Benchmark

November 30, 2019

1 Introdução

Esta rotina inspeciona as séries temporais para ajustar um modelo VAR em que será testado se o investimento residencial (I_h) depende da taxa própria de juros dos imóveis, ou seja,

$$I_h = f(r_{mo}, p_h)$$

em que

- I_h Investimento residencial
  - **Série:** PRFI
  - Com ajuste sazonal
  - Trimestral
- r_mo taxa de juros das hipotecas
  - **Série:** MORTGAGE30US
  - Sem ajuste sazonal
  - Semanal (encerrado às quintas feiras)
- p_h Inflação de imóveis: Índice Case-Shiller
  - **Série:** CSUSHPISA
  - Com ajuste sazonal, Jan 2000 = 100
  - Mensal

Nota: Uma vez que pretende-se utilizar os resultados obtidos deste modelo em um trabalho futuro, os resultados serão checados tanto em python quanto em gretl, ambos softwares livres.

2 Carregando pacotes

[1]:
```
%config InlineBackend.figure_format = 'retina'

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

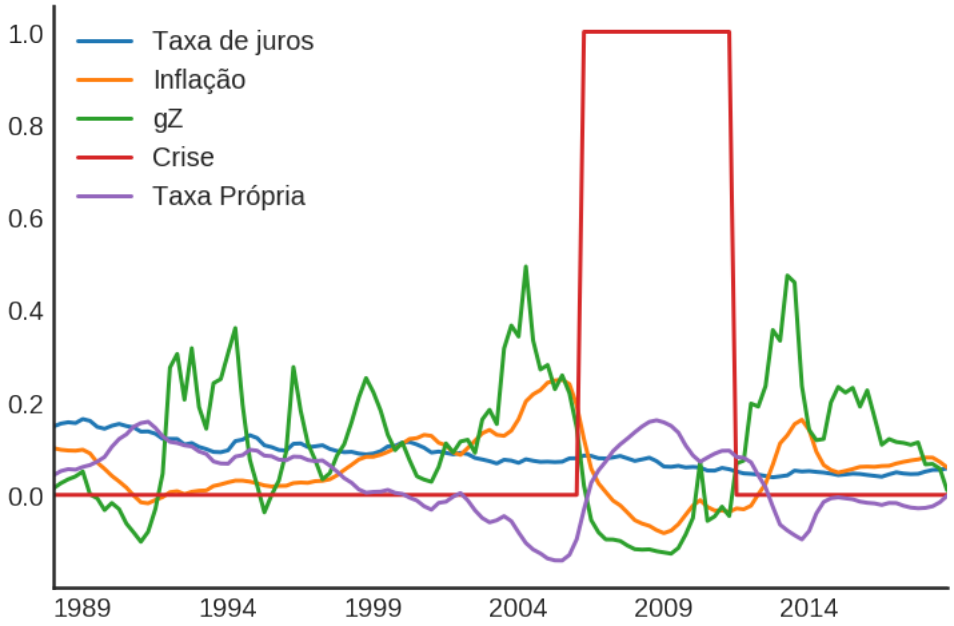
```
import pandas_datareader.data as web
import datetime

from statsmodels.tsa.stattools import adfuller, kpss, grangercausalitytests,␣
 ↪q_stat, coint
from statsmodels.tsa.vector_ar.var_model import VAR
from statsmodels.tsa.api import SVAR
from statsmodels.tsa.vector_ar.vecm import coint_johansen, CointRankResults,␣
 ↪VECM
from statsmodels.stats.diagnostic import acorr_breusch_godfrey, acorr_ljungbox,␣
 ↪het_arch, het_breuschpagan, het_white
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

from scipy.stats import yeojohnson

from arch.unitroot import PhillipsPerron, ZivotAndrews, DFGLS, KPSS, ADF

plt.style.use('seaborn-white')
start = datetime.datetime(1987, 1, 1)
end = datetime.datetime(2019, 1, 1)
```

3 Importando dados

[2]:
```
df = web.DataReader(
    [
        "PRFI",
        "CSUSHPISA",
        "MORTGAGE30US",
        "PCDG",
        "FLTOTALSL"
    ],
    'fred',
    start,
    end
)
df.columns = [
    "Investimento residencial",
    "Preço dos imóveis",
    "Taxa de juros",
    "Duráveis",
    "Crédito"
]
df.index.name = ""
df['Taxa de juros'] = df['Taxa de juros'].divide(100)
df = df.resample('Q').mean()
```

```
df['Preço dos imóveis'] = df['Preço dos imóveis']*df['Preço dos␣
 ↪imóveis']["1999-12-31"]/df['Preço dos imóveis'][0]/100
df["gDuráveis"], *_ = yeojohnson(df["Duráveis"].pct_change(4))
df["gCrédito"], *_ = yeojohnson(df["Crédito"].pct_change(4))
df["Inflação"], *_ = yeojohnson(df["Preço dos imóveis"].pct_change(4))
df['gZ'], *_ = yeojohnson(df["Investimento residencial"].pct_change(4))
df["Taxa de juros"], *_ = yeojohnson(df["Taxa de juros"])
df["Taxa Própria"] = ((1+df["Taxa de juros"])/(1+df["Inflação"])) -1

df["Crise"] = [0 for i in range(len(df["gZ"]))]
for i in range(len(df["Crise"])):
    if df.index[i] > datetime.datetime(2006,4,1) and df.index[i] < datetime.
 ↪datetime(2011,7,1):
        df["Crise"][i] = 1

df.to_csv("Dados_yeojohnson.csv", )

df = df[["Taxa de juros", "Inflação", "gZ", "Crise", "Taxa Própria"]]

df = df.dropna()
df.plot()
sns.despine()
plt.show()
```

```
/home/gpetrini/.local/lib/python3.6/site-packages/scipy/stats/morestats.py:1358:
RuntimeWarning: invalid value encountered in greater_equal
  pos = x >= 0  # binary mask
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:34:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```



4 Teste de cointegração

4.1 Engel-granger

[3]:
```
def cointegracao(ts0, ts1, trend = 'c', signif = 0.05):
  result = coint(ts0, ts1, trend = trend, autolag='bic')
  print('Null Hypothesis: there is NO cointegration')
  print('Alternative Hypothesis: there IS cointegration')
  print('t Statistic: %f' % result[0])
  print('p-value: %f' % result[1])
  if result[1] < signif:
    print('REJECT null Hypothesis: there IS cointegration')
  else:
    print('FAIL to reject Null Hypothesis: there is NO cointegration')
```

[4]:
```
cointegracao(ts0 = df['gZ'], ts1 = df[['Inflação']])
```

```
Null Hypothesis: there is NO cointegration
Alternative Hypothesis: there IS cointegration
t Statistic: -2.700328
p-value: 0.199461
FAIL to reject Null Hypothesis: there is NO cointegration
```

Não incluindo a taxa de juros dos imóveis no teste de hipótese, não rejeita-se a hipótese nula e, portanto, não há evidências de relação de cointegração. Será testado para o caso com a taxa de juros.

[5]:
```
cointegracao(ts0 = df['gZ'], ts1 = df[['Inflação', 'Taxa de juros']])
```

```
Null Hypothesis: there is NO cointegration
Alternative Hypothesis: there IS cointegration
t Statistic: -2.479682
p-value: 0.494677
FAIL to reject Null Hypothesis: there is NO cointegration
```

Não rejeita-se a hipótese nula a um nível de significância de 5% e, portanto, **não** há evidência de relação de cointegração.

4.1.1 Conclusão

Adotando um valor crítico de 5%, não rejeita-se a hipótese nula para as variáveis individuais e para os resíduos. Portanto, não há evidências de relação de cointegração supondo sem a inclusão dos juros. As conclusões são as mesmas com a inclusão dos juros. Portanto, a partir do teste de Engel-Granger, conclui-se que não existe relação de cointegração entre as variáveis.

4.2 Johansen

O teste de Engel-Granger não capta cointegração conjunta entre as variáveis e, portanto, deve ser utilizado o teste de Johansen. **OBS:** Dada a importância das implicações dos resultados deste teste, será adotado um nível de significância mais "rigoroso" de 1%. Testando sucessivamente começando em r = 1 (max. = 3).

4.2.1 Python

[6]:
```
def johansen(ts, det_order = 1, k_ar_diff = 2, signif = 0.05):
  """
  ts = ts dataframe
  det_order = similar to coint_johansen function
    - (-1): no deterministic terms
    - 0: constant term
    - 1: linear trend
  k_ar_diff = similar to coint_johansen function
  signif = 0.1, 0.05, 0.01
  """
  var = len(ts.columns)
  result = coint_johansen(
      endog = ts,
      det_order = det_order,
      k_ar_diff = k_ar_diff
  )

  p = [i for i in range(var)]

  print("Null hypothesis: number of cointegration vectors is r* = {}".
↪format(k_ar_diff))
```

```
  print('Alternative (Trace): r* > {}'.format(k_ar_diff))
  print('\n')
  for i in range(var):
    print("Test for variable " + str(ts.columns[i]).upper())
    print("TRACE test")
    print('Statistic: %f' % result.lr1[i])
    print('Critical Values:')
    print('\t 10%: ', result.cvt[i][0])
    print('\t 5%: ', result.cvt[i][1])
    print('\t 1%: ', result.cvt[i][2])

    if signif == 0.05:
      p[i] = result.cvt[i][1]
    elif signif == 0.01:
      p[i] = result.cvt[i][2]
    else:
      p[i] = result.cvt[i][0]

  print("For a significant level of {}, REJECT null hypothesis? {}".
↪format(signif, np.sum(result.lr1 > p) == var))
```

Incluindo taxa de juros dos imóveis.

[7]:
```
johansen(df[['Taxa de juros', 'Inflação', 'gZ']], det_order = 1, k_ar_diff = 0,␣
↪signif = 0.05)
```

```
Null hypothesis: number of cointegration vectors is r* = 0
Alternative (Trace): r* > 0


Test for variable TAXA DE JUROS
TRACE test
Statistic: 102.641172
Critical Values:
         10%:  32.0645
         5%:  35.0116
         1%:  41.0815
Test for variable INFLAÇÃO
TRACE test
Statistic: 44.943513
Critical Values:
         10%:  16.1619
         5%:  18.3985
         1%:  23.1485
Test for variable GZ
TRACE test
Statistic: 6.580712
Critical Values:
         10%:  2.7055
```

```
        5%:  3.8415
        1%:  6.6349
For a significant level of 0.05, REJECT null hypothesis? True
```

[8]:
```
johansen(df[['Taxa de juros', 'Inflação', 'gZ']], det_order = 1, k_ar_diff = 1,␣
↪signif = 0.05)
```

```
Null hypothesis: number of cointegration vectors is r* = 1
Alternative (Trace): r* > 1


Test for variable TAXA DE JUROS
TRACE test
Statistic: 51.521904
Critical Values:
        10%:  32.0645
        5%:  35.0116
        1%:  41.0815
Test for variable INFLAÇÃO
TRACE test
Statistic: 26.954884
Critical Values:
        10%:  16.1619
        5%:  18.3985
        1%:  23.1485
Test for variable GZ
TRACE test
Statistic: 12.056876
Critical Values:
        10%:  2.7055
        5%:  3.8415
        1%:  6.6349
For a significant level of 0.05, REJECT null hypothesis? True
```

[9]:
```
johansen(df[['Taxa de juros', 'Inflação', 'gZ']], det_order = 1, k_ar_diff = 2,␣
↪signif = 0.05)
```

```
Null hypothesis: number of cointegration vectors is r* = 2
Alternative (Trace): r* > 2


Test for variable TAXA DE JUROS
TRACE test
Statistic: 42.877391
Critical Values:
        10%:  32.0645
        5%:  35.0116
```

```
         1%:  41.0815
Test for variable INFLAÇÃO
TRACE test
Statistic: 20.961551
Critical Values:
         10%:  16.1619
         5%:  18.3985
         1%:  23.1485
Test for variable GZ
TRACE test
Statistic: 8.051458
Critical Values:
         10%:  2.7055
         5%:  3.8415
         1%:  6.6349
For a significant level of 0.05, REJECT null hypothesis? True
```

Conclusão: A hipótese nula do teste não é rejeitada a um nível de significância de 1% para $r \geq 2$. Portanto, há envidências de relação de cointegração entre as variávies.

Testando sem os juros (determinado exogenamente).

[10]:
```
johansen(df[['Inflação', 'gZ']], det_order = 1, k_ar_diff = 0, signif = 0.05)
```

```
Null hypothesis: number of cointegration vectors is r* = 0
Alternative (Trace): r* > 0


Test for variable INFLAÇÃO
TRACE test
Statistic: 53.411260
Critical Values:
         10%:  16.1619
         5%:  18.3985
         1%:  23.1485
Test for variable GZ
TRACE test
Statistic: 6.071659
Critical Values:
         10%:  2.7055
         5%:  3.8415
         1%:  6.6349
For a significant level of 0.05, REJECT null hypothesis? True
```

[11]:
```
johansen(df[['Inflação', 'gZ']], det_order = 1, k_ar_diff = 1, signif = 0.05)
```

```
Null hypothesis: number of cointegration vectors is r* = 1
Alternative (Trace): r* > 1
```

```
Test for variable INFLAÇÃO
TRACE test
Statistic: 27.747761
Critical Values:
         10%:  16.1619
         5%:  18.3985
         1%:  23.1485
Test for variable GZ
TRACE test
Statistic: 13.459240
Critical Values:
         10%:  2.7055
         5%:  3.8415
         1%:  6.6349
For a significant level of 0.05, REJECT null hypothesis? True
```

Excluindo os juros (teoricamente mais factível), não há evidências de relação de cointegração a 1%.

4.2.2 Gretl

Considerando: Taxa de juros como variável exógena. Obtém-se os mesmos resultados do python.

4.2.3 Conclusão

Não é possível não rejeitar a hipótese nula tanto para o teste do traço quanto para o teste autovalor máximo. Portanto, conclui-se que as série não são estácionários mas **não** apresentam relação de correlação.

5 Teste de quebra estrutural

Para realizar os testes de raíz unitária, é preciso verificar se as séries apresentam quebras estruturais. Uma vez que parte dos procedimentos não estão implementados em python, serão utilizados pacotes escritos em R. Habilitando uso do R no python.

[12]:
```
import warnings
warnings.filterwarnings('ignore')
%load_ext rpy2.ipython
```

Carregando pacotes necessários:

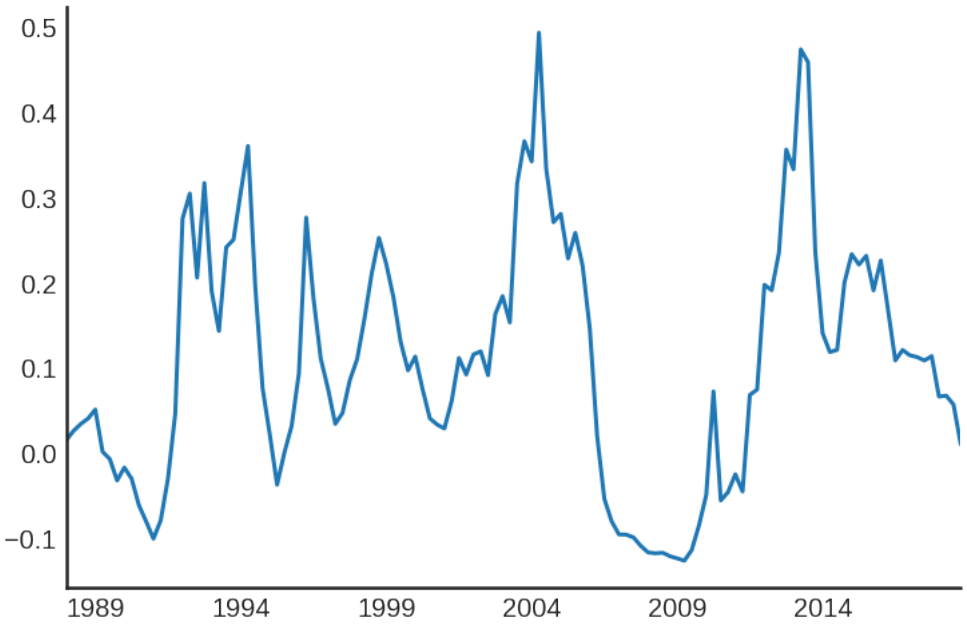
[13]:
```
%%R
library(strucchange)
library(urca)
```

```
R[write to console]: Carregando pacotes exigidos: zoo

R[write to console]:
```

```
Attaching package: zoo


R[write to console]: The following objects are masked from package:base:

    as.Date, as.Date.numeric


R[write to console]: Carregando pacotes exigidos: sandwich
```

5.1 Investimento residencial (g_Z)

[14]:
```
df["gZ"].plot()
sns.despine()
plt.show()
```



Destaque será dado para os testes com intercepto e sem tendência.

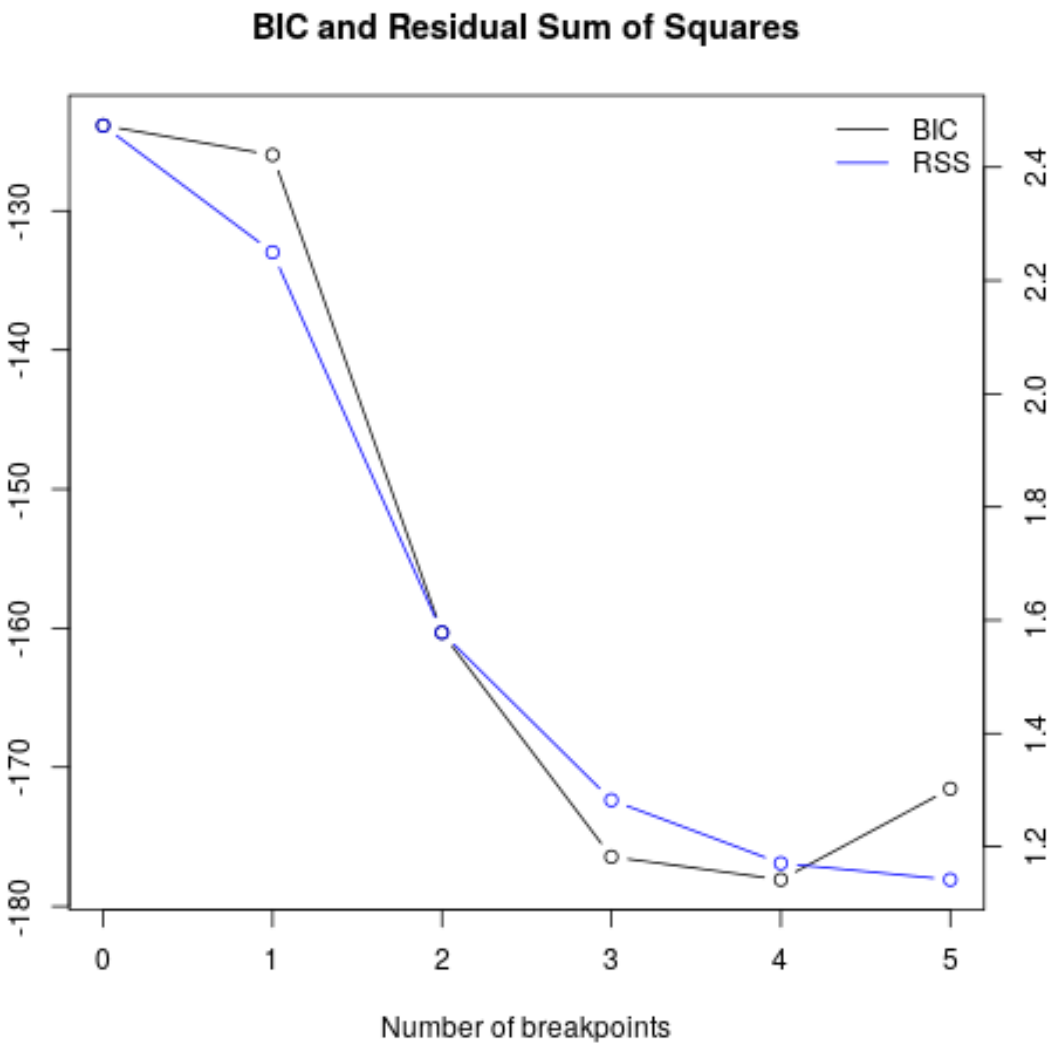
5.1.1 Teste de Chow

- H0: Sem quebra estrutural
- H1: Quebra estrutural

[15]:
```
%%R -i df
df = df$gZ
df = ts(df, start = c(1987,4), frequency = 4)
```

```
df.chow = Fstats(df ~ 1, from = 0.15)
print(breakpoints(df.chow)) #mostra o ponto da quebra quebra estrutural
print(sctest(df.chow))
print("P-Valor do teste:")
df.bp = breakpoints(df ~ 1, h = 0.15) #nota-se que a função "breakpoints"␣
 ↪aponta, no máximo, cinco quebras estruturais, se houver
plot(summary(df.bp))
```

```
         Optimal 2-segment partition:

Call:
breakpoints.Fstats(obj = df.chow)

Breakpoints at observation number:
96

Corresponding to breakdates:
2011(3)

        supF test

data:  df.chow
sup.F = 12.138, p-value = 0.009905

[1] "P-Valor do teste:"
```



Conclusão: A um nível de significância de 5% e pelo teste de Chow, conclui-se que a série possui quebra estrutural.

5.1.2 Teste ADF

Será adotado o método BIC dada a maior parcimônia.

[16]:
```
ADF(df['gZ'], method = 'BIC', trend = 'ct').summary()
```

[16]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
   Augmented Dickey-Fuller Results
====================================
Test Statistic                 -2.357
P-value                         0.403
```

```
Lags                                4
-------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -4.04 (1%), -3.45 (5%), -3.15 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

Não rejeita-se a hipótese nula para a série em nível adotando um nível de signficância de 5%, mas rejeita-se para a variável em primeira diferença. Portanto, pelo teste ADF e utilizando o crítério BIC de informação, a série é estacionária em primeira diferença. Como a série possui uma quebra estrutural, o valor crítico a ser adotado para a série em nível é -3.95. No entanto, isso não altera os resultados do teste, qual seja, não rejeição da hipótese nula para a série em nível.

OBS: Os mesmos resultados foram obtidos utilizando o gretl. Mais detalhes no script.

[17]:
```
print(ADF(df["gZ"], trend='ct').summary())
print(ADF(df["gZ"].diff().dropna(), trend='c').summary())
```

```
   Augmented Dickey-Fuller Results
=====================================
Test Statistic                 -3.303
P-value                         0.066
Lags                                7
-------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -4.04 (1%), -3.45 (5%), -3.15 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
   Augmented Dickey-Fuller Results
=====================================
Test Statistic                 -7.629
P-value                         0.000
Lags                                3
-------------------------------------

Trend: Constant
Critical Values: -3.49 (1%), -2.89 (5%), -2.58 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
```

Conclusão: Série é fracamente estacionária em primeira diferença.

5.1.3 Teste DFGLS

[18]:
```
print(DFGLS(df["gZ"], trend='ct').summary())
print(DFGLS(df["gZ"].diff().dropna(), trend='c').summary())
```

```
      Dickey-Fuller GLS Results
=====================================
Test Statistic                 -3.155
P-value                         0.022
Lags                                7
-------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -3.59 (1%), -3.01 (5%), -2.72 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
      Dickey-Fuller GLS Results
=====================================
Test Statistic                 -7.546
P-value                         0.000
Lags                                3
-------------------------------------

Trend: Constant
Critical Values: -2.73 (1%), -2.11 (5%), -1.80 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
```

Conclusão: Série em nível e em primeira diferença são fracamente estacionárias.

5.1.4 Teste KPSS

[19]:
```
print(KPSS(df["gZ"], trend = 'ct').summary())
print(KPSS(df["gZ"].diff().dropna(), trend = 'c').summary())
```

```
    KPSS Stationarity Test Results
=====================================
Test Statistic                  0.072
P-value                         0.326
Lags                                6
-------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: 0.22 (1%), 0.15 (5%), 0.12 (10%)
Null Hypothesis: The process is weakly stationary.
Alternative Hypothesis: The process contains a unit root.
    KPSS Stationarity Test Results
=====================================
Test Statistic                  0.045
P-value                         0.904
Lags                                2
-------------------------------------
```

```
Trend: Constant
Critical Values: 0.74 (1%), 0.46 (5%), 0.35 (10%)
Null Hypothesis: The process is weakly stationary.
Alternative Hypothesis: The process contains a unit root.
```

Conclusão: Série em nível e em primeira diferença são fracamente estacionária.

5.1.5 Teste de Phillips-Perron

[20]:
```
print(PhillipsPerron(df["gZ"], trend='ct').summary())
print(PhillipsPerron(df["gZ"].diff().dropna(), trend='c').summary())
```

```
     Phillips-Perron Test (Z-tau)
=====================================
Test Statistic                -2.830
P-value                        0.186
Lags                              13
-------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -4.03 (1%), -3.45 (5%), -3.15 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
     Phillips-Perron Test (Z-tau)
=====================================
Test Statistic                -9.753
P-value                        0.000
Lags                              13
-------------------------------------

Trend: Constant
Critical Values: -3.49 (1%), -2.89 (5%), -2.58 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
```

Conclusão: A um nível de significância de 5%, não rejeita-se a hipótese nula do teste Phillips-Perron e, portanto, suspeita-se que o processo contenha uma raíz unitária. Em primeira difenrença, a série é fracamente estacionária.

