```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
```

Precio del Dolar

```
df_dolar = pd.read_excel(r'tipo-de-cambio.xlsx')
print(df_dolar.head())

df_dolar = df_dolar[["Fecha",
                     "Cambio a 48 horas Cierre Compra"]].rename(columns={df_dolar.co
                         df_dolar.columns[1]:"dolar"})

plt.plot(df_dolar["Fecha"], df_dolar["dolar"])
plt.title("Precio del Dolar")
```

```
       Fecha  Cambio a 48 horas Cierre Compra  Cambio a 48 horas Cierre Venta
0 1992-01-02                           3.0745                          3.0755
1 1992-01-03                           3.0723                          3.0733
2 1992-01-06                           3.0673                          3.0683
3 1992-01-07                           3.0650                          3.0660
4 1992-01-08                           3.0650                          3.0660
Text(0.5, 1.0, 'Precio del Dolar')
```

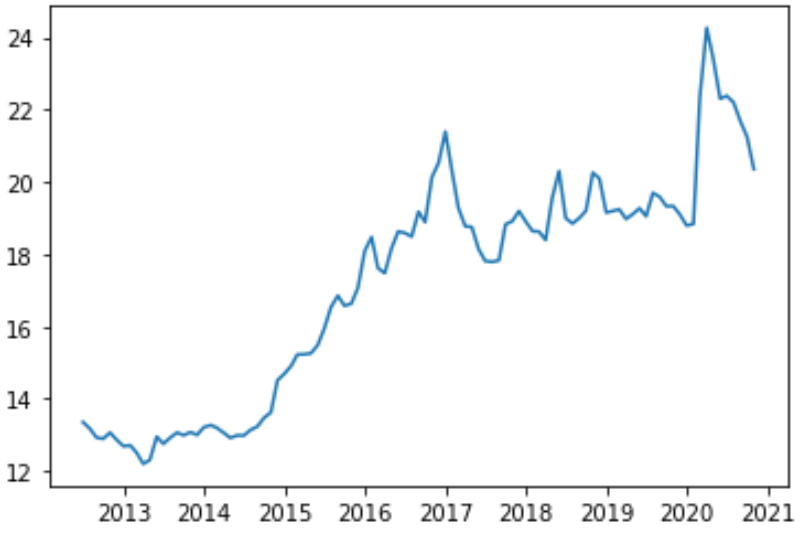

```
### Estadisticas de estos datos:
```

```
stat_dolar = df_dolar['dolar'].describe()
stat_dolar
```

```
count    7500.000000
mean       11.823985
std         4.888754
min         3.060000
25%         9.245000
50%        11.037000
75%        13.687350
max        25.105000
Name: dolar, dtype: float64
```

```
# Tambien podemos sacar los datos del dolar pero promediados por mes para hacerlos m
df_dolar_mes = df_dolar.groupby(pd.PeriodIndex(df_dolar['Fecha'],
                                freq="M"))['dolar'].mean().reset_index()
fecha_i = pd.Timestamp('2012-07-01')
fecha_f = pd.Timestamp('2020-11-01')


df_dolar_mes['Fecha'] = [pd.Timestamp(str(j)) for j in df_dolar_mes["Fecha"]]

df_dolar_mes = df_dolar_mes[(df_dolar_mes['Fecha']>=fecha_i) & (df_dolar_mes['Fecha'

plt.plot(df_dolar_mes['Fecha'],df_dolar_mes['dolar'])
```

```
[<matplotlib.lines.Line2D at 0x7fc802e4eb50>]
```




```
# Tambien podemos sacar los datos del dolar pero promediados por mes para hacerlos m
df_oil = pd.read_excel(r'Crude-Oil.xlsx')
print(df_oil.head())

df_oil_mes = df_oil.groupby(pd.PeriodIndex(df_oil['Fecha'],
```

```
                                freq="M"))['Crude Oil Price'].mean().reset_index()

fecha_i = pd.Timestamp('2012-07-01')
fecha_f = pd.Timestamp('2020-11-01')


df_oil_mes['Fecha'] = [pd.Timestamp(str(j)) for j in df_oil_mes["Fecha"]]

df_oil_mes = df_oil_mes[(df_oil_mes['Fecha']>=fecha_i) & (df_oil_mes['Fecha']<=fecha_

plt.plot(df_oil_mes['Fecha'],df_oil_mes['Crude Oil Price'])
```

```
        Fecha  Crude Oil Price
0 1986-01-02            25.56
1 1986-01-03            26.00
2 1986-01-06            26.53
3 1986-01-07            25.85
4 1986-01-08            25.87
[<matplotlib.lines.Line2D at 0x7fc7fd04aa10>]
```


```
# Tambien podemos sacar los datos del dolar pero promediados por mes para hacerlos m
df_petroleo = pd.read_excel(r'Petroleo.xlsx')
df_petroleo.replace(to_replace = "N/E",value = np.nan, inplace=True)
print(df_petroleo.head())

df_petroleo_mes = df_petroleo.groupby(pd.PeriodIndex(df_petroleo['Fecha'],
                                freq="M"))['Precio del Petroleo Mexicano'].mean().rese

fecha_i = pd.Timestamp('2012-07-01')
fecha_f = pd.Timestamp('2020-11-01')


df_petroleo_mes['Fecha'] = [pd.Timestamp(str(j)) for j in df_petroleo_mes["Fecha"]]
```

```
df_petroleo_mes = df_petroleo_mes[(df_petroleo_mes['Fecha']>=fecha_i) & (df_petroleo_

plt.plot(df_petroleo_mes['Fecha'],df_petroleo_mes['Precio del Petroleo Mexicano'])
df_petroleo_mes.head()
```

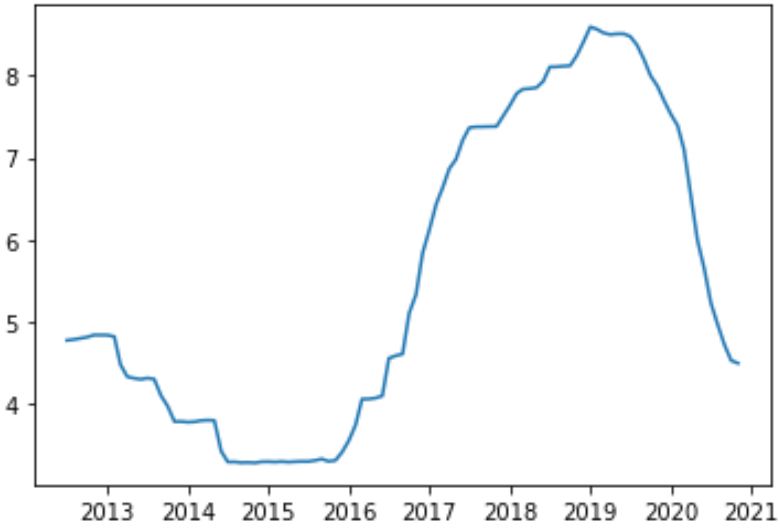
```
       Fecha  Precio del Petroleo Mexicano
0 1996-01-01                           NaN
1 1996-01-02                           NaN
2 1996-01-03                         17.40
3 1996-01-04                         17.41
4 1996-01-05                         17.70
```

	Fecha	Precio del Petroleo Mexicano
198	2012-07-01	94.661905
199	2012-08-01	101.877500
200	2012-09-01	102.696842
201	2012-10-01	99.875000
202	2012-11-01	94.886667


```
# Tambien podemos sacar los datos del dolar pero promediados por mes para hacerlos m
df_interes = pd.read_excel(r'interes.xlsx')
# Reemplazamos los "N/E" que vienen en los datos por nan
# y nos quedamos solamente con la TIIE a 28 dias
df_interes.replace(to_replace = "N/E",value = np.nan, inplace=True)
df_interes = df_interes[['Fecha', 'TIIE a 28 dias']]

df_interes_mes = df_interes.groupby(pd.PeriodIndex(df_interes['Fecha'],
                                freq="M"))['TIIE a 28 dias'].mean().reset_index()

fecha_i = pd.Timestamp('2012-07-01')
```

```
fecha_f = pd.Timestamp('2020-11-01')


df_interes_mes['Fecha'] = [pd.Timestamp(str(j)) for j in df_interes_mes["Fecha"]]

df_interes_mes = df_interes_mes[(df_interes_mes['Fecha']>=fecha_i) & (df_interes_mes

plt.plot(df_interes_mes['Fecha'],df_interes_mes['TIIE a 28 dias'])
df_interes_mes.head()
```

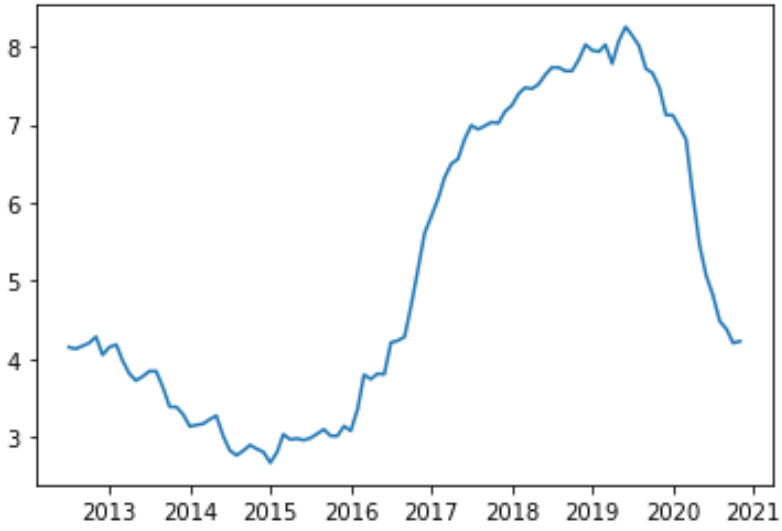
	Fecha	TIIE a 28 dias
208	2012-07-01	4.777891
209	2012-08-01	4.788165
210	2012-09-01	4.802305
211	2012-10-01	4.816974
212	2012-11-01	4.842805




```
df_cetes = pd.read_excel(r'cetes.xlsx')

df_cetes_mes = df_cetes.groupby(pd.PeriodIndex(df_cetes['Fecha'],
                                freq="M"))['Cetes a 28 Dias'].mean().reset_index()

fecha_i = pd.Timestamp('2012-07-01')
fecha_f = pd.Timestamp('2020-11-01')


df_cetes_mes['Fecha'] = [pd.Timestamp(str(j)) for j in df_cetes_mes["Fecha"]]
```

```
df_cetes_mes = df_cetes_mes[(df_cetes_mes['Fecha']>=fecha_i) & (df_cetes_mes['Fecha'

plt.plot(df_cetes_mes['Fecha'],df_cetes_mes['Cetes a 28 Dias'])
df_cetes_mes.head()
```

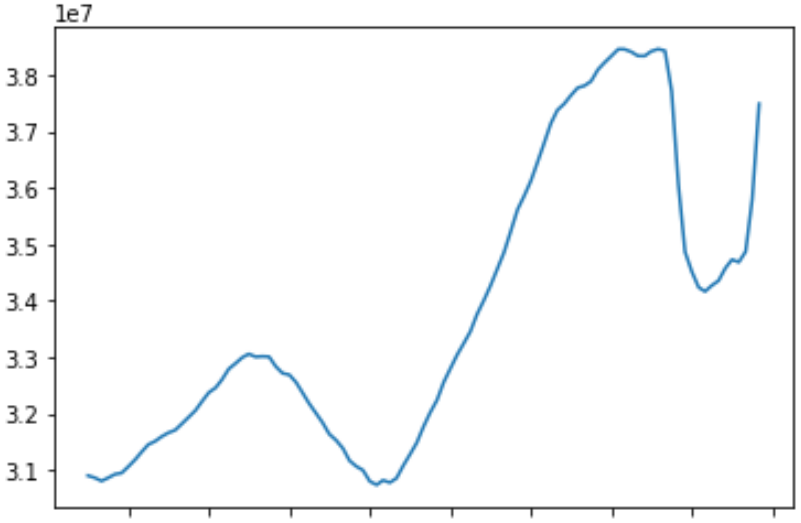
	Fecha	Cetes a 28 Dias
329	2012-07-01	4.1525
330	2012-08-01	4.1300
331	2012-09-01	4.1675
332	2012-10-01	4.2050
333	2012-11-01	4.2860




```
df_exportaciones = pd.read_excel(r'Exportaciones.xlsx')
result=seasonal_decompose(df_exportaciones['Exportaciones'], model='additive', perio
df_exportaciones['Trend Exportaciones'] = result.trend
df_exportaciones_trend_mes = df_exportaciones.groupby(pd.PeriodIndex(df_exportacione
                             freq="M"))['Trend Exportaciones'].mean().reset_index()


df_exportaciones_trend_mes['Fecha'] = [pd.Timestamp(str(j)) for j in df_exportacione

print(df_exportaciones_trend_mes.head())
df_exportaciones_trend_mes = df_exportaciones_trend_mes[(df_exportaciones_trend_mes[
plt.plot(df_exportaciones_trend_mes['Fecha'],df_exportaciones_trend_mes['Trend Expor
df_exportaciones_trend_mes.head()
```

```
       Fecha  Trend Exportaciones
0 1993-01-01                  NaN
1 1993-02-01                  NaN
2 1993-03-01                  NaN
3 1993-04-01                  NaN
4 1993-05-01                  NaN
```

