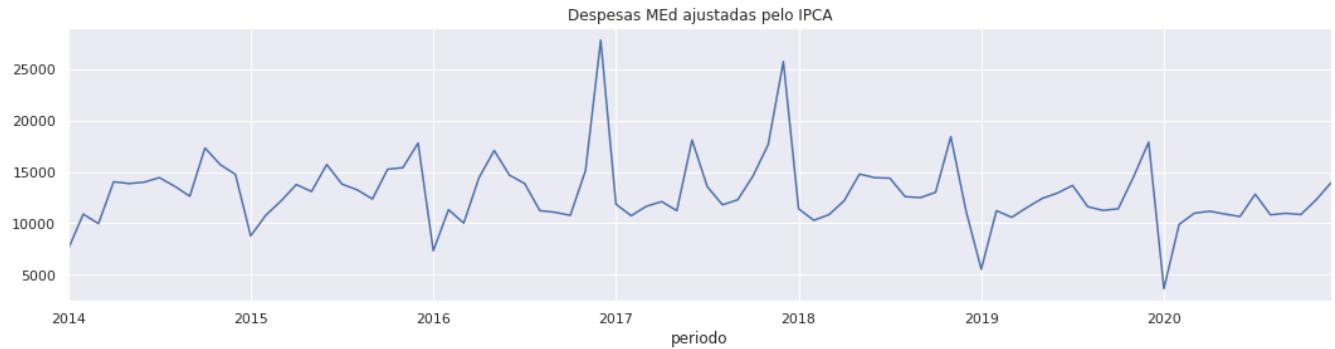


```
Teste Estatistico Dickey Fuller      -3.1597
Valor-P                           0.0224
Lags Usados                      12.0000
Número de observações usadas     71.0000
Valores Críticos (1%)             -3.5260
Valores Críticos (5%)              -2.9032
Valores Críticos (10%)             -2.5890
dtype: float64
```

```
adfuller_test(bkp_gastos['Despesa MS Ajustada'],title='Despesas MS ajustadas pelo IPCA')
```



```
adfuller_test(bkp_gastos[ 'Despesa MEd Ajustada' ],title='Despesas MEd ajustadas pelo IPCA')
```

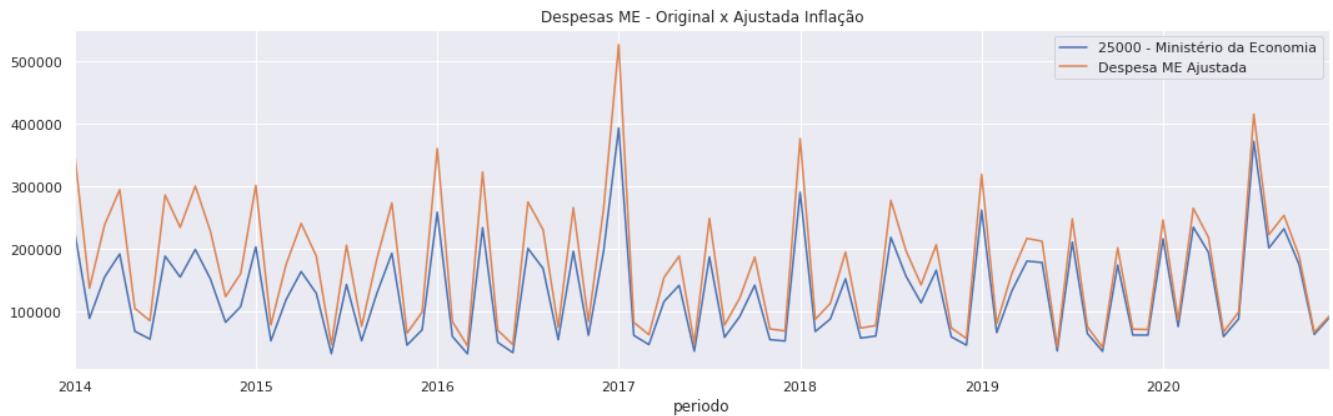


Teste Estatistico Dickey Fuller	-0.0392
Valor-P	0.9552
Lags Usados	11.0000
Número de observações usadas	72.0000
Valores Críticos (1%)	-3.5246
Valores Críticos (5%)	-2.9026
Valores Críticos (10%)	-2.5887
dtype:	float64

Como pode ser visto, a tendência de alta desapareceu apenas para o gráfico do Ministério da Economia (ME), ficando as oscilações sazonais. O teste de Dickey Fuller também confirma que a série para o ME agora é estacionária, contudo, necessitamos de um segundo método que transforme os demais Ministérios em dados Estacionários

Apenas para curiosidade, o gráfico a seguir representa as Despesas do ME original x Ajustada pela Inflação

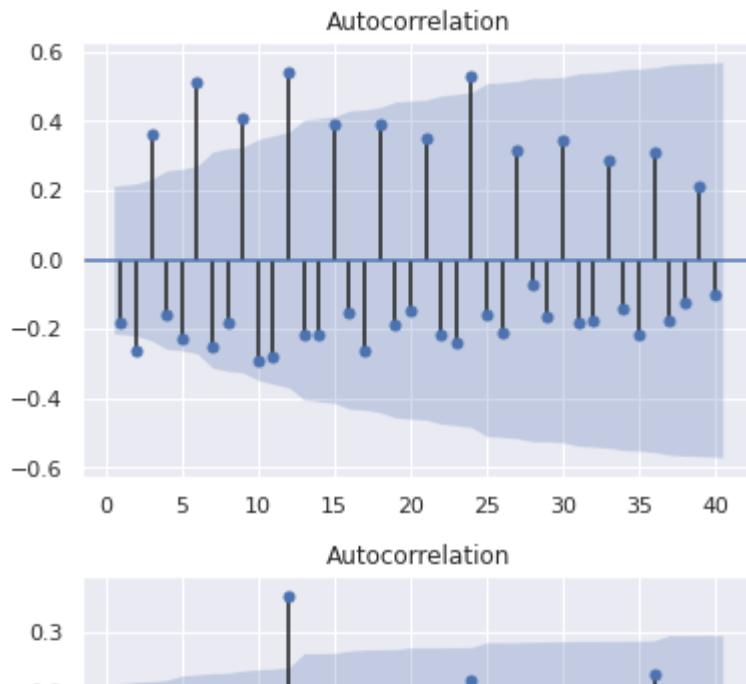
```
bkp_gastos.loc[:,['25000 - Ministério da Economia','Despesa ME Ajustada']].plot(figsize=(18,5))
plt.show()
```



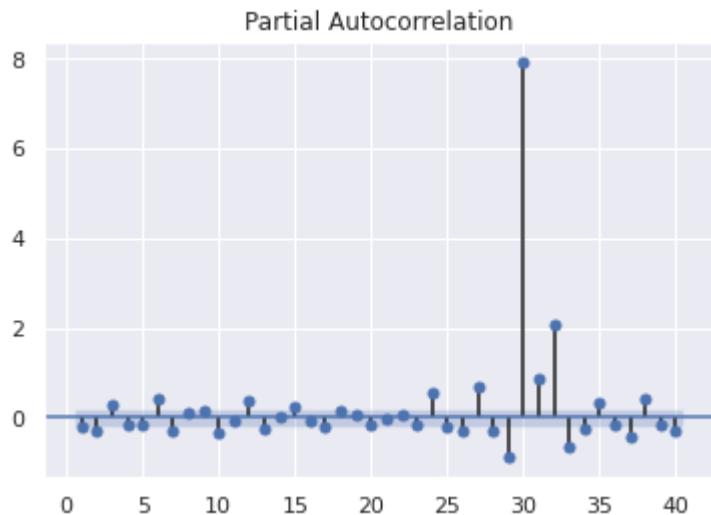
ACF e PACF dos dados ajustados pela Inflação

Para ter previsibilidade uma série com uma única variável deve ser auto correlacionada, ou seja, o período atual deve ter explicação com base em um período anterior (lag)

```
plot_acf(bkp_gastos['Despesa ME Ajustada'], lags=40, zero=False);
plot_acf(bkp_gastos['Despesa MS Ajustada'], lags=40, zero=False);
plot_acf(bkp_gastos['Despesa MEd Ajustada'], lags=40, zero=False);
```



```
plot_pacf(bkp_gastos['Despesa ME Ajustada'], lags=40, zero=False);
plot_pacf(bkp_gastos['Despesa MS Ajustada'], lags=40, zero=False);
plot_pacf(bkp_gastos['Despesa MEd Ajustada'], lags=40, zero=False);
```



Segundo método: DIFERENCIAMENTO

A diferenciação é usada para remover os sinais de tendências e reduzir a variância. É simplesmente a diferença do valor do período T com o valor do período anterior T-1

```
-100
```

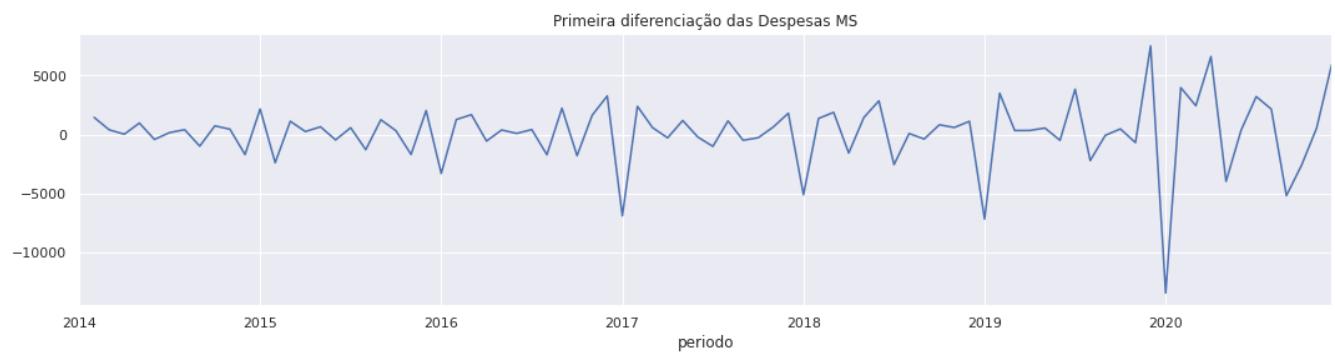
```
bkp_gastos['ME_dif'] = bkp_gastos['25000 - Ministério da Economia'].diff().dropna()
bkp_gastos['MS_dif'] = bkp_gastos['36000 - Ministério da Saúde'].diff().dropna()
bkp_gastos['MEd_dif'] = bkp_gastos['26000 - Ministério da Educação'].diff().dropna()
```

```
-200
```

```
bkp_gastos['ME_dif'].plot(figsize=(18,4), title='Primeira diferenciação das Despesas ME')
plt.show()
print('Mostrando as 10 primeiras diferenciações')
print(bkp_gastos['ME_dif'].dropna().head(10))
```



```
bkp_gastos['MS_dif'].plot(figsize=(18,4), title='Primeira diferenciação das Despesas MS')
plt.show()
print('Mostrando as 10 primeiras diferenciações')
print(bkp_gastos['MS_dif'].dropna().head(10))
```



Mostrando as 10 primeiras diferenciações
periodo

2014-02-01	1444.681519
2014-03-01	392.407853
2014-04-01	18.172951
2014-05-01	965.242530
2014-06-01	-429.404894
2014-07-01	152.931970
2014-08-01	401.534979
2014-09-01	-996.323304
2014-10-01	728.544828
2014-11-01	444.813586

Freq: MS, Name: MS_dif, dtype: float64

```
bkp_gastos['MEd_dif'].plot(figsize=(18,4), title='Primeira diferenciação das Despesas MEd')
plt.show()
print('Mostrando as 10 primeiras diferenciações')
print(bkp_gastos['MEd_dif'].dropna().head(10))
```



Mostrando as 10 primeiras diferenciações periodo

2014-02-01	2248.224797
2014-03-01	-582.494512
2014-04-01	2661.810438
2014-05-01	-92.269764
2014-06-01	119.221323
2014-07-01	334.237969
2014-08-01	-504.360727
2014-09-01	-628.625575
2014-10-01	3173.599074
2014-11-01	-1026.747955