5.1.6 Teste de Zivot-Andrews

[21]:
```
print(ZivotAndrews(df["gZ"], trend = 'ct').summary(),"\n")
print(ZivotAndrews(df["gZ"].diff().dropna(), trend = 'c').summary(),"\n")
```

```
        Zivot-Andrews Results
=====================================
Test Statistic                -4.056
```

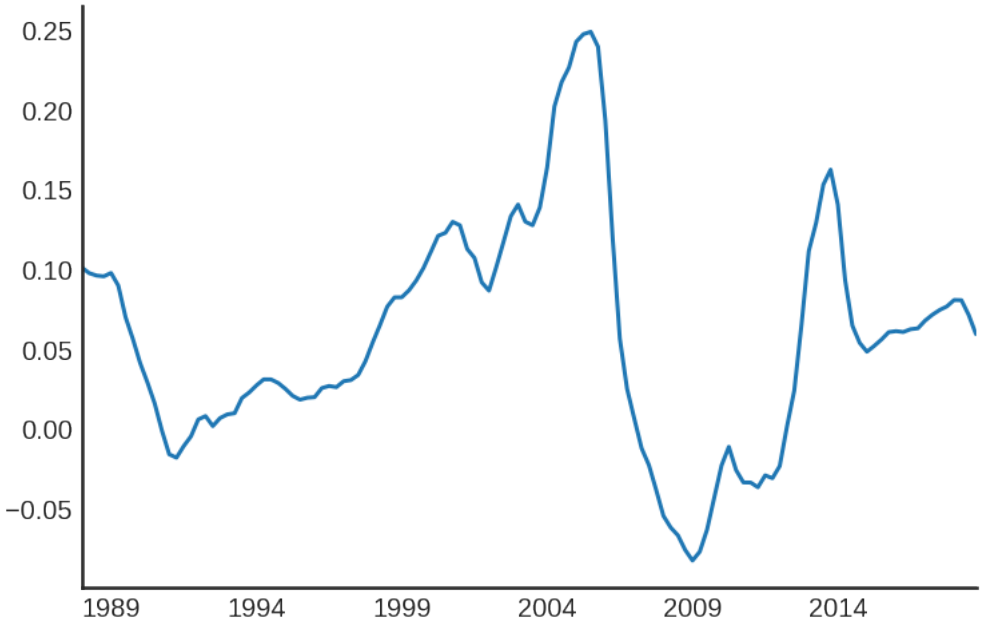
```
P-value                         0.466
Lags                                7
-------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -5.58 (1%), -5.07 (5%), -4.83 (10%)
Null Hypothesis: The process contains a unit root with a single structural
break.
Alternative Hypothesis: The process is trend and break stationary.

        Zivot-Andrews Results
=====================================
Test Statistic                -7.868
P-value                        0.000
Lags                               3
-------------------------------------

Trend: Constant
Critical Values: -5.28 (1%), -4.81 (5%), -4.57 (10%)
Null Hypothesis: The process contains a unit root with a single structural
break.
Alternative Hypothesis: The process is trend and break stationary.
```

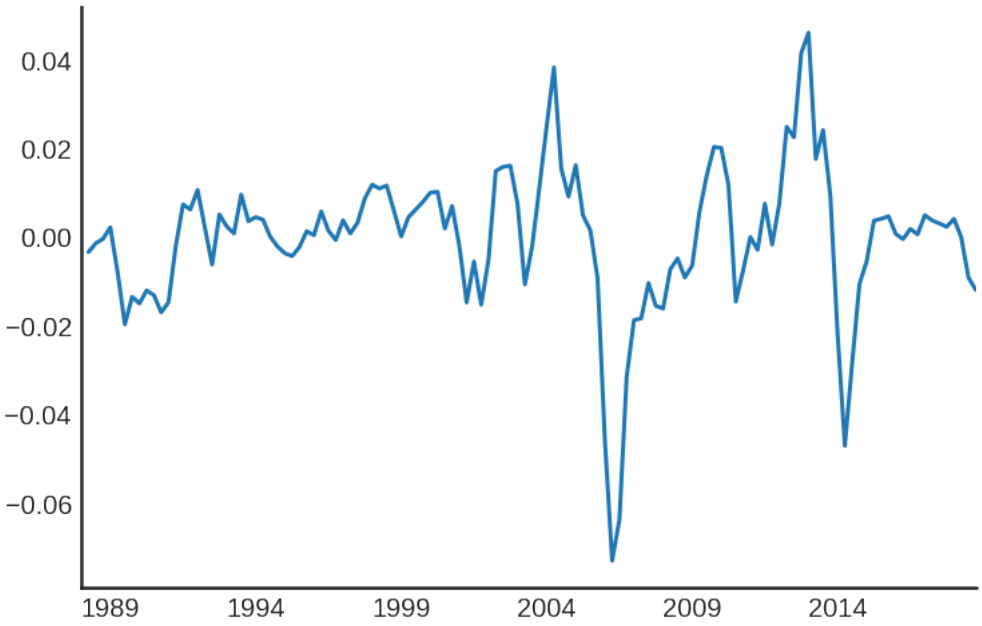
Conclusão: Não rejeita-se, a 5% de significância, a hipótese nula do teste Zivot e Andrews (1992), ou seja, existe uma quebra estrutural na série analizada. Portanto, os testes de raíz unitária desta série estão comprometidos. Em outras palavras, na presenca de quebra estrutural, os testes são viesados na direcao da não rejeicao da hipotese de raiz unitaria. Em primeira diferença, a série é estacionária.

Investimento residencial: Conclusão Há suspeitas de quebra estrutural na taxa de crescimento do investimento residencial e, portanto, os testes de raíz unitária podem ser comprometidos e são viesados na direção de não-rejeitação da hipótese de raíz unitária.

5.2 Inflação do preço dos imóveis

[22]:
```
df["Inflação"].plot()
sns.despine()
plt.show()

df["Inflação"].diff().plot()
sns.despine()
plt.show()
```





Destaque será dado para os testes com intercepto e sem tendência.

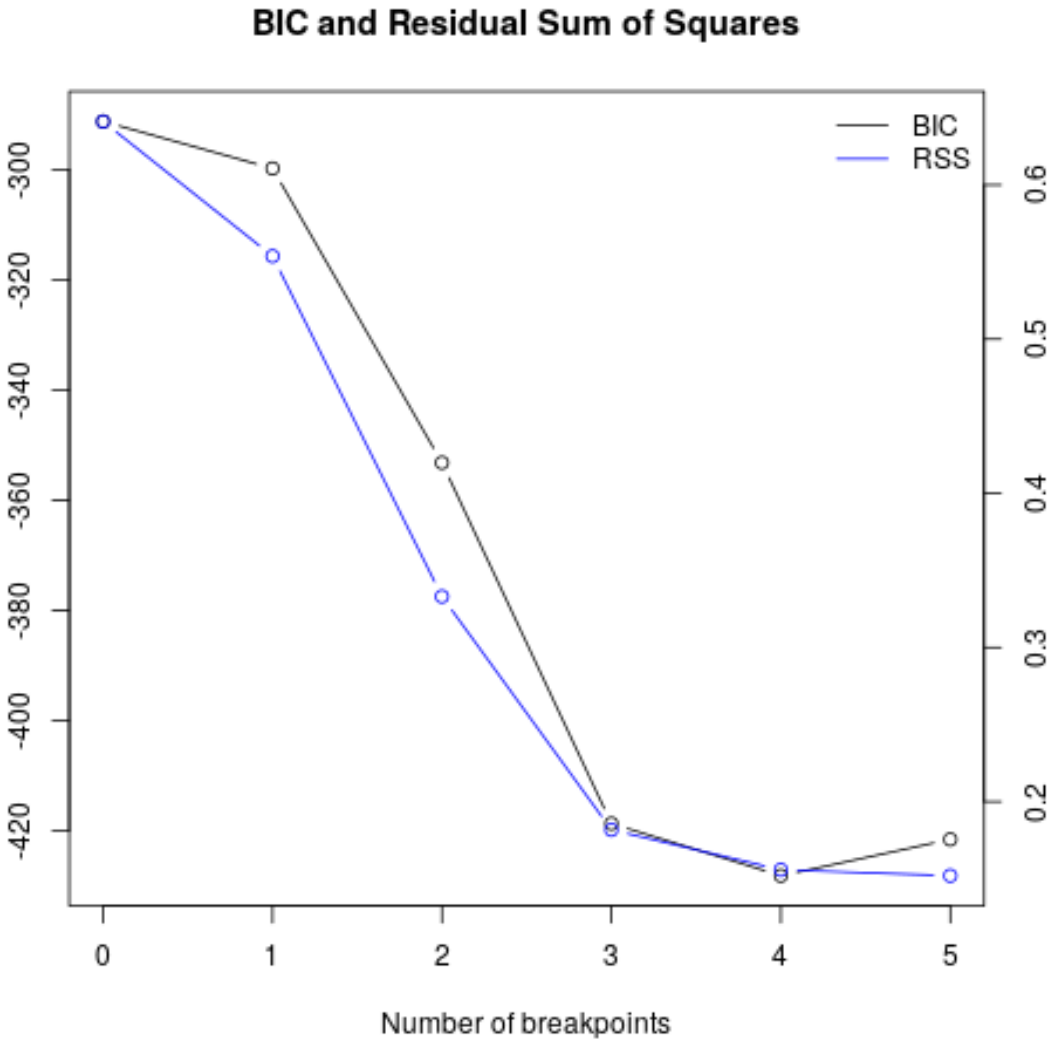
5.2.1 Teste de Chow

- H0: Sem quebra estrutural

- H1: Quebra estrutural

[23]:
```
%%R -i df
df = df[,2]
df = ts(df, start = c(1987,4), frequency = 4)
df.chow = Fstats(df ~ 1, from = 0.15)
print(breakpoints(df.chow)) #mostra o ponto da quebra quebra estrutural
print("P-Valor do teste:")
print(sctest(df.chow))
df.bp = breakpoints(df ~ 1, h = 0.15) #nota-se que a função "breakpoints"␣
 ↪aponta, no máximo, cinco quebras estruturais, se houver
plot(summary(df.bp))
```

```
         Optimal 2-segment partition:

Call:
breakpoints.Fstats(obj = df.chow)

Breakpoints at observation number:
75

Corresponding to breakdates:
2006(2)
[1] "P-Valor do teste:"

        supF test

data:  df.chow
sup.F = 19.237, p-value = 0.0003311
```



Conclusão: Rejeita-se, a 5% de significância, a hipótese nula do teste de Chow e, portanto, a série apresenta uma quebra estrutural.

5.2.2 Teste de Phillips-Perron

[24]:
```
print(PhillipsPerron(df["Inflação"], trend='ct').summary())
print(PhillipsPerron(df["Inflação"].diff().dropna(), trend='c').summary())
```

```
     Phillips-Perron Test (Z-tau)
=====================================
Test Statistic                -2.272
P-value                        0.449
Lags                              13
```

```
-------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -4.03 (1%), -3.45 (5%), -3.15 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
     Phillips-Perron Test (Z-tau)
=====================================
Test Statistic                -3.197
P-value                        0.020
Lags                              13
-------------------------------------

Trend: Constant
Critical Values: -3.49 (1%), -2.89 (5%), -2.58 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
```

Conclusão: A um nível de significância de 5%, não rejeita-se a hipótese nula do teste Phillips-Perron e, portanto, suspeita-se que o processo contenha uma raíz unitária. Em primeira diferença, a série é fracamente estacionária.

5.2.3 Teste de Zivot-Andrews

[25]:
```
print(ZivotAndrews(df["Inflação"], trend = 'ct').summary(),"\n")
print(ZivotAndrews(df["Inflação"].diff().dropna(), trend = 'c').summary(),"\n")
```

```
        Zivot-Andrews Results
=====================================
Test Statistic                -5.371
P-value                        0.021
Lags                               6
-------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -5.58 (1%), -5.07 (5%), -4.83 (10%)
Null Hypothesis: The process contains a unit root with a single structural
break.
Alternative Hypothesis: The process is trend and break stationary.

        Zivot-Andrews Results
=====================================
Test Statistic                -4.945
P-value                        0.035
Lags                               5
-------------------------------------

Trend: Constant
```

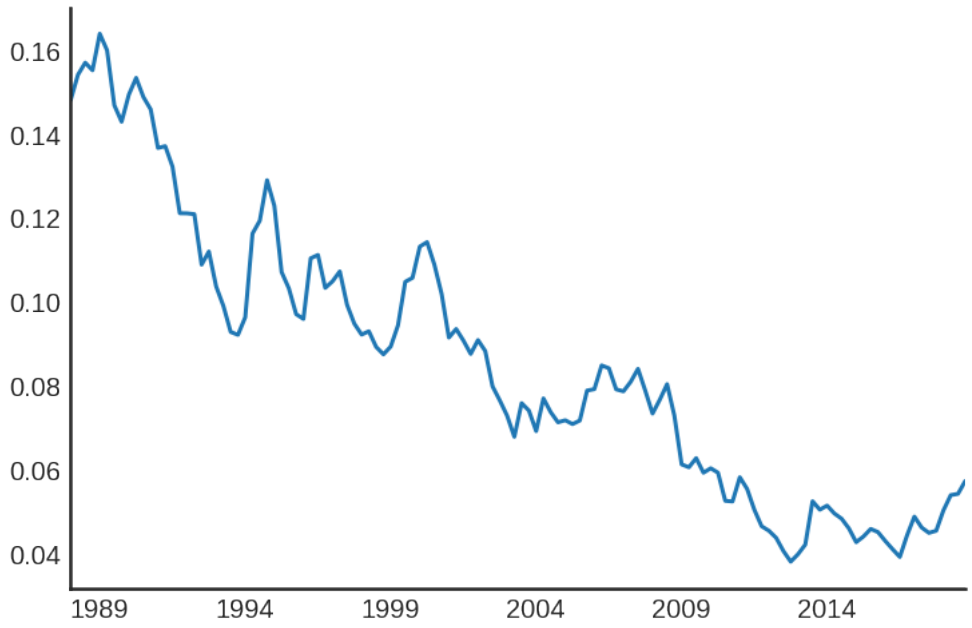
```
Critical Values: -5.28 (1%), -4.81 (5%), -4.57 (10%)
Null Hypothesis: The process contains a unit root with a single structural
break.
Alternative Hypothesis: The process is trend and break stationary.
```

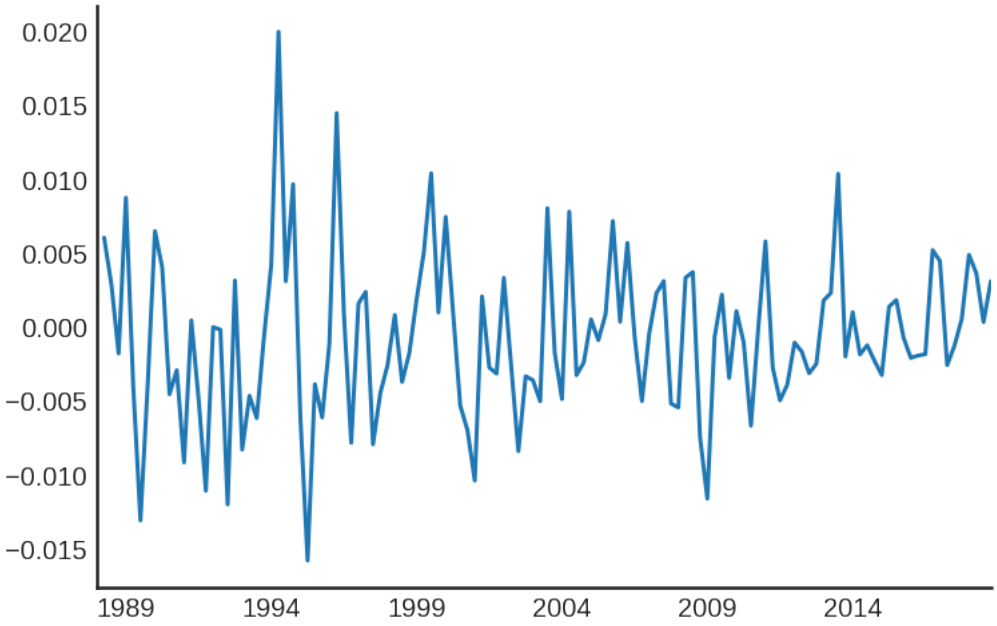
Conclusão: Rejeita-se, a 5% de significância, a hipótese nula do teste Zivot e Andrews (1992), ou seja, **não** existe uma quebra estrutural na série analizada. O mesmo vale para a série em primeira diferença.

Inflação dos imóveis: Conclusão Suspeita-se de quebra estrutural na série.

5.3 Taxa de juros das hipotecas

[26]:
```
df["Taxa de juros"].plot()
sns.despine()
plt.show()

df["Taxa de juros"].diff().plot()
sns.despine()
plt.show()
```





Destaque será dado para o teste com constante e tendência.

5.3.1 Teste de Phillips-Perron

[27]:
```
print(PhillipsPerron(df["Taxa de juros"], trend='ct').summary())
```

```
     Phillips-Perron Test (Z-tau)
=====================================
Test Statistic                 -2.209
P-value                         0.485
Lags                               13
-------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -4.03 (1%), -3.45 (5%), -3.15 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
```

Conclusão: A um nível de significância de 5%, não rejeita-se a hipótese nula do teste Phillips-Perron e, portanto, suspeita-se que o processo contenha uma raíz unitária.

5.3.2 Teste de Zivot-Andrews

[28]:
```
print(ZivotAndrews(df["Taxa de juros"], trend = 'c').summary(),"\n")
print(ZivotAndrews(df["Taxa de juros"], trend = 't').summary(),"\n")
print(ZivotAndrews(df["Taxa de juros"], trend = 'ct').summary(),"\n") ###␣
 ↪Destaque
```

```
        Zivot-Andrews Results
====================================
Test Statistic                -3.732
P-value                        0.525
Lags                               8
------------------------------------

Trend: Constant
Critical Values: -5.28 (1%), -4.81 (5%), -4.57 (10%)
Null Hypothesis: The process contains a unit root with a single structural
break.
Alternative Hypothesis: The process is trend and break stationary.

        Zivot-Andrews Results
====================================
Test Statistic                -3.954
P-value                        0.149
Lags                               8
------------------------------------

Trend: Linear Time Trend (No Constant)
Critical Values: -5.03 (1%), -4.41 (5%), -4.14 (10%)
Null Hypothesis: The process contains a unit root with a single structural
break.
Alternative Hypothesis: The process is trend and break stationary.

        Zivot-Andrews Results
====================================
Test Statistic                -4.141
P-value                        0.411
Lags                               8
------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -5.58 (1%), -5.07 (5%), -4.83 (10%)
Null Hypothesis: The process contains a unit root with a single structural
break.
Alternative Hypothesis: The process is trend and break stationary.
```

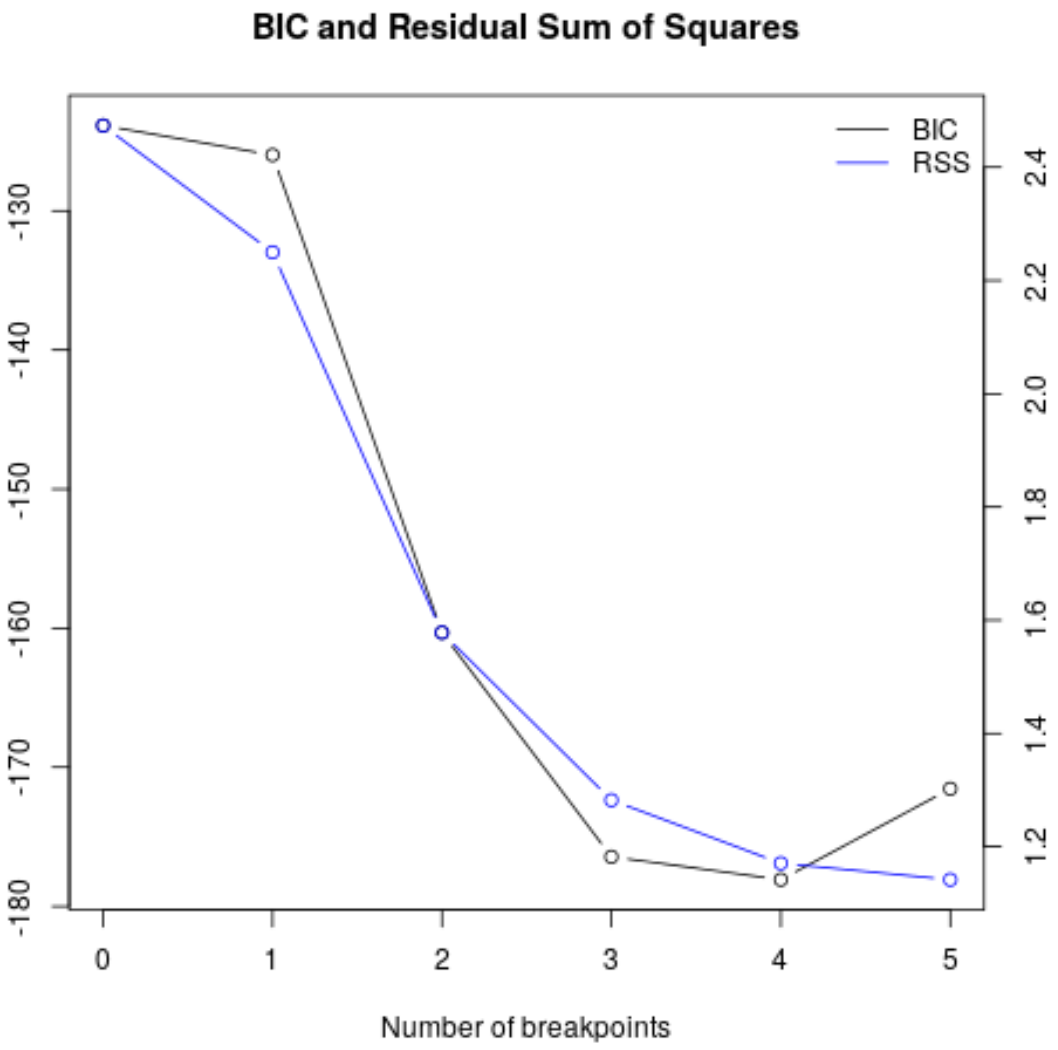
Conclusão: Não rejeita-se, a 5% de significância, a hipótese nula do teste Zivot e Andrews (1992), ou seja, **existe** uma quebra estrutural na série analizada.

5.3.3 Teste de Chow

- H0: Sem quebra estrutural
- H1: Quebra estrutural

[29]:
```
%%R -i df
df = df[,3]
df = ts(df, start = c(1987,4), frequency = 4)
df.chow = Fstats(df ~ 1, from = 0.15)
print(breakpoints(df.chow)) #mostra o ponto da quebra quebra estrutural
print("P-Valor do teste:")
print(sctest(df.chow))
df.bp = breakpoints(df ~ 1, h = 0.15) #nota-se que a função "breakpoints"␣
 ↪aponta, no máximo, cinco quebras estruturais, se houver
plot(summary(df.bp))
```

```
         Optimal 2-segment partition:

Call:
breakpoints.Fstats(obj = df.chow)

Breakpoints at observation number:
96

Corresponding to breakdates:
2011(3)
[1] "P-Valor do teste:"

        supF test

data:  df.chow
sup.F = 12.138, p-value = 0.009905
```



Conclusão: Rejeita-se, a 5% de significância, a hipótese nula do teste de Chow e, portanto, a série apresenta uma quebra estrutural.

Taxa de juros hipotecárias: Conclusão Suspeita-se de quebra estrutural na série.

5.3.4 Zivot-Andrews

[30]:
```
ZivotAndrews(df['gZ'])
```

[30]:
```
<class 'arch.unitroot.unitroot.ZivotAndrews'>
"""
        Zivot-Andrews Results
=====================================
Test Statistic                -4.024
```

```
P-value                         0.337
Lags                                7
-------------------------------------

Trend: Constant
Critical Values: -5.28 (1%), -4.81 (5%), -4.57 (10%)
Null Hypothesis: The process contains a unit root with a single structural
break.
Alternative Hypothesis: The process is trend and break stationary.
"""
```

[31]:
```
ZivotAndrews(df['gZ'].diff().dropna())
```

[31]:
```
<class 'arch.unitroot.unitroot.ZivotAndrews'>
"""
        Zivot-Andrews Results
=====================================
Test Statistic                -7.868
P-value                        0.000
Lags                               3
-------------------------------------

Trend: Constant
Critical Values: -5.28 (1%), -4.81 (5%), -4.57 (10%)
Null Hypothesis: The process contains a unit root with a single structural
break.
Alternative Hypothesis: The process is trend and break stationary.
"""
```

Conclusão: Processo possui uma quebra estrutural e uma raíz unitária em nível. Em primeira diferença, a série é estacionária em torno da tendência.

5.3.5 Teste DFGLS

[32]:
```
DFGLS(df['gZ'], method = "BIC").summary()
```

[32]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
      Dickey-Fuller GLS Results
=====================================
Test Statistic                -2.084
P-value                        0.037
Lags                               4
-------------------------------------

Trend: Constant
Critical Values: -2.73 (1%), -2.11 (5%), -1.80 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
```

```
"""
```

[33]:
```
DFGLS(df['gZ'].diff().dropna(), method = "BIC").summary()
```

[33]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
      Dickey-Fuller GLS Results
====================================
Test Statistic                 -7.546
P-value                         0.000
Lags                                3
------------------------------------

Trend: Constant
Critical Values: -2.73 (1%), -2.11 (5%), -1.80 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

Conclusão: Já pelo teste de DFGLS conclui-se que a série é fracamente estacionária tanto em nível quanto em primeira diferença adotando um nível de significância de 5%.

5.3.6 Teste KPSS

[34]:
```
KPSS(df['gZ']).summary()
```

[34]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
    KPSS Stationarity Test Results
====================================
Test Statistic                  0.081
P-value                         0.689
Lags                                6
------------------------------------

Trend: Constant
Critical Values: 0.74 (1%), 0.46 (5%), 0.35 (10%)
Null Hypothesis: The process is weakly stationary.
Alternative Hypothesis: The process contains a unit root.
"""
```

[35]:
```
KPSS(df['gZ'].diff().dropna()).summary()
```

[35]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
    KPSS Stationarity Test Results
====================================
Test Statistic                  0.045
P-value                         0.904
Lags                                2
```

```
-------------------------------------

Trend: Constant
Critical Values: 0.74 (1%), 0.46 (5%), 0.35 (10%)
Null Hypothesis: The process is weakly stationary.
Alternative Hypothesis: The process contains a unit root.
"""
```

Conclusão: Pelo teste de KPSS conclui-se que a série é fracamente estacionária tanto em nível quanto em primeira diferença adotando um nível de significância de 5%.

5.3.7 Teste de Phillip Perron

[36]:
```
PhillipsPerron(df["gZ"]).summary()
```

[36]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
     Phillips-Perron Test (Z-tau)
=====================================
Test Statistic                -2.876
P-value                        0.048
Lags                              13
-------------------------------------

Trend: Constant
Critical Values: -3.48 (1%), -2.89 (5%), -2.58 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

[37]:
```
PhillipsPerron(df["gZ"].diff().dropna()).summary()
```

[37]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
     Phillips-Perron Test (Z-tau)
=====================================
Test Statistic                -9.753
P-value                        0.000
Lags                              13
-------------------------------------

Trend: Constant
Critical Values: -3.49 (1%), -2.89 (5%), -2.58 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

Conclusão: Série é estacionária em primeira diferença.

Investimento residencial - Estacionariedade: Conclusão Uma vez que a série possui uma quebra estrutural, os resultados do testes são viesados no sentido de apontar uma raíz unitária em uma série que é estacionária (ver p. 14). Desse modo, pelos testes KPSS e DFGLS indicarem que a série é fracamente estacionária enquanto o ADF, Phillip-Perron e Zivot-Andrews indicam o oposto (provavelmente pela quebra estrutural), adota-se a série tomada em primeira diferença.