	Fecha	Trend Exportaciones
234	2012-07-01	3.089820e+07
235	2012-08-01	3.086041e+07
236	2012-09-01	3.080217e+07
237	2012-10-01	3.085861e+07
238	2012-11-01	3.092272e+07

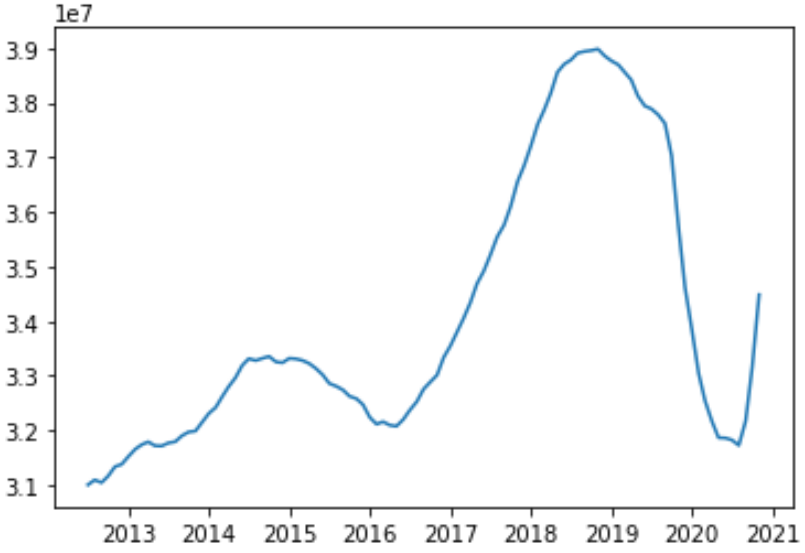



```
df_exportaciones = pd.read_excel(r'Exportaciones.xlsx')
result=seasonal_decompose(df_exportaciones['Exportaciones'], model='additive', perio
df_exportaciones['Trend Exportaciones'] = result.trend
df_exportaciones_trend_mes = df_exportaciones.groupby(pd.PeriodIndex(df_exportacione
                             freq="M"))['Trend Exportaciones'].mean().reset_index()


df_exportaciones_trend_mes['Fecha'] = [pd.Timestamp(str(j)) for j in df_exportacione

print(df_exportaciones_trend_mes.head())
df_exportaciones_trend_mes = df_exportaciones_trend_mes[(df_exportaciones_trend_mes[
plt.plot(df_exportaciones_trend_mes['Fecha'],df_exportaciones_trend_mes['Trend Expor
df_exportaciones_trend_mes.head()



df_importaciones = pd.read_excel(r'Importaciones.xlsx')
```

```
result=seasonal_decompose(df_importaciones['Importaciones'], model='additive', perio
df_importaciones['Trend Importaciones'] = result.trend
df_importaciones_trend_mes = df_importaciones.groupby(pd.PeriodIndex(df_importacione
                             freq="M"))['Trend Importaciones'].mean().reset_index()


df_importaciones_trend_mes['Fecha'] = [pd.Timestamp(str(j)) for j in df_importacione

print(df_importaciones_trend_mes.head())
df_importaciones_trend_mes = df_importaciones_trend_mes[(df_importaciones_trend_mes[
plt.plot(df_importaciones_trend_mes['Fecha'],df_importaciones_trend_mes['Trend Impor
df_importaciones_trend_mes.head()
```

```
       Fecha  Trend Importaciones
0 1993-01-01                  NaN
1 1993-02-01                  NaN
2 1993-03-01                  NaN
3 1993-04-01                  NaN
4 1993-05-01                  NaN
```

	Fecha	Trend Importaciones
234	2012-07-01	3.100446e+07
235	2012-08-01	3.109416e+07
236	2012-09-01	3.104571e+07
237	2012-10-01	3.116630e+07
238	2012-11-01	3.133753e+07



Haz doble clic (o ingresa) para editar

Variables

```
df = pd.read_excel(r'lab1 mensuales.xlsx')

df = df.rename(columns={df.columns[0]:"Fecha"})
df = df.set_index('Fecha')

df = pd.merge(df, df_oil_mes, on="Fecha",how="left")
df = pd.merge(df, df_petroleo_mes, on="Fecha",how="left")
df = pd.merge(df, df_cetes_mes, on="Fecha",how="left")
df = pd.merge(df, df_interes_mes, on="Fecha",how="left")
df = pd.merge(df, df_exportaciones_trend_mes, on="Fecha",how="left")
df = pd.merge(df, df_importaciones_trend_mes, on="Fecha",how="left")


df = pd.merge(df, df_dolar_mes, on="Fecha",how="left")


df = df.set_index('Fecha')

df.tail()
```

	confianza	exportaciones	importaciones	incertidumbre política	INPC Mensual	Infla Subyac Men
Fecha						

```
#Normalizamos los datos
for a in df.columns:
  df[a] = (df[a]-np.min(df[a]))/ (np.max(df[a]) - np.min(df[a]))
df.head()
```

	confianza	exportaciones	importaciones	incertidumbre política	INPC Mensual	Infla Subyac Mer
Fecha						
2012-07-01	0.584822	0.510949	0.401792	0.18	0.579336	0.41
2012-08-01	0.591384	0.568249	0.487114	0.20	0.483395	0.32
2012-09-01	0.525223	0.468375	0.329829	0.17	0.535055	0.27
2012-10-01	0.578894	0.663895	0.615805	0.13	0.560886	0.33
2012-11-01	0.566421	0.560227	0.490390	0.10	0.623616	0.13

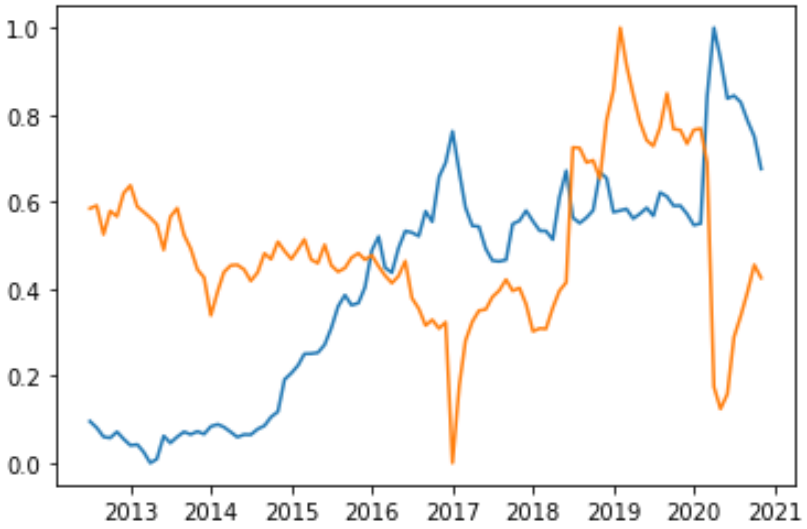
```
# Estadistica descriptiva
df.describe()
```

	confianza	exportaciones	importaciones	incertidumbre política	INPC Mensual	In Sub
count	101.000000	101.000000	101.000000	101.000000	101.000000	101
mean	0.496308	0.657719	0.549298	0.227426	0.492894	0
std	0.179545	0.170270	0.161462	0.189165	0.136236	0
min	0.000000	0.000000	0.000000	0.000000	0.000000	0
25%	0.394694	0.560227	0.441206	0.080000	0.428044	0
50%	0.463806	0.642770	0.540851	0.170000	0.512915	0
75%	0.585466	0.787235	0.634260	0.320000	0.575646	0
max	1.000000	1.000000	1.000000	1.000000	1.000000	1

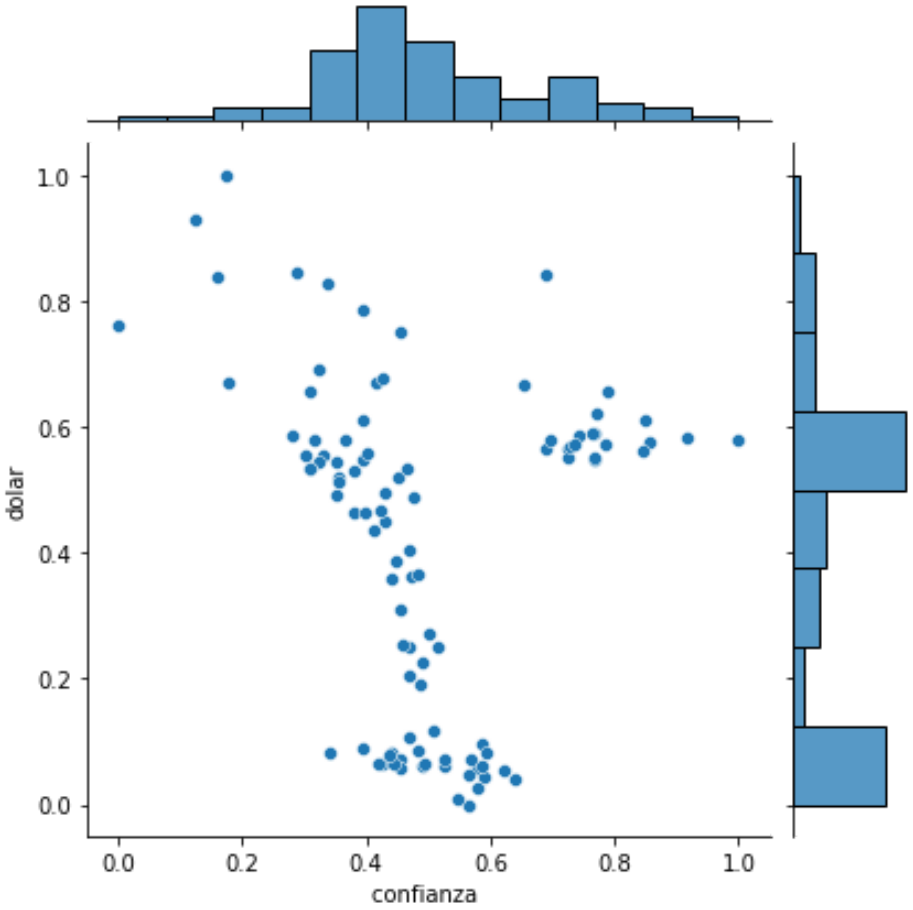
```
for col in df.columns:
  if col == 'dolar':
    continue
  plt.plot(df['dolar'])
  plt.plot(df[col])
  print('Variable: ' + col )
  print('\n  Gráfica como función del tiempo')
  plt.show()

  print('\n Grafica de variable vs dolar e histogramas')
  sns.jointplot(x=df[col], y=df['dolar'])
  plt.show()

  print('\n\n\n')
```