Freq: MS, Name: MEd_dif, dtype: float64

Normalmente só é necessário uma diferenciação para transformar uma série em estacionária, mas caso seja necessário, pode-se aplicar uma segunda diferenciação. Neste caso a diferenciação será sobre a primeira diferenciação (dificilmente haverá casos com mais do que 2 diferenciações)

Testando Dickey-Fuller nas colunas com diferenciação

```
from statsmodels.tsa.stattools import adfuller

def adf_test(series,title=''):
    """
    Passar uma série temporal e um título opcional, retorna um relatório ADF
    """
    print(f'Teste de Dickey-Fuller Aumentado: {title}')
    result = adfuller(series.dropna(),autolag='AIC') # .dropna() para lidar com diferentes da
    labels = ['ADF teste estatístico','p-value','# lags used','# observações']
    out = pd.Series(result[0:4],index=labels)

    for key,val in result[4].items():
        out[f'valor crítico ({key})']=val

    print(out.to_string())

    if result[1] <= 0.05:
```

```

print("Fortes evidências contra a hipótese nula")
print("Rejeita a hipótese nula")
print("É estacionário")
else:
    print("Fracas evidências contra a hipótese nula")
    print("Falha ao rejeitar a hipótese nula")
    print("É não-estacionária")

```

```
adf_test(bkp_gastos['ME_dif'].dropna())
```

Teste de Dickey-Fuller Aumentado:

ADF teste estatístico	-4.582948
p-value	0.000139
# lags used	10.000000
# observações	72.000000
valor crítico (1%)	-3.524624
valor crítico (5%)	-2.902607
valor crítico (10%)	-2.588679

Fortes evidências contra a hipótese nula
 Rejeita a hipótese nula
 É estacionário

```
adf_test(bkp_gastos['MS_dif'].dropna())
```

Teste de Dickey-Fuller Aumentado:

ADF teste estatístico	-2.405226
p-value	0.140267
# lags used	12.000000
# observações	70.000000
valor crítico (1%)	-3.527426
valor crítico (5%)	-2.903811
valor crítico (10%)	-2.589320

Fracas evidências contra a hipótese nula
 Falha ao rejeitar a hipótese nula
 É não-estacionária

```
adf_test(bkp_gastos['MEd_dif'].dropna())
```

Teste de Dickey-Fuller Aumentado:

ADF teste estatístico	-8.013862e+00
p-value	2.165045e-12
# lags used	1.000000e+01
# observações	7.200000e+01
valor crítico (1%)	-3.524624e+00
valor crítico (5%)	-2.902607e+00
valor crítico (10%)	-2.588679e+00

Fortes evidências contra a hipótese nula
 Rejeita a hipótese nula
 É estacionário

Foi possível observar que, ao aplicar a primeira diferenciação para cada Ministério, vemos os Min. da Economia (e seu respectivo valor ajustado pela inflação) e da Educação com dados totalmente estacionários, contudo, o Min. da Saúde permanece com dados não estacionários. Para isso, faz-se necessário utilizar a segunda diferenciação.

Segunda DIFERENCIACÃO para o Ministério da Saúde

```
bkp_gastos['MS_dif'] = bkp_gastos['36000 - Ministério da Saúde'].diff().diff().dropna()
```

```
bkp_gastos['MS_dif'].plot(figsize=(18,4), title='Segunda diferenciação das Despesas MS')
plt.show()
```

```
print('Mostrando as 10 primeiras diferenciações')
```

```
print(bkp_gastos['MS_dif'].dropna().head(10))
```



Mostrando as 10 primeiras diferenciações
periodo

2014-03-01	-1052.273666
2014-04-01	-374.234903
2014-05-01	947.069580
2014-06-01	-1394.647424
2014-07-01	582.336864
2014-08-01	248.603009
2014-09-01	-1397.858283
2014-10-01	1724.868132
2014-11-01	-283.731241
2014-12-01	-2159.735998

Freq: MS, Name: MS_dif, dtype: float64

```
adf_test(bkp_gastos['MS_dif'].dropna())
```

Teste de Dickey-Fuller Aumentado:

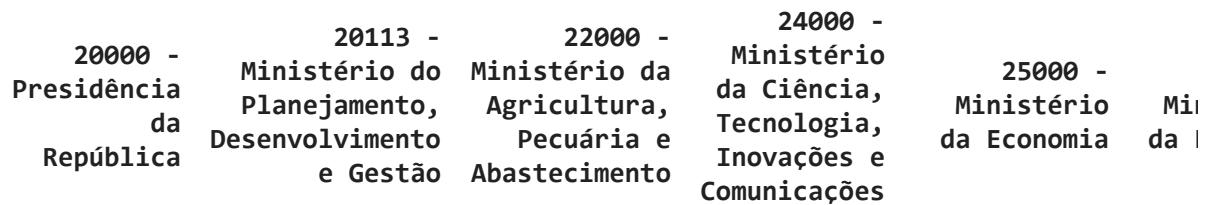
ADF teste estatístico	-8.246901e+00
p-value	5.522832e-13
# lags used	1.200000e+01
# observações	6.900000e+01
valor crítico (1%)	-3.528890e+00

```
valor crítico (5%)      -2.904440e+00
valor crítico (10%)     -2.589656e+00
Fortes evidências contra a hipótese nula
Rejeita a hipótese nula
É estacionário
```

A partir deste momento, temos todas as bases com dados ditos ESTACIONÁRIOS. Desta forma, seguiremos neste estudo com aplicações de modelos de previsão de Série Temporal utilizando ora dados estacionários (AUTO-ARIMA), ora não estacionários (Holt-Winter e Facebook Prophet)

```
train = bkp_gastos.loc[:'2018-12-01']
test = bkp_gastos.loc['2019-01-01':]
```

```
train.tail()
```



periodo	20000 - Presidência da República	20113 - Ministério do Planejamento, Desenvolvimento e Gestão	22000 - Ministério da Agricultura, Pecuária e Abastecimento	24000 - Ministério da Ciência, Tecnologia, Inovações e Comunicações	25000 - Ministério da Economia	Mi
2018-08-01	179.633502	NaN	3912.966257	758.529457	156909.845711	1000
2018-09-01	138.130955	NaN	1423.205008	800.204873	114243.254717	998
2018-10-01	180.869165	NaN	1489.824164	944.997731	166459.092205	1046
2018-11-01	194.644036	NaN	1601.758228	790.515420	59949.969240	1488
2018-12-01	389.158138	NaN	975.330678	1852.017885	46712.864643	914

```
test.head()
```

20000 - Presidência da República	20113 - Ministério do Planejamento, Desenvolvimento e Gestão	22000 - Ministério da Agricultura, Pecuária e Abastecimento	24000 - Ministério da Ciência, Tecnologia, Inovações e Comunicações	25000 - Ministério da Economia	Mi
----------------------------------	--	---	---	--------------------------------	----

periodo

2019-01-01	22.915742	NaN	145.789711	187.508521	262311.577100	454
2019-02-01	103.069368	NaN	792.327481	418.411024	66735.795897	926
2019-03-01	124.694829	NaN	769.692799	578.983409	133261.769754	875
2019-						

▼ AUTO-ARIMA

Utilizamos o pmdarima.auto_arima para indicar o melhor modelo para cada base estudada

```
!pip install pmdarima
from pmdarima import auto_arima
```

```
Collecting pmdarima
```

```
  Downloading https://files.pythonhosted.org/packages/e4/a8/bdf15174e35d072e145d16388b1
|██████████| 1.5MB 11.3MB/s
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.7/dist-packages (Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.7/dist-packages (Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-packages Collecting statsmodels!=0.12.0,>=0.11
  Downloading https://files.pythonhosted.org/packages/da/69/8eef30a6237c54f3c0b524140e2
|██████████| 9.5MB 42.5MB/s
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (Requirement already satisfied: scinv>=1.3.2 in /usr/local/lib/python3.7/dist-packages (
```

▼ Indicação Ministério da Economia com preço ajustado pela Inflação

```
Requirement already satisfied: setupools!=50.0.0,>=58.0.0 in /usr/local/lib/python3.7/dist-packages (auto_arima(bkp_gastos['Despesa ME Ajustada'].dropna(), seasonal=True, m=12).summary()
```

Statespace Model Results

Dep. Variable:	y	No. Observations:	84
Model:	SARIMAX(3, 0, 5)x(2, 0, 0, 12)	Log Likelihood	-1056.220
Date:	Wed, 14 Apr 2021	AIC	2136.440
Time:	14:51:47	BIC	2165.610
Sample:	0 - 84	HQIC	2148.166

Covariance Type: opg

	coef	std err	z	P> z	[0.025	0.975]
intercept	2.654e+05	1.25e-05	2.12e+10	0.000	2.65e+05	2.65e+05
ar.L1	-1.0125	0.509	-1.989	0.047	-2.010	-0.015
ar.L2	-1.0103	0.502	-2.014	0.044	-1.993	-0.027
ar.L3	-0.0146	0.504	-0.029	0.977	-1.003	0.974
ma.L1	1.1738	0.557	2.109	0.035	0.083	2.265
ma.L2	1.1366	0.694	1.639	0.101	-0.223	2.496
ma.L3	0.0287	0.724	0.040	0.968	-1.391	1.448
ma.L4	-0.1499	0.305	-0.492	0.623	-0.747	0.447
ma.L5	-0.1483	0.198	-0.748	0.455	-0.537	0.241
ar.S.L12	0.1639	0.178	0.922	0.356	-0.184	0.512
ar.S.L24	0.3319	0.175	1.901	0.057	-0.010	0.674
sigma2	5.44e+09	1.55e-10	3.51e+19	0.000	5.44e+09	5.44e+09

Ljung-Box (Q): 21.86 **Jarque-Bera (JB):** 7.89

Prob(Q): 0.99 **Prob(JB):** 0.02

Heteroskedasticity (H): 0.95 **Skew:** 0.71

Prob(H) (two-sided): 0.89 **Kurtosis:** 3.48

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

▼ Indicação Ministério da Saúde com diferenciação

```
auto_arima(bkp_gastos.MS_dif.dropna(), seasonal=True, m=12).summary()
```

Statespace Model Results

Dep. Variable:	y	No. Observations:	82
Model:	SARIMAX(1, 0, 1)x(1, 0, 0, 12)	Log Likelihood	-747.801
Date:	Wed, 14 Apr 2021	AIC	1505.602
Time:	14:52:08	BIC	1517.635
Sample:	0 - 82	HQIC	1510.433

Covariance Type: opg

	coef	std err	z	P> z 	[0.025	0.975]
intercept	-0.1219	6.588	-0.019	0.985	-13.034	12.790
ar.L1	-0.4673	0.078	-6.029	0.000	-0.619	-0.315
ma.L1	-0.9787	0.091	-10.802	0.000	-1.156	-0.801
ar.S.L12	0.6792	0.110	6.152	0.000	0.463	0.896
sigma2	4.198e+06	5.83e+05	7.205	0.000	3.06e+06	5.34e+06

Ljung-Box (Q): 70.43 **Jarque-Bera (JB):** 25.13
Prob(Q): 0.00 **Prob(JB):** 0.00
Heteroskedasticity (H): 5.68 **Skew:** 0.50
Prob(H) (two-sided): 0.00 **Kurtosis:** 5.52

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex step)

▼ Indicação Ministério da Educação com diferenciação

```
auto_arima(bkp_gastos.MEd_dif.dropna(), seasonal=True, m=12).summary()
```

Statespace Model Results

Dep. Variable: y **No. Observations:** 83
Model: SARIMAX(5, 0, 1)x(1, 0, 1, 12) **Log Likelihood:** -754.147
Date: Wed, 14 Apr 2021 **AIC:** 1528.295
Time: 14:53:42 **BIC:** 1552.483
Sample: 0 **HQIC:** 1538.012
- 83

Covariance Type: opg

	coef	std err	z	P> z	[0.025	0.975]
intercept	2.0279	16.299	0.124	0.901	-29.918	33.974
ar.L1	0.0358	0.255	0.140	0.888	-0.465	0.537

▼ Ajustando os modelos nas bases de treino

ar.I 5 -0.0830 0.260 -0.320 0.749 -0.592 0.426

Ministério da Economia com despesas ajustadas pela Inflação

ar.I 5 -0.0830 0.260 -0.320 0.749 -0.592 0.426

```
model_sarima_ME = SARIMAX(train['Despesa ME Ajustada'].dropna(), order=(3,0,5), seasonal_order=(0,0,0,0))
results_sarima_ME = model_sarima_ME.fit()
results_sarima_ME.summary()
```