```
df["gZ"] = df["gZ"].diff()
```

5.4 Inflação dos imóveis

5.4.1 Autocorrelação e autoccorelação parcial

{{plot_acf(df[“Inflação”]);sns.despine(); plt.show()}} {{plot_pacf(df[“Inflação”]);sns.despine(); plt.show()}}

5.4.2 Inspeção gráfica

{{df[“Inflação”].plot();sns.despine(); plt.show()}}

A análise do gráfico da taxa de inflação indica que a série não possui tendência mas tem intercepto. Deste modo, o teste a ser considerado é aquele cujo valor crítico é τ^B.

[38]:
```
ADF(df['Inflação'], method = 'BIC').summary()
```

[38]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
   Augmented Dickey-Fuller Results
=====================================
Test Statistic                 -3.147
P-value                         0.023
Lags                                5
-------------------------------------

Trend: Constant
Critical Values: -3.49 (1%), -2.89 (5%), -2.58 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

[39]:
```
ADF(df['Inflação'].diff().dropna(), method = 'BIC').summary()
```

[39]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
   Augmented Dickey-Fuller Results
=====================================
Test Statistic                 -5.094
P-value                         0.000
Lags                                1
-------------------------------------

Trend: Constant
Critical Values: -3.49 (1%), -2.89 (5%), -2.58 (10%)
```

```
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

Conclusão: Adotando um nível de significância de 5%, conclui-se pelo teste ADF que a série é fracamente estacionária. Tal resultado permanece diante da possibilidade de quebra estrutural uma vez que os resultados são viasados no sentido da não rejeição da hipótese nula do teste.

5.4.3 Teste Zivot-Andrews

[40]:
```
ZivotAndrews(df["Inflação"], trend = 'c').summary()
```

[40]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
        Zivot-Andrews Results
====================================
Test Statistic                -5.501
P-value                        0.004
Lags                               6
------------------------------------

Trend: Constant
Critical Values: -5.28 (1%), -4.81 (5%), -4.57 (10%)
Null Hypothesis: The process contains a unit root with a single structural
break.
Alternative Hypothesis: The process is trend and break stationary.
"""
```

[41]:
```
ZivotAndrews(df["Inflação"].diff().dropna(), trend = 'c').summary()
```

[41]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
        Zivot-Andrews Results
====================================
Test Statistic                -4.945
P-value                        0.035
Lags                               5
------------------------------------

Trend: Constant
Critical Values: -5.28 (1%), -4.81 (5%), -4.57 (10%)
Null Hypothesis: The process contains a unit root with a single structural
break.
Alternative Hypothesis: The process is trend and break stationary.
"""
```

Conclusão: Processo é fracamente estacionário em nível.

5.4.4 Teste DFGLS

[42]:
```
DFGLS(df['Inflação'], method = 'BIC').summary()
```

[42]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
      Dickey-Fuller GLS Results
====================================
Test Statistic                 -2.791
P-value                         0.005
Lags                                5
------------------------------------

Trend: Constant
Critical Values: -2.73 (1%), -2.11 (5%), -1.80 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

[43]:
```
DFGLS(df['Inflação'].diff().dropna(), method = 'BIC').summary()
```

[43]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
      Dickey-Fuller GLS Results
====================================
Test Statistic                 -5.042
P-value                         0.000
Lags                                1
------------------------------------

Trend: Constant
Critical Values: -2.73 (1%), -2.11 (5%), -1.79 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

Conclusão: Obtém-se as mesmas conclusões do teste anterior, qual seja, a série é fracamente estacionária.

5.4.5 Teste KPSS

[44]:
```
KPSS(df['Inflação']).summary()
```

[44]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
    KPSS Stationarity Test Results
====================================
Test Statistic                  0.119
P-value                         0.501
Lags                                6
```

```
------------------------------------

Trend: Constant
Critical Values: 0.74 (1%), 0.46 (5%), 0.35 (10%)
Null Hypothesis: The process is weakly stationary.
Alternative Hypothesis: The process contains a unit root.
"""
```

[45]:
```
KPSS(df['Inflação'].diff().dropna()).summary()
```

[45]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
    KPSS Stationarity Test Results
====================================
Test Statistic                  0.055
P-value                         0.845
Lags                                5
------------------------------------

Trend: Constant
Critical Values: 0.74 (1%), 0.46 (5%), 0.35 (10%)
Null Hypothesis: The process is weakly stationary.
Alternative Hypothesis: The process contains a unit root.
"""
```

Conclusão: Obtém-se as mesmas conclusões do teste ADF e DFGLS, qual seja, a série é fracamente estacionária.

5.4.6 Teste Phillip-Perron

[46]:
```
PhillipsPerron(df["Inflação"]).summary()
```

[46]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
     Phillips-Perron Test (Z-tau)
====================================
Test Statistic                -2.271
P-value                        0.182
Lags                              13
------------------------------------

Trend: Constant
Critical Values: -3.48 (1%), -2.89 (5%), -2.58 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

[47]:
```
PhillipsPerron(df["Inflação"].diff().dropna()).summary()
```

[47]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
     Phillips-Perron Test (Z-tau)
=====================================
Test Statistic                 -3.197
P-value                         0.020
Lags                               13
-------------------------------------

Trend: Constant
Critical Values: -3.49 (1%), -2.89 (5%), -2.58 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

Conclusão: Série fracamente estacionária em primeira diferença.

Inflação de imóveis - Estacionariedade: Conclusão Pelos testes realizados anteriormente (exceto Phillips-Perron), conclui-se que há indícios que a série seja fracamente estacionária em nível a um nível de significância de 5%. Os mesmos resultados foram obtidos via gretl e podem ser viasualizados nos resultados do script.

```
df["Inflação"] = df["Inflação"]
```

5.5 Taxa de juros das hipotecas

{{df[“Taxa de juros”].plot();sns.despine();plt.show()}}

Pela análise gráfica, verifica-se que a série possui intercepto e tendência e isso será incluido nos testes. Além disso, dada a presença de quebra estrutural, será considerando $\tau^c = -4,24$.

5.5.1 Teste ADF

[48]:
```
ADF(df['Taxa de juros'], method='BIC', trend='ct').summary()
```

[48]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
   Augmented Dickey-Fuller Results
=====================================
Test Statistic                 -3.171
P-value                         0.090
Lags                                1
-------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -4.03 (1%), -3.45 (5%), -3.15 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

[49]:
```
ADF(df['Taxa de juros'].diff().dropna(), method='BIC', trend='ct').summary()
```

[49]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
   Augmented Dickey-Fuller Results
====================================
Test Statistic                -9.167
P-value                        0.000
Lags                               0
------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -4.03 (1%), -3.45 (5%), -3.15 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

Conclusão: Considerando os valores críticos para o teste com presença de raíz unitária fornecidos por Bueno (2005), conclui-se a 5% de significância que a série em nível possui uma raíz unitária enquanto a série em primeira diferença é estacionária.

5.5.2 Teste Zivot Andrews

[50]:
```
ZivotAndrews(df["Taxa de juros"], trend='ct').summary()
```

[50]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
        Zivot-Andrews Results
====================================
Test Statistic                -4.141
P-value                        0.411
Lags                               8
------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -5.58 (1%), -5.07 (5%), -4.83 (10%)
Null Hypothesis: The process contains a unit root with a single structural
break.
Alternative Hypothesis: The process is trend and break stationary.
"""
```

[51]:
```
ZivotAndrews(df["Taxa de juros"].diff().dropna(), trend='c').summary()
```

[51]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
        Zivot-Andrews Results
====================================
Test Statistic                -7.457
P-value                        0.000
Lags                               3
```

```
-------------------------------------

Trend: Constant
Critical Values: -5.28 (1%), -4.81 (5%), -4.57 (10%)
Null Hypothesis: The process contains a unit root with a single structural
break.
Alternative Hypothesis: The process is trend and break stationary.
"""
```

Conclusão: Série é fracamente estacionária em primeira diferença.

5.5.3 Teste DFGLS

[52]:
```
DFGLS(df['Taxa de juros'], method='BIC', trend='ct').summary()
```

[52]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
      Dickey-Fuller GLS Results
====================================
Test Statistic                -2.980
P-value                        0.036
Lags                               1
-------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -3.58 (1%), -3.00 (5%), -2.71 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

[53]:
```
DFGLS(df['Taxa de juros'].diff().dropna(), method='BIC', trend='ct').summary()
```

[53]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
      Dickey-Fuller GLS Results
====================================
Test Statistic                -7.681
P-value                        0.000
Lags                               0
-------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -3.58 (1%), -3.00 (5%), -2.71 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

Conclusão: Adotando um nível de significância de 5%, conclui-se que a série é fracamente estacionária pelo teste DFGLS e o mesmo vale para a primeira diferença.

5.5.4 Teste KPSS

[54]:
```
KPSS(df['Taxa de juros'], trend='ct').summary()
```

[54]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
    KPSS Stationarity Test Results
====================================
Test Statistic                  0.139
P-value                         0.061
Lags                                6
------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: 0.22 (1%), 0.15 (5%), 0.12 (10%)
Null Hypothesis: The process is weakly stationary.
Alternative Hypothesis: The process contains a unit root.
"""
```

[55]:
```
KPSS(df['Taxa de juros'].diff().dropna(), trend='ct').summary()
```

[55]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
    KPSS Stationarity Test Results
====================================
Test Statistic                  0.036
P-value                         0.786
Lags                                0
------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: 0.22 (1%), 0.15 (5%), 0.12 (10%)
Null Hypothesis: The process is weakly stationary.
Alternative Hypothesis: The process contains a unit root.
"""
```

Conclusão: Obtém-se os mesmos resultados do teste anterior adotando um nível de significância de 5%, ou seja, a série é fracamente estacionária.

Teste Phillips-Perron

[56]:
```
PhillipsPerron(df["Taxa de juros"]).summary()
```

[56]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
     Phillips-Perron Test (Z-tau)
====================================
Test Statistic                 -1.701
P-value                         0.430
Lags                               13
------------------------------------
```

```
Trend: Constant
Critical Values: -3.48 (1%), -2.89 (5%), -2.58 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

[57]:
```
PhillipsPerron(df["Taxa de juros"].diff().dropna()).summary()
```

[57]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
     Phillips-Perron Test (Z-tau)
====================================
Test Statistic                -8.999
P-value                        0.000
Lags                              13
------------------------------------

Trend: Constant
Critical Values: -3.49 (1%), -2.89 (5%), -2.58 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

Conclusão: Série é fracamente estacionária em primeira diferença.

Taxa de juros das hipotecas - Estacionariedade: Conclusão Dada a presença de quebra estrutural, os testes de raíz unitária tendem a apontar que uma série estacionária é não estacionária, ou seja, os resultados dos testes são viesados. Desse modo, pelos testes KPSS e DFGLS, conclui-se que a série é fracamente estacionária enquanto o teste ADF afirma o oposto. No entanto, pela inspeção gráfica da série e das funções de autocorrelação e autocorreção parcial (abaixo), conclui-se a série deve ser tomada em primeira diferença. Os testes de Phillips-Perron e Zivot-Andrews corroboram para tal decisão.

```
df["Taxa de juros"] = df["Taxa de juros"].diff()
```

5.5.5 Autocorrelação e autoccorelação parcial - série em nível

{{plot_acf(df[“Taxa de juros”]);sns.despine(); plt.show()}} {{plot_pacf(df[“Taxa de juros”]);sns.despine(); plt.show()}}

5.5.6 Autocorrelação e autoccorelação parcial - série em diferença

{{plot_acf(df[“Taxa de juros”].diff().dropna());sns.despine(); plt.show()}} {{plot_pacf(df[“Taxa de juros”].diff().dropna());sns.despine(); plt.show()}}

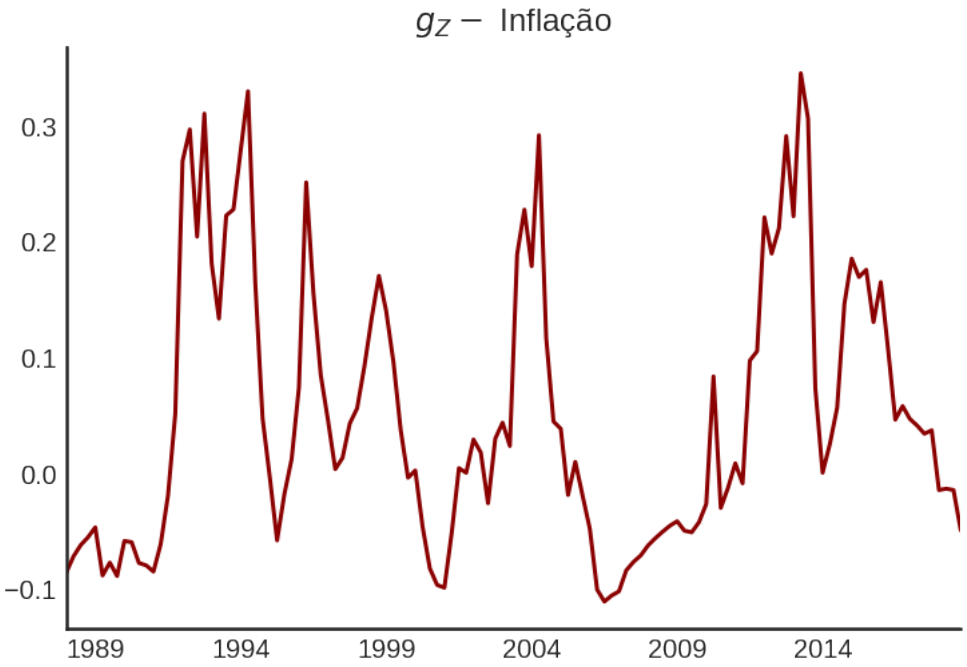
5.6 Investimento residencial - Inflação de imóveis

Resta testar se existe alguma relação de longo prazo entre inflação de imóveis e investimento residencial.

Testando:

$$y = g_Z - \text{Infla}$$

[58]:
```
(df["gZ"] - df["Inflação"]).plot(color = 'darkred', title ='$g_Z - $ Inflação')
sns.despine()
plt.show()
```



5.6.1 Teste ADF

[59]:
```
ADF((df['gZ']-df["Inflação"]), method = 'BIC').summary()
```

[59]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
   Augmented Dickey-Fuller Results
=====================================
Test Statistic                 -2.681
P-value                         0.077
Lags                                4
-------------------------------------

Trend: Constant
Critical Values: -3.49 (1%), -2.89 (5%), -2.58 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
```

```
"""
```

[60]:
```
DFGLS((df['gZ']-df["Inflação"]), method = 'BIC').summary()
```

[60]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
      Dickey-Fuller GLS Results
====================================
Test Statistic                 -1.779
P-value                         0.074
Lags                                4
------------------------------------

Trend: Constant
Critical Values: -2.73 (1%), -2.11 (5%), -1.80 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

[61]:
```
KPSS((df['gZ']-df["Inflação"])).summary()
```

[61]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
    KPSS Stationarity Test Results
====================================
Test Statistic                  0.093
P-value                         0.620
Lags                                6
------------------------------------

Trend: Constant
Critical Values: 0.74 (1%), 0.46 (5%), 0.35 (10%)
Null Hypothesis: The process is weakly stationary.
Alternative Hypothesis: The process contains a unit root.
"""
```

[62]:
```
ZivotAndrews((df['gZ']-df["Inflação"])).summary()
```

[62]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
        Zivot-Andrews Results
====================================
Test Statistic                 -3.249
P-value                         0.819
Lags                                4
------------------------------------

Trend: Constant
Critical Values: -5.28 (1%), -4.81 (5%), -4.57 (10%)
Null Hypothesis: The process contains a unit root with a single structural
```

```
break.
Alternative Hypothesis: The process is trend and break stationary.
"""
```

[63]:
```
PhillipsPerron((df['gZ']-df["Inflação"]))
```

[63]:
```
<class 'arch.unitroot.unitroot.PhillipsPerron'>
"""
     Phillips-Perron Test (Z-tau)
=====================================
Test Statistic                 -3.123
P-value                         0.025
Lags                               13
-------------------------------------

Trend: Constant
Critical Values: -3.48 (1%), -2.89 (5%), -2.58 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
"""
```

Investimento - Inflação: Conclusão A diferença entre as séries não é estacionária.

6 Seleção das variávies

[64]:
```
df["Taxa de juros"] = df["Taxa de juros"].diff()
df["gZ"] = df["gZ"].diff()
#df["Inflação"] = df["Inflação"].diff()
df = df[["Taxa de juros", "Crise", "Inflação", "gZ"]].dropna() # Ordenação de␣
 ↪Cholesky
df.head()
```

[64]:

	Taxa de juros	Crise	Inflação	gZ
1988-06-30	0.006016	0	0.097437	0.010722
1988-09-30	0.002879	0	0.095949	0.008294
1988-12-31	-0.001817	0	0.095454	0.006252
1989-03-31	0.008713	0	0.097561	0.010689
1989-06-30	-0.003937	0	0.089601	-0.049557

7 VAR

Dúvida: Variável exógena do VAR deve ser estacionária também?

[65]:
```
model = VAR(
    df[["Inflação", "gZ"]],
    exog = df[["Taxa de juros", "Crise"]]
```

```
)
print(model.select_order(maxlags=12).summary())
```

```
 VAR Order Selection (* highlights the minimums)
================================================
       AIC         BIC         FPE         HQIC
------------------------------------------------
0       -10.87      -10.72   1.900e-05      -10.81
1       -13.85      -13.61   9.664e-07      -13.75
2       -14.70      -14.36   4.124e-07      -14.56
3       -14.77      -14.33   3.839e-07      -14.60
4       -15.00     -14.47*   3.055e-07      -14.78
5      -15.06*      -14.43  2.873e-07*     -14.81*
6       -15.04      -14.31   2.942e-07      -14.75
7       -15.01      -14.18   3.051e-07      -14.67
8       -14.96      -14.03   3.213e-07      -14.58
9       -14.90      -13.88   3.401e-07      -14.49
10      -14.88      -13.76   3.496e-07      -14.42
11      -14.83      -13.61   3.671e-07      -14.34
12      -14.77      -13.46   3.910e-07      -14.24
------------------------------------------------
```

Adotando o BIC como critério de seleção dada a parciomônia, estima-se uma VAR de ordem 5.

7.1 Estimação

[66]:
```
results = model.fit(ic='bic')
print(results.summary())
```

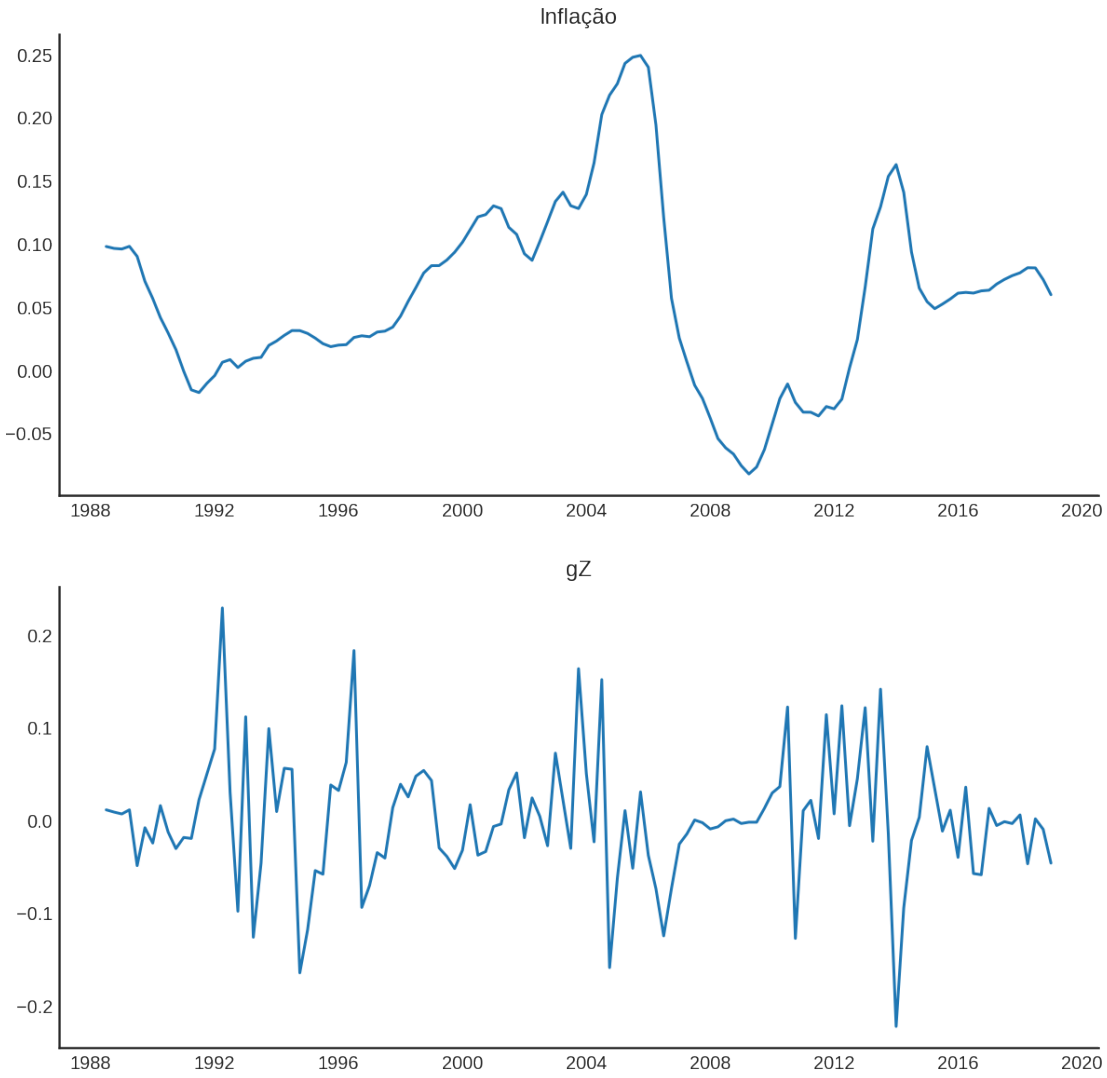
```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           sex, 08, nov, 2019
Time:                      15:07:07
--------------------------------------------------------------------
No. of Equations:         2.00000    BIC:                   -14.5805
Nobs:                     119.000    HQIC:                  -14.8857
Log likelihood:           582.404    FPE:                2.78666e-07
AIC:                     -15.0943    Det(Omega_mle):     2.33502e-07
--------------------------------------------------------------------
Results for equation Inflação
================================================================================
                 coefficient       std. error           t-stat            prob
--------------------------------------------------------------------------------
const               0.003964         0.001363            2.908           0.004
```

```
exog0              0.203873         0.160205           1.273           0.203
exog1             -0.005955         0.002583          -2.305           0.021
L1.Inflação        2.013206         0.102854          19.573           0.000
L1.gZ              0.001137         0.013586           0.084           0.933
L2.Inflação       -1.501162         0.222618          -6.743           0.000
L2.gZ             -0.001571         0.013424          -0.117           0.907
L3.Inflação        0.554472         0.221051           2.508           0.012
L3.gZ              0.020898         0.013487           1.549           0.121
================================================================================

Results for equation gZ
================================================================================
                 coefficient       std. error           t-stat            prob
--------------------------------------------------------------------------------
const             -0.117168         0.099422          -1.178           0.239
exog0             -0.023045         0.013303          -1.732           0.083
exog1              0.023575         0.008959           2.631           0.009
L1.Inflação        1.821987         1.053138           1.730           0.084
L1.gZ             -0.030866         0.016981          -1.818           0.069
L2.Inflação       -0.011107         0.676132          -0.016           0.987
L2.gZ              0.077869         0.089310           0.872           0.383
L3.Inflação        0.623563         1.463424           0.426           0.670
L3.gZ             -0.105347         0.088244          -1.194           0.233
================================================================================

Correlation matrix of residuals
            Inflação        gZ
Inflação    1.000000  0.355068
gZ          0.355068  1.000000
```

7.2 Inspeção

7.2.1 Gráfico dos inputs

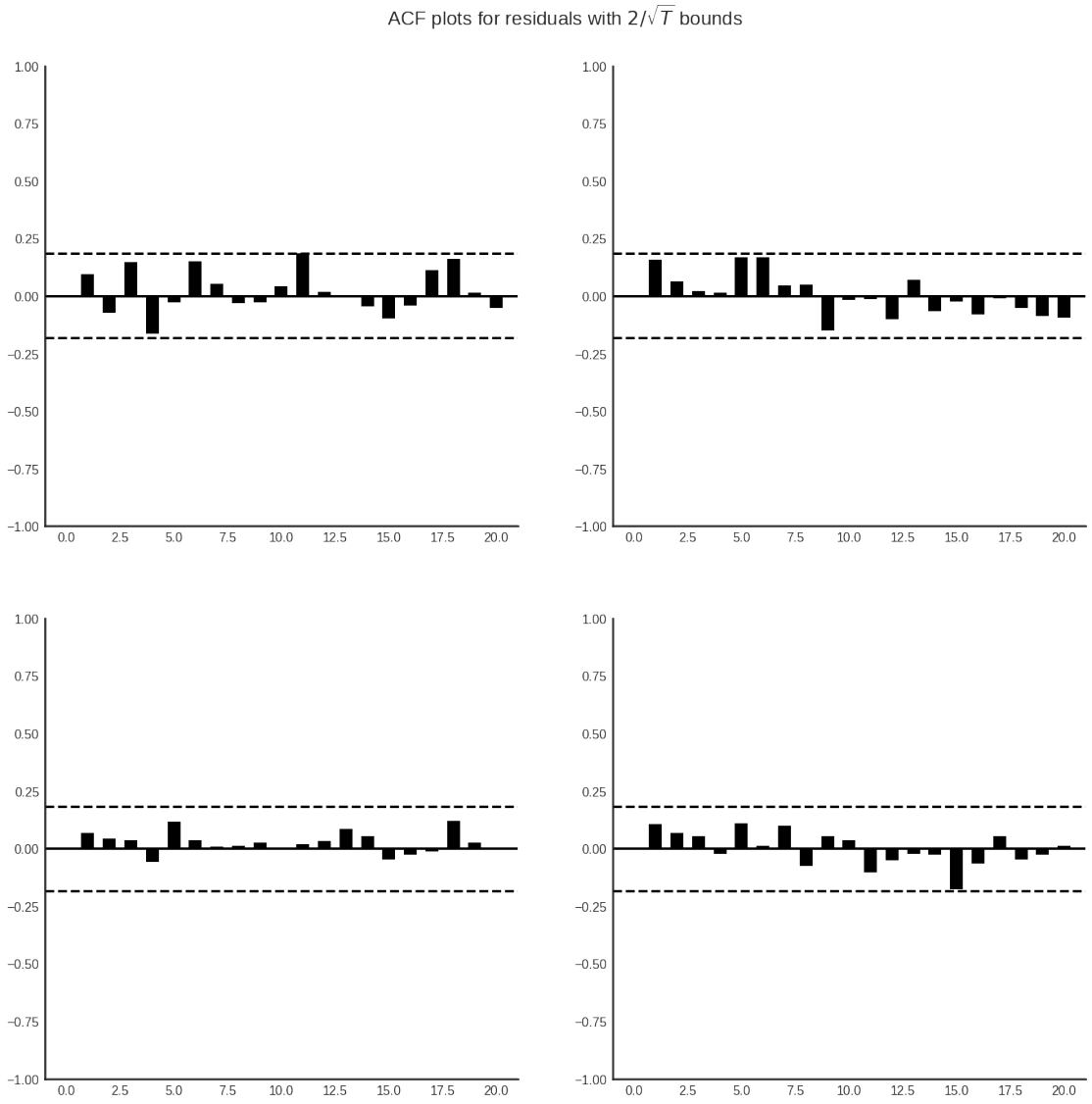
[67]:
```
results.plot()
sns.despine()
plt.show()
```



7.2.2 Autocorrelação dos resíduos

OBS: série consigo mesma na diagonal principal.