Variable: confianza

Gráfica como función del tiempo

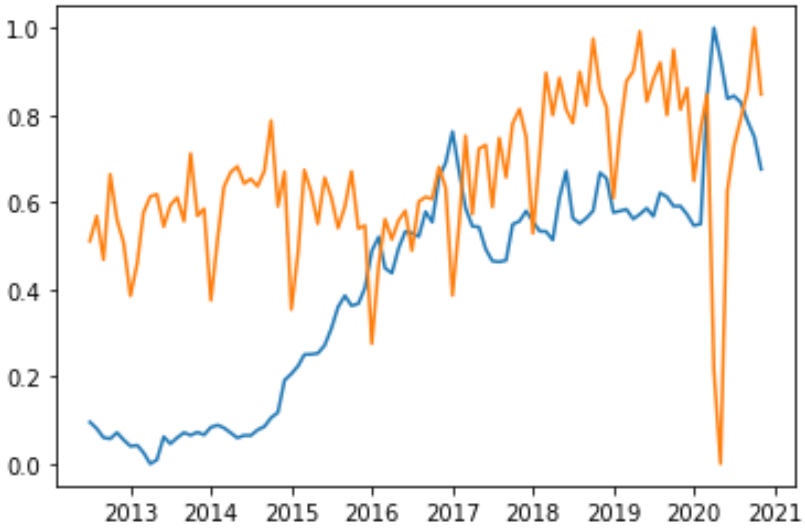


Grafica de variable vs dolar e histogramas

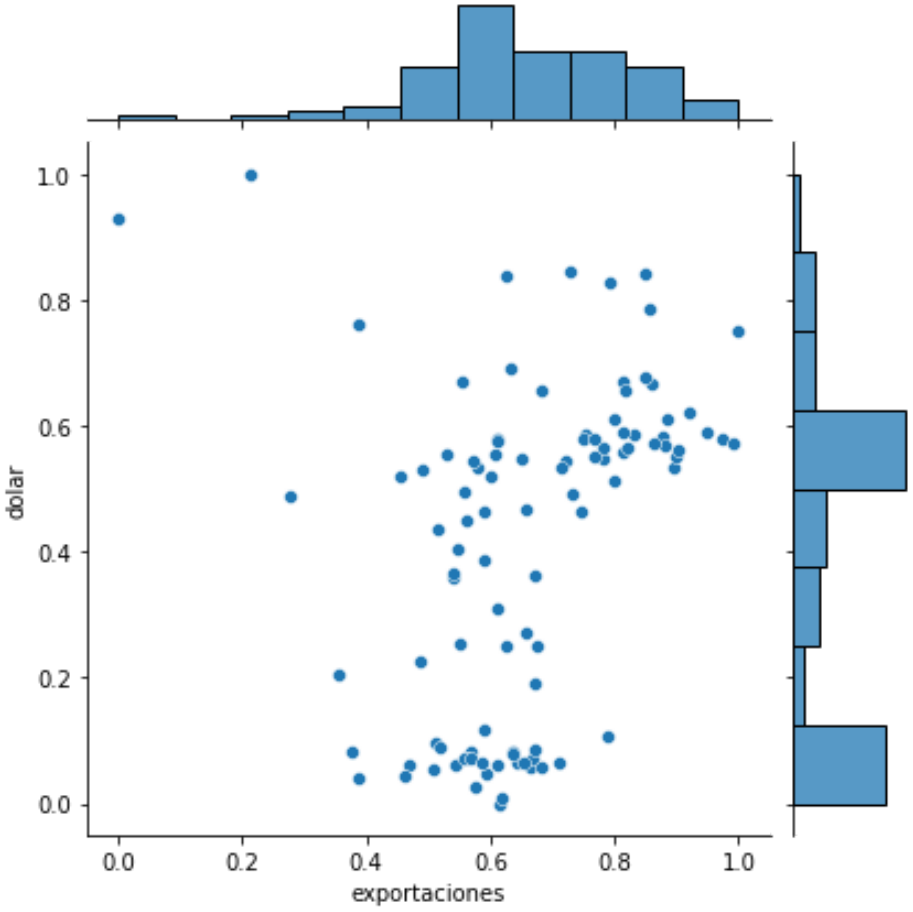


Variable: exportaciones

Gráfica como función del tiempo

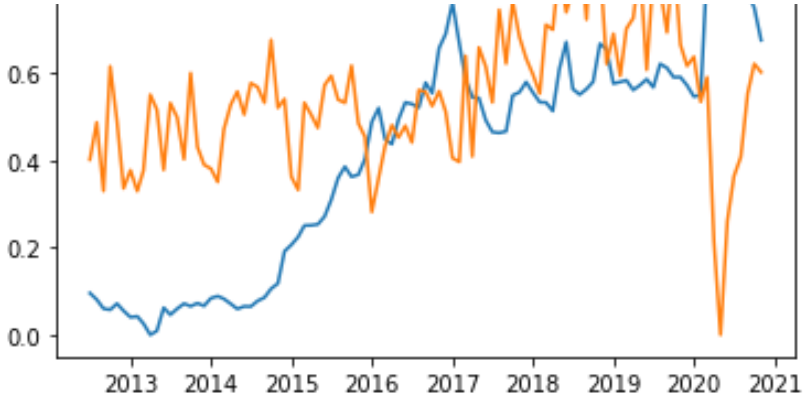


Grafica de variable vs dolar e histogramas

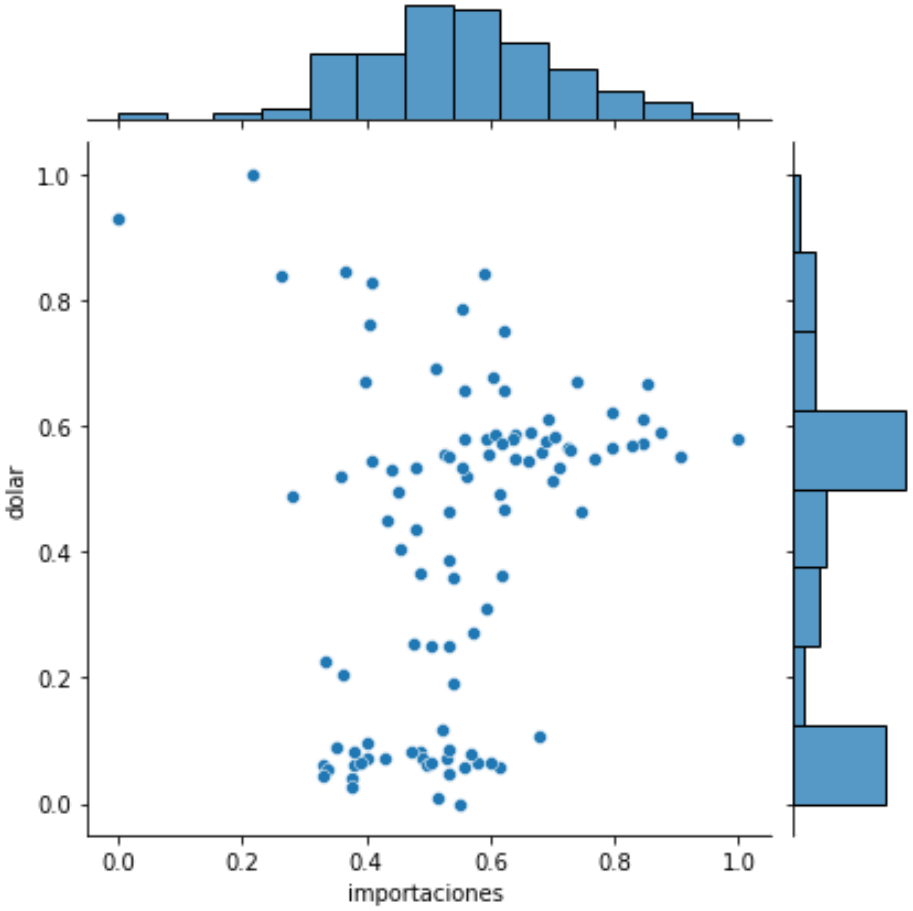


Variable: importaciones

Gráfica como función del tiempo

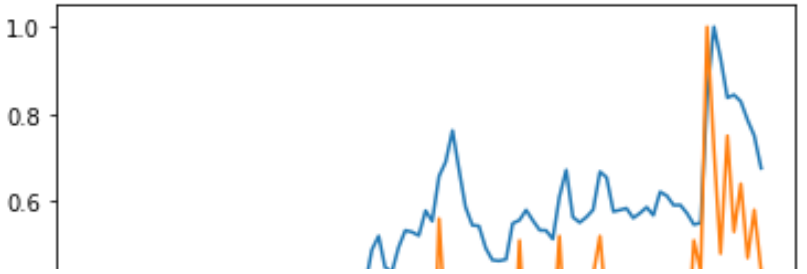


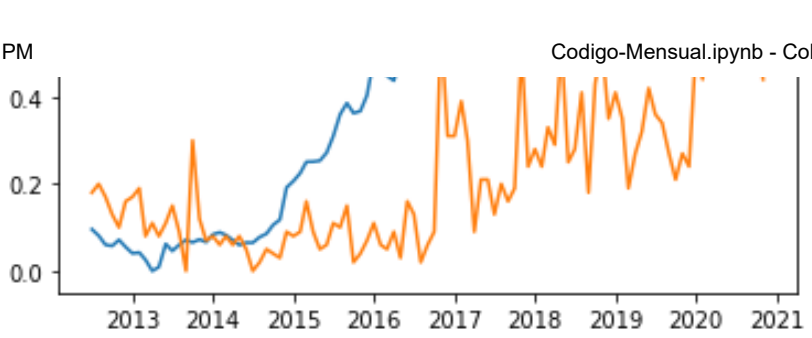
Grafica de variable vs dolar e histogramas