Statespace Model Results

Dep. Variable: Despesa ME Ajustada **No. Observations:** 60
Model: SARIMAX(3 0 5)x(2 0 0 12) **Log Likelihood:** -761.701

```
# Obtendo a previsão
inicio = len(train)
fim = len(train)+len(test)-1
predictions_sarima_ME = results_sarima_ME.predict(start=inicio, end=fim, dynamic=False, typ=''
Covariance Type: opg

# Comparando a previsão com os valores esperados
for i in range(len(predictions_sarima_ME)):
    print(f"predicted={predictions_sarima_ME[i]:<11.10}, expected={test['Despesa ME Ajustada'"

predicted=350595.3866, expected=319517.2232448776
predicted=116727.0768, expected=80933.61795470581
predicted=87551.161 , expected=161081.20693436387
predicted=159417.0595, expected=217200.637926258
predicted=134588.6093, expected=212603.4247543776
predicted=68789.58631, expected=44223.490176924286
predicted=234813.0035, expected=248303.91988798845
predicted=128640.0762, expected=76191.63543070751
predicted=132093.4715, expected=42975.54550985554
predicted=176150.0395, expected=202221.5698529857
predicted=84196.87875, expected=72182.13020551435
predicted=78905.43073, expected=71759.66229953362
predicted=297319.6558, expected=246642.26739943962
predicted=107893.5253, expected=86810.25742145017
predicted=103683.5592, expected=265275.78239159624
predicted=161342.1257, expected=219275.49920923475
predicted=108326.2114, expected=67744.57409044722
predicted=84317.34736, expected=99513.53711038354
predicted=219386.4827, expected=415611.8561092594
predicted=149545.818 , expected=223234.0923045261
predicted=131834.3007, expected=254070.03345019068
predicted=171180.771 , expected=188774.30170309034
predicted=88746.22299, expected=67183.3932489177
predicted=80654.28413, expected=93455.06681117052

# Plotar previsões em relação aos valores conhecidos
title = 'Comparação de Previsão e real - ME'
ylabel='Despesa'
xlabel=''

ax = test['Despesa ME Ajustada'].plot(legend=True, figsize=(15,6), title=title)
predictions_sarima_ME.plot(legend=True)
ax.autoscale(axis='x', tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel);
```



Ministério da Saúde com despesas ajustadas por Diferenciação

```
model_sarima_MS = SARIMAX(train.MS_dif.dropna(),order=(1,0,1),seasonal_order=(1,0,0,12))
results_sarima_MS = model_sarima_MS.fit(disp = -1)
results_sarima_MS.summary()
```

Statespace Model Results

Dep. Variable:	MS_dif	No. Observations:	58
Model:	SARIMAX(1, 0, 1)x(1, 0, 0, 12)	Log Likelihood	-502.953
Date:	Wed, 14 Apr 2021	AIC	1013.905
Time:	14:53:46	BIC	1022.147
Sample:	03-01-2014 - 12-01-2018	HQIC	1017.115

Covariance Type: opg

coef	std err	z	P> z	[0.025	0.975]	
ar.L1	-0.5553	0.087	-6.365	0.000	-0.726	-0.384
ma.L1	-1.0000	0.161	-6.202	0.000	-1.316	-0.684
ar.S.L12	0.5210	0.125	4.183	0.000	0.277	0.765
sigma2	1.736e+06	9.29e-08	1.87e+13	0.000	1.74e+06	1.74e+06

Ljung-Box (Q): 48.12 **Jarque-Bera (JB):** 4.86
Prob(Q): 0.18 **Prob(JB):** 0.09
Heteroskedasticity (H): 1.72 **Skew:** 0.09
Prob(H) (two-sided): 0.25 **Kurtosis:** 4.41

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

[2] Covariance matrix is singular or near singular, with condition number 2.12e+28. Standard errors may be

```
# Obtendo a previsão
inicio = len(train)-2
fim = len(train)+len(test)-3
predictions_sarima_MS = results_sarima_MS.predict(start=inicio, end=fim, dynamic=False, typ='
```

<https://colab.research.google.com/drive/1jG6GB8N-LOL-IK2YNcD6rp6O2n8ezH-q#scrollTo=xKDF8wHvcTgq&printMode=true>

```
# Comparando a previsão com os valores esperados
for i in range(len(predictions_sarima_MS)):
    print(f"predicted={predictions_sarima_MS[i]}: {test.MS_dif[i]}")

predicted=-3808.866954, expected=-8311.539567070002
predicted=3489.199128, expected=10701.293028910002
predicted=209.4486129, expected=-3175.845992540001
predicted=-1771.151751, expected=7.794346609998684
predicted=1555.752499, expected=208.17342762000408
predicted=751.1683067, expected=-1048.5109703500038
predicted=-2830.827399, expected=4346.227741530001
predicted=1379.8696 , expected=-6064.11620479
predicted=-245.5777839, expected=2146.087364119998
predicted=631.2037573, expected=543.9239639600037
predicted=-118.9914089, expected=-1166.5870854500026
predicted=273.5503338, expected=8222.91605653
predicted=-1984.551312, expected=-21009.616575499997
predicted=1817.932468, expected=17466.136017589997
predicted=109.0686755, expected=-1547.3376730900009
predicted=-922.7239469, expected=4193.352381530003
predicted=810.5158771, expected=-10622.981964630006
predicted=391.3602852, expected=4359.350612180006
predicted=-1474.838478, expected=2856.284621419998
predicted=718.9013823, expected=-1071.8511100800024
predicted=-127.9451609, expected=-7353.086648009994
predicted=328.851863 , expected=2579.598455589996
predicted=-61.9938365, expected=3156.6808704799987
predicted=142.5173235, expected=5415.396301070003

# Plotar previsões em relação aos valores conhecidos
title = 'Comparação de Previsão e real - MS'
ylabel='Despesa'
xlabel=''

ax = test.MS_dif.plot(legend=True, figsize=(15,6), title=title)
predictions_sarima_MS.plot(legend=True)
ax.autoscale(axis='x', tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel);
```



Ministério da Educação com despesas ajustadas por Diferenciação

```
-15000 15000
```

```
model_sarima_MEd = SARIMAX(train.MEd_dif.dropna(), order=(5,0,1), seasonal_order=(1,0,1,12))
results_sarima_MEd = model_sarima_MEd.fit(disp = -1)
results_sarima_MEd.summary()
```

Statespace Model Results

Dep. Variable:	MEd_dif	No. Observations:	59
Model:	SARIMAX(5, 0, 1)x(1, 0, 1, 12)	Log Likelihood	-535.082
Date:	Wed, 14 Apr 2021	AIC	1088.163
Time:	14:53:47	BIC	1106.861
Sample:	02-01-2014 - 12-01-2018	HQIC	1095.462

Covariance Type: opg

	coef	std err	z	P> z 	[0.025	0.975]
ar.L1	-0.1282	0.224	-0.573	0.566	-0.567	0.310
ar.L2	-0.2303	0.223	-1.033	0.302	-0.667	0.207
ar.L3	-0.2441	0.214	-1.140	0.254	-0.664	0.175
ar.L4	-0.2798	0.253	-1.104	0.270	-0.777	0.217
ar.L5	-0.2600	0.159	-1.637	0.102	-0.571	0.051
ma.L1	-0.8602	0.133	-6.485	0.000	-1.120	-0.600
ar.S.L12	0.3929	0.176	2.237	0.025	0.049	0.737
ma.S.L12	0.2687	0.309	0.869	0.385	-0.337	0.874
sigma2	3.852e+06	7.2e+05	5.349	0.000	2.44e+06	5.26e+06

Ljung-Box (Q): 17.56 **Jarque-Bera (JB):** 56.40

Prob(Q): 1.00 **Prob(JB):** 0.00

Heteroskedasticity (H): 3.03 **Skew:** 0.41

Prob(H) (two-sided): 0.02 **Kurtosis:** 7.72

Warnings:

```
# Obtendo a previsão
inicio = len(train)-1
fim = len(train)+len(test)-2
predictions_sarima_MEd = results_sarima_MEd.predict(start=inicio, end=fim, dynamic=False, typ

# Comparando a previsão com os valores esperados
```

```
for i in range(len(predictions_sarima_MEd)):  
    print(f"predicted={predictions_sarima_MEd[i]}: <11.10}, expected={test.MEd_dif[i]}")
```

```
predicted=599.600686 , expected=-4593.187947239998  
predicted=338.1910899, expected=4713.263698669998  
predicted=-65.52085402, expected=-507.28933427000084  
predicted=641.267488 , expected=855.6261464300042  
predicted=1946.740645, expected=825.994034399997  
predicted=-2909.781016, expected=513.3927634600022  
predicted=671.1324683, expected=695.5561645100006  
predicted=-347.8243641, expected=-1722.349718650008  
predicted=203.7533016, expected=-262.03040865  
predicted=209.1066037, expected=194.67803906000154  
predicted=2886.322358, expected=2751.818577619997  
predicted=-4762.079347, expected=3040.0617074700076  
predicted=1602.500769, expected=-12441.3238728  
predicted=193.4328333, expected=5497.379551929999  
predicted=-63.68421289, expected=1036.3415515600027  
predicted=311.4744375, expected=207.30684875000225  
predicted=1019.790194, expected=-223.98533295000198  
predicted=-1552.792867, expected=-234.45555048999813  
predicted=237.8340794, expected=2003.8762933600028  
predicted=-108.0431856, expected=-1702.4073977100052  
predicted=95.5050844 , expected=255.21006951999516  
predicted=128.2153167, expected=49.0143375200023  
predicted=1231.315528, expected=1585.395764019997  
predicted=-1899.15515, expected=1880.878959350006
```

```
# Plotar previsões em relação aos valores conhecidos
```

```
title = 'Comparação de Previsão e real - MEd'
```

```
ylabel='Despesa'
```

```
xlabel=''
```

```
ax = test.MEd_dif.plot(legend=True, figsize=(15,6), title=title)  
predictions_sarima_MEd.plot(legend=True)  
ax.autoscale(axis='x', tight=True)  
ax.set(xlabel=xlabel, ylabel=ylabel);
```



▼ Previsão de futuro utilizando AUTO-ARIMA

Previsão final (12 meses) das despesas do Ministério da Economia com valores ajustados pela Inflação

```
modelo_final_sarimax_ME = SARIMAX(bkp_gastos[ 'Despesa ME Ajustada' ],order=(3,0,5),seasonal_order=(0,1,1,12))
resultado_final_sarimax_ME = modelo_final_sarimax_ME.fit()
previsao_final_sarimax_ME = resultado_final_sarimax_ME.predict(len(bkp_gastos),len(bkp_gastos))
```

```
# Plotar previsões de 12 meses para frente
title = 'Previsão de 12 meses ME'
ylabel='Despesa'
xlabel=''

ax = bkp_gastos[ 'Despesa ME Ajustada' ].plot(legend=True,figsize=(20,6),title=title)
previsao_final_sarimax_ME.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel);
```



Previsão final (12 meses) das despesas do Ministério da Saúde com valores ajustados por Diferenciação