[68]:
```
results.plot_acorr(nlags = 20)
sns.despine()
plt.show()
```



Conclusão: Pela inspeção gráfica, o modelo não apresenta autocorrelação seria dos resíduos.

7.2.3 Estabilidade

[69]:
```
print("Estável:", results.is_stable(verbose=True))
```

```
Eigenvalues of VAR(1) rep
0.8358933660193383
0.8358933660193383
0.4805177123909676
0.4805177123909676
0.7961783387946113
0.7961783387946113
```

```
0.8056412828348928
0.8056412828348928
Estável: True
```

OBS: Apesar de estar escrito VAR(1), os resultados acima correspondem ao VAR(p)

7.2.4 Teste de Autocorrelacao dos Resduos

Teste de Portmanteau: Verificar se as autocorrelacoes multivariadas sao nulas

- usada para testar que as autocorrelacoes e correlacoes cruzadas nao sao significativas, para um dado nvel de significancia
- H0: Não correlação serial até o lag n
- H1: Há correlação serial até o lag n

[70]:
```
results.test_whiteness(nlags=12).summary()
```

[70]:
```
<class 'statsmodels.iolib.table.SimpleTable'>
```

Conclusão: Os resíduos **não** apresentam correlação serial.

[71]:
```
results.test_whiteness(nlags=12, adjusted=True).summary()
```

[71]:
```
<class 'statsmodels.iolib.table.SimpleTable'>
```

OBS: Com adjusted = True é feito o teste de Breusch-Godfrey
Conclusão: Os resíduos **não** apresentam correlação serial.

Teste de Ljung-Box

[72]:
```
def LjungBox_Pierce(resid, signif = 0.05, boxpierce = False, k = 4):
  """
  resid = residuals df
  signif = signif. level
  """
  var = len(resid.columns)
  print("H0: autocorrelations up to lag k equal zero")
  print('H1: autocorrelations up to lag k not zero')
  print("Box-Pierce: ", boxpierce)

  for i in range(var):
    print("Testing for ", resid.columns[i].upper(), ". Considering a␣
↪significance level of",  signif*100,"%")
    result = acorr_ljungbox(x = resid.iloc[:,i-1], lags = k, boxpierce =␣
↪boxpierce)[i-1] < signif
    for j in range(k):
      print("Reject H0 on lag " ,j+1,"? ", result[j])
    print("\n")
```

[73]:
```
LjungBox_Pierce(results.resid, k = 12, boxpierce=False)
```

```
H0: autocorrelations up to lag k equal zero
H1: autocorrelations up to lag k not zero
Box-Pierce:  False
Testing for  INFLAÇÃO . Considering a significance level of 5.0 %
Reject H0 on lag  1 ?  False
Reject H0 on lag  2 ?  False
Reject H0 on lag  3 ?  False
Reject H0 on lag  4 ?  False
Reject H0 on lag  5 ?  False
Reject H0 on lag  6 ?  False
Reject H0 on lag  7 ?  False
Reject H0 on lag  8 ?  False
Reject H0 on lag  9 ?  False
Reject H0 on lag  10 ?  False
Reject H0 on lag  11 ?  False
Reject H0 on lag  12 ?  False


Testing for  GZ . Considering a significance level of 5.0 %
Reject H0 on lag  1 ?  False
Reject H0 on lag  2 ?  False
Reject H0 on lag  3 ?  False
Reject H0 on lag  4 ?  False
Reject H0 on lag  5 ?  False
Reject H0 on lag  6 ?  False
Reject H0 on lag  7 ?  False
Reject H0 on lag  8 ?  False
Reject H0 on lag  9 ?  False
Reject H0 on lag  10 ?  False
Reject H0 on lag  11 ?  False
Reject H0 on lag  12 ?  False
```

7.2.5 Normalidade

[74]:
```
results.test_normality().summary()
```

[74]:
```
<class 'statsmodels.iolib.table.SimpleTable'>
```

7.3 Inspeção dos resíduos

[75]:
```
residuals = pd.DataFrame(results.resid)
residuals.columns = ["Inflação", 'gZ']
residuals.tail()
```

[75]:
```
            Inflação        gZ
```

```
2017-12-31  0.000225  0.007173
2018-03-31  0.001325 -0.062426
2018-06-30 -0.004309 -0.003512
2018-09-30 -0.007815 -0.020811
2018-12-31 -0.002562 -0.042838
```

7.3.1 ARCH-LM

[76]:
```
def ARCH_LM(resid, signif = 0.05, autolag = 'bic'):
  """
  df = residuals df
  signif = signif. level
  """
  var = len(resid.columns)
  print("H0: Residuals are homoscedastic")
  print('H1: Residuals are heteroskedastic')

  for i in range(var):
    print("Testing for ", resid.columns[i].upper())
    result = het_arch(resid = resid.iloc[:,i], autolag = autolag)
    print('LM p-value: ', result[1])
    print("Reject H0? ", result[1] < signif)
    print('F p-value: ', result[3])
    print("Reject H0? ", result[3] < signif)
    print('\n')
```

[77]:
```
ARCH_LM(residuals)
```

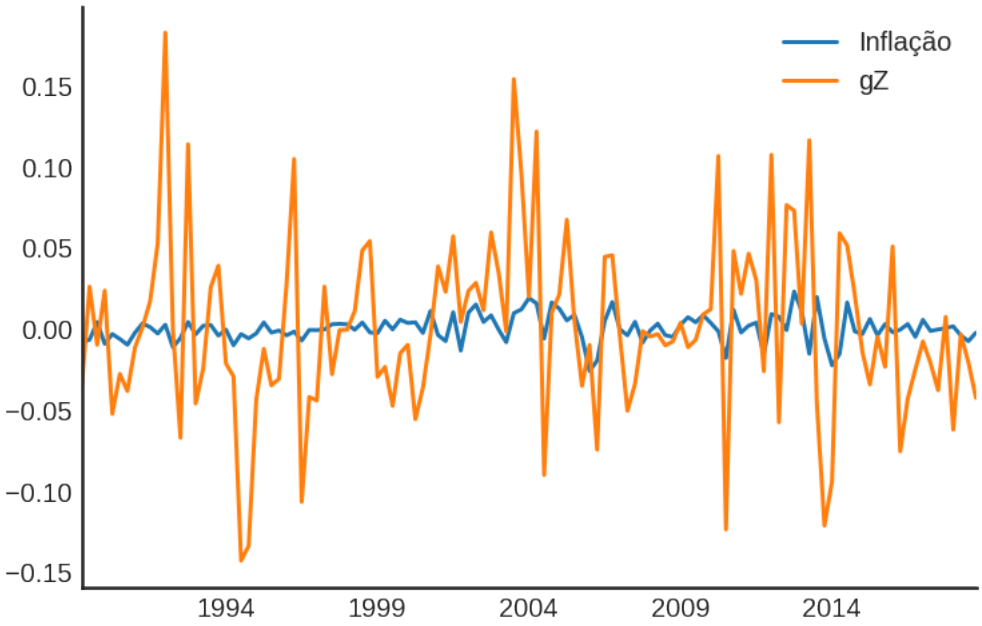
```
H0: Residuals are homoscedastic
H1: Residuals are heteroskedastic
Testing for  INFLAÇÃO
LM p-value:  0.0014357797890077253
Reject H0?  True
F p-value:  0.0012609712969068455
Reject H0?  True


Testing for  GZ
LM p-value:  0.051744826683253135
Reject H0?  False
F p-value:  0.05234845104088973
Reject H0?  False
```

Conclusão: Os resíduos apresentam heterocedasticidade condicional, impossibilitando a inferência estatística. Uma forma de contornar tal problema é por meio de estimadores robustos, o

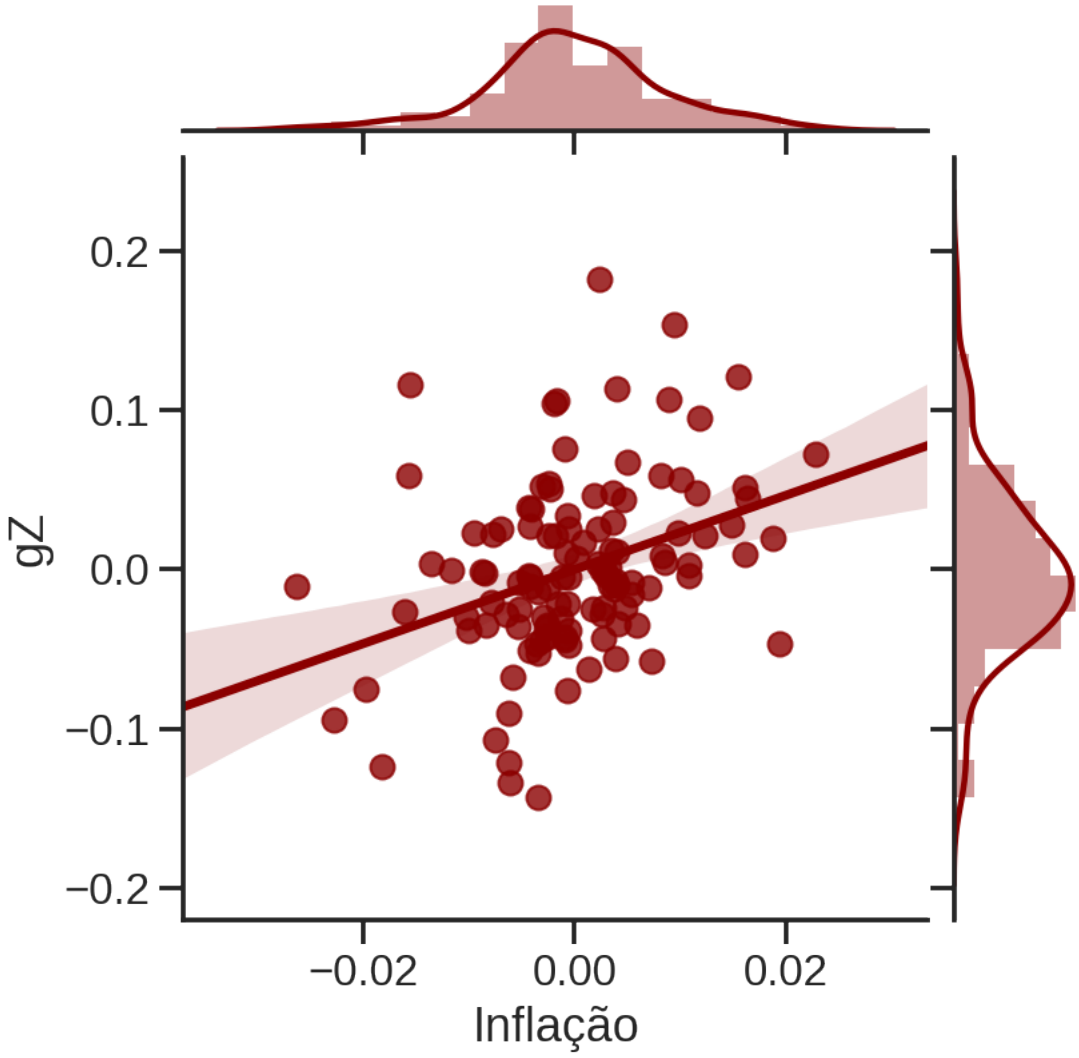
que não está implementado no python e será feito no gretl.

7.3.2 Conjunto

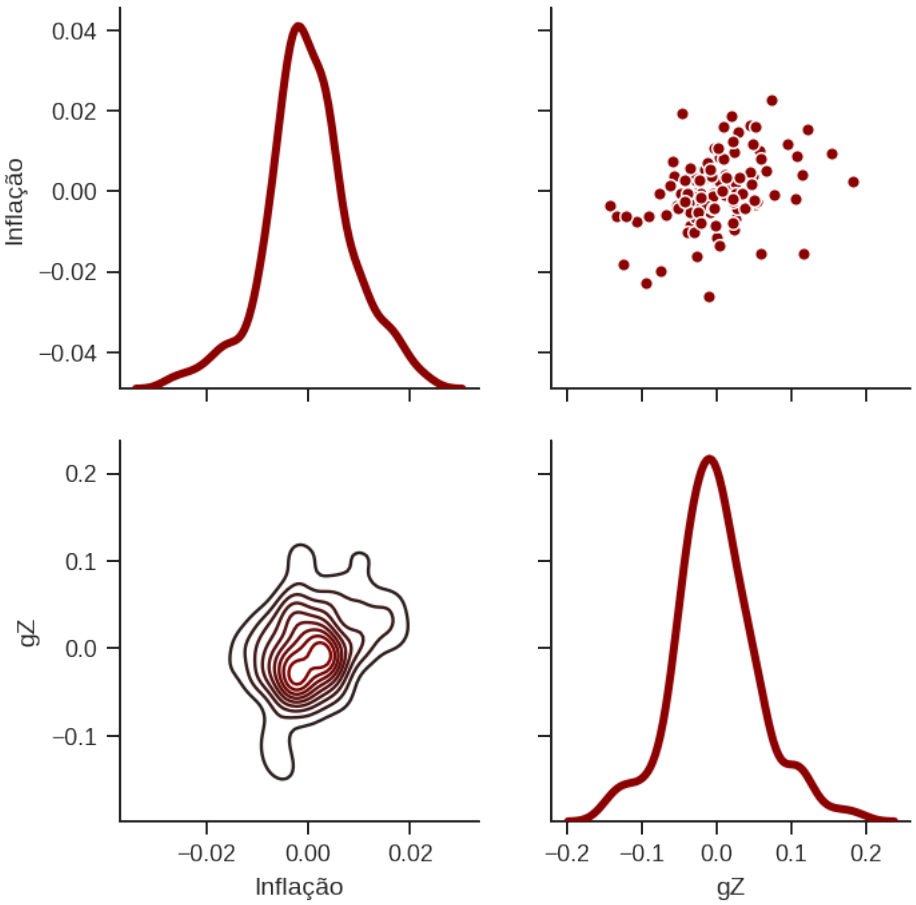
[78]:
```
residuals.plot()
sns.despine()
plt.show()
```



[79]:
```
sns.set_context('talk')
ax = sns.jointplot(
    x = 'Inflação',
    y = 'gZ',
#    data = residuals**2, kind="kde", space = 0, color = 'black',
    data = residuals, color = 'darkred', kind="reg",
)
plt.show()
```

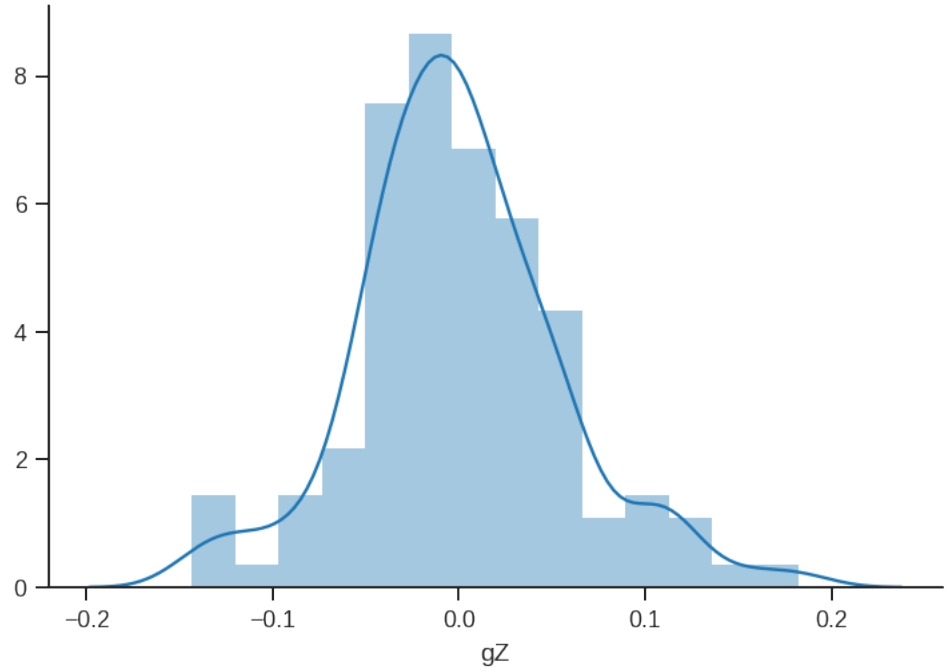


[80]:
```
sns.set_context('paper')
g = sns.PairGrid(residuals, diag_sharey=False)
g.map_lower(sns.kdeplot, color = 'darkred')
g.map_upper(sns.scatterplot, color = 'darkred')
g.map_diag(sns.kdeplot, lw=3, color = 'darkred')
plt.show()
g.savefig("./Escrita/Figs/Residuos_4.png", dpi=300)
```

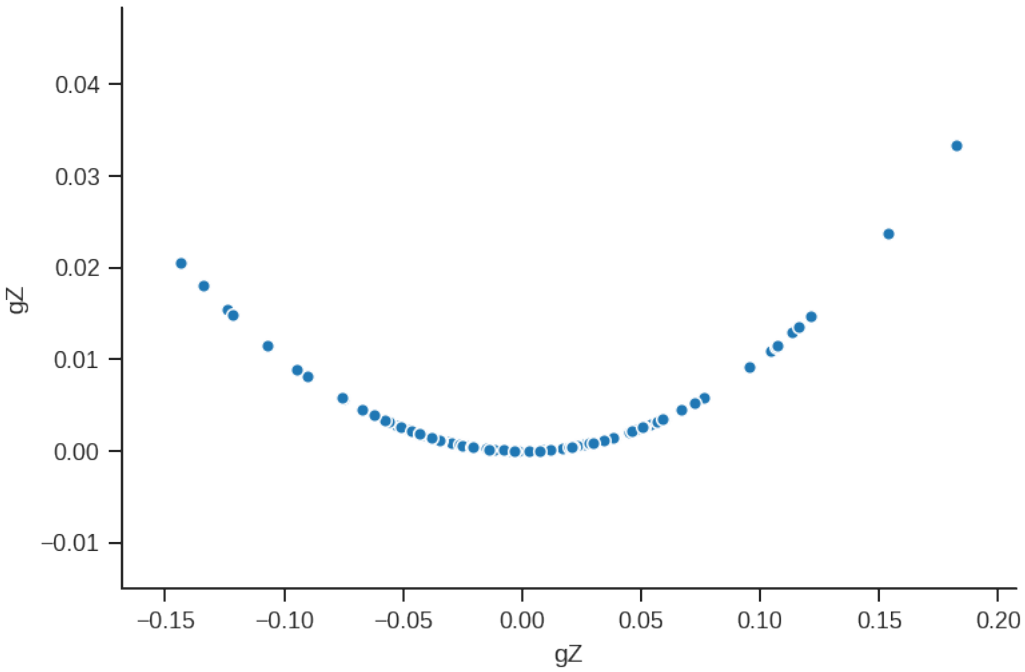


7.3.3 Investimento residencial

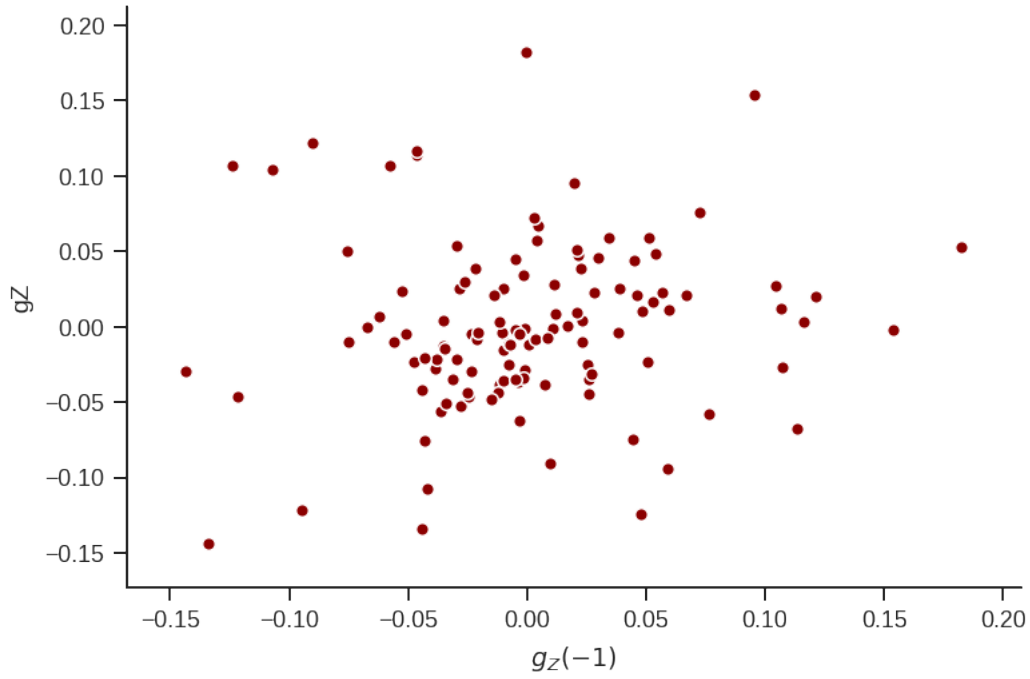
[81]:
```
sns.distplot(residuals["gZ"])
sns.despine()
plt.show()
```



[82]:
```
sns.scatterplot(x = residuals['gZ'], y = residuals['gZ']**2)
sns.despine()
plt.show()
```

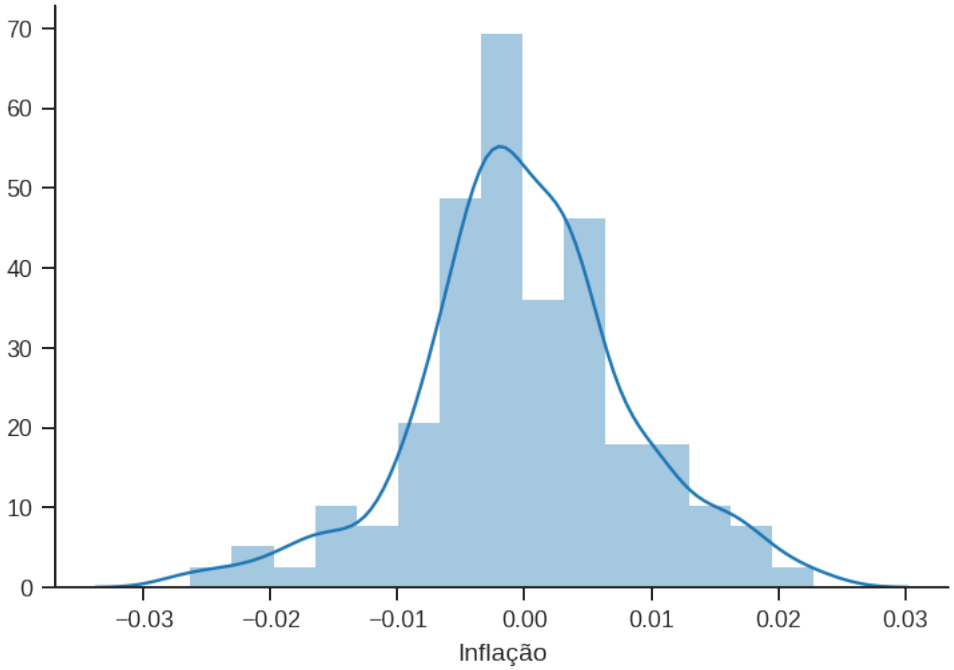


[83]:
```
sns.scatterplot(
    y = residuals['gZ'],
    x = residuals['gZ'].shift(-1),
    color = 'darkred'
)
sns.despine()
plt.xlabel("$g_Z(-1)$")
plt.show()
```