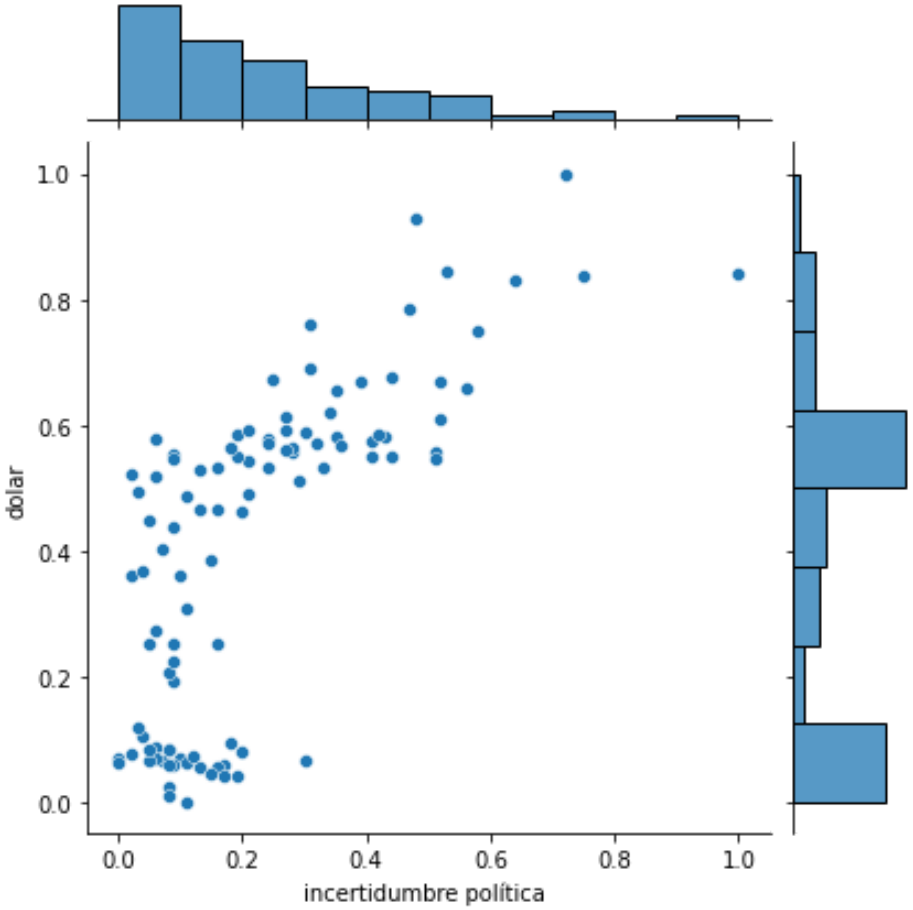
Variable: incertidumbre política

Gráfica como función del tiempo



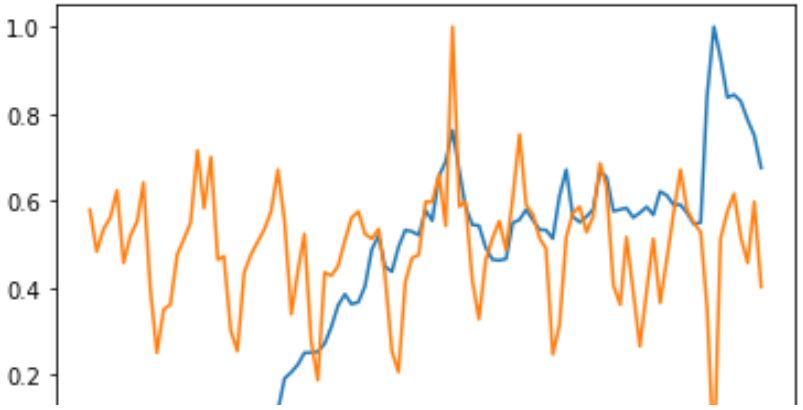


Grafica de variable vs dolar e histogramas

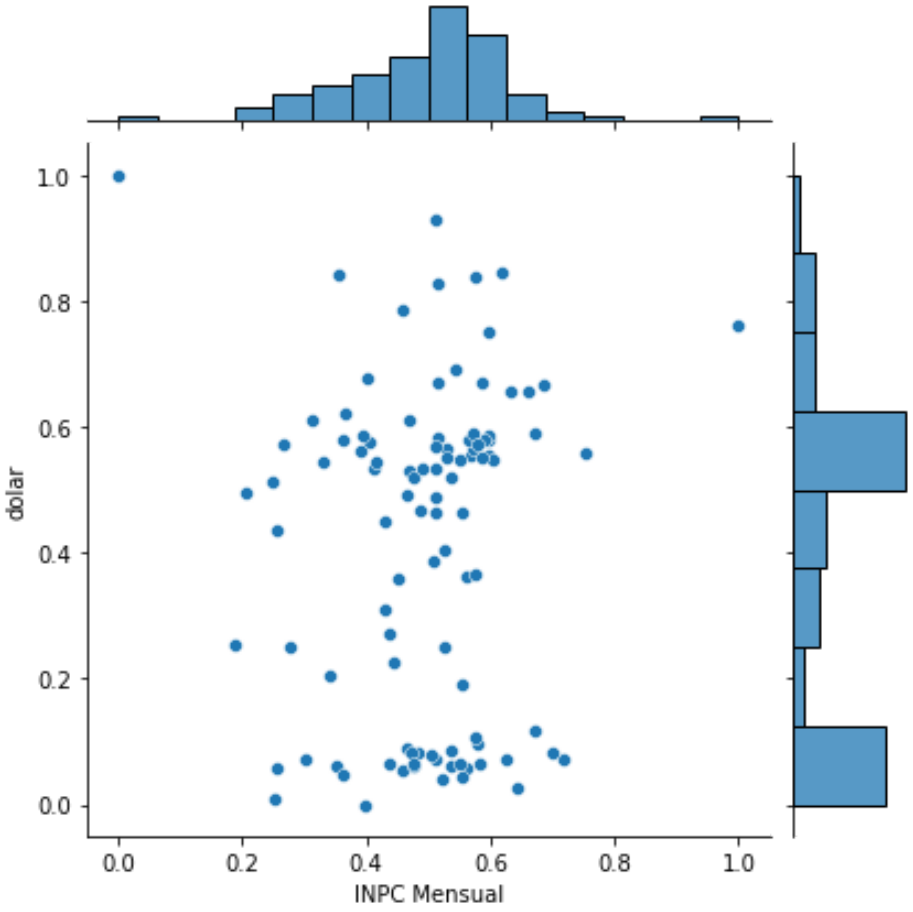


Variable: INPC Mensual

Gráfica como función del tiempo

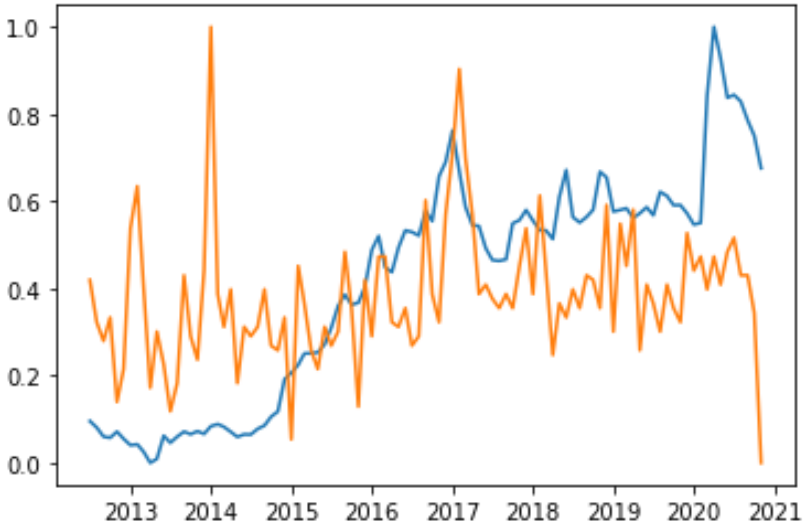


Grafica de variable vs dolar e histogramas

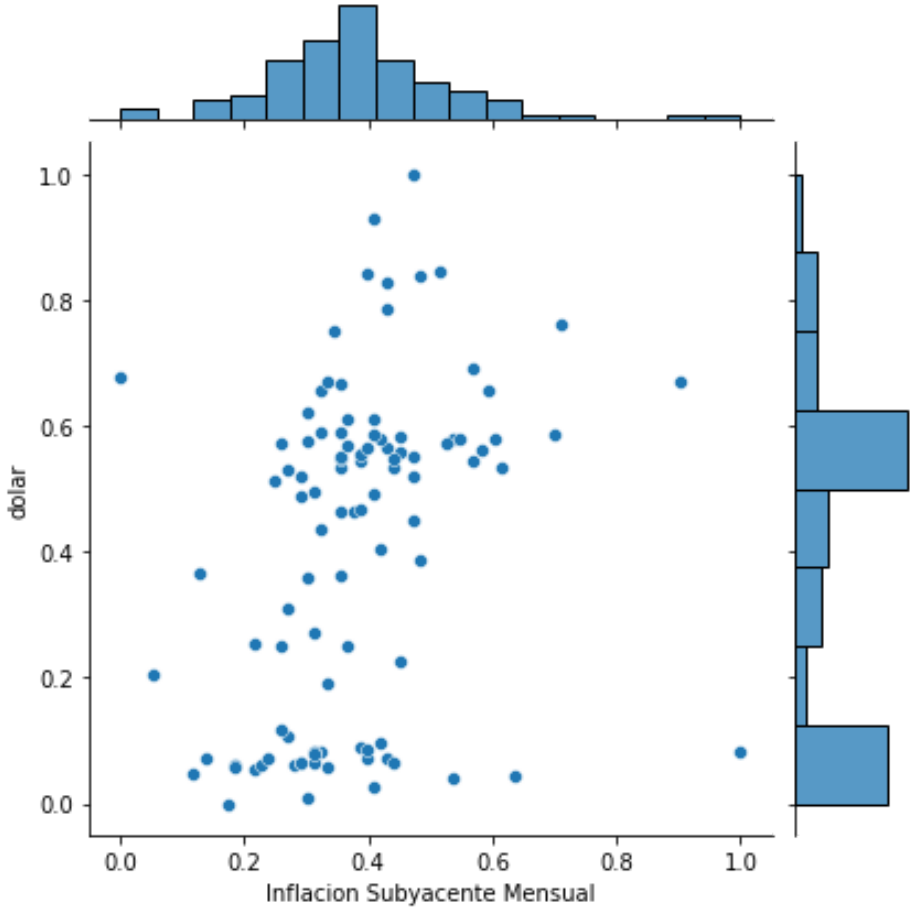


Variable: Inflacion Subyacente Mensual

Gráfica como función del tiempo

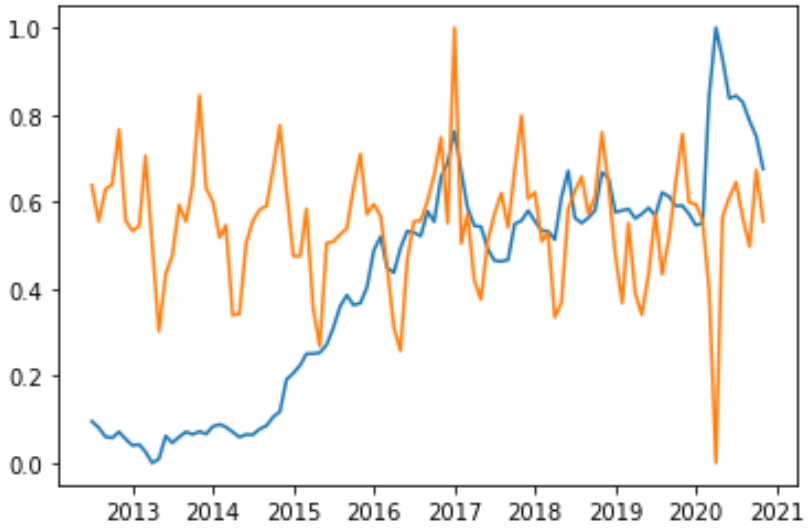


Grafica de variable vs dolar e histogramas

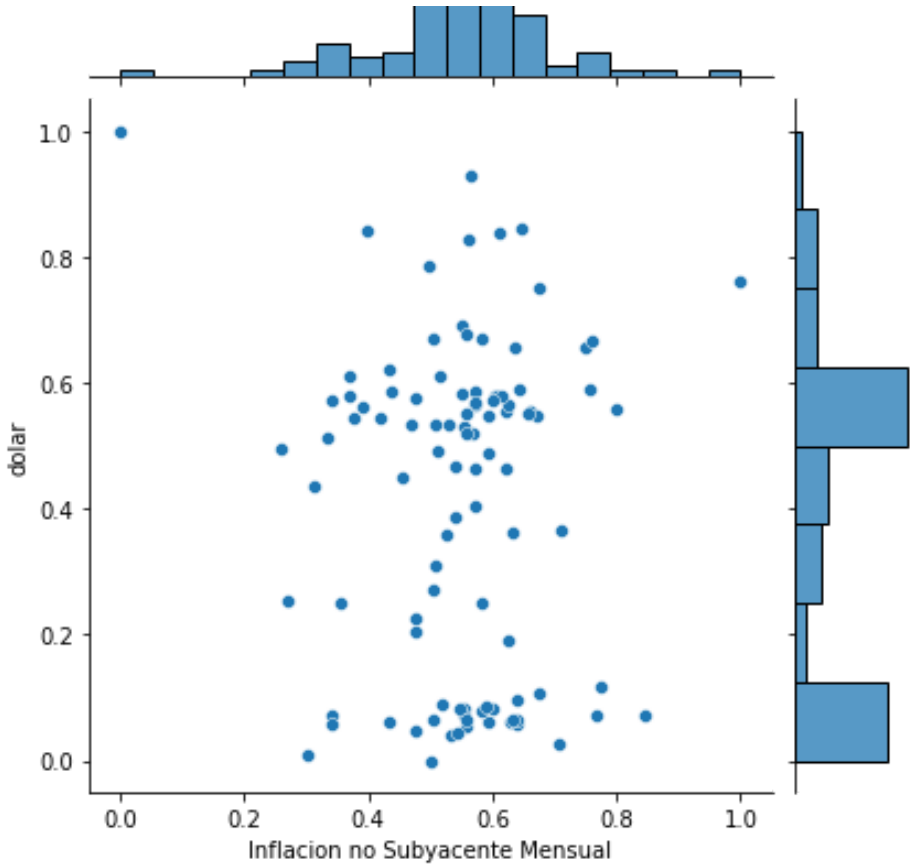


Variable: Inflacion no Subyacente Mensual

Gráfica como función del tiempo

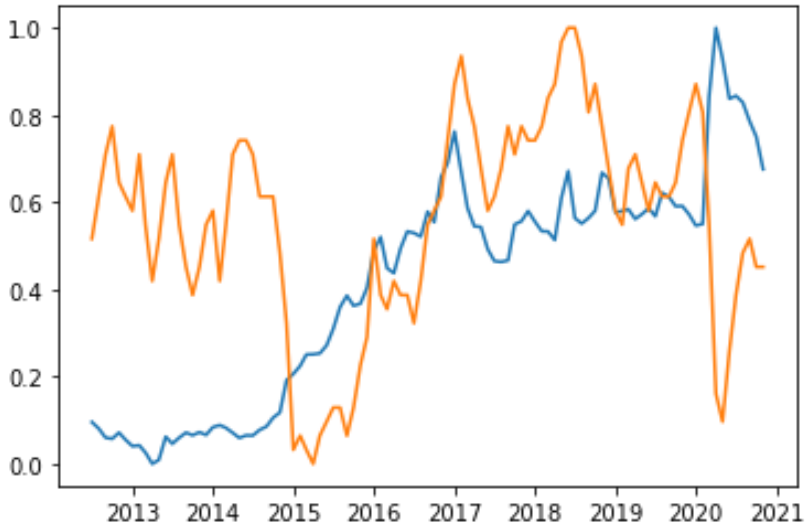