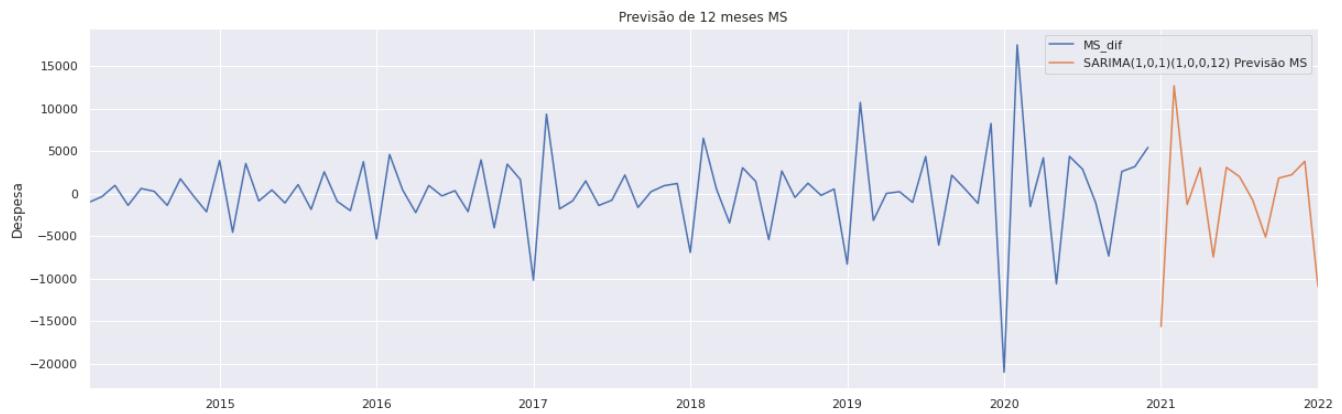
```

modelo_final_sarimax_MS = SARIMAX(bkp_gastos.MS_dif,order=(1,0,1),seasonal_order=(1,0,0,12))
resultado_final_sarimax_MS = modelo_final_sarimax_MS.fit()
previsao_final_sarimax_MS = resultado_final_sarimax_MS.predict(len(bkp_gastos),len(bkp_gastos))

# Plotar previsões de 12 meses para frente
title = 'Previsão de 12 meses MS'
ylabel='Despesa'
xlabel=''

ax = bkp_gastos.MS_dif.plot(legend=True,figsize=(20,6),title=title)
previsao_final_sarimax_MS.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel);

```



Previsão final (12 meses) das despesas do Ministério da Educação com valores ajustados por Diferenciação

```

modelo_final_sarimax_MEd = SARIMAX(bkp_gastos.MEd_dif,order=(5,0,1),seasonal_order=(1,0,1,12))
resultado_final_sarimax_MEd = modelo_final_sarimax_MEd.fit()
previsao_final_sarimax_MEd = resultado_final_sarimax_MEd.predict(len(bkp_gastos),len(bkp_gastos))

# Plotar previsões de 12 meses para frente
title = 'Previsão de 12 meses MEd'
ylabel='Despesa'
xlabel=''

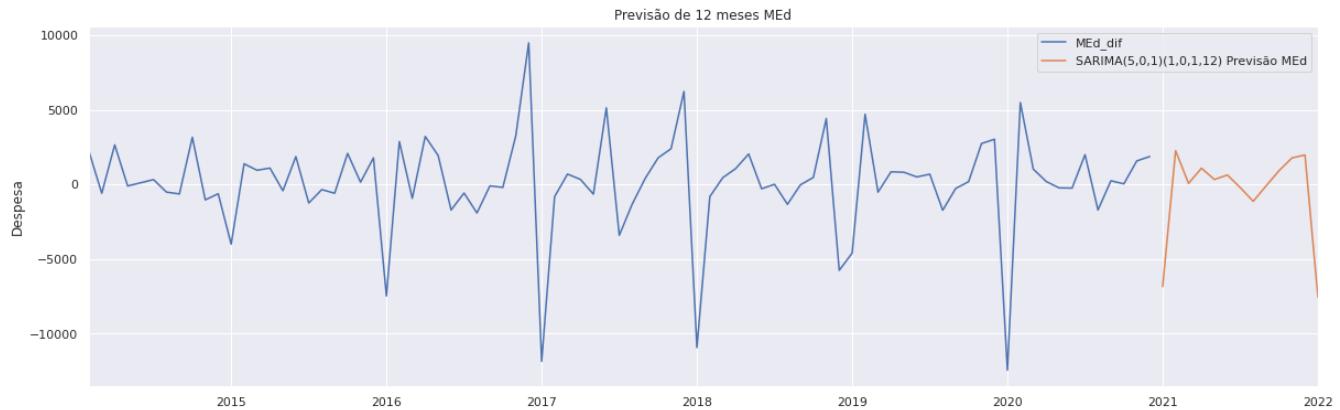
ax = bkp_gastos.MEd_dif.plot(legend=True,figsize=(20,6),title=title)
previsao_final_sarimax_MEd.plot(legend=True)

```

```

    ax.autoscale(axis='x',tight=True)
    ax.set(xlabel=xlabel, ylabel=ylabel);

```



▼ HOLT WINTERS

O método elaborado por Holt (1957) e Winters (1960) tem como pressuposto fundamental lidar com sazonalidades. Ele possui 3 equações básicas: uma para ajuste de nível, outra para ajustar o crescimento e outra para sazonalidade

fonte: <https://otexts.com/fpp2/holt-winters.html>

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

▼ Treinando a base de despesas do Ministério da Economia com valores ajustados pela Inflação

```
fitted_model_ME = ExponentialSmoothing(train[ 'Despesa ME Ajustada' ],trend='add',seasonal='add')
```

```
test_predictions_ME = fitted_model_ME.forecast(24).rename('Previsão - Holt-Winters - Despesa')
test_predictions_ME
```

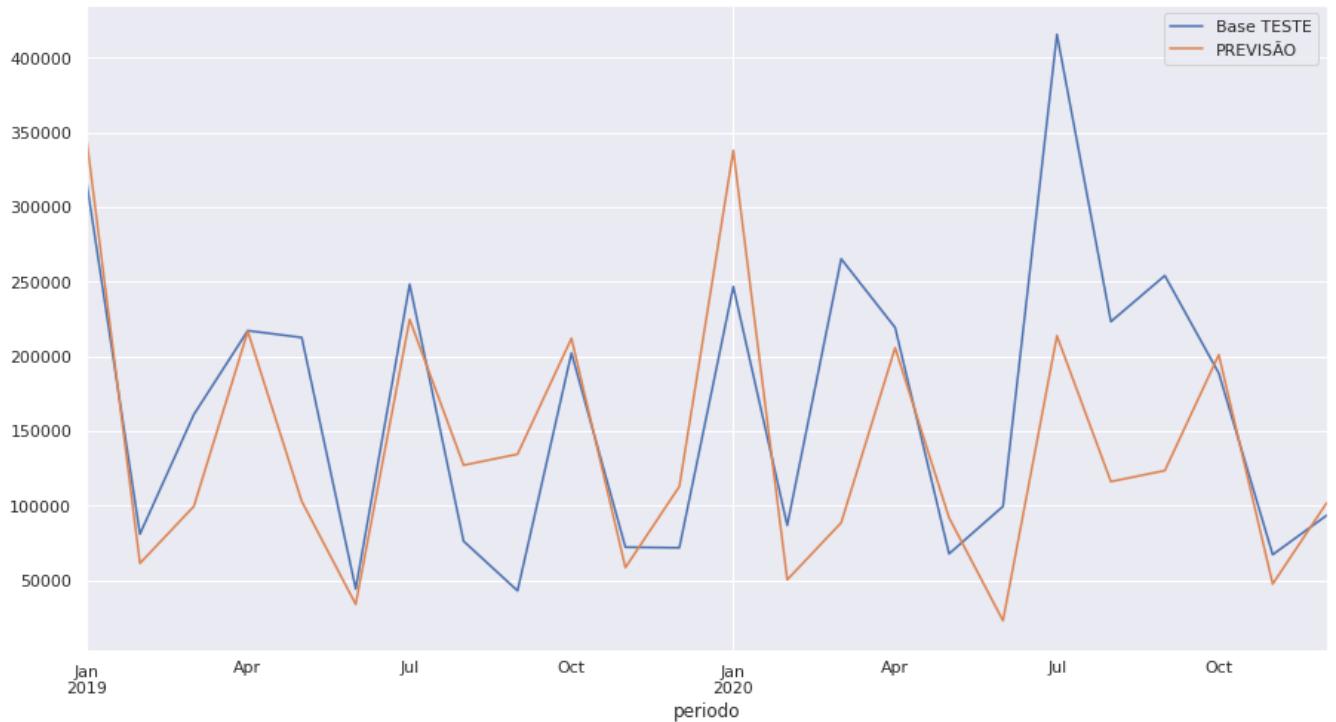
2019-01-01	348837.620144
2019-02-01	61363.375890

2019-03-01	99535.604487
2019-04-01	216729.253641
2019-05-01	102889.377840
2019-06-01	33927.750142
2019-07-01	224727.265628
2019-08-01	127100.520423
2019-09-01	134444.948949
2019-10-01	212022.147025
2019-11-01	58522.403311
2019-12-01	112813.343535
2020-01-01	337841.738222
2020-02-01	50367.493968
2020-03-01	88539.722565
2020-04-01	205733.371719
2020-05-01	91893.495919
2020-06-01	22931.868220
2020-07-01	213731.383706
2020-08-01	116104.638501
2020-09-01	123449.067028
2020-10-01	201026.265104
2020-11-01	47526.521389
2020-12-01	101817.461613

Freq: MS, Name: Previsão - Holt-Winters - Despesa ME Ajustada, dtype: float64

```
train['Despesa ME Ajustada'].plot(legend=True,label=' Base TREINO')
test['Despesa ME Ajustada'].plot(legend=True,label='Base TESTE',figsize=(15,8))
test_predictions_ME.plot(legend=True,label='PREVISÃO');
```

```
test['Despesa ME Ajustada'].plot(legend=True,label='Base TESTE',figsize=(15,8))
test_predictions_ME.plot(legend=True,label='PREVISÃO',xlim=['2019-01-01','2020-12-01']);
```



```
from sklearn.metrics import mean_squared_error,mean_absolute_error

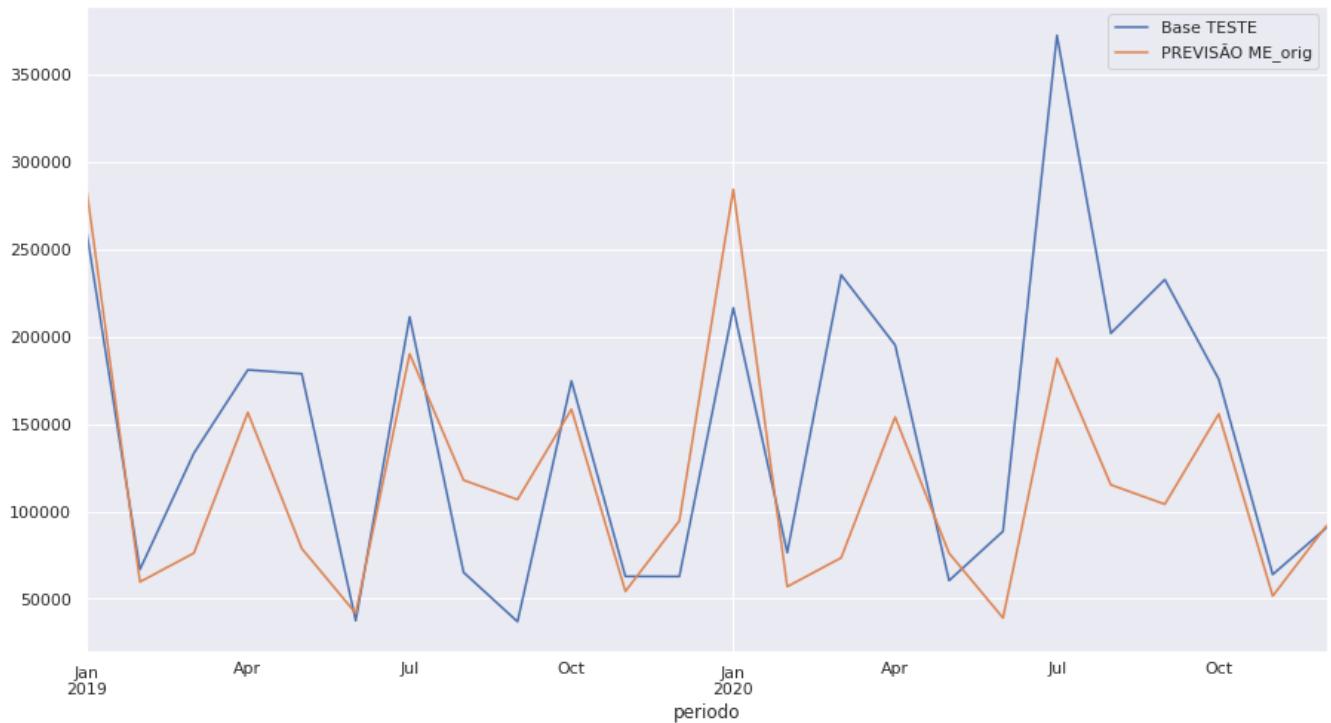
import warnings
warnings.filterwarnings("ignore")

mae = mean_absolute_error(test['Despesa ME Ajustada'],test_predictions_ME)
mse = mean_squared_error(test['Despesa ME Ajustada'],test_predictions_ME)
rmse = np.sqrt(mean_squared_error(test['Despesa ME Ajustada'],test_predictions_ME))
print(f'erro MAE: {mae}')
print(f'erro MSE: {mse}')
print(f'erro RMSE: {rmse}')

erro MAE: 56664.14481839077
erro MSE: 6169553611.222234
erro RMSE: 78546.50604083057
```