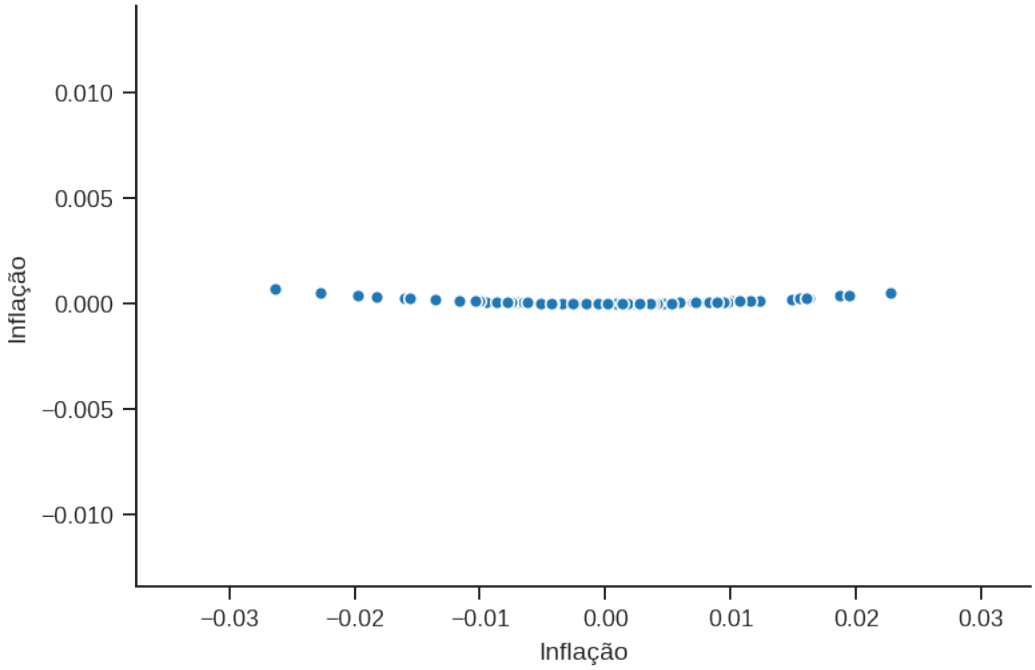


7.3.4 Inflação

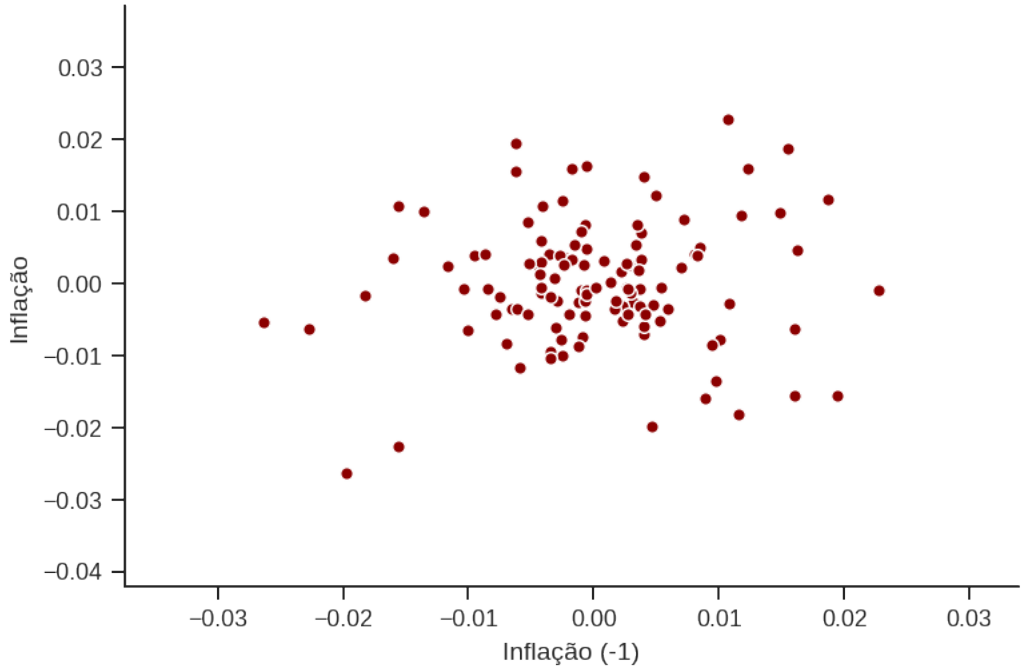
[84]:
```
sns.distplot(residuals["Inflação"])
sns.despine()
plt.show()
```



[85]:
```
sns.scatterplot(x = residuals['Inflação'], y = residuals['Inflação']**2)
sns.despine()
plt.show()
```

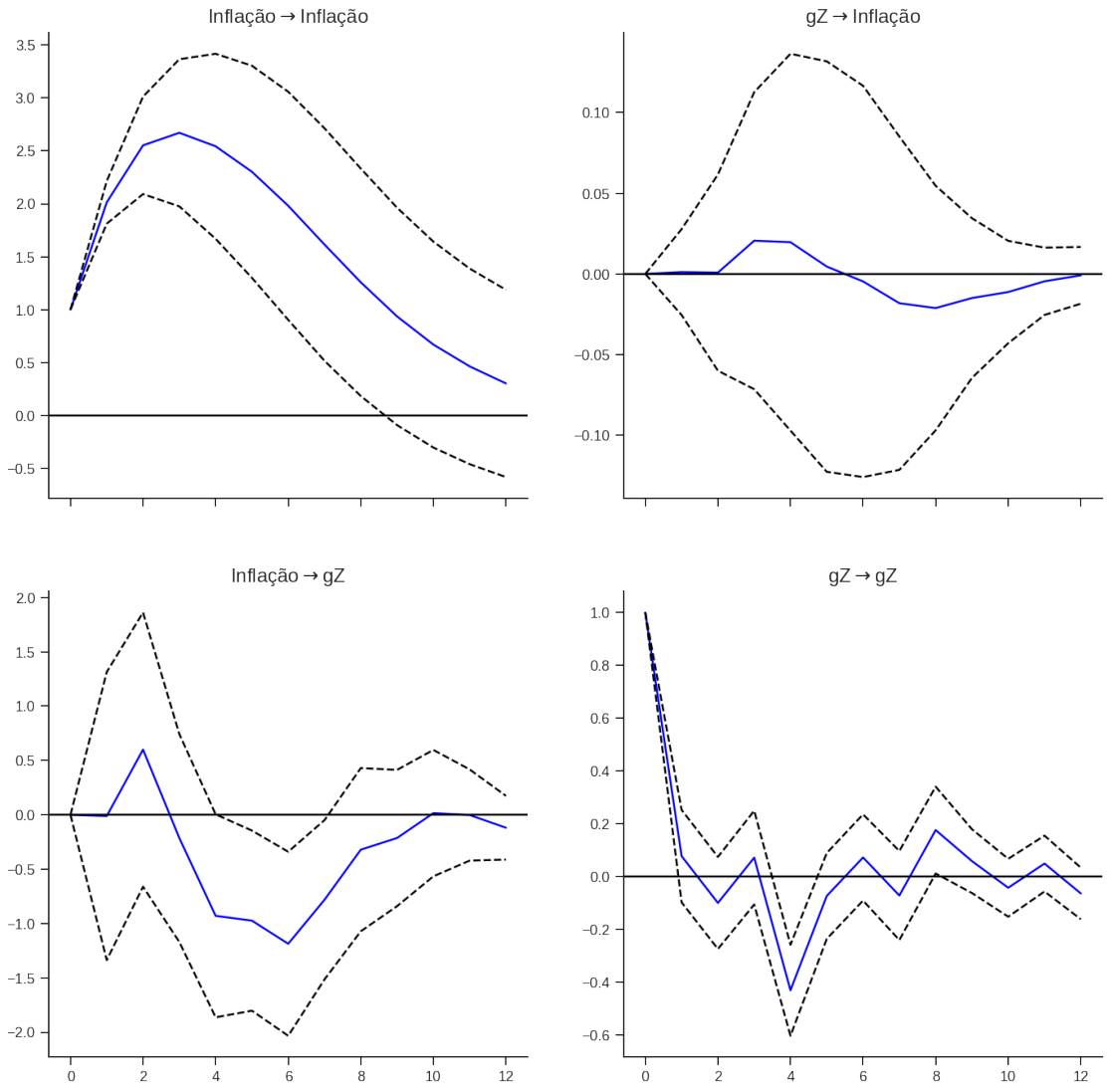


[86]:
```
sns.scatterplot(y = residuals['Inflação'], x = residuals['Inflação'].shift(-1),␣
↪color = 'darkred')
sns.despine()
plt.xlabel("Inflação (-1)")
plt.show()
```

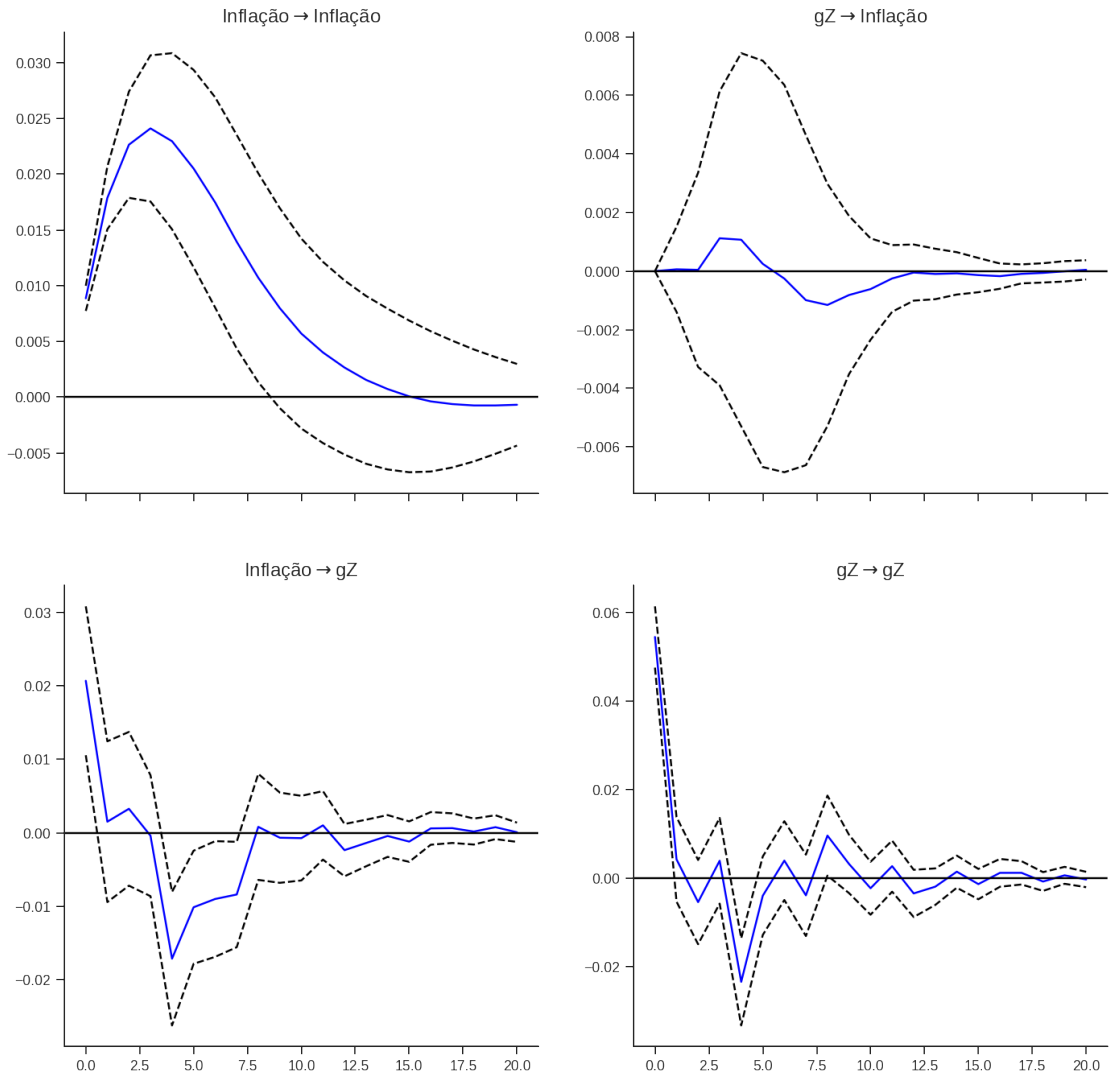


7.4 Função resposta ao impulso

[87]:
```
p = results.irf(12).plot(orth=False)
p.suptitle("")
sns.despine()
plt.show()
p.savefig("./Escrita/Figs/Impulso.png", dpi = 300)
```

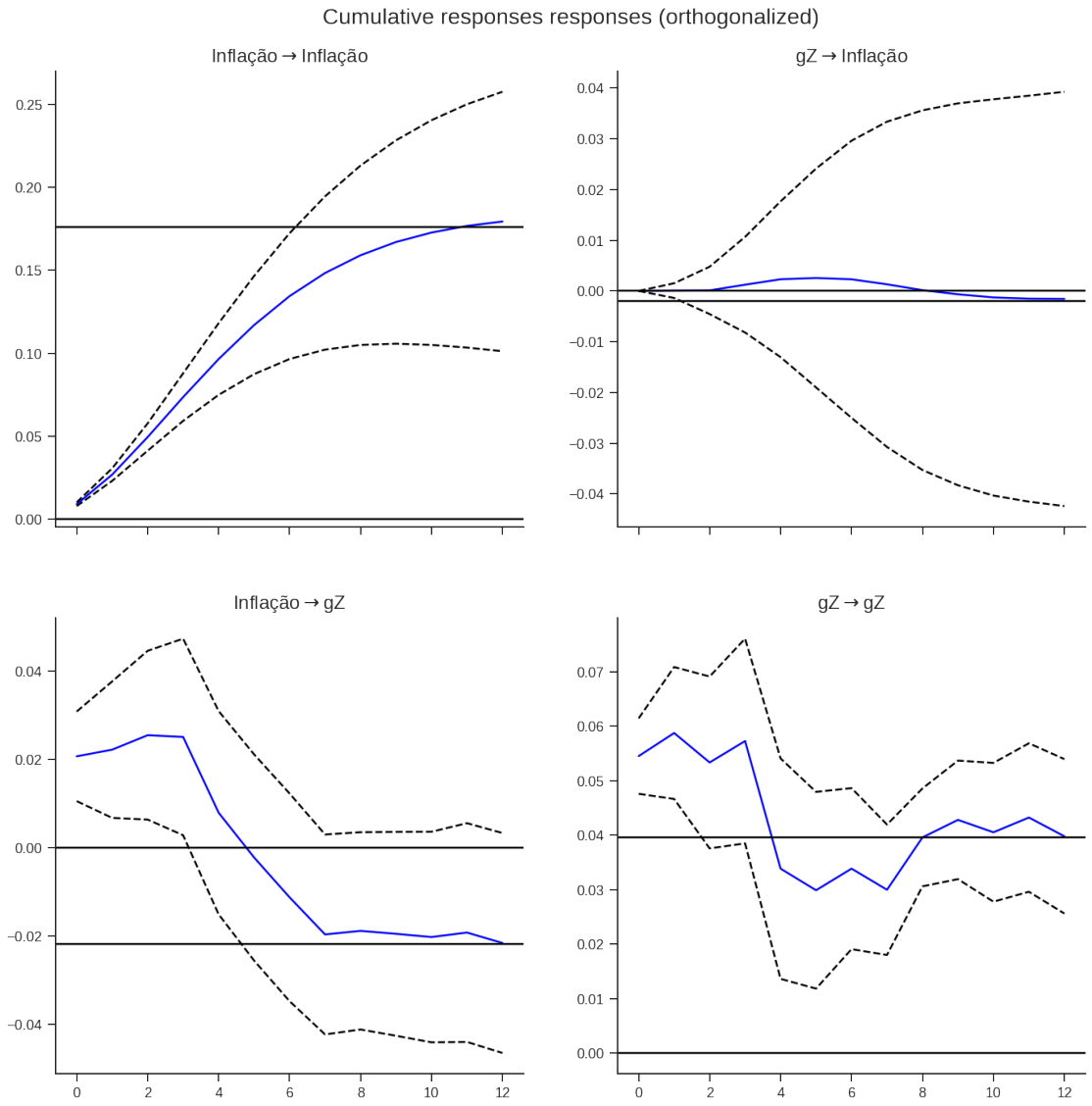


[88]:
```
p = results.irf(20).plot(orth=True)
p.suptitle("")
sns.despine()
plt.show()
p.savefig("./Escrita/Figs/Impulso_Orth.png", dpi = 300)
```



7.5 Efeito cumulativo

[89]:
```
p = results.irf(12).plot_cum_effects(orth=True)
sns.despine()
plt.show()
p.savefig("./Escrita/Figs/Impulso_Cum.png", dpi = 300)
```



7.6 Teste Causalidade de Granger

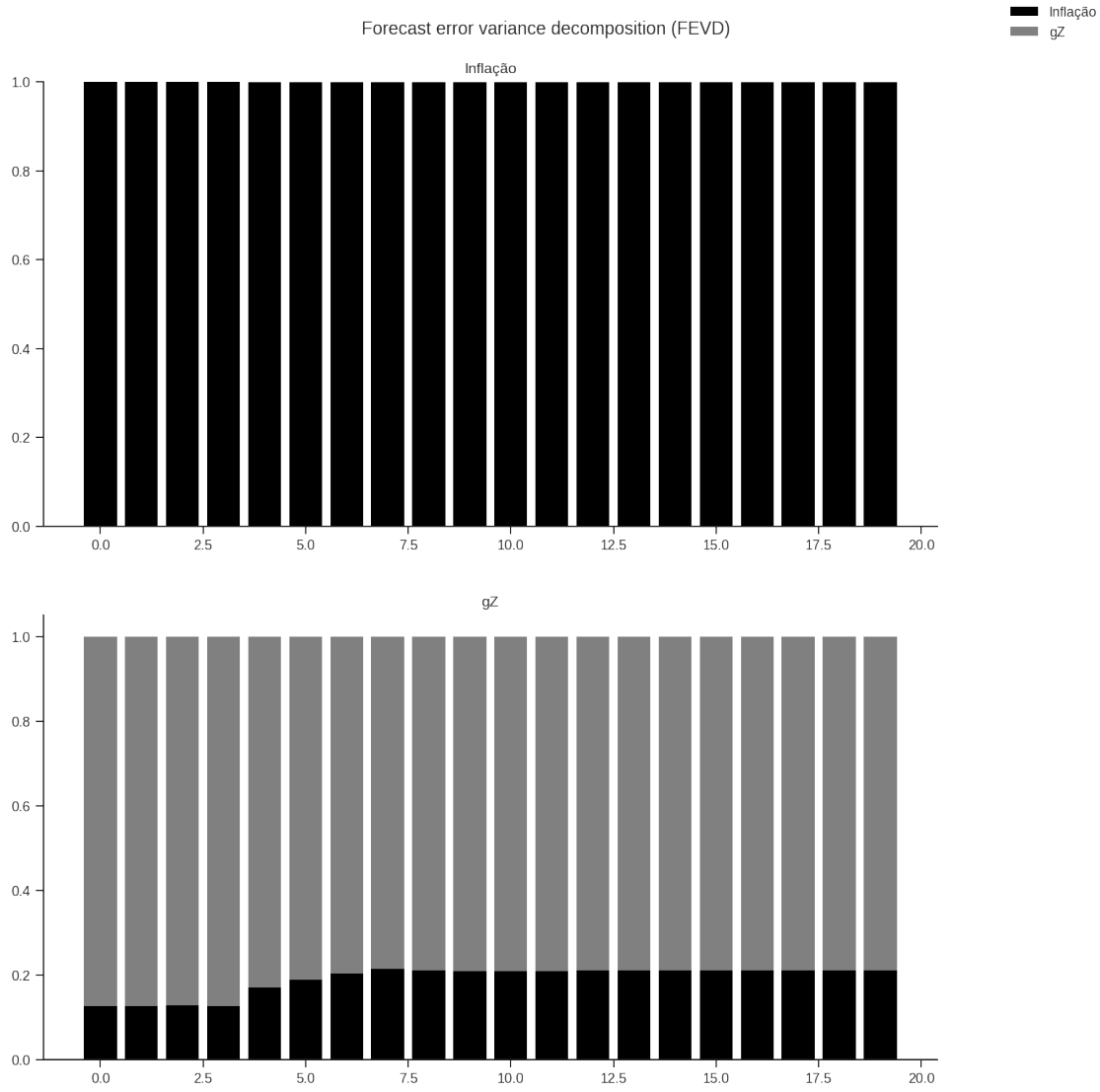
[90]:
```
results.test_causality(caused = 'gZ', causing = 'Inflação', signif = 0.05).
↪summary()
```

[90]:
```
<class 'statsmodels.iolib.table.SimpleTable'>
```

Conclusão: A um nível de significância de 5%, Inflação granger causa g_Z

7.7 Decomposição da variância

[91]:
```
p = results.fevd(20).plot()
sns.despine()
plt.show()
p.savefig("./Escrita/Figs/DecompVar.png", dpi = 300)
```



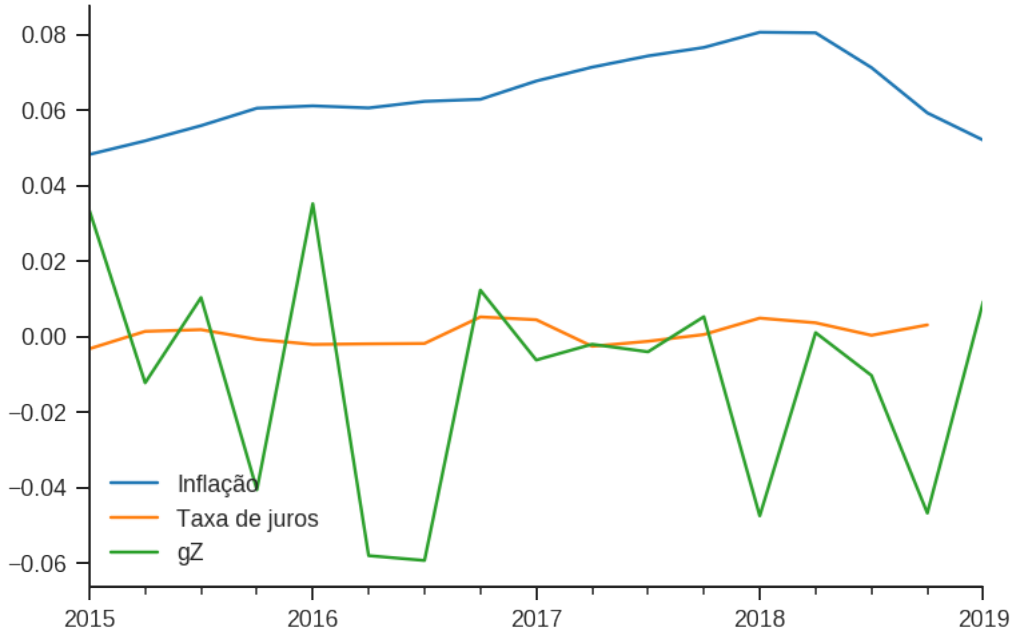
8 Previsão

8.1 Coletando dados exógenos

[92]:
```
exog = web.DataReader(
    [
        "MORTGAGE30US"
    ],
    'fred',
    start = datetime.datetime(2018,12,1),
    end = datetime.datetime(2019,4,1)
)
exog = exog.resample('Q').mean()/100
exog["Crise"] = [0 for i in range(len(exog))]
exog["Yeojohnson"], *_ = yeojohnson(exog["MORTGAGE30US"])
exog["Yeojohnson"] = exog["Yeojohnson"].diff()
exog = exog[["Yeojohnson", "Crise"]].dropna()
exog
```

[92]:
```
            Yeojohnson  Crise
DATE
2019-03-31   -0.002523      0
```

[93]:
```
previsao = df.drop("Crise", axis="columns")
prev = pd.DataFrame(
    results.forecast(y = results.y, steps = len(exog["Yeojohnson"]),␣
 ↪exog_future = exog),
    columns=["Inflação", "gZ"],
    index = exog.index
)
previsao.append(prev)["2015":].plot()
sns.despine()
plt.show()
```



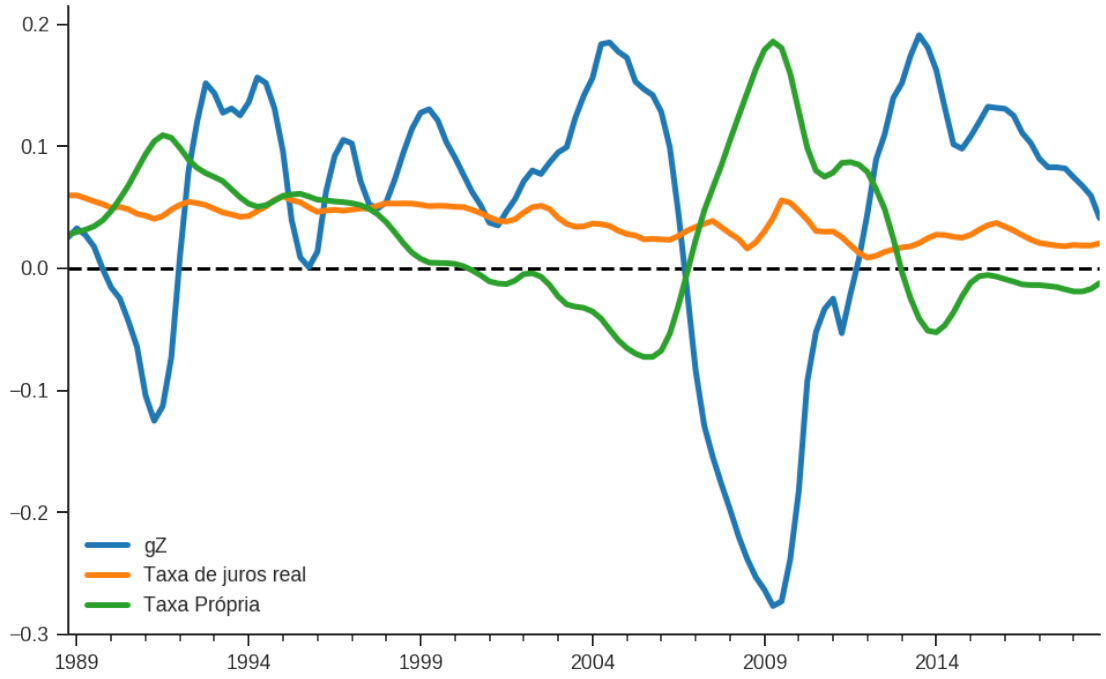
8.1.1 MSE

[94]:
```
lags = [4, 5]

for i in lags:
    model = VAR(
        endog=df[["Inflação", "gZ"]],
        exog = df[["Taxa de juros", "Crise"]],
    )
    results = model.fit(i)
    ajustado = pd.DataFrame(results.fittedvalues, columns = ["Inflação", "gZ"])
    ajustado.index = df.index[results.k_ar:]
    print("Calculando para o lag {}".format(i))
    print("MSE para Inflação:", np.mean((df['Inflação'].iloc[results.k_ar:] -␣
 ↪ajustado["Inflação"])**2).round(5))
    print("MSE para gZ:", np.mean((df['gZ'].iloc[results.k_ar:] -␣
 ↪ajustado["gZ"])**2).round(5), "\n")
```

```
Calculando para o lag 4
MSE para Inflação: 7e-05
MSE para gZ: 0.00308

Calculando para o lag 5
MSE para Inflação: 6e-05
MSE para gZ: 0.00309
```

