oussama-camil-baptiste-time-series

April 5, 2023

[104]:
```python
import pandas as pd
from scipy.optimize import minimize
from statsmodels.tsa.ar_model import AutoReg
from sklearn.metrics import r2_score
from scipy.stats import norm

import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt

from scipy.stats import t

import plotly.subplots as sp
import plotly.graph_objects as go

from statsmodels.tsa.seasonal import seasonal_decompose
```

[2]:
```python
df = pd.read_excel('data_exam.xlsx', index_col='Unnamed: 0')
df
```

[2]:

	VIX	Parkinson	Squared returns
2000-01-04	27.01	0.001406	0.000612
2000-01-05	26.41	0.000002	0.000326
2000-01-06	25.73	0.000045	0.000169
2000-01-07	21.72	0.001577	0.000723
2000-01-10	21.71	0.000109	0.000111
…	…	…	…
2023-02-08	19.63	0.000117	0.000070
2023-02-09	20.71	0.000088	0.000179
2023-02-10	20.53	0.000004	0.000041
2023-02-13	20.34	0.000134	0.000110
2023-02-14	18.94	0.000019	0.000110

[5822 rows x 3 columns]

0.0.1 1. Explain the concept behind each column. What are these time series meant to represent? Transform these time series so that they are comparable in scale and order

- **Le VIX est une mesure de la volatilité attendue des prix des actions du S&P 500. Elle est calculée en fonction des prix des options d'achat et de vente sur cet indice. Le VIX est souvent utilisé comme indicateur de l'état d'anxiété du marché et de la perception des investisseurs quant aux risques à venir.**
- **Le Parkinson est une mesure de la volatilité réalisée des prix des actions. Elle est calculée en fonction des écarts entre les prix les plus élevés et les plus bas sur une période donnée. Cette mesure est plus sensible aux mouvements soudains des prix.**
- **Les rendements au carré, quant à eux, sont une mesure de la volatilité réalisée des prix des actions qui est calculée en prenant les carrés des rendements quotidiens des prix des actions. Cette mesure est souvent utilisée pour évaluer la volatilité des prix des actions sur des périodes plus longues que le Parkinson. Elle est moins sensible aux mouvements brusques des prix et peut être utilisée pour évaluer le risque d'un portefeuille d'investissement.**

0.0.2 Les trois séries ne sont pas comparable, il faut donc les mettre à la même échelle

[3]:
```python
# On fait 3 graphiques pour se rendre compte de la différence d'échelle et de␣
↪l'allure des courbes
fig = sp.make_subplots(rows=3, cols=1, shared_xaxes=True, vertical_spacing=0.05)

fig.add_trace(go.Scatter(x=df.index, y=df['VIX'], name='VIX'), row=1, col=1)
fig.add_trace(go.Scatter(x=df.index, y=df['Parkinson'], name='Parkinson'),␣
↪row=2, col=1)
fig.add_trace(go.Scatter(x=df.index, y=df['Squared returns'], name='Rendements␣
↪au carré'), row=3, col=1)

fig.update_yaxes(title_text='VIX', row=1, col=1)
fig.update_yaxes(title_text='Parkinson', row=2, col=1)
fig.update_yaxes(title_text='Rendements au carré', row=3, col=1)
fig.update_xaxes(title_text='Temps', row=3, col=1)

fig.update_layout(height=800, width=800, title={
    'text': "Graphiques financiers",
    'y':0.95,
    'x':0.5,
    'xanchor': 'center',
    'yanchor': 'top'})

# Afficher la figure
fig.show()
```

0.0.3 les séries Squared returns et Parkinson sont des mesures intra day, ce qui explique leur différente echelle et faibles valeurs. On transforme donc ces séries en serie de volatilité annuelle comparable en échelle et en ordre.

[43]:
```
# Transformation des series
ann_vol = (df['Squared returns'] ** 0.5) * (252 ** 0.5) * 100
Parkinson_ann = (df['Parkinson'] ** 0.5) * (252 ** 0.5) * 100

fig = go.Figure()

fig.add_trace(go.Scatter(x=df.index, y=Parkinson_ann, name='Parkinson annuel'))
fig.add_trace(go.Scatter(x=df.index, y=ann_vol, name='Volatilité annuelle'))
fig.add_trace(go.Scatter(x=df.index, y=df['VIX'], name='VIX'))

fig.update_layout(title='Indicateurs de volatilité', xaxis_title='Temps',␣
↪yaxis_title='Volatilité',
                  legend=dict(yanchor="top", y=0.99, xanchor="left", x=0.01))

fig.show()
```