Treinando a base de despesas do Ministério da Economia com os valores originais

```
fitted_model_ME_orig = ExponentialSmoothing(train['25000 - Ministério da Economia'],trend='ac')
test_predictions_ME_orig = fitted_model_ME_orig.forecast(24).rename('Previsão - Holt-Winters')
test['25000 - Ministério da Economia'].plot(legend=True,label='Base TESTE',figsize=(15,8))
test_predictions_ME_orig.plot(legend=True,label='PREVISÃO ME_orig',xlim=['2019-01-01','2020-12-31'])
```

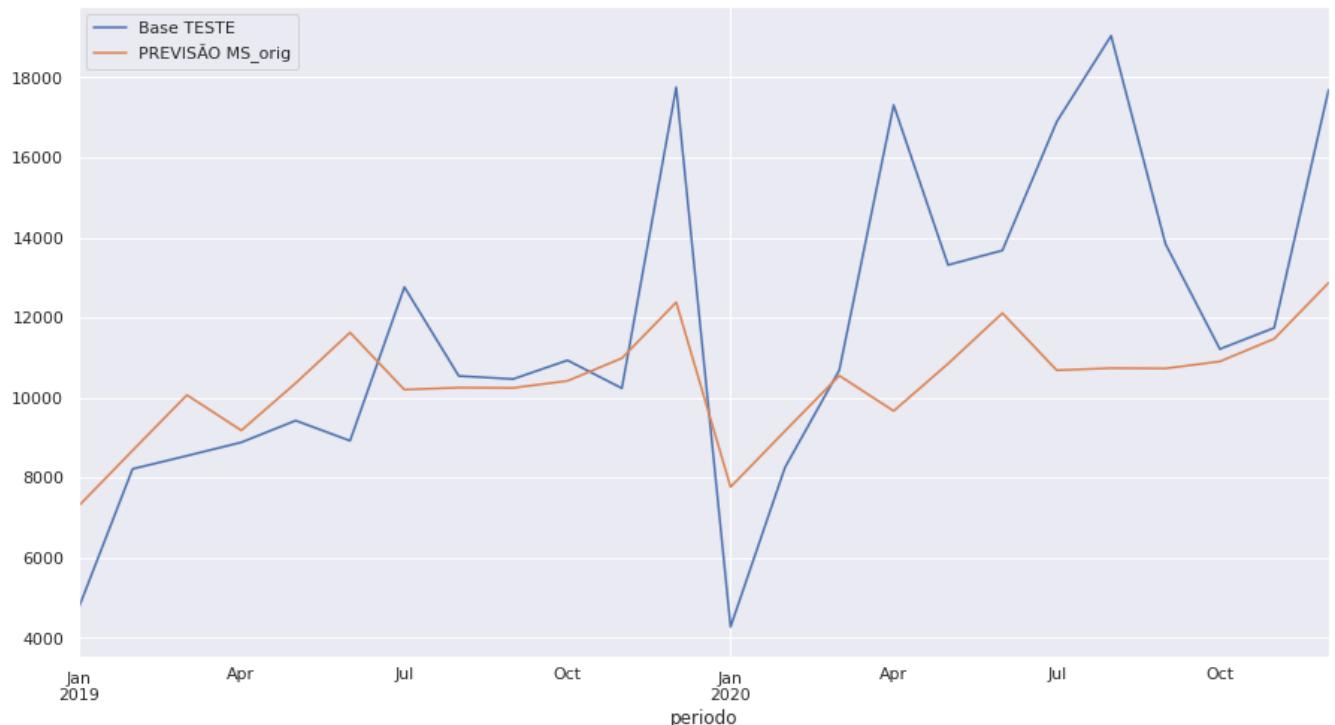


```
mae = mean_absolute_error(test['25000 - Ministério da Economia'],test_predictions_ME_orig)
mse = mean_squared_error(test['25000 - Ministério da Economia'],test_predictions_ME_orig)
rmse = np.sqrt(mean_squared_error(test['25000 - Ministério da Economia'],test_predictions_ME_orig))
print(f'erro MAE: {mae}')
print(f'erro MSE: {mse}')
print(f'erro RMSE: {rmse}')

erro MAE: 50270.186455678224
erro MSE: 4928456597.607062
erro RMSE: 70202.96715671683
```

Treinando a base de despesas do Ministério da Saúde com os valores originais

```
fitted_model_MS_orig = ExponentialSmoothing(train['36000 - Ministério da Saúde'],trend='add',  
test_predictions_MS_orig = fitted_model_MS_orig.forecast(24).rename('Previsão - Holt-Winters  
test['36000 - Ministério da Saúde'].plot(legend=True,label='Base TESTE',figsize=(15,8))  
test_predictions_MS_orig.plot(legend=True,label='PREVISÃO MS_orig',xlim=['2019-01-01','2020-1
```



```
mae = mean_absolute_error(test['36000 - Ministério da Saúde'],test_predictions_MS_orig)  
mse = mean_squared_error(test['36000 - Ministério da Saúde'],test_predictions_MS_orig)  
rmse = np.sqrt(mean_squared_error(test['36000 - Ministério da Saúde'],test_predictions_MS_orig))  
print(f'erro MAE: {mae}')  
print(f'erro MSE: {mse}')  
print(f'erro RMSE: {rmse}')
```

```
erro MAE: 2394.268761213631  
erro MSE: 11442697.93742841  
erro RMSE: 3382.705712507136
```

Treinando a base de despesas do Ministério da Educação com os valores originais

```
fitted_model_MEd_orig = ExponentialSmoothing(train['26000 - Ministério da Educação'],trend='a')
test_predictions_MEd_orig = fitted_model_MEd_orig.forecast(24).rename('Previsão - Holt-Winter')
test['26000 - Ministério da Educação'].plot(legend=True,label='Base TESTE',figsize=(15,8))
test_predictions_MEd_orig.plot(legend=True,label='PREVISÃO MEd_orig',xlim=['2019-01-01','2020-01-01'])
```



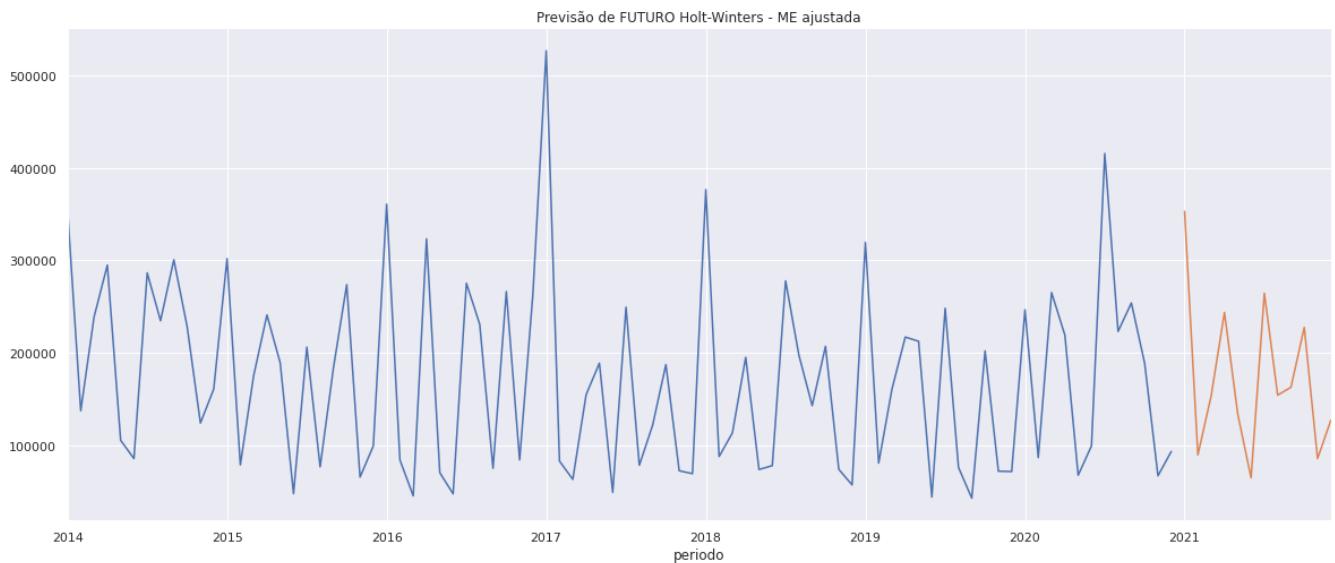
```
mae = mean_absolute_error(test['26000 - Ministério da Educação'],test_predictions_MEd_orig)
mse = mean_squared_error(test['26000 - Ministério da Educação'],test_predictions_MEd_orig)
rmse = np.sqrt(mean_squared_error(test['26000 - Ministério da Educação'],test_predictions_MEd_orig))
print(f'erro MAE: {mae}')
print(f'erro MSE: {mse}')
print(f'erro RMSE: {rmse}')

erro MAE: 1371.775626642618
erro MSE: 3436136.861761887
erro RMSE: 1853.6819742776502
```

▼ Previsão de futuro utilizando Holt-Winters

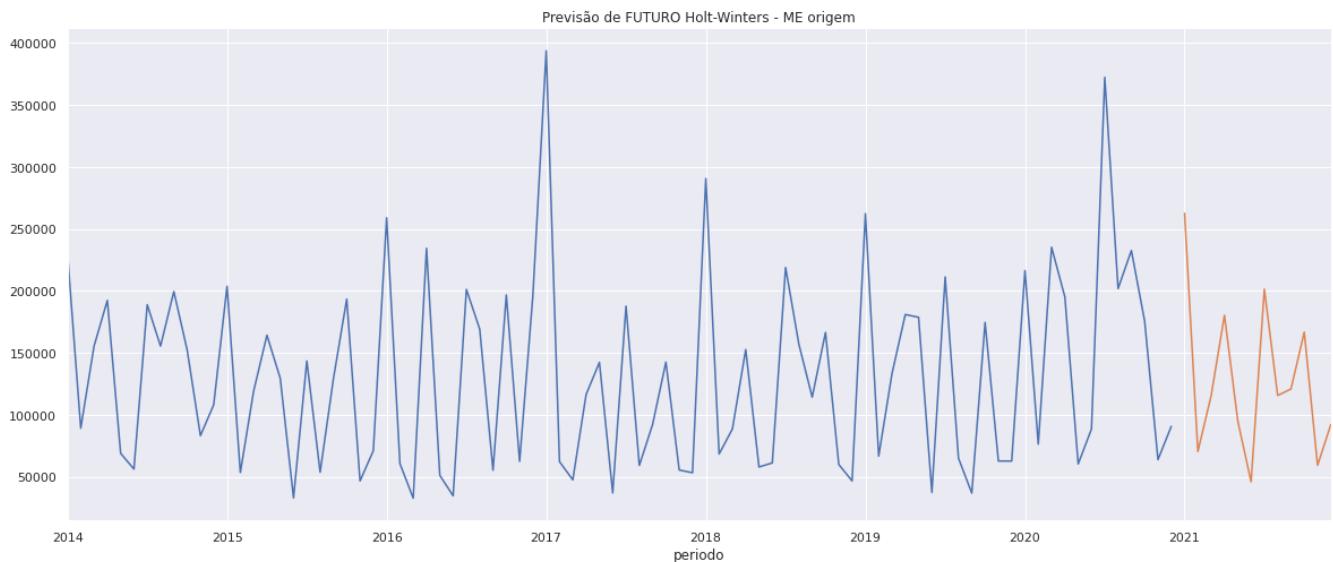
Previsão final (12 meses) das despesas do Ministério da Economia com valores ajustados pela Inflação

```
modelo_HW_futuro_ME = ExponentialSmoothing(bkp_gastos['Despesa ME Ajustada'], trend='add', seasonal='add')
predição_HW_ME = modelo_HW_futuro_ME.forecast(12)
bkp_gastos['Despesa ME Ajustada'].plot(figsize=(20,8), title = 'Previsão de FUTURO Holt-Winters - ME ajustada')
predição_HW_ME.plot();
```