Grafica de variable vs dolar e histogramas

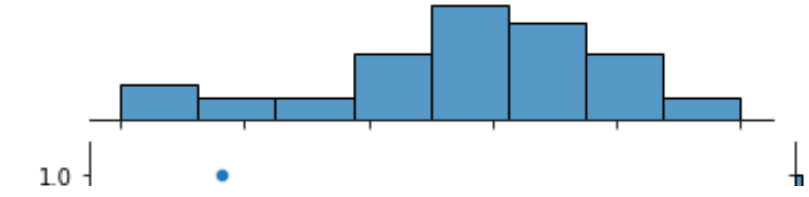


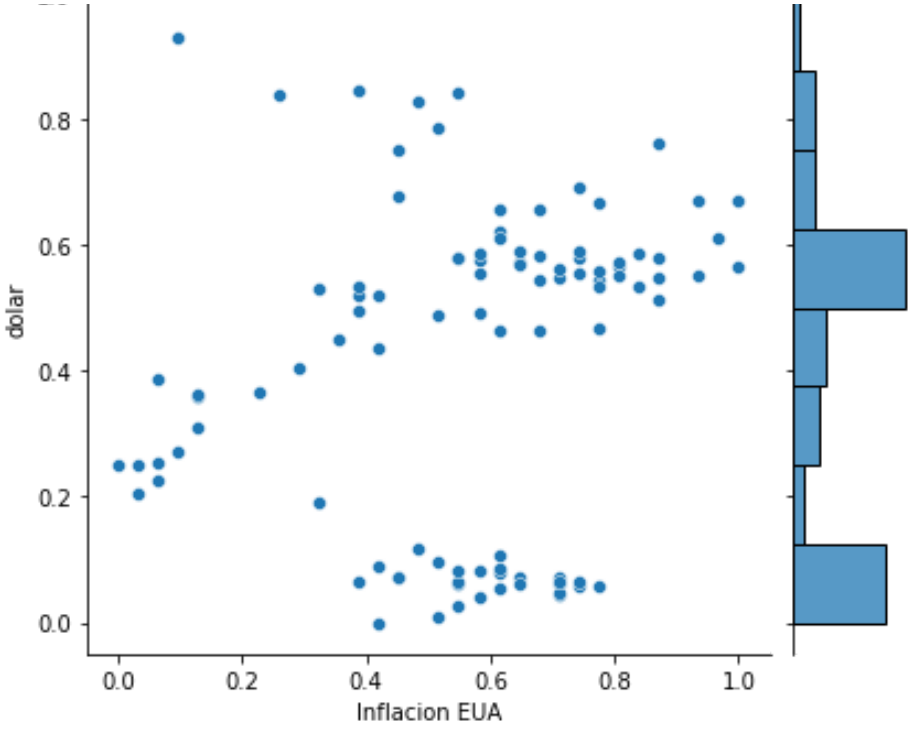
Variable: Inflacion EUA

Gráfica como función del tiempo



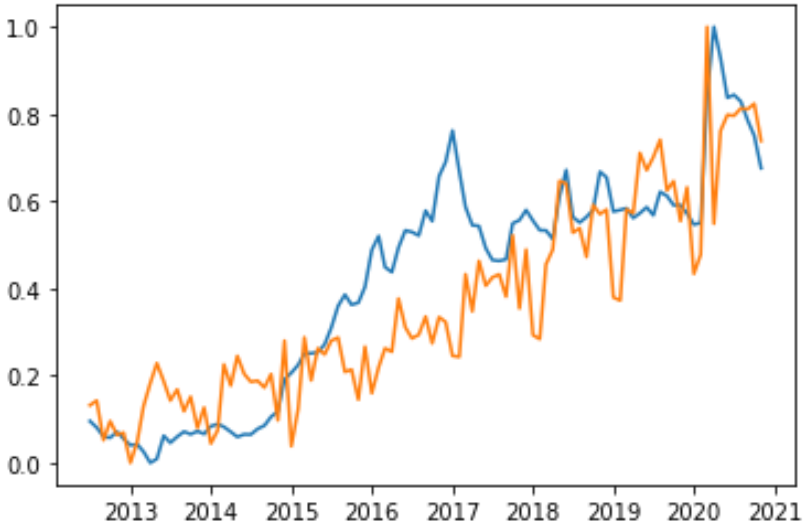
Grafica de variable vs dolar e histogramas



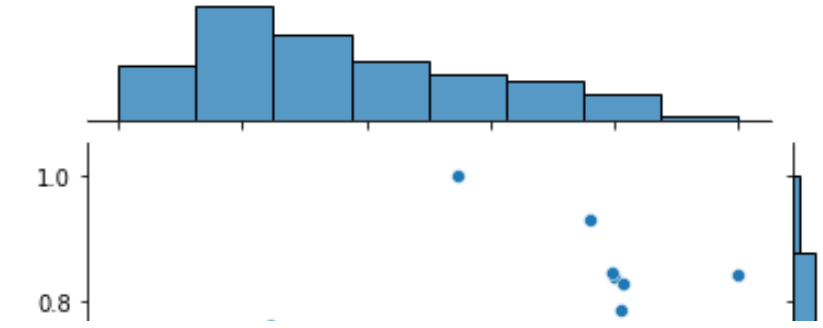


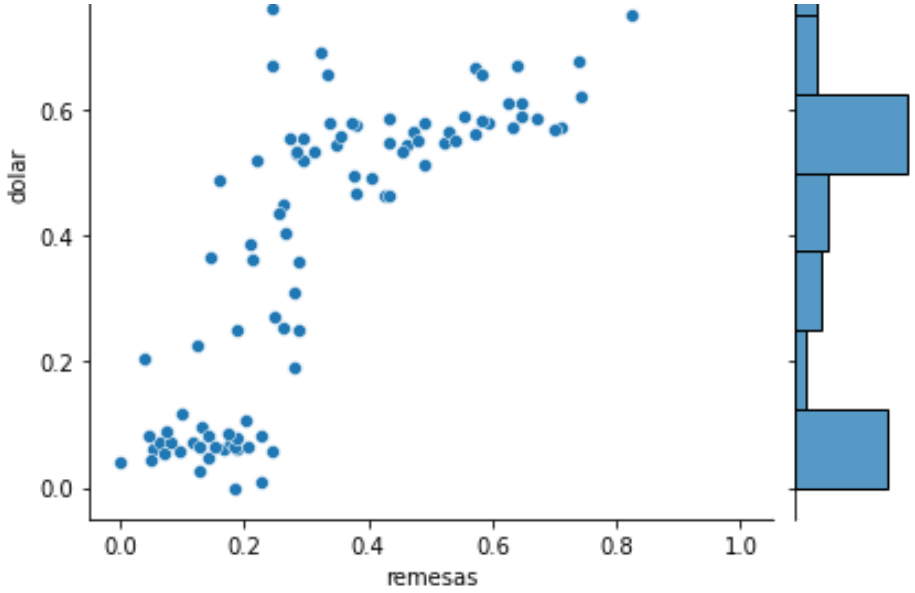
Variable: remesas

Gráfica como función del tiempo



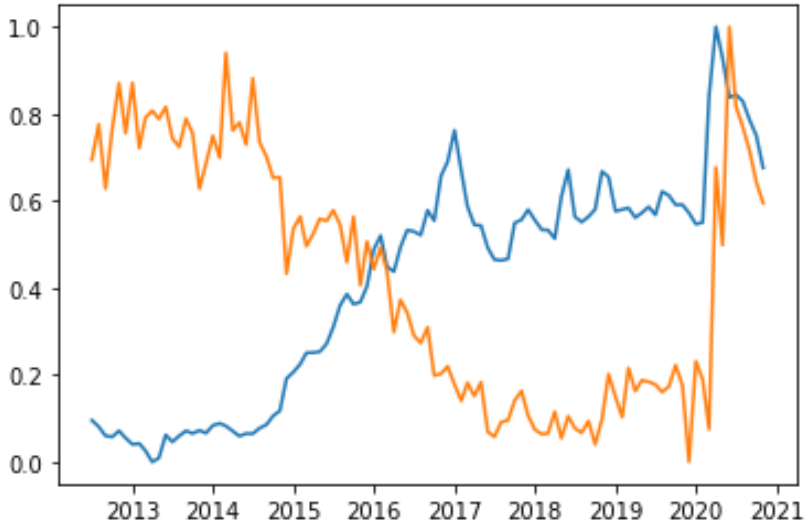
Grafica de variable vs dolar e histogramas



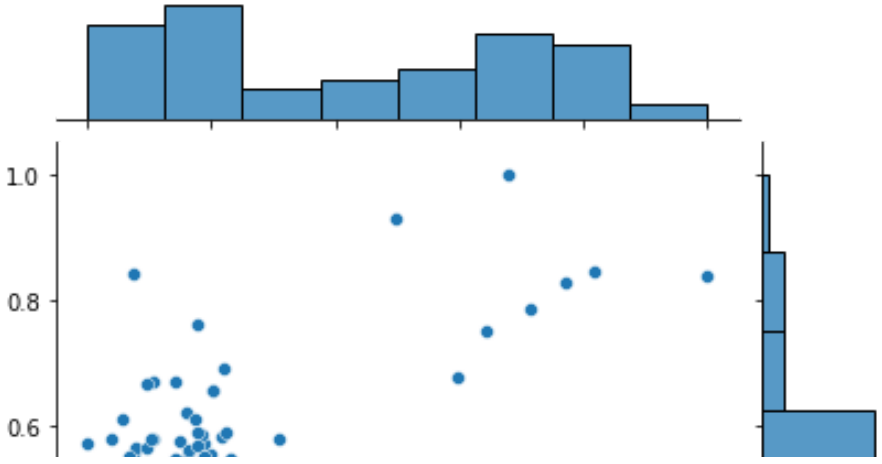


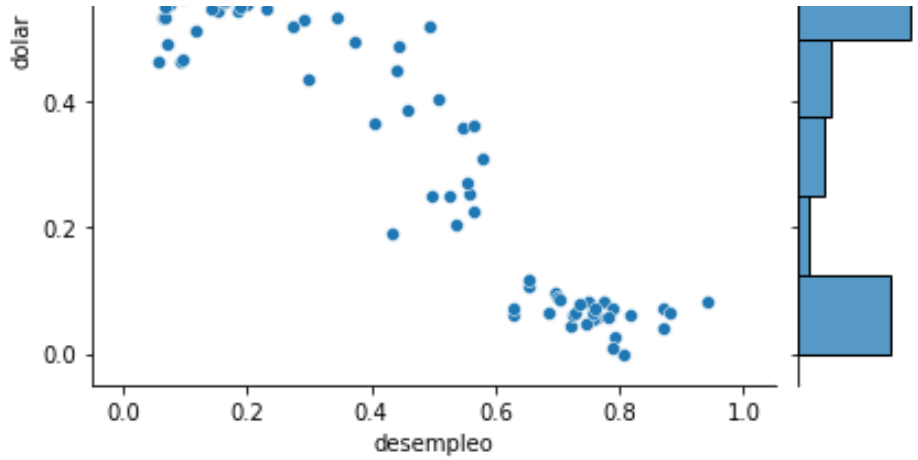
Variable: desempleo

Gráfica como función del tiempo



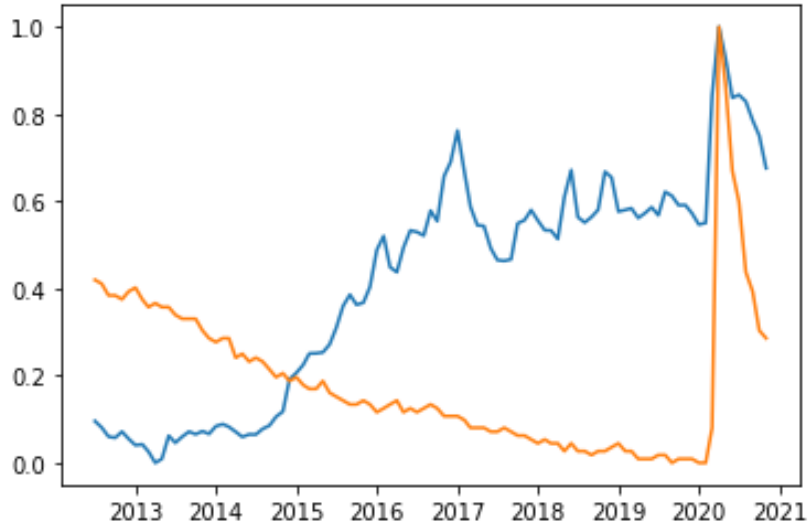
Grafica de variable vs dolar e histogramas