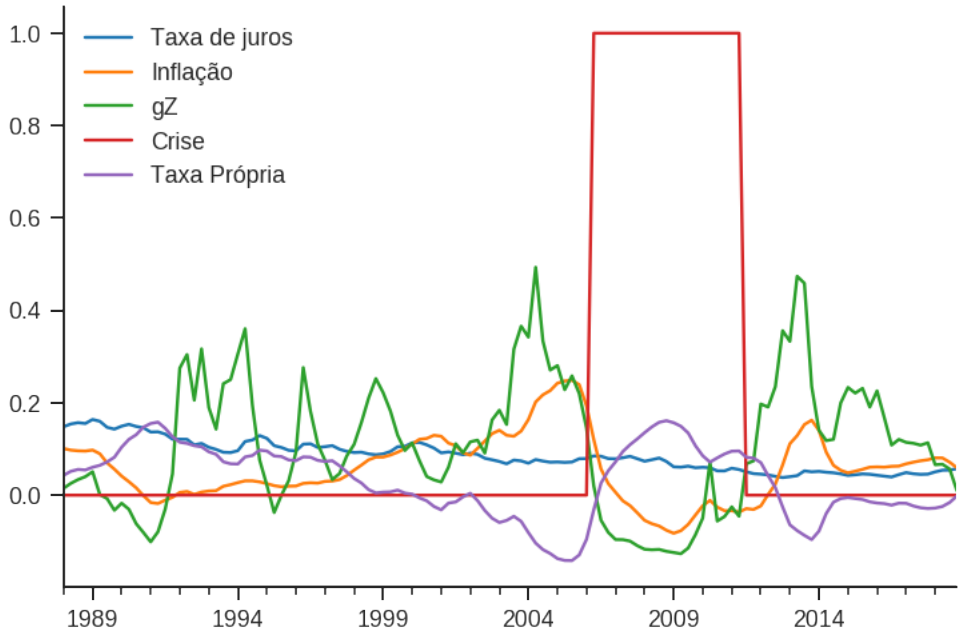
9 VECM

[95]:
```
df = web.DataReader(
    [
        "PRFI",
        "CSUSHPISA",
        "MORTGAGE30US",
        "CPIAUCSL"
    ],
    'fred',
    start,
    end
)
df.columns = [
    "Investimento residencial",
    "Preço dos imóveis",
    "Taxa de juros",
    "Inflação",
]
df.index.name = ""
df['Taxa de juros'] = df['Taxa de juros'].divide(100)
df = df.resample('Q').mean()
df['Preço dos imóveis'] = df['Preço dos imóveis']*df['Preço dos␣
 ↪imóveis']["1999-12-31"]/df['Preço dos imóveis'][0]/100
df['Inflação'] = (df['Inflação']*df['Inflação']["1999-12-31"]/df['Inflação'][0]/
 ↪100).pct_change(4)
df["Inflação imóveis"] = df["Preço dos imóveis"].pct_change(4)
df['gZ'] = df["Investimento residencial"].pct_change(4)
df["Taxa de juros real"] = ((1+df["Taxa de juros"])/(1+df["Inflação"])) -1
df["Taxa Própria"] = ((1+df["Taxa de juros"])/(1+df["Inflação imóveis"])) -1

fig, ax = plt.subplots(figsize = (8,5))

df[["gZ", "Taxa de juros real", "Taxa Própria"]].rolling(4).mean().dropna().
 ↪plot(lw=2.5, ax = ax)
ax.axhline(y=0, color = 'black', lw=1.5, ls='--', zorder=0)
sns.despine()
plt.show()
fig.savefig("./Escrita/Figs/TxPropria_Investo.png", dpi = 300, bbox_inches =␣
 ↪'tight',
    pad_inches = 0.2, transparent = True,)
```



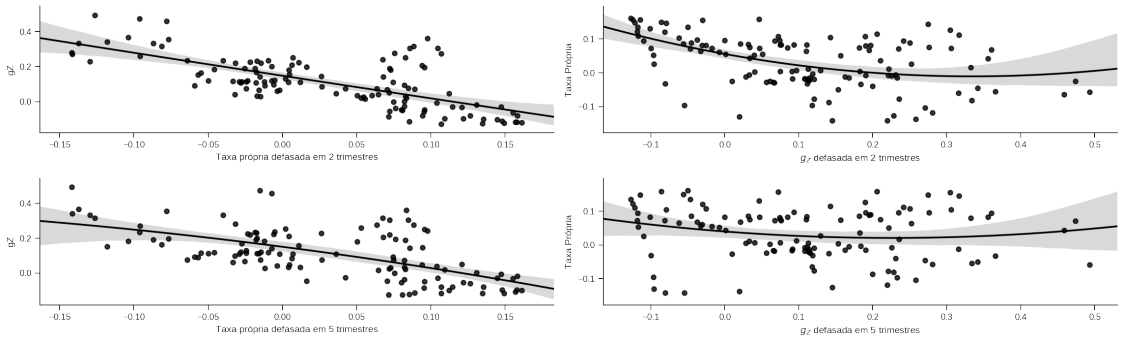
[96]:
```
df = web.DataReader(
    [
        "PRFI",
        "CSUSHPISA",
        "MORTGAGE30US",
        "PCDG",
        "FLTOTALSL"
    ],
    'fred',
    start,
    end
)
df.columns = [
    "Investimento residencial",
    "Preço dos imóveis",
    "Taxa de juros",
    "Duráveis",
    "Crédito"
]
df.index.name = ""
df['Taxa de juros'] = df['Taxa de juros'].divide(100)
df = df.resample('Q').mean()
df['Preço dos imóveis'] = df['Preço dos imóveis']*df['Preço dos␣
 ↪imóveis']["1999-12-31"]/df['Preço dos imóveis'][0]/100
df["gDuráveis"], *_ = yeojohnson(df["Duráveis"].pct_change(4))
```

```
df["gCrédito"], *_ = yeojohnson(df["Crédito"].pct_change(4))
df["Inflação"], *_ = yeojohnson(df["Preço dos imóveis"].pct_change(4))
df['gZ'], *_ = yeojohnson(df["Investimento residencial"].pct_change(4))
df["Taxa de juros"], *_ = yeojohnson(df["Taxa de juros"])
df["Taxa Própria"] = ((1+df["Taxa de juros"])/(1+df["Inflação"])) -1

df["Crise"] = [0 for i in range(len(df["gZ"]))]
for i in range(len(df["Crise"])):
    if df.index[i] > datetime.datetime(2006,4,1) and df.index[i] < datetime.
 ↪datetime(2011,7,1):
        df["Crise"][i] = 1

df.to_csv("Dados_yeojohnson.csv", )

df = df[["Taxa de juros", "Inflação", "gZ", "Crise", "Taxa Própria"]]

df = df.dropna()
df.plot()
sns.despine()
plt.show()
```



[97]:
```
fig, ax = plt.subplots(2,2, figsize = (16,5))

trimestres = 2
```

```
sns.regplot(y = df["gZ"], x = df["Taxa Própria"].shift(-trimestres), color =␣
 ↪'black', ax = ax[0,0], order = 2)
ax[0,0].set_xlabel('Taxa própria defasada em {} trimestres'.format(trimestres))

sns.regplot(x = df["gZ"].shift(-trimestres), y = df["Taxa Própria"], color =␣
 ↪'black', ax = ax[0,1], order = 2)
ax[0,1].set_xlabel('$g_Z$ defasada em {} trimestres'.format(trimestres))


trimestres = 5

sns.regplot(y = df["gZ"], x = df["Taxa Própria"].shift(-trimestres), color =␣
 ↪'black', ax = ax[1,0], order = 2)
ax[1,0].set_xlabel('Taxa própria defasada em {} trimestres'.format(trimestres))

sns.regplot(x = df["gZ"].shift(-trimestres), y = df["Taxa Própria"], color =␣
 ↪'black', ax = ax[1,1], order = 2)
ax[1,1].set_xlabel('$g_Z$ defasada em {} trimestres'.format(trimestres))

sns.despine()
fig.tight_layout()
plt.show()
fig.savefig("./Escrita/Figs/Scatter_VECM.png", dpi = 300, bbox_inches = 'tight',
    pad_inches = 0.2, transparent = True,)
```



[98]:
```
from statsmodels.tsa.vector_ar.vecm import select_order

with open('./Escrita/lag_order.tex','w') as fh:
    fh.write(select_order(df[["Taxa Própria", "gZ"]], maxlags=15).summary().
 ↪as_latex_tabular(tile = "Selação ordem do VECM"))

select_order(df[["Taxa Própria", "gZ"]], maxlags=15).summary()
```

[98]:
```
<class 'statsmodels.iolib.table.SimpleTable'>
```

[99]:
```
model = VECM(
    endog = df[["Taxa Própria", "gZ"]],
    #k_ar_diff=1
    #k_ar_diff=4
    k_ar_diff=5
    #k_ar_diff=9
)
results = model.fit()

with open('./Escrita/ajuste.tex','w') as fh:
    fh.write(results.summary().as_latex())

results.summary()
```

[99]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
Det. terms outside the coint. relation & lagged endog. parameters for equation
Taxa Própria
================================================================================
===
                       coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
---
L1.Taxa Própria     0.9309      0.089     10.483      0.000       0.757
1.105
L1.gZ              -0.0542      0.014     -3.756      0.000      -0.083
-0.026
L2.Taxa Própria    -0.2790      0.122     -2.287      0.022      -0.518
-0.040
L2.gZ               0.0035      0.013      0.270      0.787      -0.022
0.029
L3.Taxa Própria     0.0036      0.127      0.028      0.977      -0.245
0.253
L3.gZ              -0.0128      0.012     -1.028      0.304      -0.037
0.012
L4.Taxa Própria    -0.0633      0.126     -0.503      0.615      -0.310
0.183
L4.gZ              -0.0032      0.012     -0.256      0.798      -0.028
0.021
L5.Taxa Própria     0.0153      0.093      0.166      0.868      -0.166
0.197
L5.gZ              -0.0472      0.014     -3.321      0.001      -0.075
-0.019
 Det. terms outside the coint. relation & lagged endog. parameters for equation
gZ
================================================================================
```

```
===
                       coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
---
L1.Taxa Própria    -0.8490      0.553     -1.536      0.125      -1.932
0.234
L1.gZ               0.1291      0.090      1.436      0.151      -0.047
0.305
L2.Taxa Própria    -1.6341      0.759     -2.152      0.031      -3.122
-0.146
L2.gZ              -0.0597      0.081     -0.737      0.461      -0.218
0.099
L3.Taxa Própria     1.5472      0.791      1.956      0.050      -0.003
3.097
L3.gZ               0.1096      0.077      1.415      0.157      -0.042
0.261
L4.Taxa Própria    -0.5664      0.784     -0.723      0.470      -2.102
0.969
L4.gZ              -0.4589      0.078     -5.901      0.000      -0.611
-0.306
L5.Taxa Própria    -0.3342      0.576     -0.580      0.562      -1.464
0.795
L5.gZ               0.0290      0.089      0.327      0.744      -0.145
0.202
            Loading coefficients (alpha) for equation Taxa Própria
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ec1           -0.0098      0.007     -1.335      0.182      -0.024       0.005
               Loading coefficients (alpha) for equation gZ
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ec1            0.1385      0.046      3.032      0.002       0.049       0.228
         Cointegration relations for loading-coefficients-column 1
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
beta.1         1.0000          0          0      0.000       1.000       1.000
beta.2        -0.4869      0.231     -2.112      0.035      -0.939      -0.035
==============================================================================
"""
```

[100]:
```
results.test_whiteness(nlags=15).summary()
```

[100]:
```
<class 'statsmodels.iolib.table.SimpleTable'>
```

[101]:
```
results.test_whiteness(nlags=15, adjusted=True).summary()
```

[101]:
```
<class 'statsmodels.iolib.table.SimpleTable'>
```

[102]:
```
results.test_normality().summary()
```

[102]:
```
<class 'statsmodels.iolib.table.SimpleTable'>
```

[103]:
```
residuals = pd.DataFrame(results.resid, columns = ["Taxa Própria", "gZ"])
LjungBox_Pierce(residuals, k = 12, boxpierce=False)
```

```
H0: autocorrelations up to lag k equal zero
H1: autocorrelations up to lag k not zero
Box-Pierce:  False
Testing for  TAXA PRÓPRIA . Considering a significance level of 5.0 %
Reject H0 on lag  1 ?  False
Reject H0 on lag  2 ?  False
Reject H0 on lag  3 ?  False
Reject H0 on lag  4 ?  False
Reject H0 on lag  5 ?  False
Reject H0 on lag  6 ?  False
Reject H0 on lag  7 ?  False
Reject H0 on lag  8 ?  False
Reject H0 on lag  9 ?  False
Reject H0 on lag  10 ?  False
Reject H0 on lag  11 ?  False
Reject H0 on lag  12 ?  False


Testing for  GZ . Considering a significance level of 5.0 %
Reject H0 on lag  1 ?  False
Reject H0 on lag  2 ?  False
Reject H0 on lag  3 ?  False
Reject H0 on lag  4 ?  False
Reject H0 on lag  5 ?  False
Reject H0 on lag  6 ?  False
Reject H0 on lag  7 ?  False
Reject H0 on lag  8 ?  False
Reject H0 on lag  9 ?  False
Reject H0 on lag  10 ?  False
Reject H0 on lag  11 ?  False
Reject H0 on lag  12 ?  False
```

[104]:
```
LjungBox_Pierce(residuals, k = 12, boxpierce=True)
```

```
H0: autocorrelations up to lag k equal zero
H1: autocorrelations up to lag k not zero
```

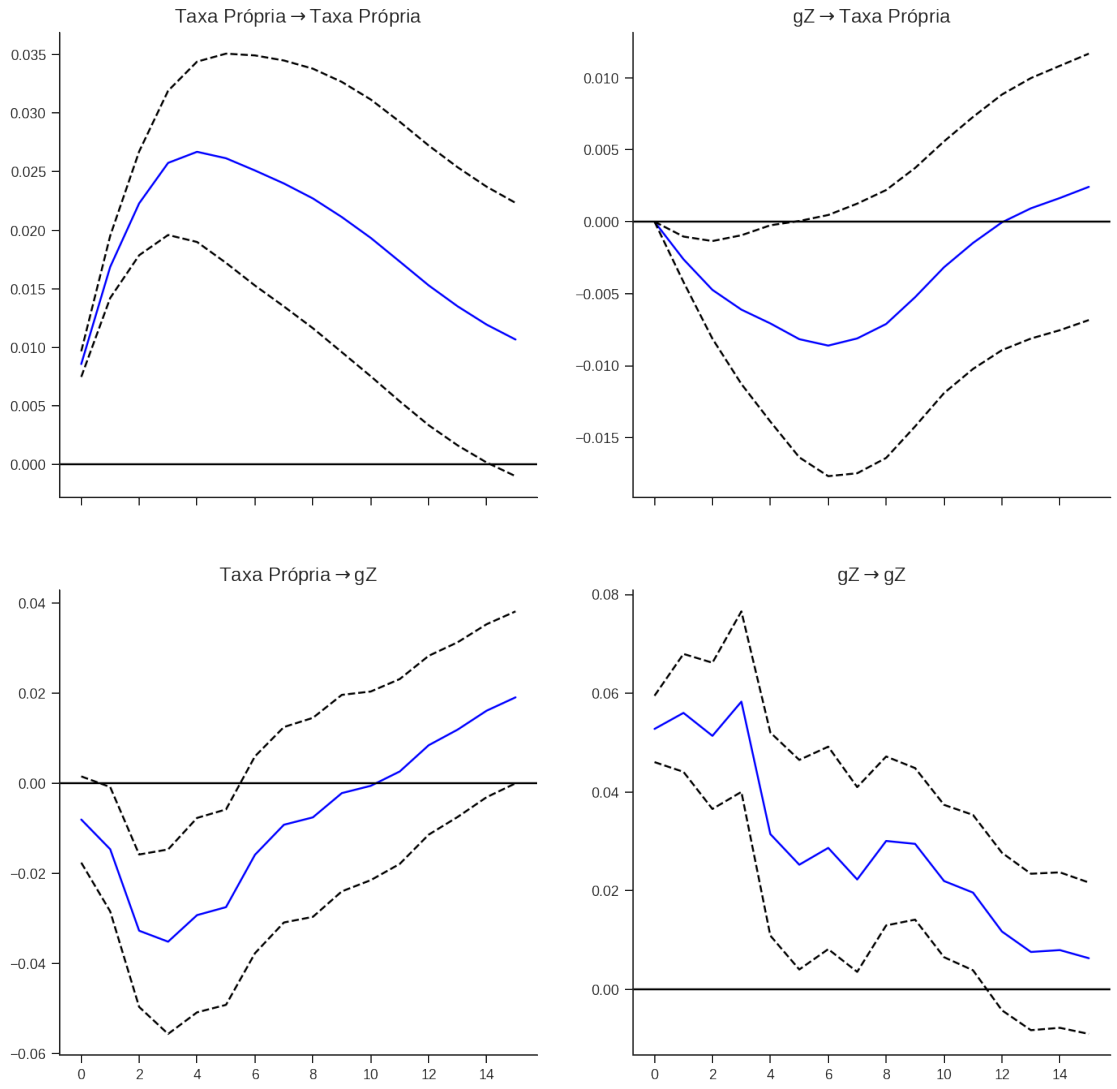
```
Box-Pierce:  True
Testing for  TAXA PRÓPRIA . Considering a significance level of 5.0 %
Reject H0 on lag  1 ?  False
Reject H0 on lag  2 ?  False
Reject H0 on lag  3 ?  False
Reject H0 on lag  4 ?  False
Reject H0 on lag  5 ?  False
Reject H0 on lag  6 ?  False
Reject H0 on lag  7 ?  False
Reject H0 on lag  8 ?  False
Reject H0 on lag  9 ?  False
Reject H0 on lag  10 ?  False
Reject H0 on lag  11 ?  False
Reject H0 on lag  12 ?  False


Testing for  GZ . Considering a significance level of 5.0 %
Reject H0 on lag  1 ?  False
Reject H0 on lag  2 ?  False
Reject H0 on lag  3 ?  False
Reject H0 on lag  4 ?  False
Reject H0 on lag  5 ?  False
Reject H0 on lag  6 ?  False
Reject H0 on lag  7 ?  False
Reject H0 on lag  8 ?  False
Reject H0 on lag  9 ?  False
Reject H0 on lag  10 ?  False
Reject H0 on lag  11 ?  False
Reject H0 on lag  12 ?  False
```

[105]:
```
ARCH_LM(residuals)
```

```
H0: Residuals are homoscedastic
H1: Residuals are heteroskedastic
Testing for  TAXA PRÓPRIA
LM p-value:  0.9577075687836497
Reject H0?  False
F p-value:  0.9581609813310835
Reject H0?  False