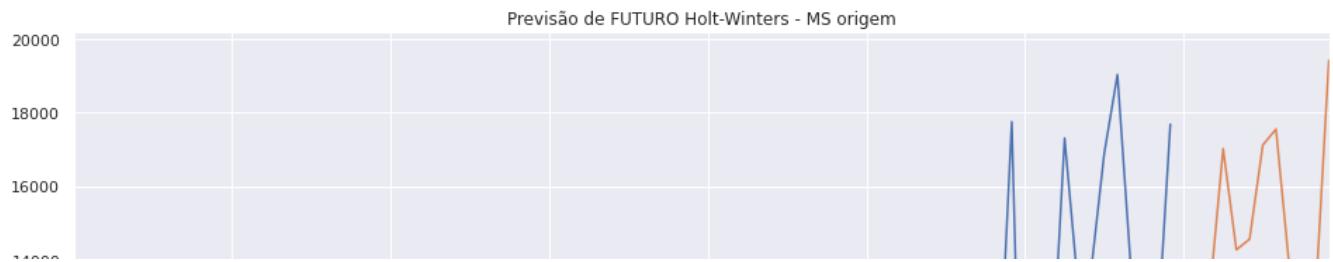
Previsão final (12 meses) das despesas do Ministério da Economia com valores originais

```
modelo_HW_futuro_ME_orig = ExponentialSmoothing(bkp_gastos['25000 - Ministério da Economia'], trend='add', seasonal='add')
predição_HW_ME_orig = modelo_HW_futuro_ME_orig.forecast(12)
bkp_gastos['25000 - Ministério da Economia'].plot(figsize=(20,8), title = 'Previsão de FUTURO Holt-Winters - ME original')
predição_HW_ME_orig.plot();
```



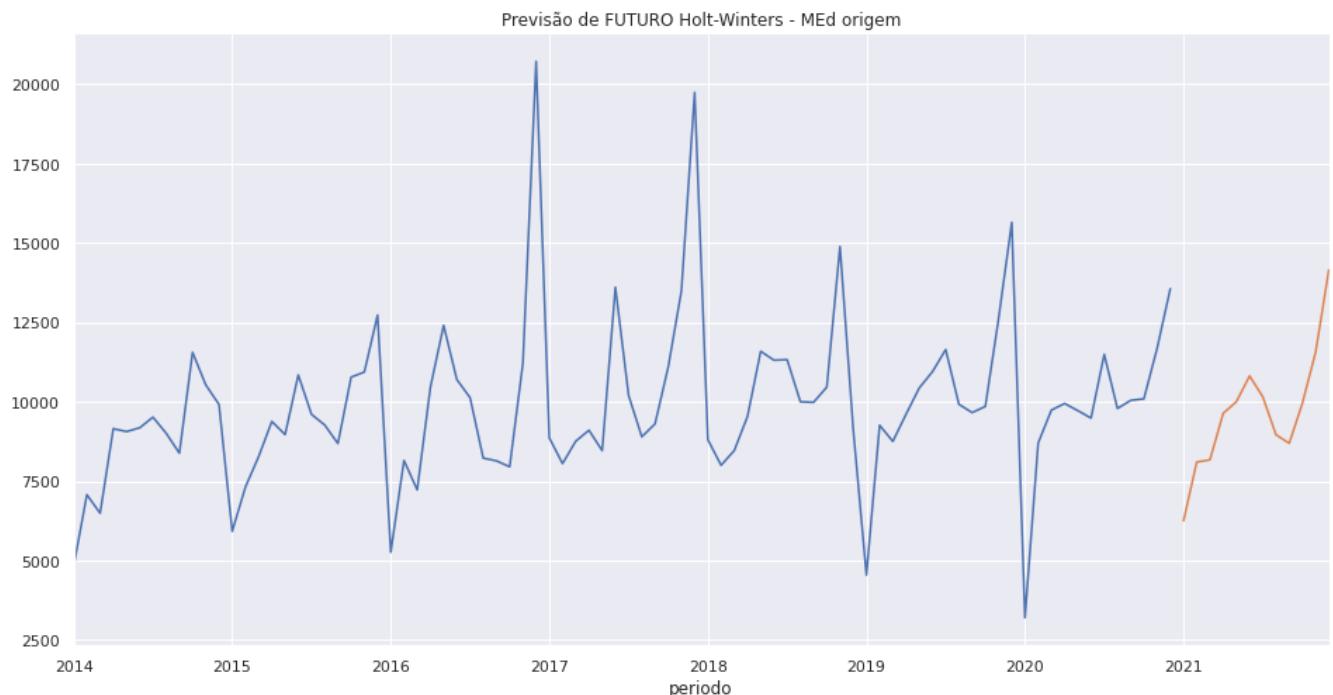
Previsão final (12 meses) das despesas do Ministério da Saúde com valores originais

```
modelo_HW_futuro_MS_orig = ExponentialSmoothing(bkp_gastos['36000 - Ministério da Saúde'], trend='add', seasonal='add')
predição_HW_MS_orig = modelo_HW_futuro_MS_orig.forecast(12)
bkp_gastos['36000 - Ministério da Saúde'].plot(figsize=(16,8), title = 'Previsão de FUTURO Hc')
predição_HW_MS_orig.plot();
```



Previsão final (12 meses) das despesas do Ministério da Educação com valores originais

```
modelo_HW_futuro_MEd_orig = ExponentialSmoothing(bkp_gastos['26000 - Ministério da Educação'])
predição_HW_MEd_orig = modelo_HW_futuro_MEd_orig.forecast(12)
bkp_gastos['26000 - Ministério da Educação'].plot(figsize=(16,8), title = 'Previsão de FUTURO')
predição_HW_MEd_orig.plot();
```



▼ Facebook Prophet

A equipe do Facebook desenvolveu um modelo que captura e interpreta a sazonalidade e a tendência em dados de séries temporais, podendo ser, por exemplo, anualmente, semanalmente e

diariamente, além de séries personalizadas. A biblioteca foi desenvolvida para a equipe de analistas do Facebook, mas também é open source.

fonte: <https://facebook.github.io/prophet/>

```
from fbprophet import Prophet
from statsmodels.tools.eval_measures import rmse
```

Preparando os dados

```
bkp_gastos_prophet = pd.DataFrame(bkp_gastos)

bkp_gastos_prophet[ 'PERIODO' ] = bkp_gastos_prophet.index
```

Treinando a base de despesas do Ministério da Economia com os valores originais

```
df_prophet_ME = pd.DataFrame(bkp_gastos_prophet, columns=['PERIODO', '25000 - Ministério da E
df_prophet_ME.head()
```

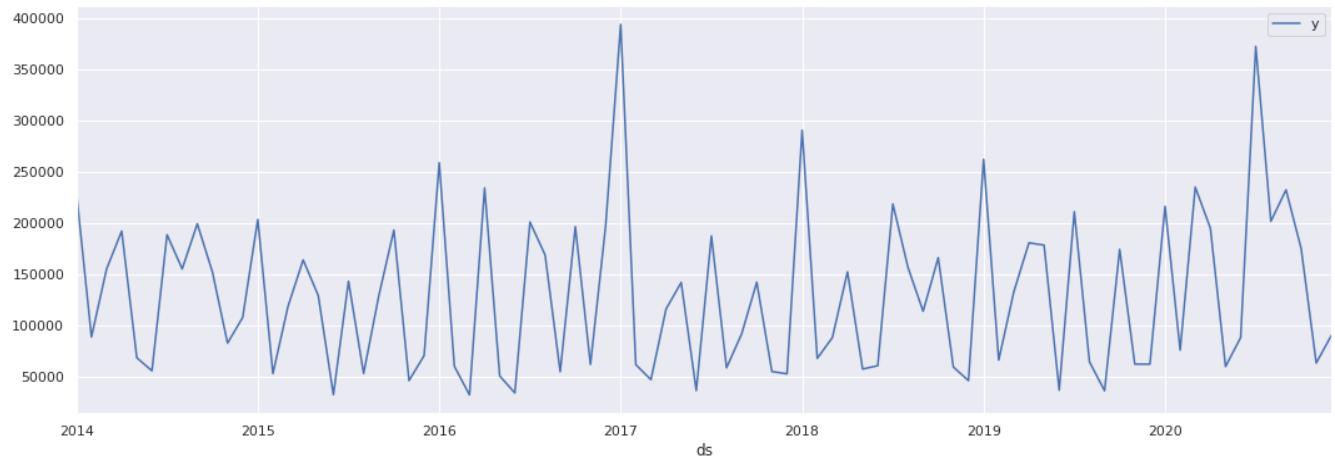
PERIODO 25000 - Ministério da Economia

periodo		
2014-01-01	2014-01-01	230707.875070
2014-02-01	2014-02-01	89260.298717
2014-03-01	2014-03-01	155450.420693
2014-04-01	2014-04-01	192347.415307
2014-05-01	2014-05-01	68952.008458

```
df_prophet_ME.columns = [ 'ds' , 'y' ]
```

```
df_prophet_ME.plot(x='ds',y='y',figsize=(18,6))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc5a8e9ad10>
```



```
len(df_prophet_ME)
```

```
84
```

```
len(df_prophet_ME)-24
```

```
60
```

```
train_prophet_ME = df_prophet_ME.iloc[:60]
test_prophet_ME = df_prophet_ME.iloc[60:]
```

```
m = Prophet()
m.fit(train_prophet_ME)
future = m.make_future_dataframe(periods=24,freq='MS')
forecast = m.predict(future)
```

```
INFO:fbprophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to
```

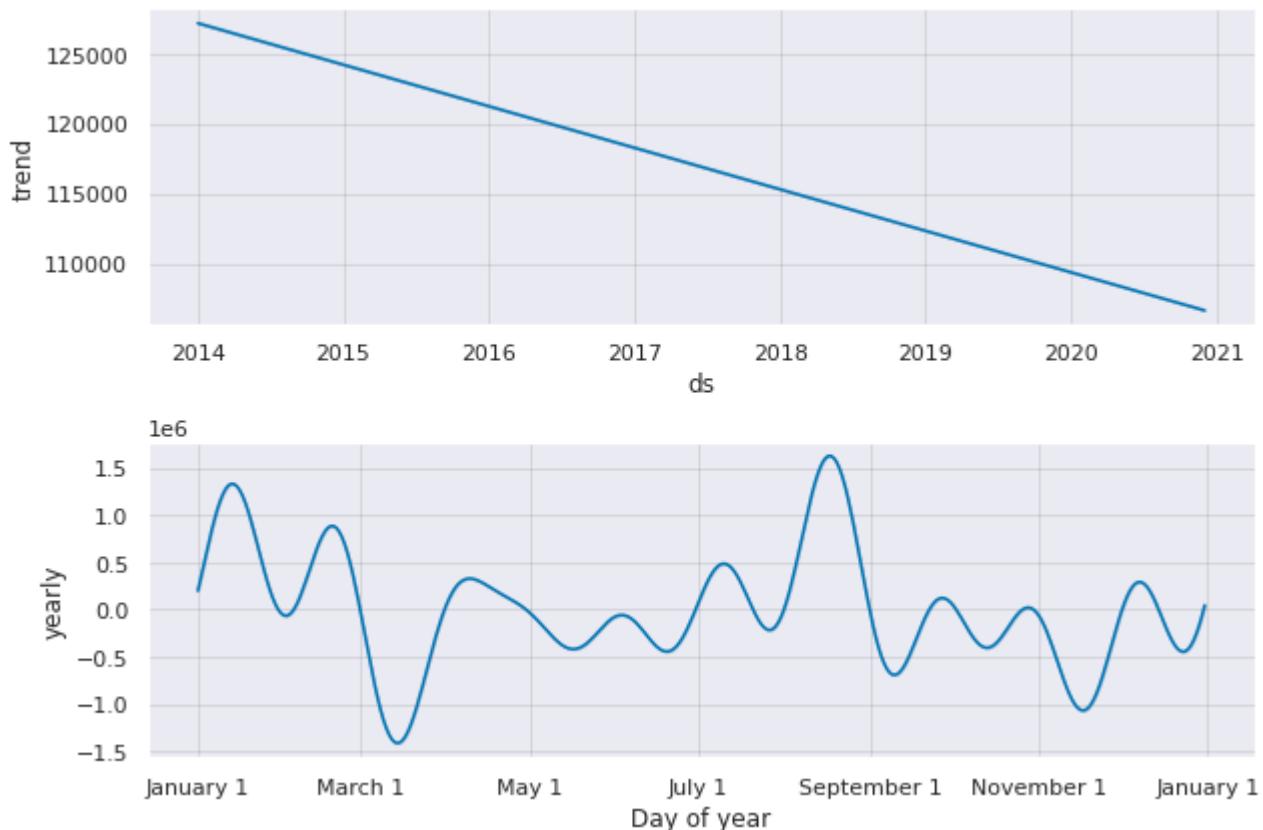
```
forecast.tail()
```

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	add
79	2020-08-01	107648.343626	98716.561529	180308.640753	107648.338730	107648.348791	3
80	2020-09-01	107396.684105	8718.789894	92716.142925	107396.678876	107396.689656	-5

```
test_prophet_ME.tail()
```