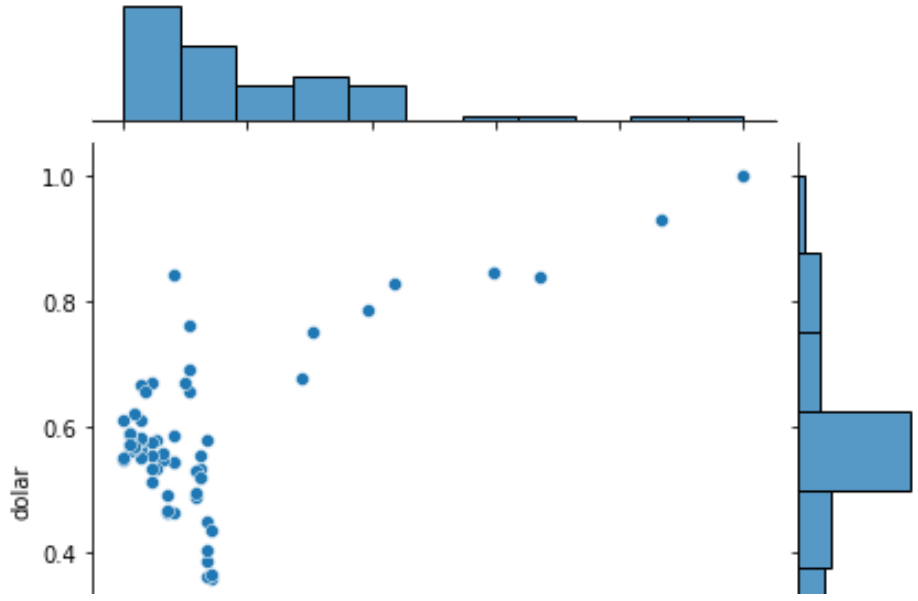


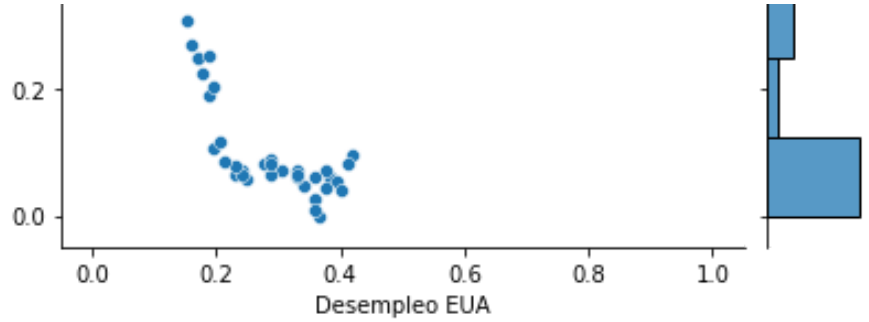
Variable: Desempleo EUA

Gráfica como función del tiempo



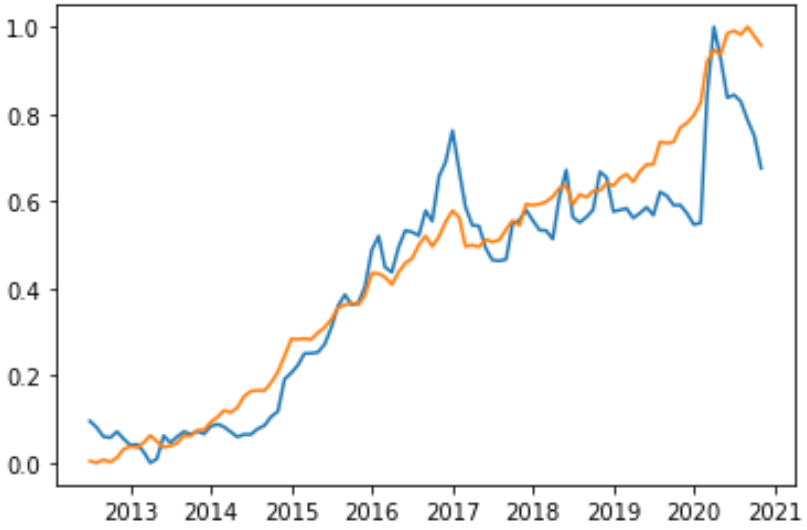
Grafica de variable vs dolar e histogramas



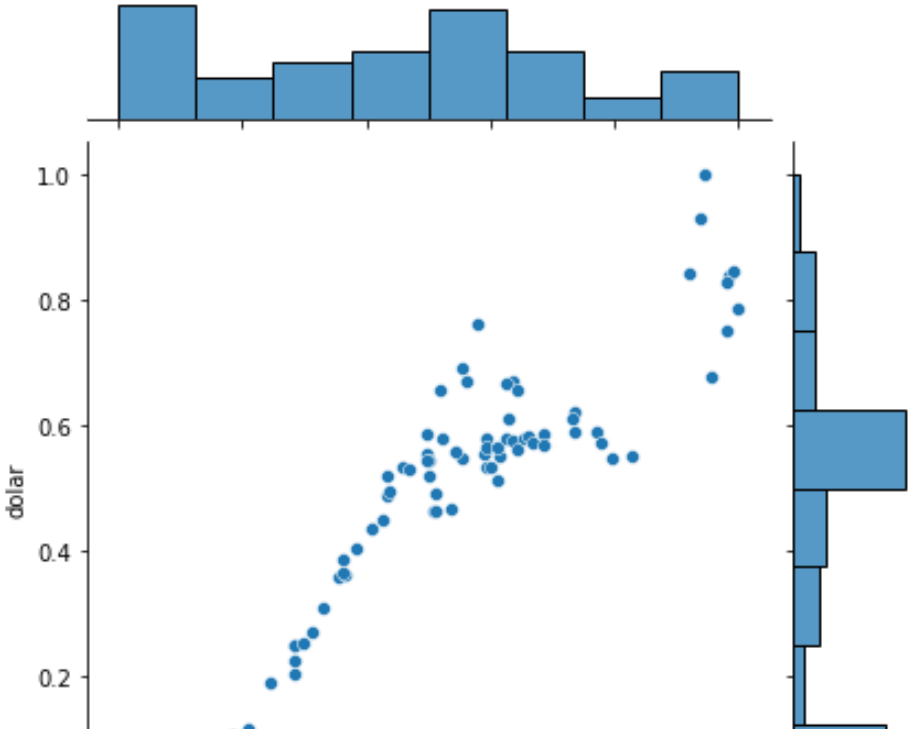


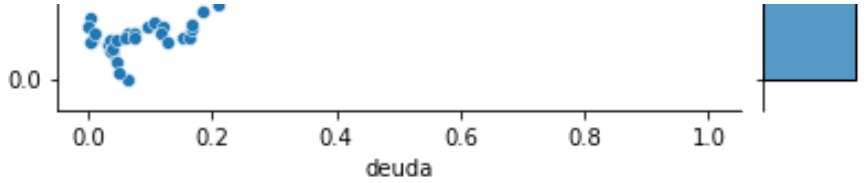
Variable: deuda

Gráfica como función del tiempo



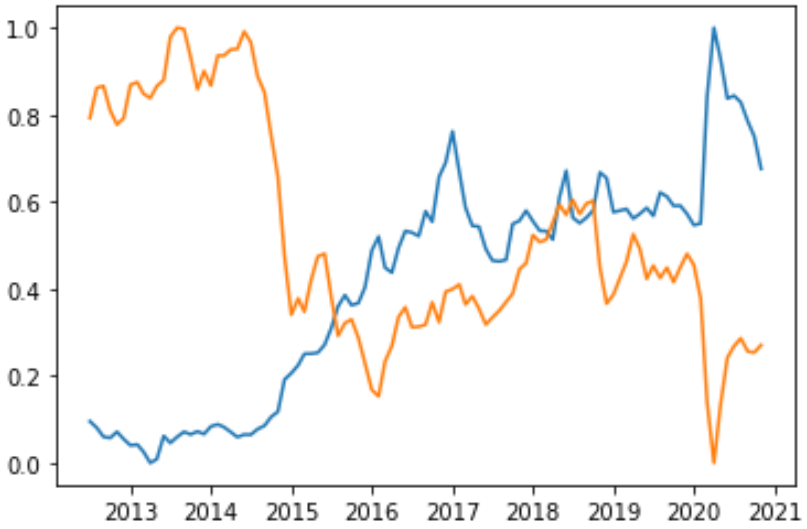
Grafica de variable vs dolar e histogramas