Testing for  GZ
LM p-value:  0.27563240736296946
Reject H0?  False
F p-value:  0.2795929028224716
```

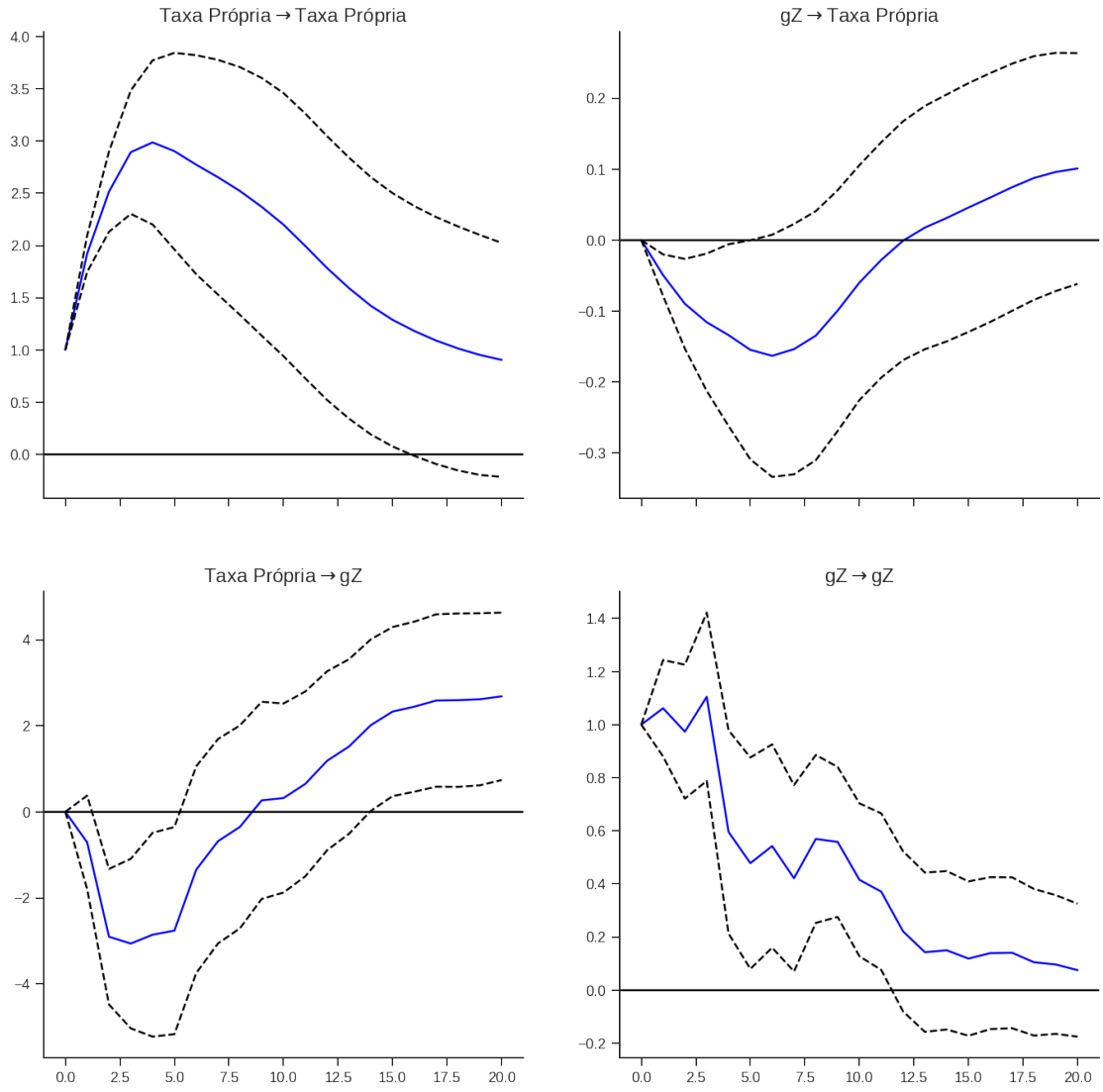
```
Reject H0?  False
```

[106]:
```
p = results.irf(15).plot(orth=True)
p.suptitle("")
sns.despine()
plt.show()
p.savefig("./Escrita/Figs/Impulso_VECMOrth.png", dpi = 300, bbox_inches =␣
 ↪'tight',
    pad_inches = 0.2, transparent = True,)
```

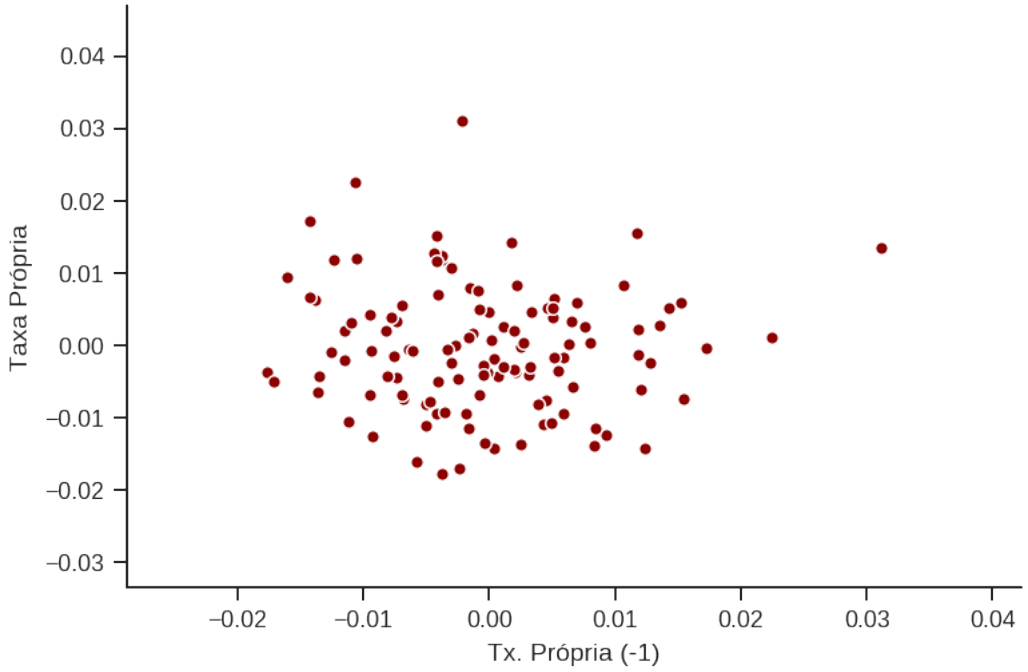


[107]:
```
p = results.irf(20).plot(orth=False)
p.suptitle("")
```

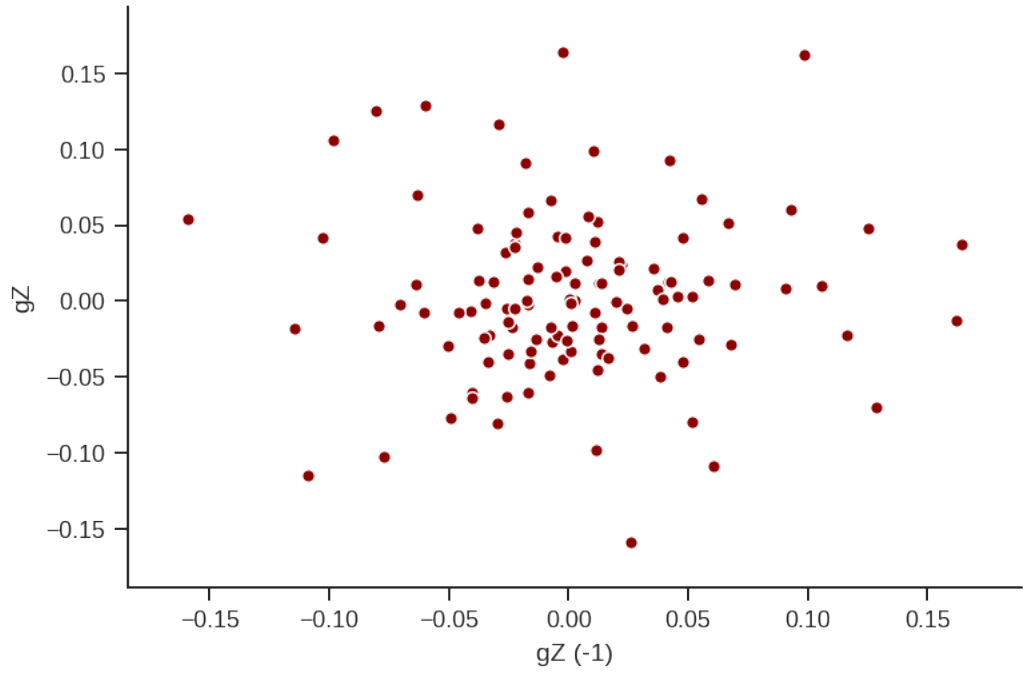
```
sns.despine()
plt.show()
p.savefig("./Escrita/Figs/Impulso_VECM.png", dpi = 300, bbox_inches = 'tight',
    pad_inches = 0.2, transparent = True,)
```



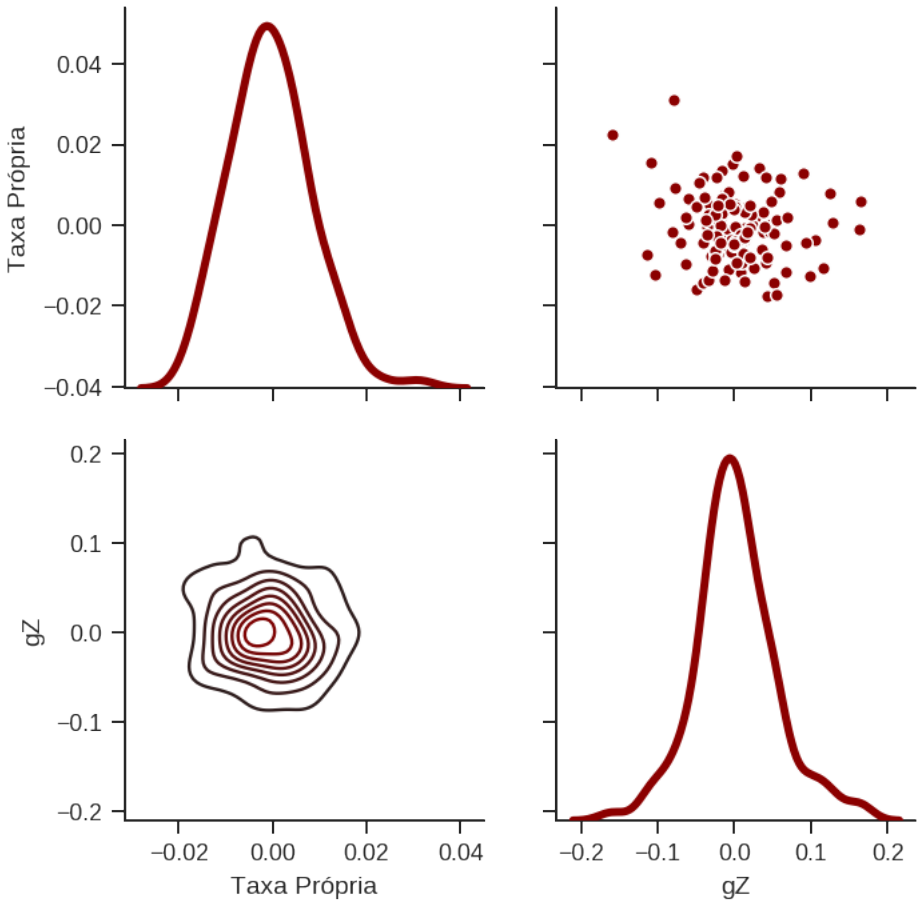
[108]:
```
sns.scatterplot(y = residuals['Taxa Própria'], x = residuals['Taxa Própria'].
 ↪shift(-1), color = 'darkred')
sns.despine()
plt.xlabel("Tx. Própria (-1)")
plt.show()
```



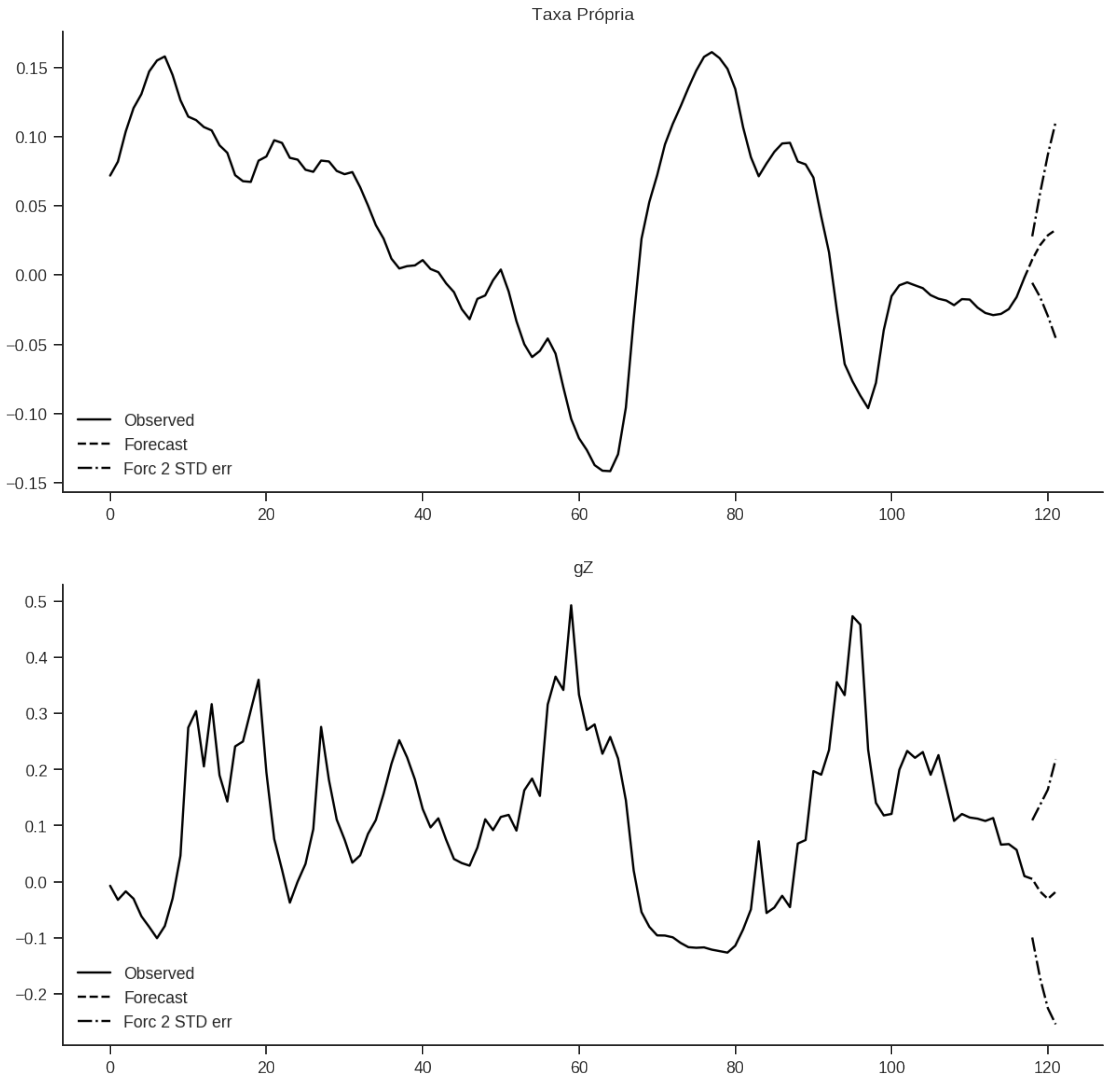
[109]:
```
sns.scatterplot(y = residuals['gZ'], x = residuals['gZ'].shift(-1), color =␣
↪'darkred')
sns.despine()
plt.xlabel("gZ (-1)")
plt.show()
```



[110]:
```
sns.set_context('paper')
g = sns.PairGrid(residuals, diag_sharey=False)
g.map_lower(sns.kdeplot, color = 'darkred')
g.map_upper(sns.scatterplot, color = 'darkred')
g.map_diag(sns.kdeplot, lw=3, color = 'darkred')
plt.show()
g.savefig("./Escrita/Figs/Residuos_4VECM.png", dpi = 300, bbox_inches = 'tight',
    pad_inches = 0.2, transparent = True,)
```



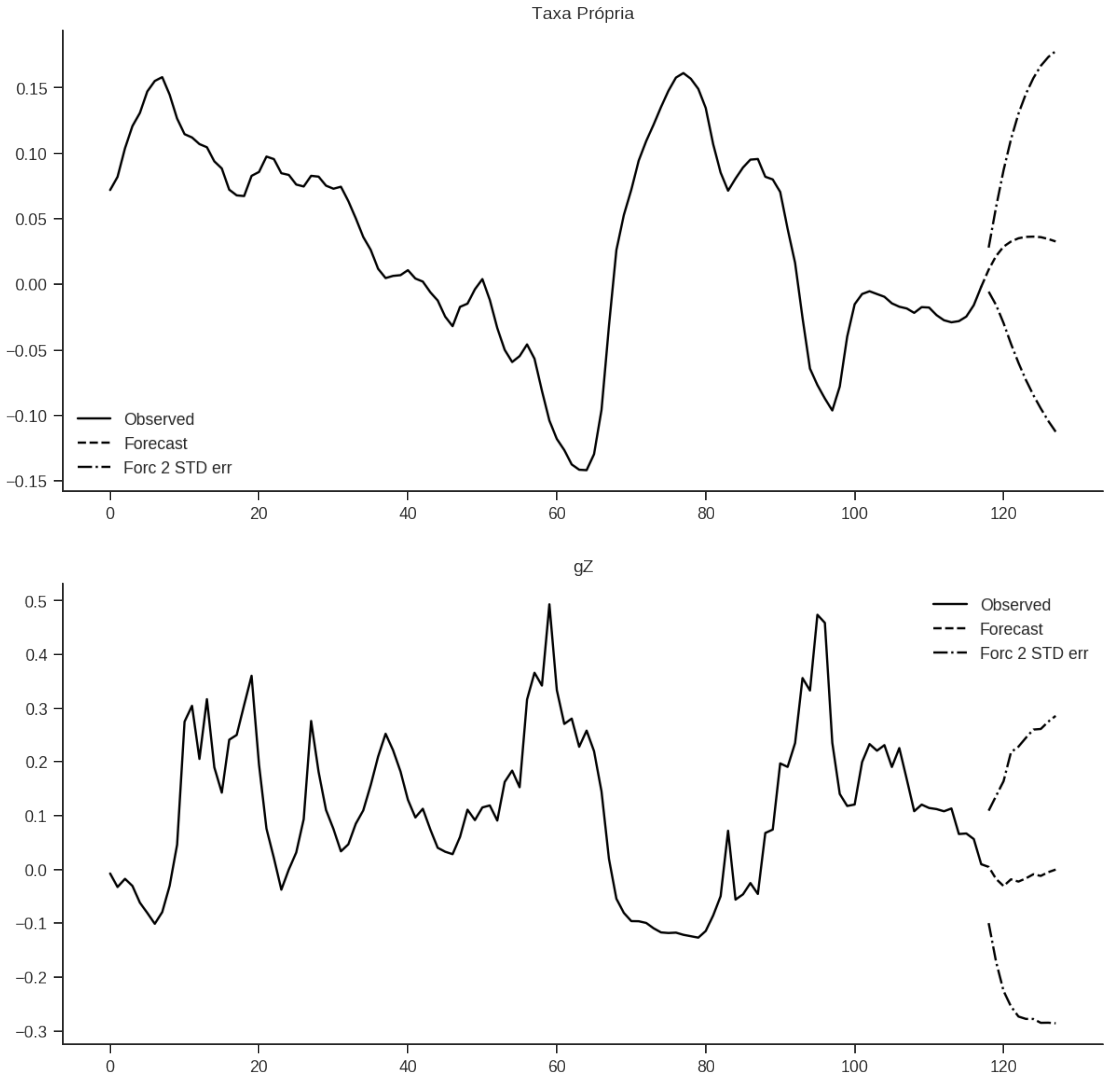
[111]:
```
sns.set_context('paper')

f = results.plot_forecast(steps=4)
sns.despine()
plt.show()
plt.savefig("./Escrita/Figs/Previsao_VECM.png", dpi = 300, bbox_inches =␣
 ↪'tight',
    pad_inches = 0.2, transparent = True,)
```




```
<Figure size 432x288 with 0 Axes>
```

[112]:
```
import statsmodels.tsa.vector_ar.plotting as plot

pred = 10
mid, lower, upper = results.predict(pred, alpha=0.05)
n_last_obs=None

y = results.y_all.T
y = y[results.k_ar:] if n_last_obs is None else y[-n_last_obs:]
#y.index = pd.date_range(start = df.index[0], end = df.index[-n_last_obs:
 ↪]+pred)
plot.plot_var_forc(y,
```

```
                    mid, lower, upper,
                    names=results.names,
                    plot_stderr=2,
                    legend_options={"loc": "best"})
sns.despine()
plt.show()
```



10 Coletando novos dados

[113]:
```
efetivo = web.DataReader(
    [
        "CSUSHPISA",
        "MORTGAGE30US",
```

```
        "PRFI"
    ],
    'fred',
    start = datetime.datetime(1987,1,1),
    end = datetime.datetime(2019,7,1)
)
efetivo.columns = [
    "Preço dos imóveis",
    "Taxa de juros",
    "Investimento Residencial"
]
efetivo['Taxa de juros'] = efetivo['Taxa de juros'].divide(100)
efetivo = efetivo.resample('Q').mean()
efetivo["Inflação"], *_ = yeojohnson(efetivo["Preço dos imóveis"].pct_change(4))
efetivo["gZ"], *_ = yeojohnson(efetivo["Investimento Residencial"].
 ↪pct_change(4))
efetivo["Taxa de juros"], *_ = yeojohnson(efetivo["Taxa de juros"])
efetivo["Taxa Própria"] = ((1+efetivo["Taxa de juros"])/
 ↪(1+efetivo["Inflação"])) -1
efetivo = efetivo.dropna()
efetivo = efetivo[["Taxa Própria", "gZ"]].tail(2)*100
efetivo
```

[113]:

DATE	Taxa Própria	gZ
2019-03-31	0.639439	0.475385
2019-06-30	0.737446	-0.600678

[114]:
```
previsao = pd.DataFrame(results.predict(1), columns = ["Taxa própria", "gZ"])
previsao.head()
```

[114]:

	Taxa própria	gZ
0	0.011221	0.004742

[115]:
```
print('erro de previsão (Taxa própria): |{}|%'.format(np.abs(efetivo["Taxa␣
 ↪Própria"].iloc[-1] - previsao["Taxa própria"].iloc[-1]).round(5)*100))
print('erro de previsão (gZ): |{}|%'.format(np.abs(efetivo["gZ"].iloc[-1] -␣
 ↪previsao["gZ"].iloc[-1]).round(5)*100))
```

```
erro de previsão (Taxa própria): |72.623|%
erro de previsão (gZ): |60.541999999999994|%
```

10.0.1 MSE

[116]:
```
lags = [2, 5, 6, 9, 11, 12]

for i in lags:
    model = VECM(
        endog=df[["Taxa Própria", "gZ"]],
```

```
        #exog = df[["Bolha", "Crise"]],
        coint_rank=1,
        k_ar_diff=i
    )
    results = model.fit()
    ajustado = pd.DataFrame(results.fittedvalues, columns = ["Taxa Própria",␣
↪"gZ"])
    ajustado.index = df.index[results.k_ar:]
    print("Calculando para o lag {}".format(i))
    print("MSE para taxa própria:", np.mean((df['Taxa Própria'].iloc[results.
↪k_ar:] - ajustado["Taxa Própria"])**2).round(5))
    print("MSE para gZ:", np.mean((df['gZ'].iloc[results.k_ar:] -␣
↪ajustado["gZ"])**2).round(5), "\n")
```

```
Calculando para o lag 2
MSE para taxa própria: 8e-05
MSE para gZ: 0.00386

Calculando para o lag 5
MSE para taxa própria: 7e-05
MSE para gZ: 0.00285

Calculando para o lag 6
MSE para taxa própria: 7e-05
MSE para gZ: 0.00282

Calculando para o lag 9
MSE para taxa própria: 7e-05
MSE para gZ: 0.00276

Calculando para o lag 11
MSE para taxa própria: 6e-05
MSE para gZ: 0.00273

Calculando para o lag 12
MSE para taxa própria: 6e-05
MSE para gZ: 0.00268
```

10.1 FEVD

[117]:
```
%%R -o fevd_gz
library(tsDyn)
library(readr)
df <- read.csv("./Dados_yeojohnson.csv", encoding="UTF-8")
df <- df[,c(10:12)]
names(df) <- c("gZ", "TaxaP", "Crise")
```

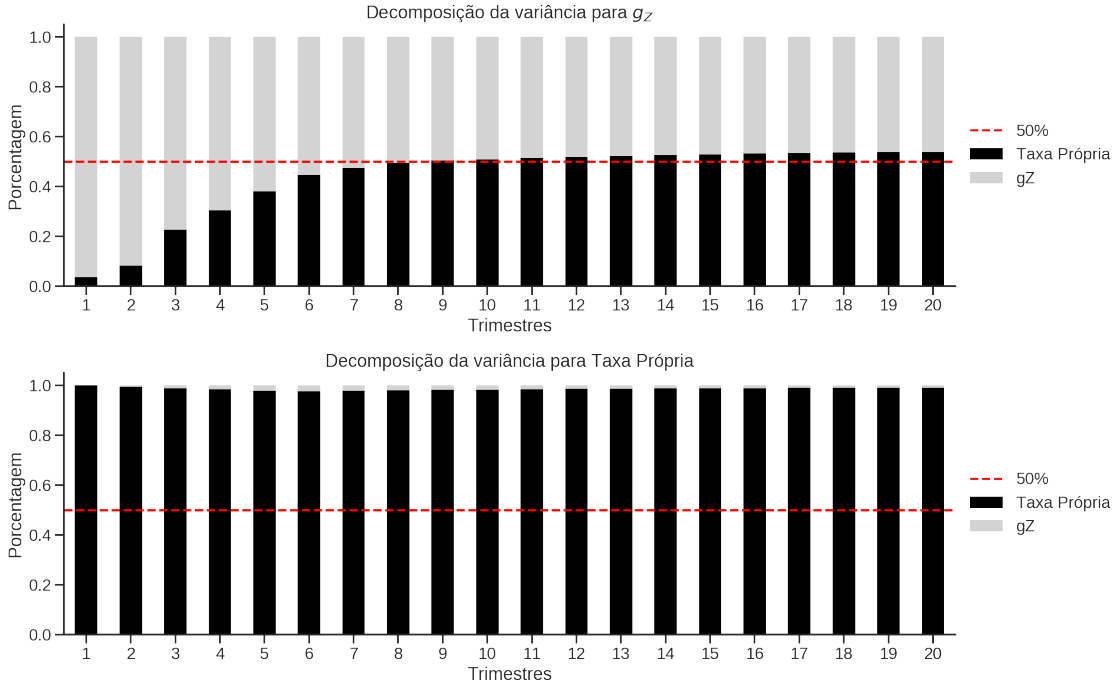
```
df <- df[,c(2,1,3)]
df <- na.omit(df[,c("TaxaP", "gZ")])
df <- ts(data = df, start = c(1987,01), frequency = 4)
model <- tsDyn::VECM(data = df, lag = 4, r = 1, estim = "ML")
fevd_gz = data.frame(tsDyn::fevd(model, 20)$gZ)
```

```
R[write to console]: Registered S3 method overwritten by 'xts':
  method     from
  as.zoo.xts zoo

R[write to console]: Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo

R[write to console]: Registered S3 methods overwritten by 'forecast':
  method             from
  fitted.fracdiff    fracdiff
  residuals.fracdiff fracdiff
```

[118]:
```
%%R -o fevd_tx
fevd_tx = data.frame(tsDyn::fevd(model, 20)$TaxaP)
```

[119]:
```
sns.set_context('talk')
fig, ax = plt.subplots(2,1, figsize = (16,10))

fevd_gz.plot(
    ax=ax[0],
    title = "Decomposição da variância para $g_Z$",
    color = ("black", "lightgray"),
    kind = 'bar', stacked = True
)
ax[0].set_xlabel('Trimestres')
ax[0].set_ylabel('Porcentagem')
ax[0].axhline(y=0.5, color = 'red', ls = '--')
ax[0].legend(loc='center left', bbox_to_anchor=(1, 0.5), labels = ("50%", "Taxa␣
 ↪Própria", "gZ"))
ax[0].set_xticklabels(ax[0].get_xticklabels(), rotation=0)


fevd_tx.plot(
    ax=ax[1],
    title = "Decomposição da variância para Taxa Própria",
    color = ("black", "lightgray"),
    kind = 'bar', stacked = True,
)
ax[1].axhline(y=0.5, color = 'red', ls = '--')
```

```
ax[1].legend(loc='center left', bbox_to_anchor=(1, 0.5), labels = ("50%", "Taxa␣
 ↪Própria", "gZ"))
ax[1].set_xlabel('Trimestres')
ax[1].set_ylabel('Porcentagem')
ax[1].set_xticklabels(ax[1].get_xticklabels(), rotation=0)

sns.despine()
plt.tight_layout()
plt.show()
fig.savefig("./Escrita/Figs/FEVD_VECM.png", dpi = 300, bbox_inches = 'tight',
    pad_inches = 0.2, transparent = True,)
```



11 TVECM

H0: VECM linear

H1: VECM com threshold

[120]:
```
%%R
tsDyn::TVECM.HStest(data = df, lag = 4)
```

```
## Test of linear versus threshold cointegration of Hansen and Seo (2002) ##

Test Statistic:  25.13664       (Maximized for threshold value: 0.3744634 )
P-Value:         0.56           ( Fixed regressor bootstrap )
```

[121]:
```
%%R
tsDyn::TVECM.HStest(data = df, lag = 5)
```

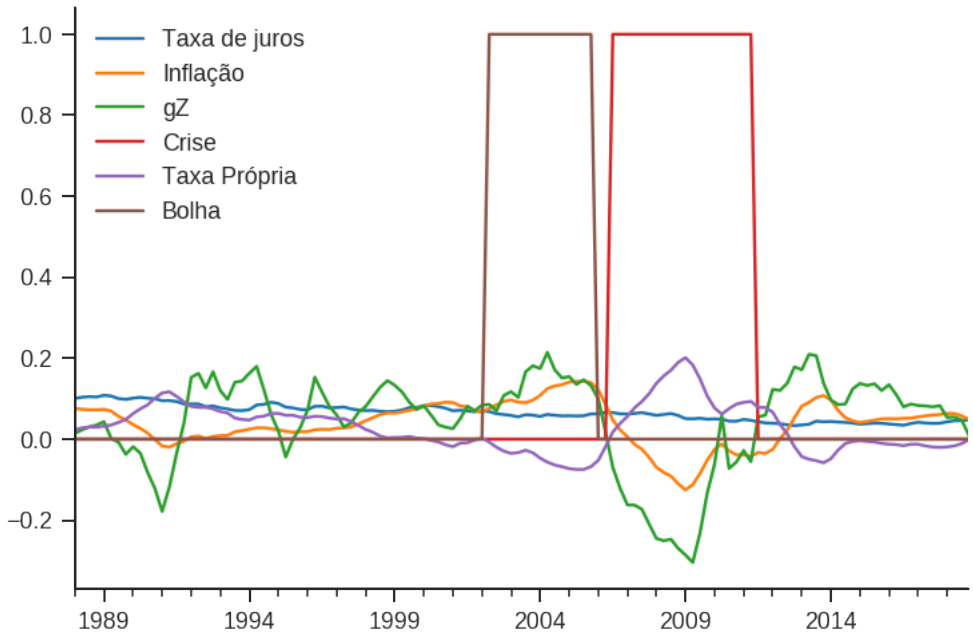
```
## Test of linear versus threshold cointegration of Hansen and Seo (2002) ##

Test Statistic:  31.75112       (Maximized for threshold value: 0.7381025 )
P-Value:         0.35           ( Fixed regressor bootstrap )
```

[122]:
```
%%R
tsDyn::TVECM.HStest(data = df, lag = 11)
```

```
## Test of linear versus threshold cointegration of Hansen and Seo (2002) ##

Test Statistic:  11.87757       (Maximized for threshold value: 8.977108 )
P-Value:         0.62           ( Fixed regressor bootstrap )
```

[123]:
```
df = web.DataReader(
    [
        "PRFI",
        "CSUSHPISA",
        "MORTGAGE30US",
        "PCDG",
        "FLTOTALSL"
    ],
    'fred',
    start,
    end
)
df.columns = [
    "Investimento residencial",
    "Preço dos imóveis",
    "Taxa de juros",
    "Duráveis",
    "Crédito"
]
df.index.name = "Data"
df['Taxa de juros'] = df['Taxa de juros'].divide(100)
df = df.resample('QS').mean()
df['Preço dos imóveis'] = df['Preço dos imóveis']*df['Preço dos␣
 ↪imóveis']["2000-01-01"]/df['Preço dos imóveis'][0]/100
df["gDuráveis"] = df["Duráveis"].pct_change(4)
df["gCrédito"] = df["Crédito"].pct_change(4)
df["Inflação"] = df["Preço dos imóveis"].pct_change(4)
df['gZ'] = df["Investimento residencial"].pct_change(4)
df["Taxa de juros"] = df["Taxa de juros"]
df["Taxa Própria"] = ((1+df["Taxa de juros"])/(1+df["Inflação"])) -1
```

```
df["Crise"] = [0 for i in range(len(df["gZ"]))]
for i in range(len(df["Crise"])):
    if df.index[i] > datetime.datetime(2006,4,1) and df.index[i] < datetime.
 ↪datetime(2011,7,1):
        df["Crise"][i] = 1

df["Bolha"] = [0 for i in range(len(df["gZ"]))]
for i in range(len(df["Crise"])):
    if df.index[i] > datetime.datetime(2002,1,1) and df.index[i] < datetime.
 ↪datetime(2006,1,1):
        df["Bolha"][i] = 1

df.index.name = ""
df = df[["Taxa de juros", "Inflação", "gZ", "Crise", "Taxa Própria", "Bolha"]]
df.to_csv("Dados.csv", )
df = df.dropna()

sns.set_context('paper')
df.plot()
sns.despine()
plt.show()
```



[124]:
```
%%R
df <- read.csv("./Dados.csv", encoding="UTF-8")
df <- df[,c(4,6)]
names(df) <- c("gZ", "TaxaP")
```