	ds	y
periodo		
2020-08-01	2020-08-01	201871.600921
2020-09-01	2020-09-01	232582.689376
2020-10-01	2020-10-01	175522.286653
2020-11-01	2020-11-01	63872.607726
2020-12-01	2020-12-01	90715.459922

```
fig = m.plot_components(forecast)
```



```
ax = forecast.plot(x='ds',y='yhat',label='Previsão',legend=True,figsize=(12,8))
```

```
test_prophet_ME.plot(x='ds',y='y',label='Real',legend=True,ax=ax,xlim=('2019-01-01','2020-12-
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc5a693ee50>
```



```
predictions = forecast.iloc[-24:]['yhat']
predictions
```

60	247992.375457
61	60621.791540
62	133091.595102
63	137987.360399
64	90173.164008
65	30450.911205
66	153931.603811
67	81593.702778
68	159580.799114
69	173698.329791
70	67258.323081
71	55107.264821
72	214591.581627
73	63812.806312
74	15836.477700
75	188230.210459
76	67736.520321
77	38039.192533
78	204843.372247
79	140661.992875
80	49391.869049
81	143819.579294

```
82      32221.677320
83      118382.773687
Name: yhat, dtype: float64
```

```
test_prophet_ME['y']
```

```
periodo
2019-01-01    262311.577100
2019-02-01    66735.795897
2019-03-01   133261.769754
2019-04-01   180928.981673
2019-05-01   178711.090419
2019-06-01   37437.501680
2019-07-01   211274.310567
2019-08-01   65069.030704
2019-09-01   36874.389777
2019-10-01   174570.988237
2019-11-01   62785.946605
2019-12-01   62743.048784
2020-01-01   216384.840962
2020-02-01   76381.468487
2020-03-01   235227.946781
2020-04-01   195099.213575
2020-05-01   60395.926599
2020-06-01   88585.647561
2020-07-01   372229.062296
2020-08-01   201871.600921
2020-09-01   232582.689376
2020-10-01   175522.286653
2020-11-01   63872.607726
2020-12-01   90715.459922
Freq: MS, Name: y, dtype: float64
```

```
rmse(predictions,test_prophet_ME['y'])
```

```
78453.0094259957
```

```
test_prophet_ME.mean()
```

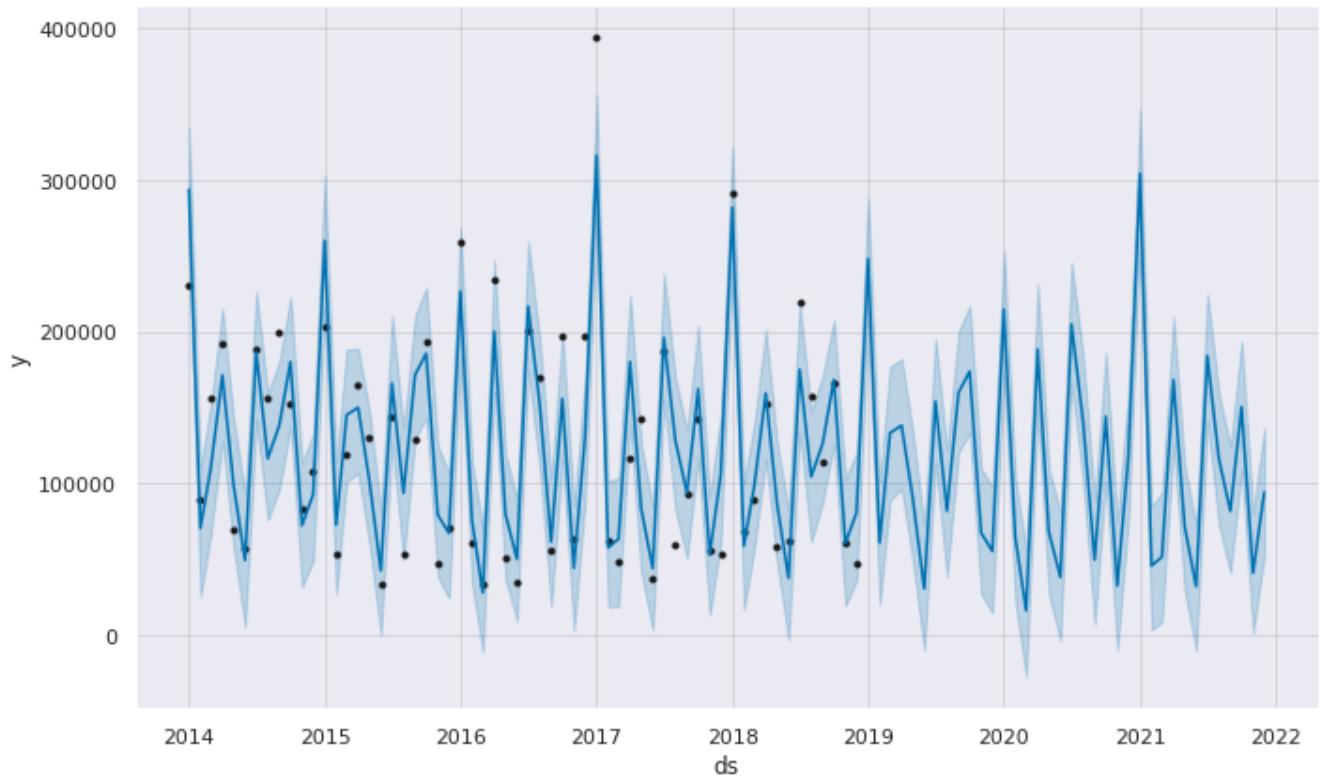
```
y      145065.549252
dtype: float64
```

Previsão de futuro da base de despesas do Ministério da Economia com os valores originais

```
m = Prophet()
m.fit(train_prophet_ME)
future = m.make_future_dataframe(periods=36,freq='MS')
forecast = m.predict(future)
```

INFO:fbprophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to

```
m.plot(forecast);
```



```
ax = forecast.plot(x='ds',y='yhat',label='Previsão',legend=True,figsize=(12,8))
```

```
test_prophet_ME.plot(x='ds',y='y',label='Real',legend=True,ax=ax,xlim=('2019-01-01','2021-12-
```

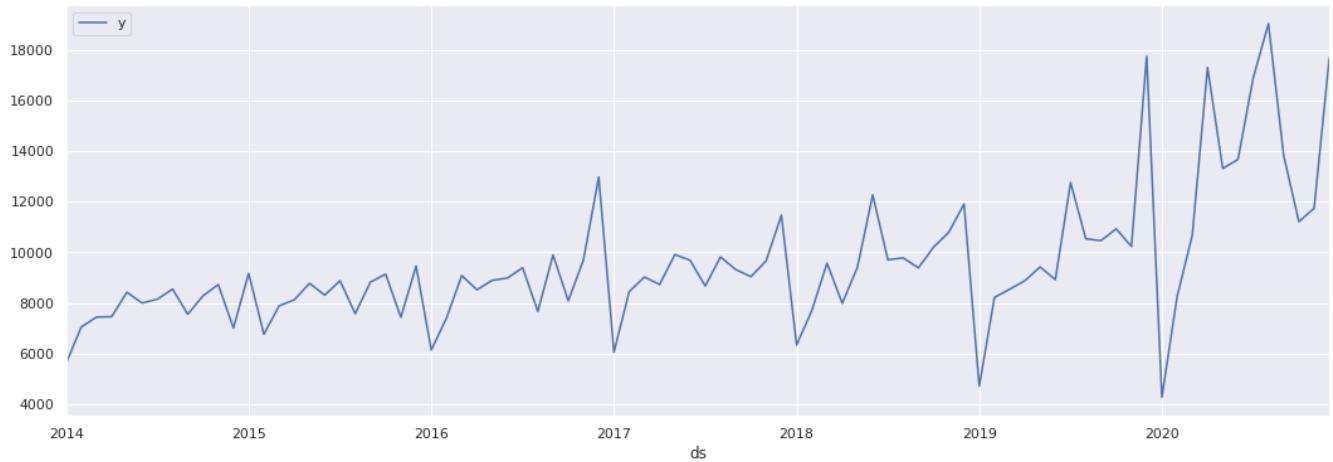
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc5a6a322d0>
```



Treinando a base de despesas do Ministério da Saúde com os valores originais

```
df_prophet_MS = pd.DataFrame(bkp_gastos_prophet, columns=['PERÍODO', '36000 - Ministério da Saúde'])
df_prophet_MS.columns = ['ds', 'y']
df_prophet_MS.plot(x='ds', y='y', figsize=(18,6))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc5a6a28f50>
```

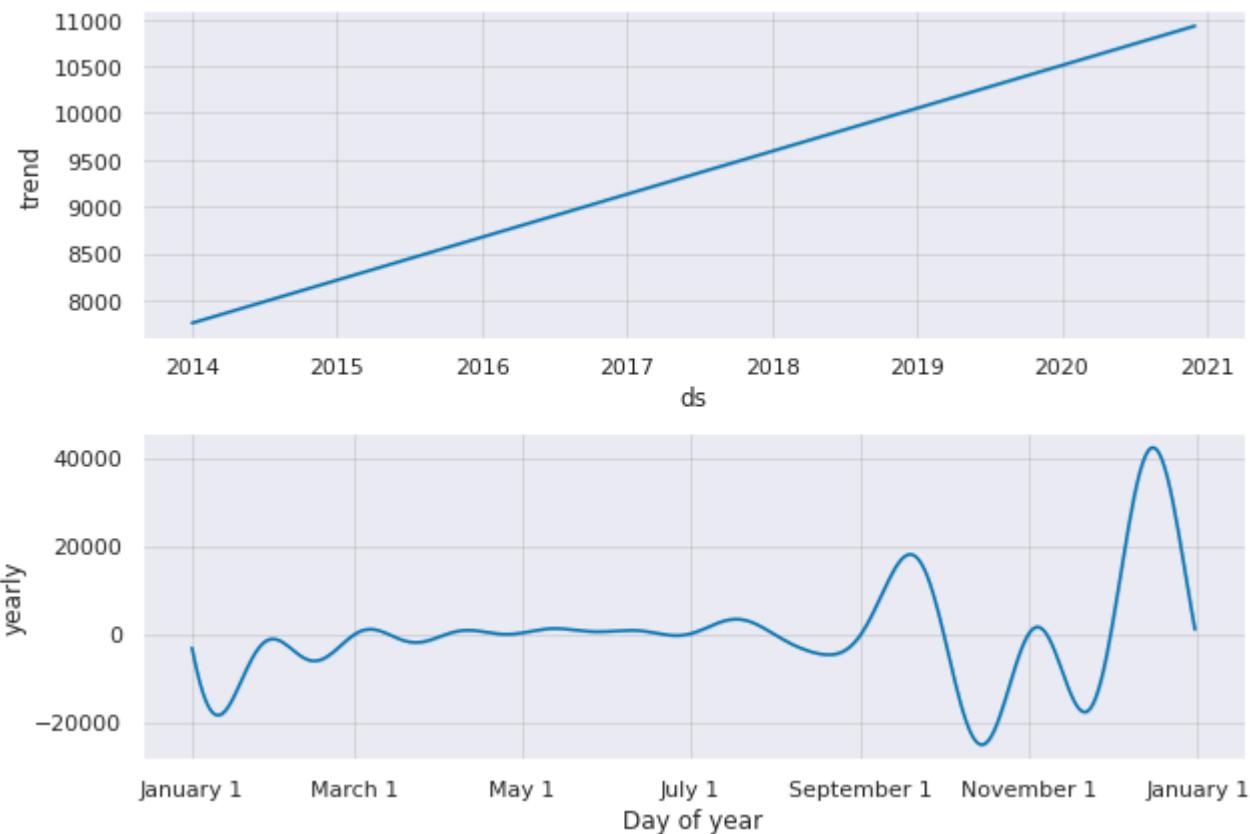