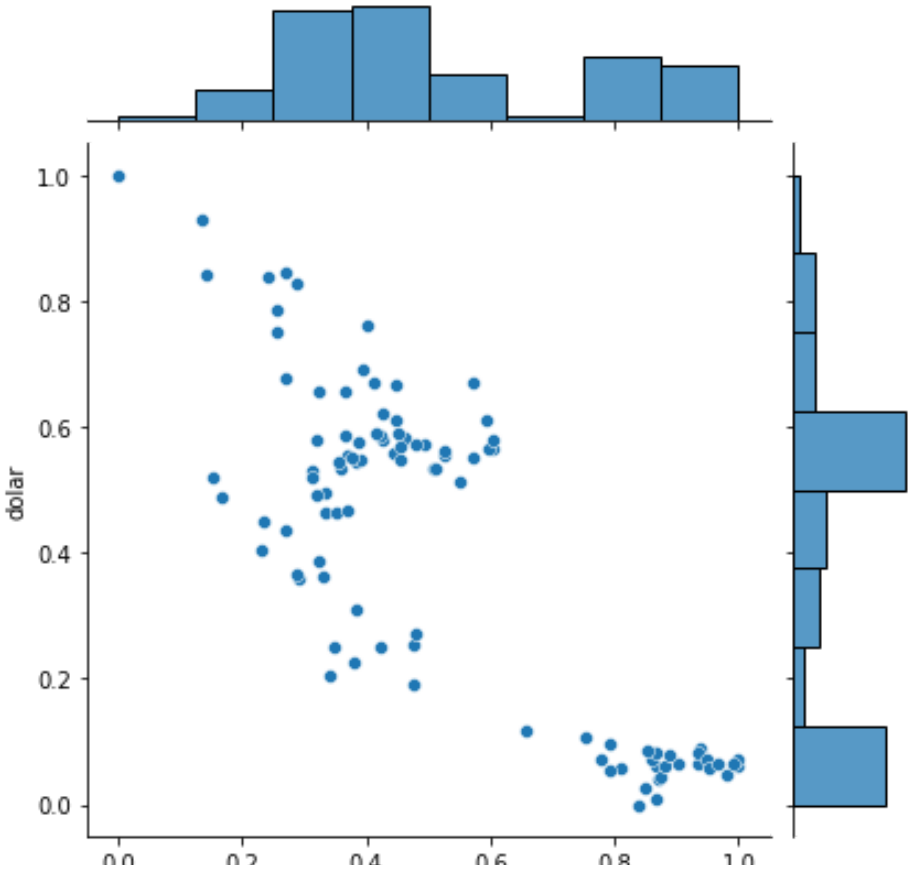


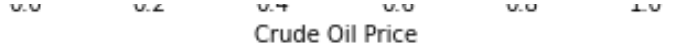
Variable: Crude Oil Price

Gráfica como función del tiempo



Grafica de variable vs dolar e histogramas

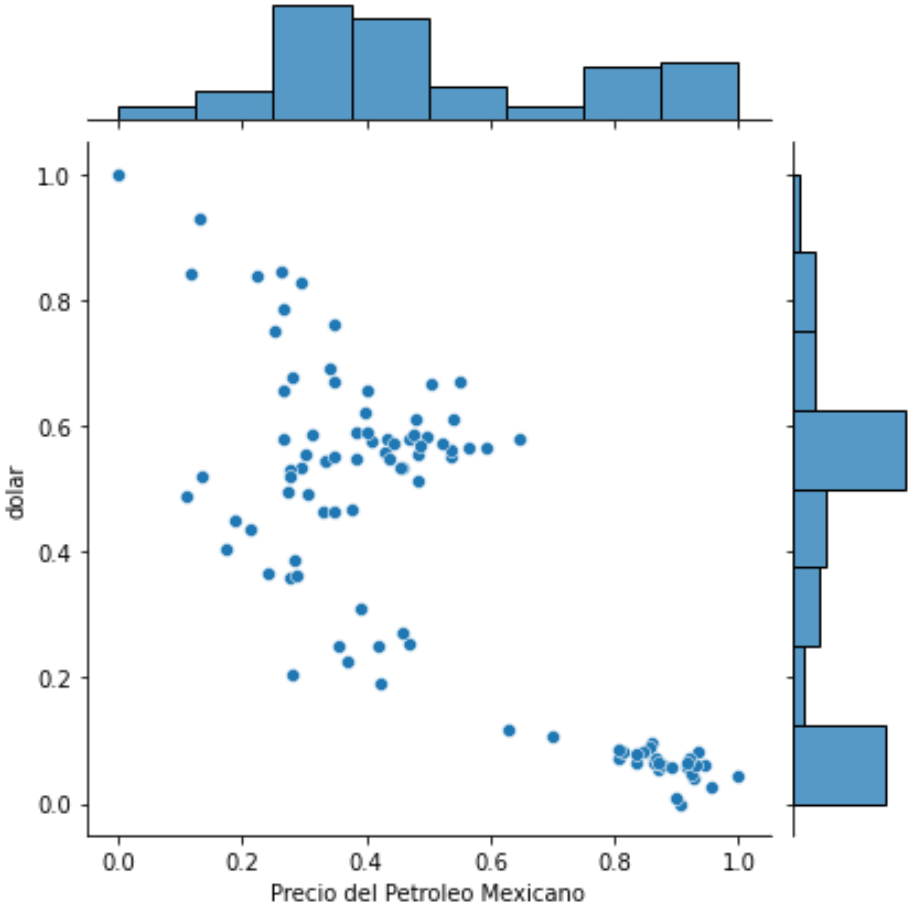




Variable: Precio del Petroleo Mexicano

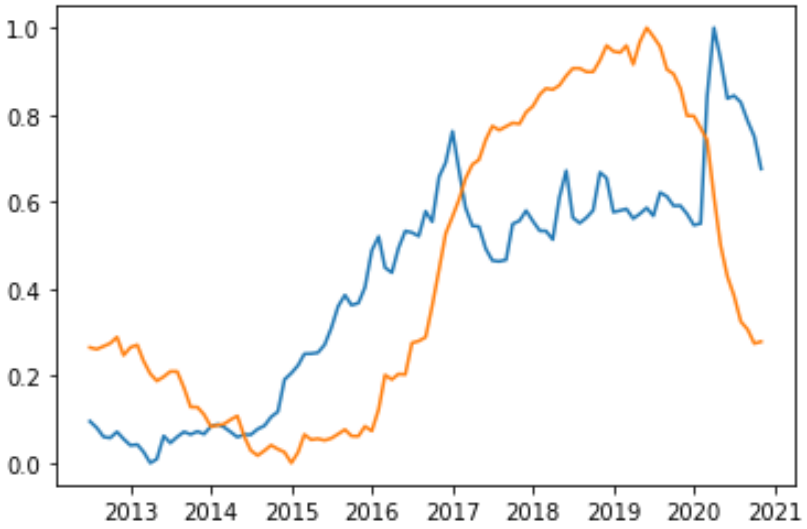
Gráfica como función del tiempo

Grafica de variable vs dolar e histogramas

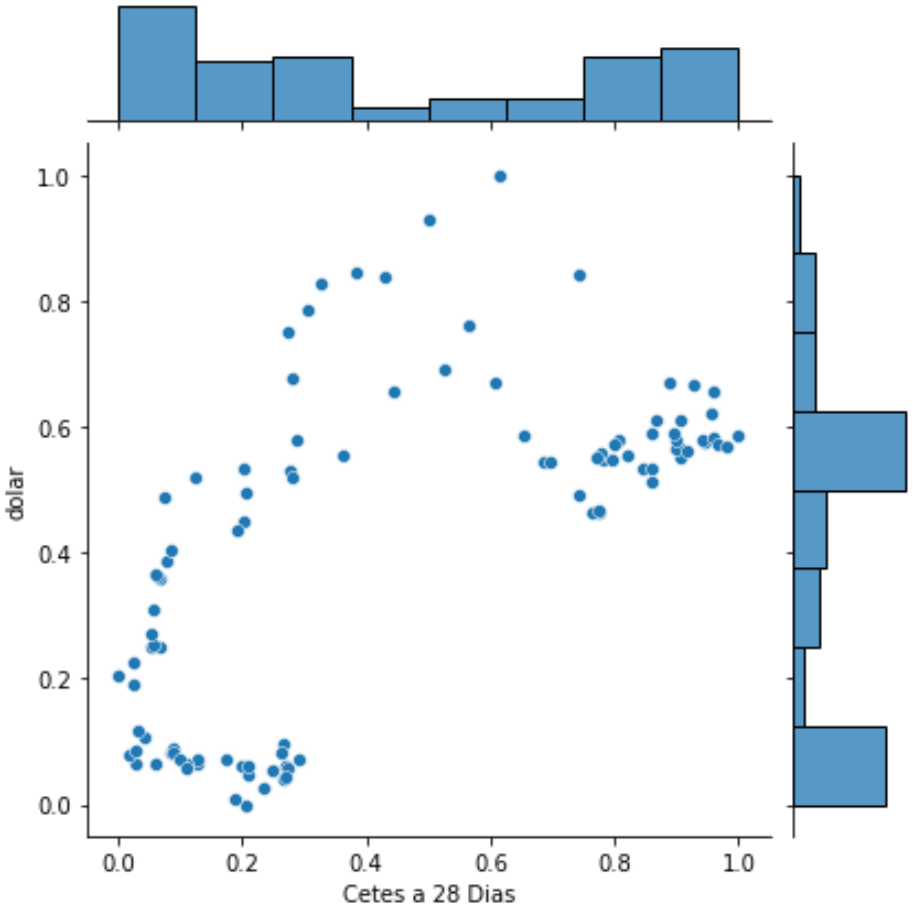


Variable: Cetes a 28 Dias

Gráfica como función del tiempo

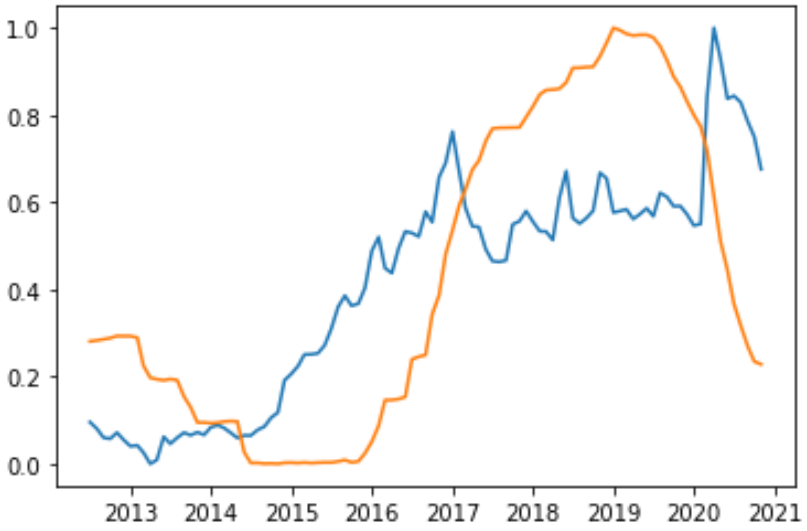


Grafica de variable vs dolar e histogramas

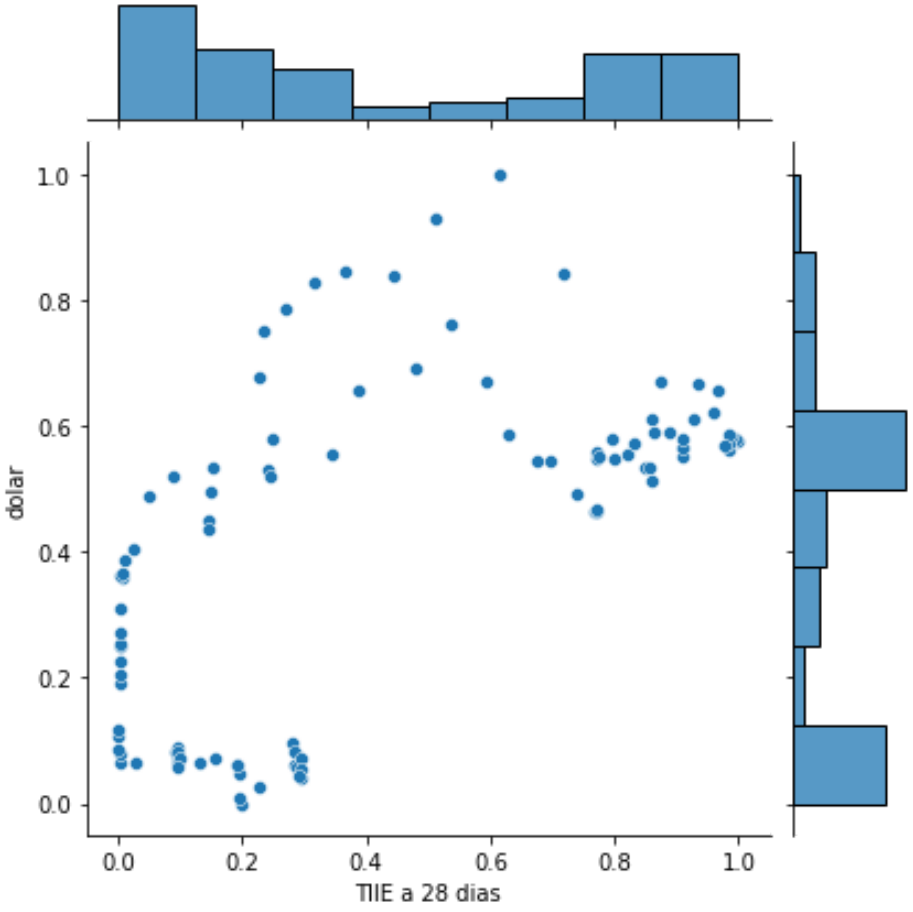


Variable: TIIE a 28 dias

Gráfica como función del tiempo

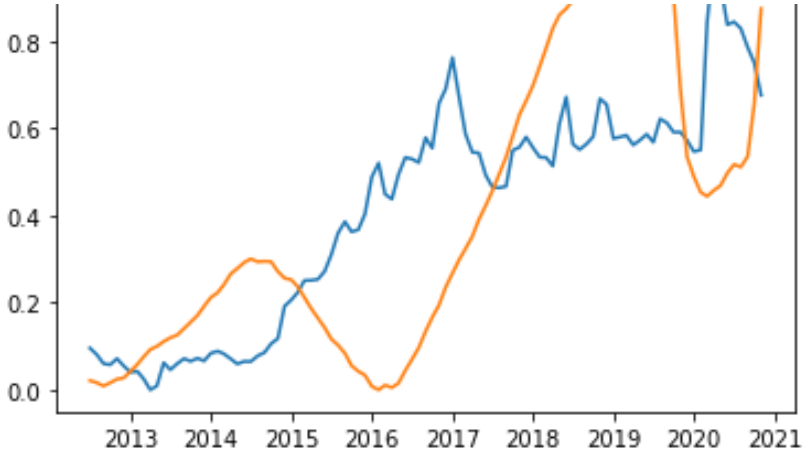


Grafica de variable vs dolar e histogramas

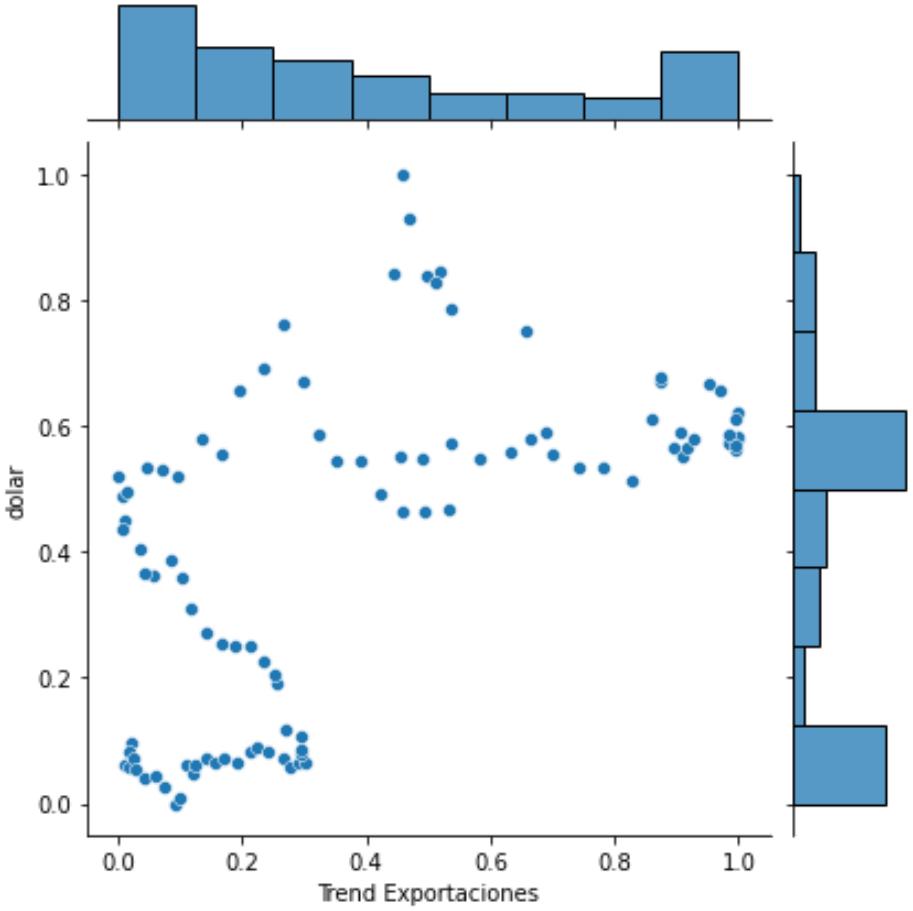


Variable: Trend Exportaciones

Gráfica como función del tiempo

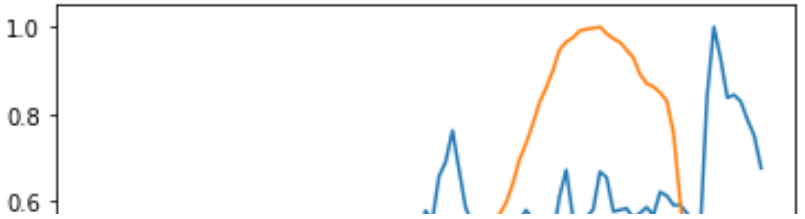


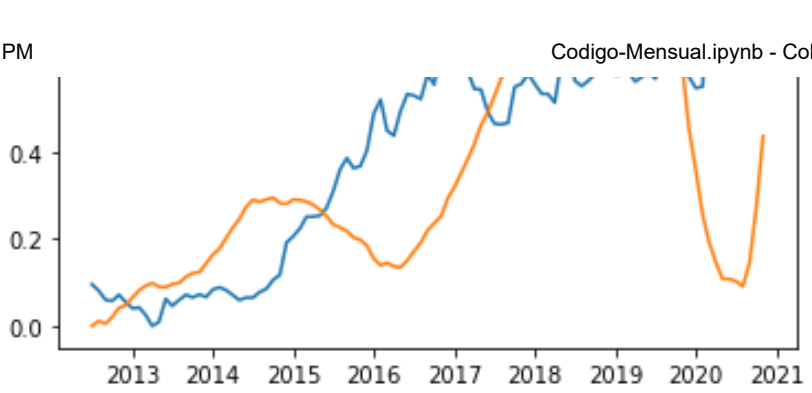
Grafica de variable vs dolar e histogramas



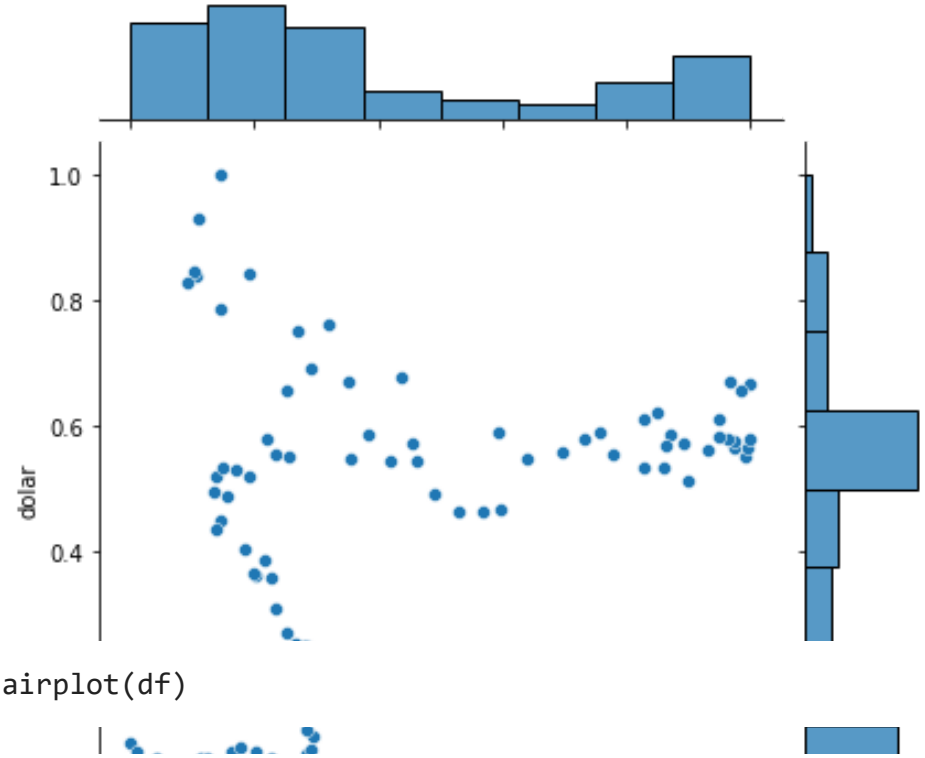
Variable: Trend Importaciones

Gráfica como función del tiempo





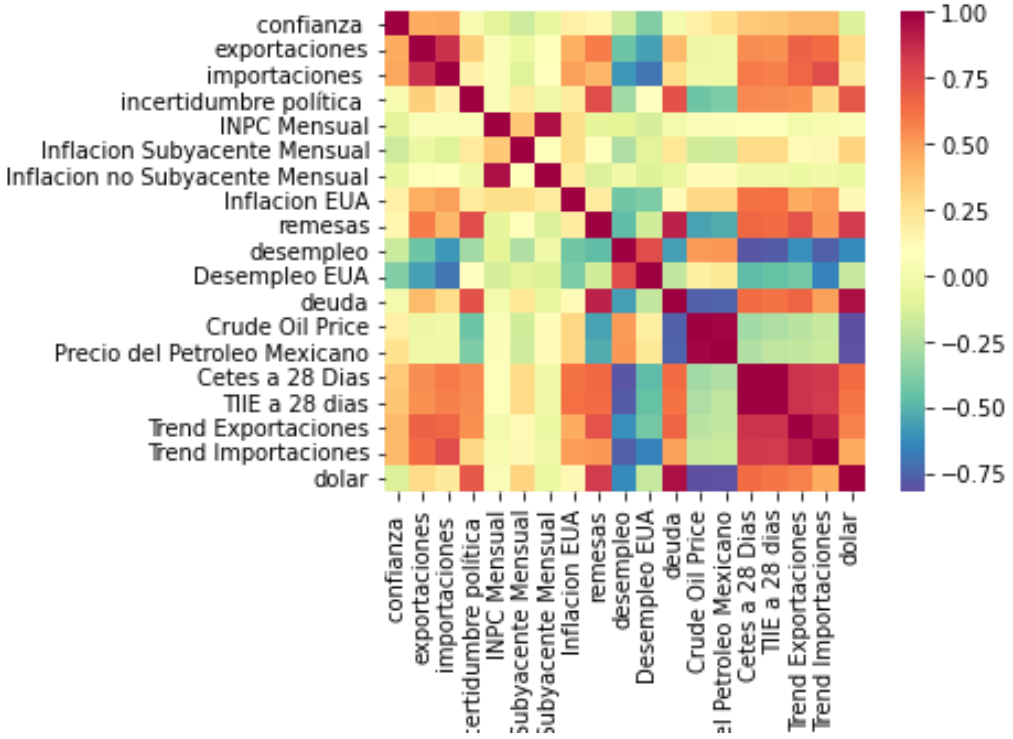
Grafica de variable vs dolar e histogramas




```
sns.pairplot(df)
```

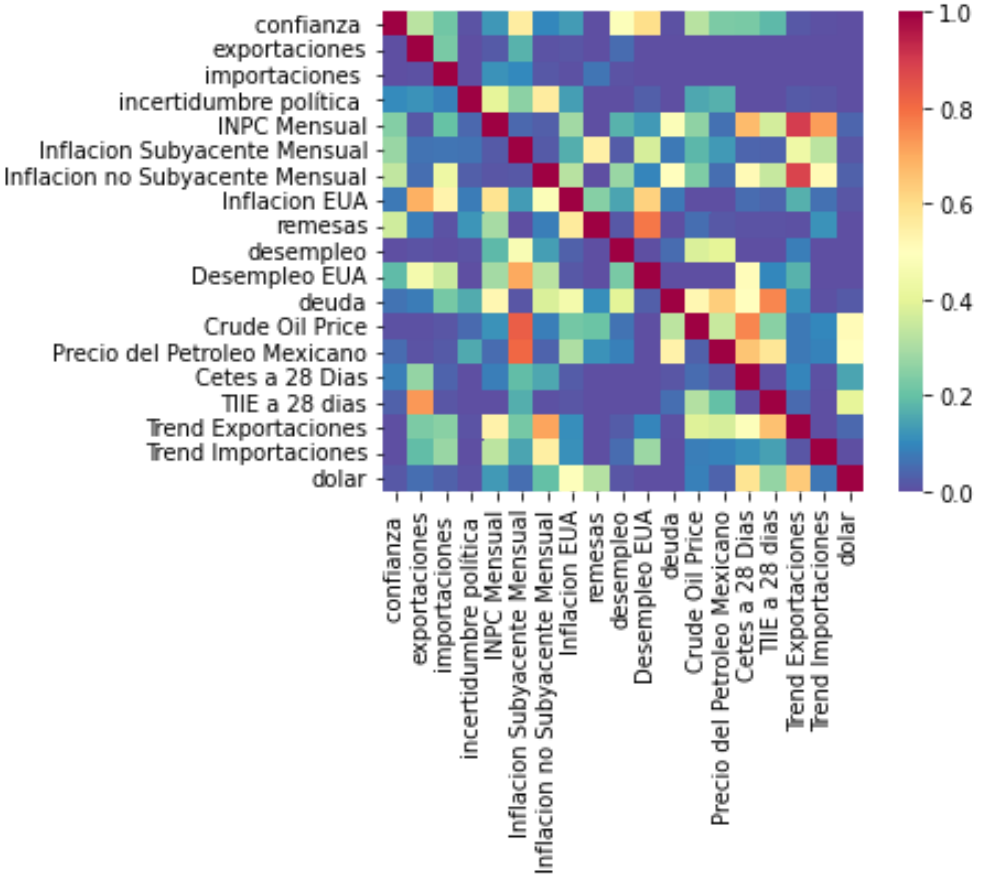


```
# Correlaciones

corrmat = df.corr()
hm = sns.heatmap(corrmat,
                 cbar=True,
                 annot=False,
                 square=True,
                 fmt='.2f', 8
                 annot_kws={'size': 10},
                 yticklabels=df.columns,
                 xticklabels=df.columns,
                 cmap="Spectral_r")
plt.show()
```




```
## Test de Causalidad
from statsmodels.tsa.stattools import grangercausalitytests