```
df <- na.omit(df[,c("gZ", "TaxaP")])
df <- ts(data = df, start = c(1987,01), frequency = 4)
df <- df[,c(2,1)]
tsDyn::rank.select(data = df, lag.max = 12, r.max = 1)
```

```
Best AIC: rank= 2  lag= 5
Best BIC: rank= 2  lag= 1
Best HQ : rank= 2  lag= 5
```

[125]:
```
%%R
tsDyn::TVECM.HStest(data = df, lag = 1, intercept = TRUE)
```

```
## Test of linear versus threshold cointegration of Hansen and Seo (2002) ##

Test Statistic:  13.92709       (Maximized for threshold value: -0.06673204 )
P-Value:         0.28           ( Fixed regressor bootstrap )
```

[126]:
```
%%R
tsDyn::TVECM.HStest(data = df, lag = 4, intercept = TRUE)
```

```
## Test of linear versus threshold cointegration of Hansen and Seo (2002) ##

Test Statistic:  23.10922       (Maximized for threshold value: 0.01623371 )
P-Value:         0.83           ( Fixed regressor bootstrap )
```

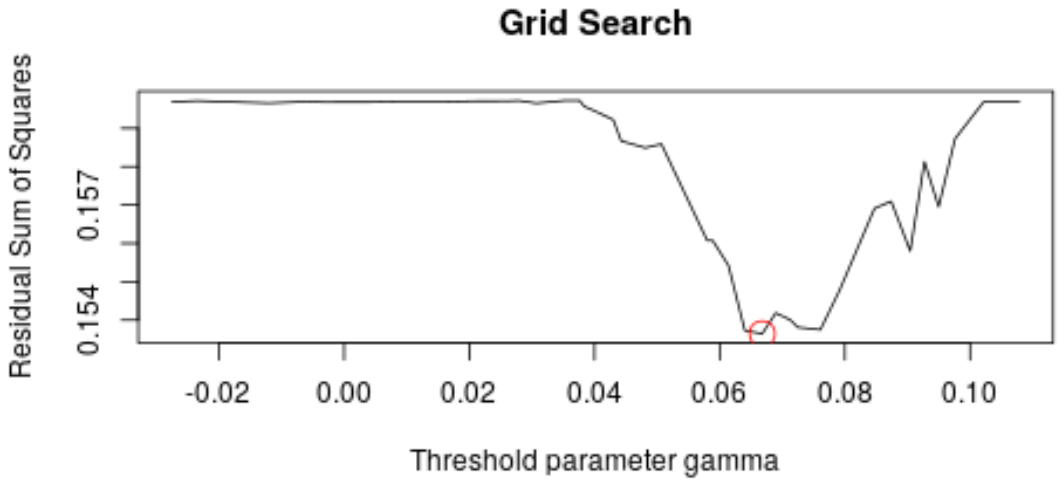
[127]:
```
%%R
tsDyn::TVECM.HStest(data = df, lag = 5, intercept = TRUE)
```

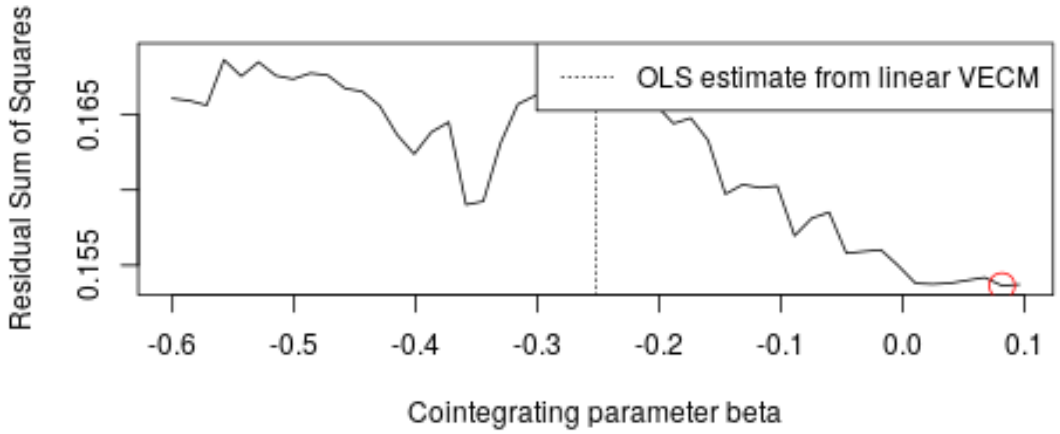
```
## Test of linear versus threshold cointegration of Hansen and Seo (2002) ##

Test Statistic:  35.32883       (Maximized for threshold value: -0.7562987 )
P-Value:         0.11           ( Fixed regressor bootstrap )
```

[128]:
```
%%R
print(summary(tsDyn::TVECM(df, 1, 1, common = 'only_ECT')))
```

```
94 (3.8%) points of the grid lead to regimes with percentage of observations <
trim and were not computed
##############
###Model TVECM
##############
Full sample size: 124   End sample size: 122
Number of variables: 2  Number of estimated parameters 10
AIC -2015.651   BIC -1984.807   SSR 0.1536455
```

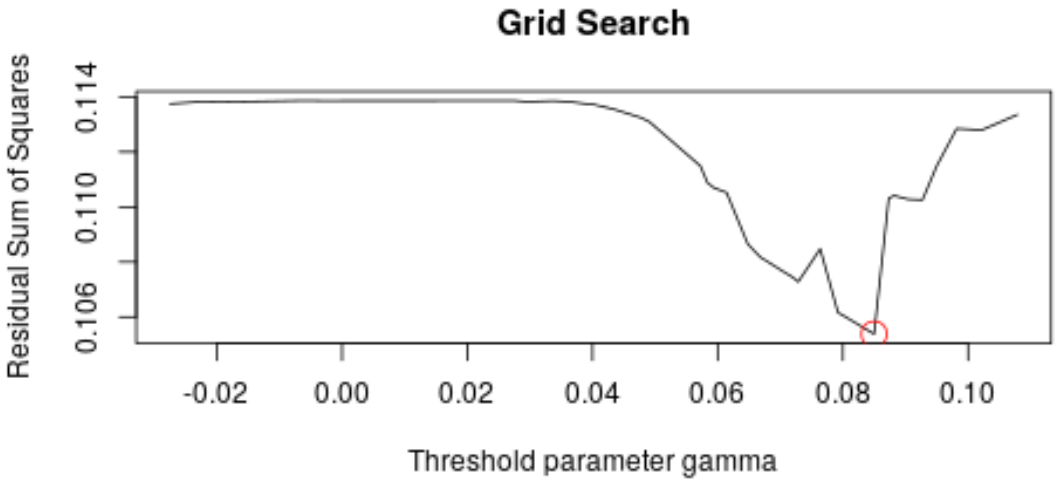
```
Cointegrating vector: (1, - 0.08146992 )
[[1]]
               ECT-              ECT+               Const
Equation TaxaP -0.0324(0.0680). -0.0371(0.0033)**  0.0009(0.2060)
Equation gZ    0.0097(0.9103)   0.2452(8.5e-05)*** -0.0072(0.0498)*
               TaxaP t -1          gZ t -1
Equation TaxaP 0.8384(1.3e-24)***  0.0067(0.7328)
Equation gZ    -1.4393(9.9e-06)*** 0.0244(0.7968)


Threshold
Values: 0.066802
Percentage of Observations in each regime 77% 23%
```

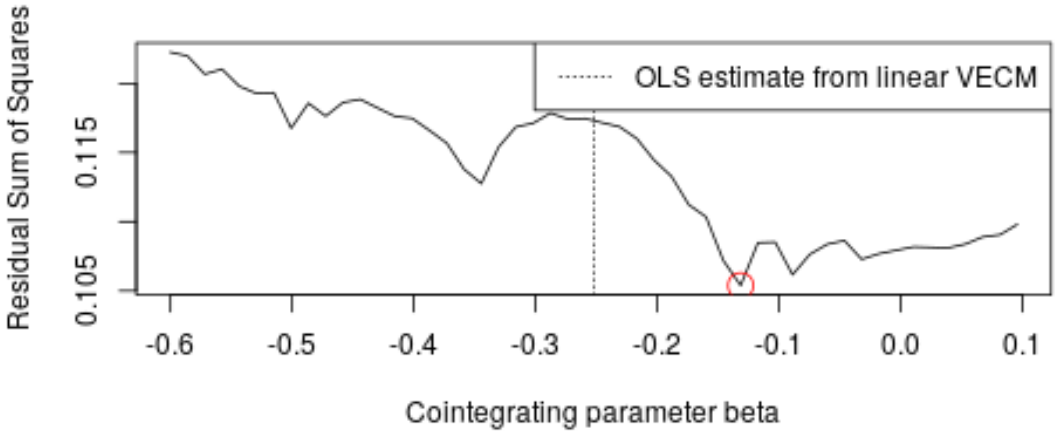




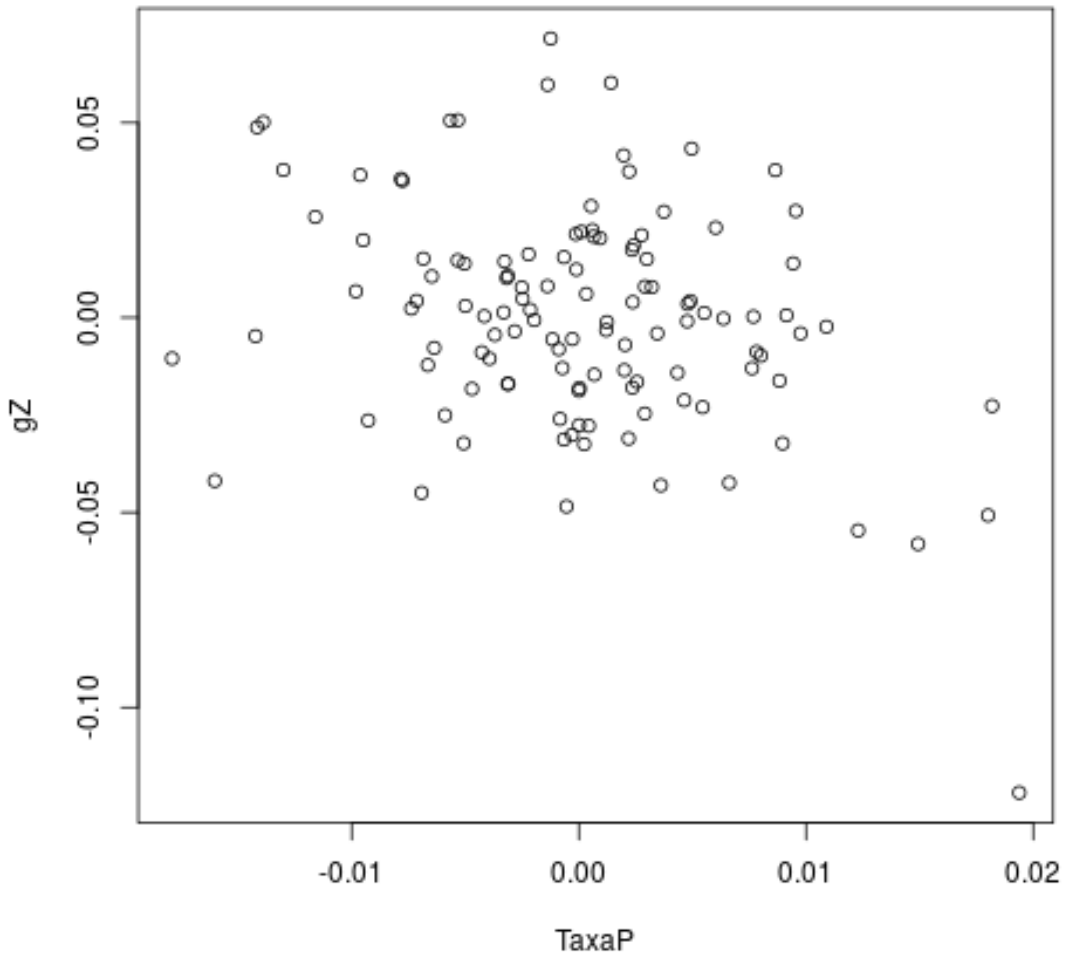
[129]:
```
%%R
model = tsDyn::TVECM(df, 4, 2, common = 'only_ECT')
print(summary(model))
```

```
77 (3.1%) points of the grid lead to regimes with percentage of observations <
trim and were not computed
Best threshold from first search 0.08498989
Best cointegrating value -0.1315781
Second best (conditionnal on the first one) 0.08498989 0.068956          SSR
0.09906713
Second step best thresholds 0.068956 0.08498989                          SSR
0.09906713
##############
###Model TVECM
##############
Full sample size: 124   End sample size: 119
Number of variables: 2  Number of estimated parameters 22
AIC -2004.594   BIC -1937.895   SSR 0.09906713


Cointegrating vector: (1, - -0.1315781 )
[[1]]
               ECT-              ECT+                Const
Equation TaxaP -0.0386(0.1082)  -0.0566(0.0038)**  0.0013(0.1084)
Equation gZ    0.2707(0.0069)** 0.3980(1.8e-06)*** -0.0098(0.0029)**
               TaxaP t -1          gZ t -1         TaxaP t -2
Equation TaxaP 1.0104(1.0e-17)*** 0.0169(0.4076)  -0.2379(0.0844).
Equation gZ    -1.0835(0.0086)**  0.0092(0.9130)  -1.0046(0.0774).
               gZ t -2           TaxaP t -3      gZ t -3
Equation TaxaP 0.0213(0.2741)    0.0673(0.6315)  -0.0172(0.3678)
Equation gZ    -0.0596(0.4583)   1.0462(0.0726). 0.0675(0.3913)
               TaxaP t -4        gZ t -4
Equation TaxaP -0.0584(0.5724)   0.0071(0.7175)
Equation gZ    -1.0646(0.0139)* -0.5502(5.2e-10)***


Threshold
Values: 0.068956 0.08498989
Percentage of Observations in each regime 69.7% 13.4% 16.8%
```





[130]:
```
%%R
plot(residuals(model))
```



12 Teste de causalidade: PIB ~ Investimento residencial

[131]:
```
df = web.DataReader(
    [
        "A011RL1Q225SBEA",
        "DPIC96",
    ],
    'fred',
    start = datetime.datetime(1947,1,1),
    end = datetime.datetime(2018,12,31)
)
df.columns = [
```

```
    "Investimento residencial",
    "Renda disponível"
]
df.index.name = ""
df["Investimento residencial"] = df["Investimento residencial"]/100
df["Renda disponível"] = df["Renda disponível"].pct_change(4)
df = df.dropna()
df.head()
```

[131]:

	Investimento residencial	Renda disponível
1948-01-01	-0.051	0.021102
1948-04-01	0.253	0.063093
1948-07-01	-0.121	0.052323
1948-10-01	-0.262	0.073347
1949-01-01	-0.264	0.032975

[132]:
```
print(ADF(df["Renda disponível"], trend='ct').summary())
print(ADF(df["Investimento residencial"], trend='ct').summary())
```

```
   Augmented Dickey-Fuller Results
=====================================
Test Statistic                 -3.617
P-value                         0.028
Lags                               12
-------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -3.99 (1%), -3.43 (5%), -3.14 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
   Augmented Dickey-Fuller Results
=====================================
Test Statistic                 -5.511
P-value                         0.000
Lags                               13
-------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -3.99 (1%), -3.43 (5%), -3.14 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
```

Ambas as taxas de crescimento são estacionárias.

[133]:
```
results, *_ = grangercausalitytests(df[["Renda disponível", "Investimento␣
 ↪residencial"]], maxlag=15)
print(results)
```

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=16.7880 , p=0.0001  , df_denom=280, df_num=1
ssr based chi2 test:   chi2=16.9679 , p=0.0000  , df=1
likelihood ratio test: chi2=16.4787 , p=0.0000  , df=1
parameter F test:         F=16.7880 , p=0.0001  , df_denom=280, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=12.2570 , p=0.0000  , df_denom=277, df_num=2
ssr based chi2 test:   chi2=24.9565 , p=0.0000  , df=2
likelihood ratio test: chi2=23.9133 , p=0.0000  , df=2
parameter F test:         F=12.2570 , p=0.0000  , df_denom=277, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=6.3698  , p=0.0003  , df_denom=274, df_num=3
ssr based chi2 test:   chi2=19.5976 , p=0.0002  , df=3
likelihood ratio test: chi2=18.9444 , p=0.0003  , df=3
parameter F test:         F=6.3698  , p=0.0003  , df_denom=274, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=4.2458  , p=0.0024  , df_denom=271, df_num=4
ssr based chi2 test:   chi2=17.5472 , p=0.0015  , df=4
likelihood ratio test: chi2=17.0193 , p=0.0019  , df=4
parameter F test:         F=4.2458  , p=0.0024  , df_denom=271, df_num=4

Granger Causality
number of lags (no zero) 5
ssr based F test:         F=4.5977  , p=0.0005  , df_denom=268, df_num=5
ssr based chi2 test:   chi2=23.9322 , p=0.0002  , df=5
likelihood ratio test: chi2=22.9609 , p=0.0003  , df=5
parameter F test:         F=4.5977  , p=0.0005  , df_denom=268, df_num=5

Granger Causality
number of lags (no zero) 6
ssr based F test:         F=3.9316  , p=0.0009  , df_denom=265, df_num=6
ssr based chi2 test:   chi2=24.7471 , p=0.0004  , df=6
likelihood ratio test: chi2=23.7069 , p=0.0006  , df=6
parameter F test:         F=3.9316  , p=0.0009  , df_denom=265, df_num=6

Granger Causality
number of lags (no zero) 7
ssr based F test:         F=3.3230  , p=0.0021  , df_denom=262, df_num=7
ssr based chi2 test:   chi2=24.5925 , p=0.0009  , df=7
likelihood ratio test: chi2=23.5614 , p=0.0014  , df=7
```

```
parameter F test:         F=3.3230  , p=0.0021  , df_denom=262, df_num=7

Granger Causality
number of lags (no zero) 8
ssr based F test:         F=2.7537  , p=0.0062  , df_denom=259, df_num=8
ssr based chi2 test:   chi2=23.4756 , p=0.0028  , df=8
likelihood ratio test: chi2=22.5304 , p=0.0040  , df=8
parameter F test:         F=2.7537  , p=0.0062  , df_denom=259, df_num=8

Granger Causality
number of lags (no zero) 9
ssr based F test:         F=3.6875  , p=0.0002  , df_denom=256, df_num=9
ssr based chi2 test:   chi2=35.6506 , p=0.0000  , df=9
likelihood ratio test: chi2=33.5219 , p=0.0001  , df=9
parameter F test:         F=3.6875  , p=0.0002  , df_denom=256, df_num=9

Granger Causality
number of lags (no zero) 10
ssr based F test:         F=3.5840  , p=0.0002  , df_denom=253, df_num=10
ssr based chi2 test:   chi2=38.8147 , p=0.0000  , df=10
likelihood ratio test: chi2=36.3003 , p=0.0001  , df=10
parameter F test:         F=3.5840  , p=0.0002  , df_denom=253, df_num=10

Granger Causality
number of lags (no zero) 11
ssr based F test:         F=3.2147  , p=0.0004  , df_denom=250, df_num=11
ssr based chi2 test:   chi2=38.6153 , p=0.0001  , df=11
likelihood ratio test: chi2=36.1172 , p=0.0002  , df=11
parameter F test:         F=3.2147  , p=0.0004  , df_denom=250, df_num=11

Granger Causality
number of lags (no zero) 12
ssr based F test:         F=2.9984  , p=0.0006  , df_denom=247, df_num=12
ssr based chi2 test:   chi2=39.6221 , p=0.0001  , df=12
likelihood ratio test: chi2=36.9891 , p=0.0002  , df=12
parameter F test:         F=2.9984  , p=0.0006  , df_denom=247, df_num=12

Granger Causality
number of lags (no zero) 13
ssr based F test:         F=2.5830  , p=0.0023  , df_denom=244, df_num=13
ssr based chi2 test:   chi2=37.2951 , p=0.0004  , df=13
likelihood ratio test: chi2=34.9423 , p=0.0009  , df=13
parameter F test:         F=2.5830  , p=0.0023  , df_denom=244, df_num=13

Granger Causality
number of lags (no zero) 14
ssr based F test:         F=2.8815  , p=0.0005  , df_denom=241, df_num=14
ssr based chi2 test:   chi2=45.1950 , p=0.0000  , df=14
```

```
likelihood ratio test: chi2=41.7878 , p=0.0001  , df=14
parameter F test:         F=2.8815  , p=0.0005  , df_denom=241, df_num=14

Granger Causality
number of lags (no zero) 15
ssr based F test:         F=2.7400  , p=0.0007  , df_denom=238, df_num=15
ssr based chi2 test:   chi2=46.4541 , p=0.0000  , df=15
likelihood ratio test: chi2=42.8522 , p=0.0002  , df=15
parameter F test:         F=2.7400  , p=0.0007  , df_denom=238, df_num=15
1
```

[134]:
```
results, *_ = grangercausalitytests(df[["Investimento residencial", "Renda␣
 ↪disponível"]], maxlag=15)
print(results)
```

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=6.2801  , p=0.0128  , df_denom=280, df_num=1
ssr based chi2 test:   chi2=6.3473  , p=0.0118  , df=1
likelihood ratio test: chi2=6.2772  , p=0.0122  , df=1
parameter F test:         F=6.2801  , p=0.0128  , df_denom=280, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=6.6123  , p=0.0016  , df_denom=277, df_num=2
ssr based chi2 test:   chi2=13.4632 , p=0.0012  , df=2
likelihood ratio test: chi2=13.1517 , p=0.0014  , df=2
parameter F test:         F=6.6123  , p=0.0016  , df_denom=277, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=5.7645  , p=0.0008  , df_denom=274, df_num=3
ssr based chi2 test:   chi2=17.7352 , p=0.0005  , df=3
likelihood ratio test: chi2=17.1980 , p=0.0006  , df=3
parameter F test:         F=5.7645  , p=0.0008  , df_denom=274, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=3.1740  , p=0.0143  , df_denom=271, df_num=4
ssr based chi2 test:   chi2=13.1174 , p=0.0107  , df=4
likelihood ratio test: chi2=12.8195 , p=0.0122  , df=4
parameter F test:         F=3.1740  , p=0.0143  , df_denom=271, df_num=4

Granger Causality
number of lags (no zero) 5
ssr based F test:         F=2.8072  , p=0.0172  , df_denom=268, df_num=5
```

```
ssr based chi2 test:   chi2=14.6120 , p=0.0122  , df=5
likelihood ratio test: chi2=14.2422 , p=0.0141  , df=5
parameter F test:         F=2.8072  , p=0.0172  , df_denom=268, df_num=5

Granger Causality
number of lags (no zero) 6
ssr based F test:         F=2.7164  , p=0.0141  , df_denom=265, df_num=6
ssr based chi2 test:   chi2=17.0980 , p=0.0089  , df=6
likelihood ratio test: chi2=16.5928 , p=0.0109  , df=6
parameter F test:         F=2.7164  , p=0.0141  , df_denom=265, df_num=6

Granger Causality
number of lags (no zero) 7
ssr based F test:         F=2.5257  , p=0.0157  , df_denom=262, df_num=7
ssr based chi2 test:   chi2=18.6924 , p=0.0092  , df=7
likelihood ratio test: chi2=18.0887 , p=0.0116  , df=7
parameter F test:         F=2.5257  , p=0.0157  , df_denom=262, df_num=7

Granger Causality
number of lags (no zero) 8
ssr based F test:         F=2.3490  , p=0.0188  , df_denom=259, df_num=8
ssr based chi2 test:   chi2=20.0253 , p=0.0102  , df=8
likelihood ratio test: chi2=19.3322 , p=0.0132  , df=8
parameter F test:         F=2.3490  , p=0.0188  , df_denom=259, df_num=8

Granger Causality
number of lags (no zero) 9
ssr based F test:         F=2.2134  , p=0.0217  , df_denom=256, df_num=9
ssr based chi2 test:   chi2=21.3990 , p=0.0110  , df=9
likelihood ratio test: chi2=20.6072 , p=0.0145  , df=9
parameter F test:         F=2.2134  , p=0.0217  , df_denom=256, df_num=9

Granger Causality
number of lags (no zero) 10
ssr based F test:         F=1.8526  , p=0.0523  , df_denom=253, df_num=10
ssr based chi2 test:   chi2=20.0635 , p=0.0287  , df=10
likelihood ratio test: chi2=19.3629 , p=0.0359  , df=10
parameter F test:         F=1.8526  , p=0.0523  , df_denom=253, df_num=10

Granger Causality
number of lags (no zero) 11
ssr based F test:         F=1.5426  , p=0.1168  , df_denom=250, df_num=11
ssr based chi2 test:   chi2=18.5294 , p=0.0701  , df=11
likelihood ratio test: chi2=17.9277 , p=0.0833  , df=11
parameter F test:         F=1.5426  , p=0.1168  , df_denom=250, df_num=11

Granger Causality
number of lags (no zero) 12
```

```
ssr based F test:         F=1.4449  , p=0.1461  , df_denom=247, df_num=12
ssr based chi2 test:   chi2=19.0940 , p=0.0863  , df=12
likelihood ratio test: chi2=18.4536 , p=0.1026  , df=12
parameter F test:         F=1.4449  , p=0.1461  , df_denom=247, df_num=12

Granger Causality
number of lags (no zero) 13
ssr based F test:         F=1.2711  , p=0.2310  , df_denom=244, df_num=13
ssr based chi2 test:   chi2=18.3531 , p=0.1446  , df=13
likelihood ratio test: chi2=17.7584 , p=0.1669  , df=13
parameter F test:         F=1.2711  , p=0.2310  , df_denom=244, df_num=13

Granger Causality
number of lags (no zero) 14
ssr based F test:         F=1.0780  , p=0.3782  , df_denom=241, df_num=14
ssr based chi2 test:   chi2=16.9080 , p=0.2611  , df=14
likelihood ratio test: chi2=16.3997 , p=0.2896  , df=14
parameter F test:         F=1.0780  , p=0.3782  , df_denom=241, df_num=14

Granger Causality
number of lags (no zero) 15
ssr based F test:         F=1.1711  , p=0.2952  , df_denom=238, df_num=15
ssr based chi2 test:   chi2=19.8539 , p=0.1776  , df=15
likelihood ratio test: chi2=19.1554 , p=0.2068  , df=15
parameter F test:         F=1.1711  , p=0.2952  , df_denom=238, df_num=15
1
```