```
train_prophet_MS = df_prophet_MS.iloc[:60]
test_prophet_MS = df_prophet_MS.iloc[60:]
```

```
m = Prophet()
m.fit(train_prophet_MS)
future = m.make_future_dataframe(periods=24, freq='MS')
```

```
forecast = m.predict(future)
fig = m.plot_components(forecast)
```

INFO:fbprophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to



```
ax = forecast.plot(x='ds',y='yhat',label='Previsão',legend=True,figsize=(12,8))
test_prophet_MS.plot(x='ds',y='y',label='Real',legend=True,ax=ax,xlim=('2019-01-01','2020-12-
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc5a6ade710>
```



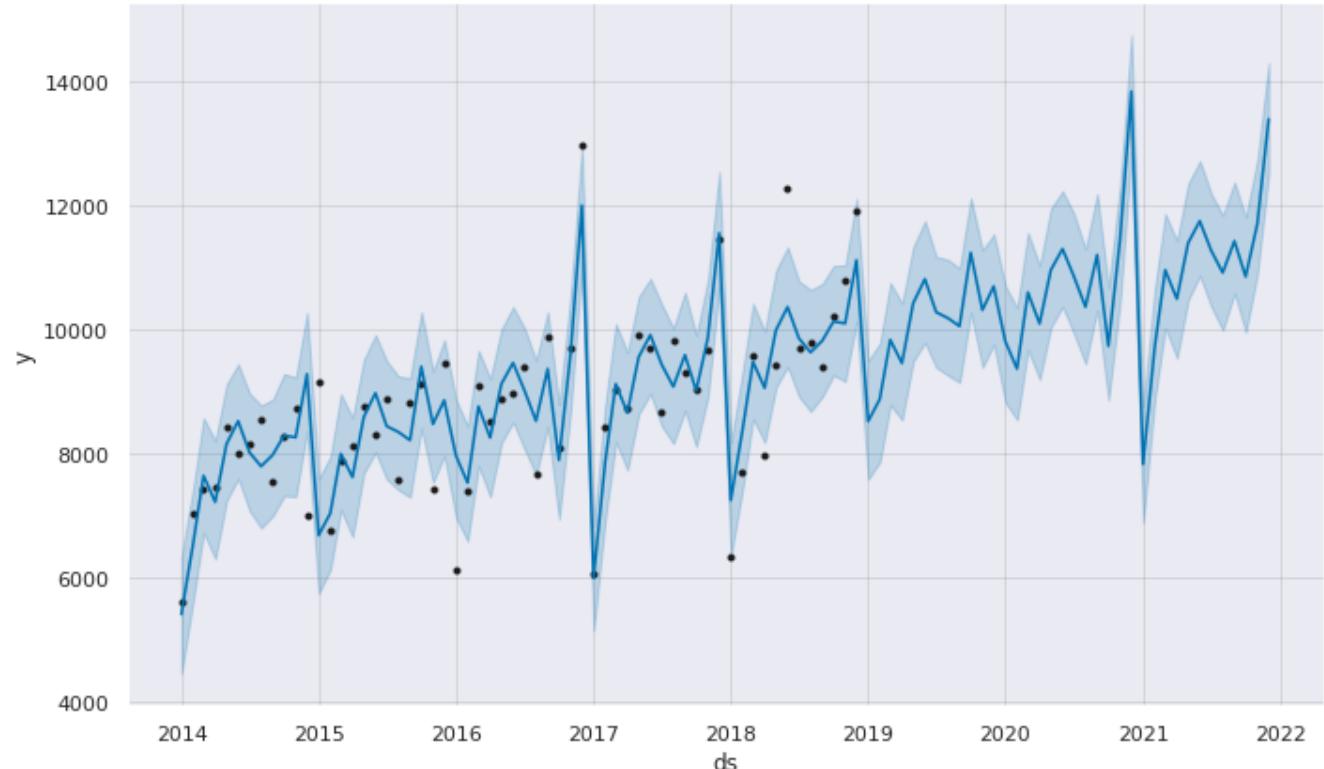
```
rmse(predictions,test_prophet_MS['y'])
```

```
118865.27598137749
```

▼ Previsão de futuro da base de despesas do Ministério da Saúde com os valores originais

```
m = Prophet()
m.fit(train_prophet_MS)
future = m.make_future_dataframe(periods=36, freq='MS')
forecast = m.predict(future)
m.plot(forecast);
```

```
INFO:fbprophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to
```



```
ax = forecast.plot(x='ds',y='yhat',label='Previsão',legend=True,figsize=(12,8))
test_prophet_MS.plot(x='ds',y='y',label='Real',legend=True,ax=ax,xlim=('2019-01-01','2021-12-
```

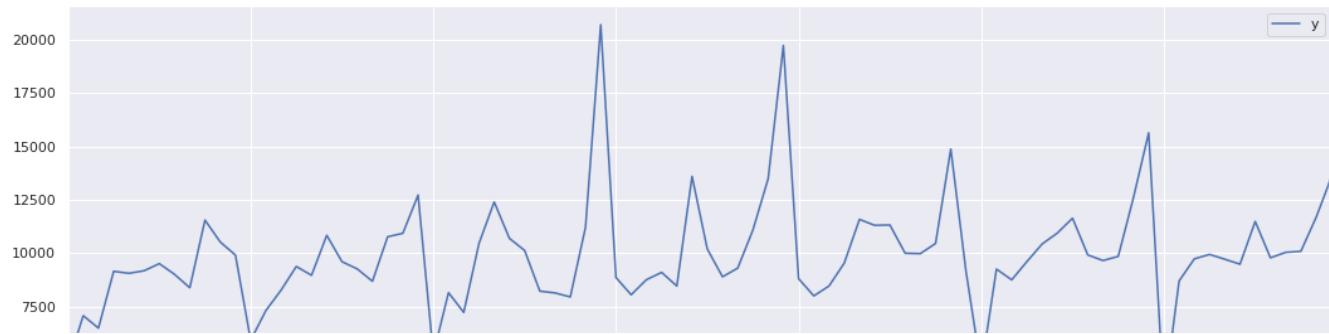
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc5a6569310>
```



Treinando a base de despesas do Ministério da Educação com os valores originais

```
df_prophet_MEd = pd.DataFrame(bkp_gastos_prophet, columns=['PERIODO', '26000 - Ministério da  
df_prophet_MEd.columns = ['ds','y']  
df_prophet_MEd.plot(x='ds',y='y',figsize=(18,6))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc5a75162d0>
```

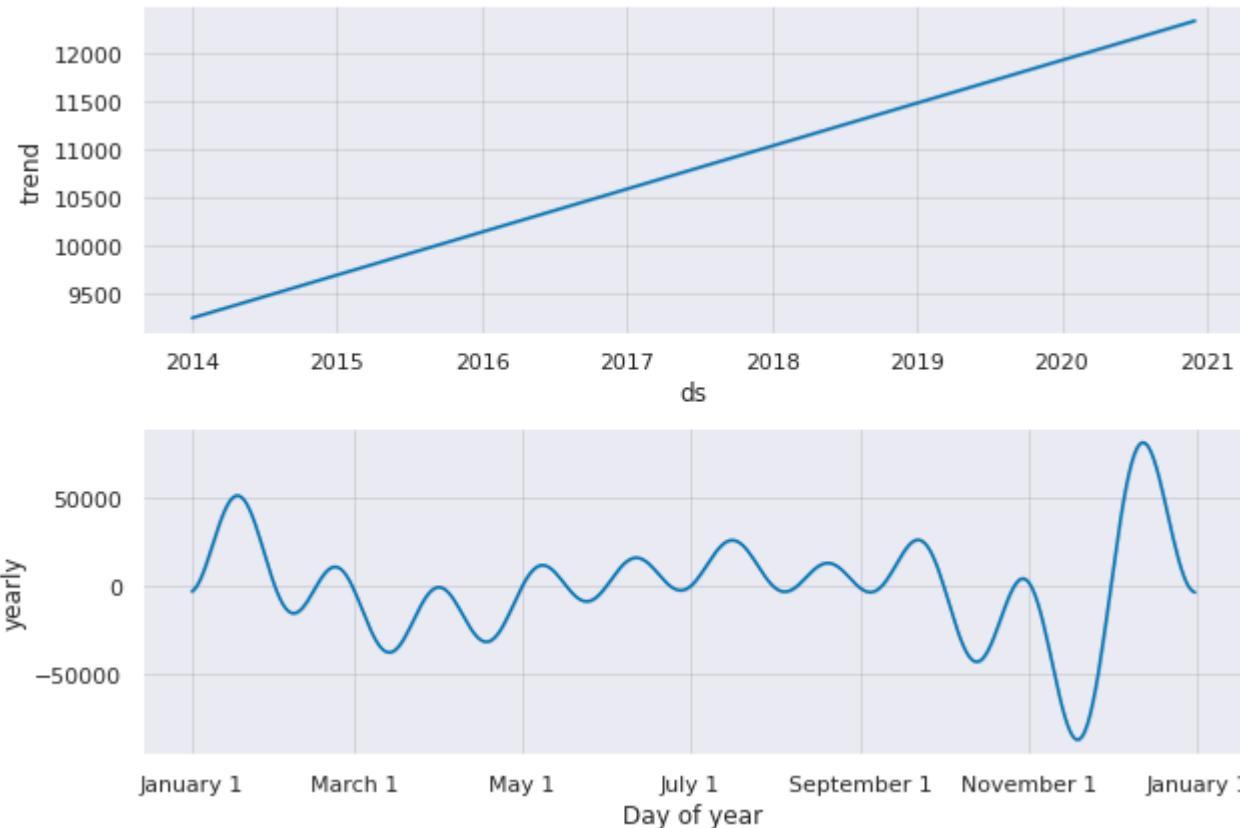


```
train_prophet_MEd = df_prophet_MEd.iloc[:60]
test_prophet_MEd = df_prophet_MEd.iloc[60:]
```

ds

```
m = Prophet()
m.fit(train_prophet_MEd)
future = m.make_future_dataframe(periods=24, freq='MS')
forecast = m.predict(future)
fig = m.plot_components(forecast)
```

INFO:fbprophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to



```
ax = forecast.plot(x='ds',y='yhat',label='Previsão',legend=True,figsize=(12,8))
test_prophet_MEd.plot(x='ds',y='y',label='Real',legend=True,ax=ax,xlim=('2019-01-01','2020-12-31'))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc5a6a153d0>
```



```
rmse(predictions,test_prophet_MEd[ 'y' ])
```

```
120572.29990218856
```

▼ Previsão de futuro da base de despesas do Ministério da Educação com os valores originais

```
m = Prophet()
m.fit(train_prophet_MEd)
future = m.make_future_dataframe(periods=36,freq='MS')
forecast = m.predict(future)
m.plot(forecast);
```

INFO:fbprophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to enable.
 INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to enable.



```
ax = forecast.plot(x='ds',y='yhat',label='Previsão',legend=True,figsize=(12,8))
test_prophet_MEd.plot(x='ds',y='y',label='Real',legend=True,ax=ax,xlim=('2019-01-01','2021-12-31'))
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc5a6492590>





0s conclusão: 11:54