def grangers_causation_matrix(data, variables, test='ssr_chi2test', verbose=False):
    maxlag=3
    """Check Granger Causality of all possible combinations of the Time series.
    The rows are the response variable, columns are predictors. The values in the tal
    are the P-Values. P-Values lesser than the significance level (0.05), implies
    the Null Hypothesis that the coefficients of the corresponding past values is
    zero, that is, the X does not cause Y can be rejected.

    data      : pandas dataframe containing the time series variables
    variables : list containing names of the time series variables.
    """
    df = pd.DataFrame(np.zeros((len(variables), len(variables))), columns=variables,
    for c in df.columns:
        for r in df.index:
            test_result = grangercausalitytests(data[[r, c]], maxlag=maxlag, verbose=
            p_values = [round(test_result[i+1][0][test][1],4) for i in range(maxlag)
            if verbose: print(f'Y = {r}, X = {c}, P Values = {p_values}')
            min_p_value = np.min(p_values)
            df.loc[r, c] = min_p_value
    df.columns = [var + '_x' for var in variables]
    df.index = [var + '_y' for var in variables]
    return df


df_caus = grangers_causation_matrix(df, variables = df.columns)
```

```
# Causalidad

hm = sns.heatmap(df_caus,
                 cbar=True,
                 annot=False,
                 square=True,
                 fmt='.2f',
                 annot_kws={'size': 10},
                 yticklabels=df.columns,
                 xticklabels=df.columns,
                 cmap="Spectral_r")
plt.show()
```



Productos pagados de Colab - Cancela los contratos aquí

✓ 1 s se ejecutó 12:23