0.0.4 2. Estimate and AR model on each of these time series. Determine the order of the AR process and show the estimates. What can you conclude from these estimations

Avant d'estimer un AR(p) il faut faire un test de stationnarité pour savoir s'il y a des transformations à faire

1 Test de stationnarité

[44]:
```
from statsmodels.tsa.stattools import adfuller
import pandas as pd

# Fonction du test augmented dickey fuller
def dickey(series, nom_de_la_serie:str):
    result = adfuller(series)

    print('ADF Statistic: %f' % result[0])
    print('p-value: %f' % result[1])

    #  HO : la série n'est pas stationnaire donc si p < 0.05 on rejette HO
    if result[1] > 0.05:
        print(f'La série {nom_de_la_serie} n\'est pas stationnaire.')
    else:
        print(f'La série {nom_de_la_serie} est stationnaire.')
```

1.1 Les 3 séries sont stationnaires il faut donc se référer au PACF

[45]:
```
dickey(df['VIX'], 'VIX')
print('#'*30)
dickey(Parkinson_ann, 'Parkinson')
print('#'*30)
dickey(ann_vol, 'returns')
```

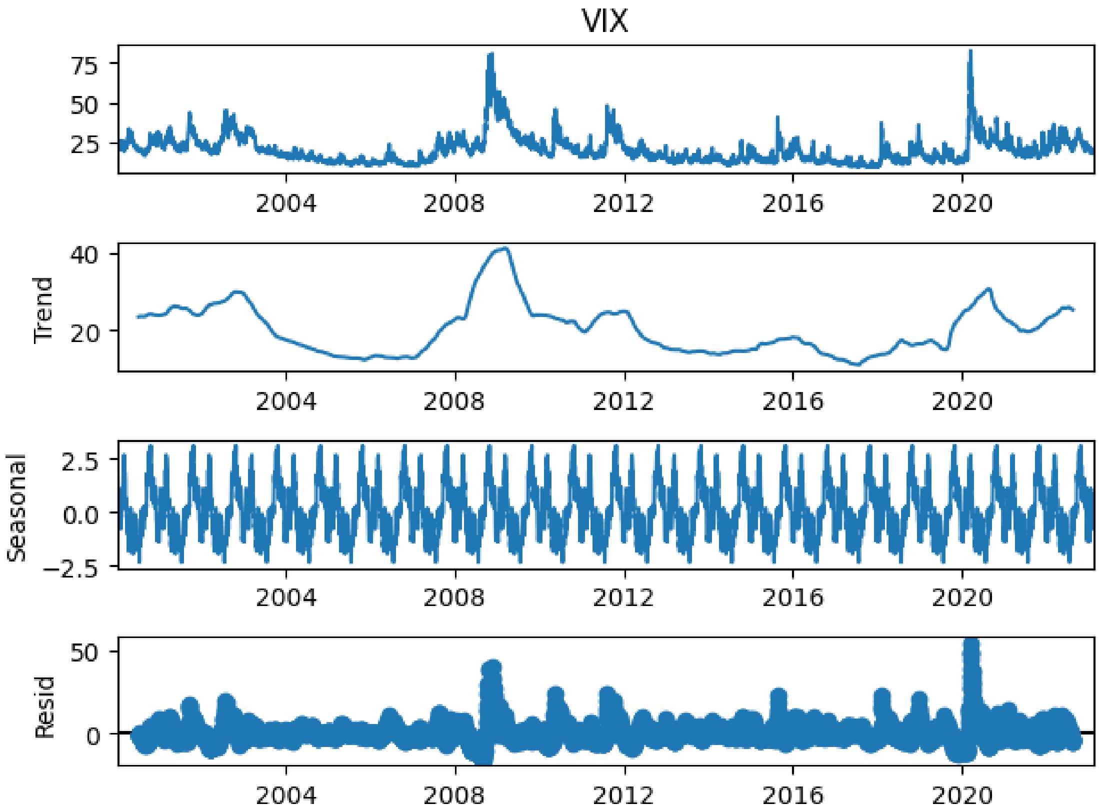
```
ADF Statistic: -5.742412
p-value: 0.000001
La série VIX est stationnaire.
##############################
ADF Statistic: -7.314537
p-value: 0.000000
La série Parkinson est stationnaire.
##############################
ADF Statistic: -6.800695
p-value: 0.000000
La série returns est stationnaire.
```

On regarde ensuite si ce sont des séries additives ou multiplicatives pour plusieurs raisons :

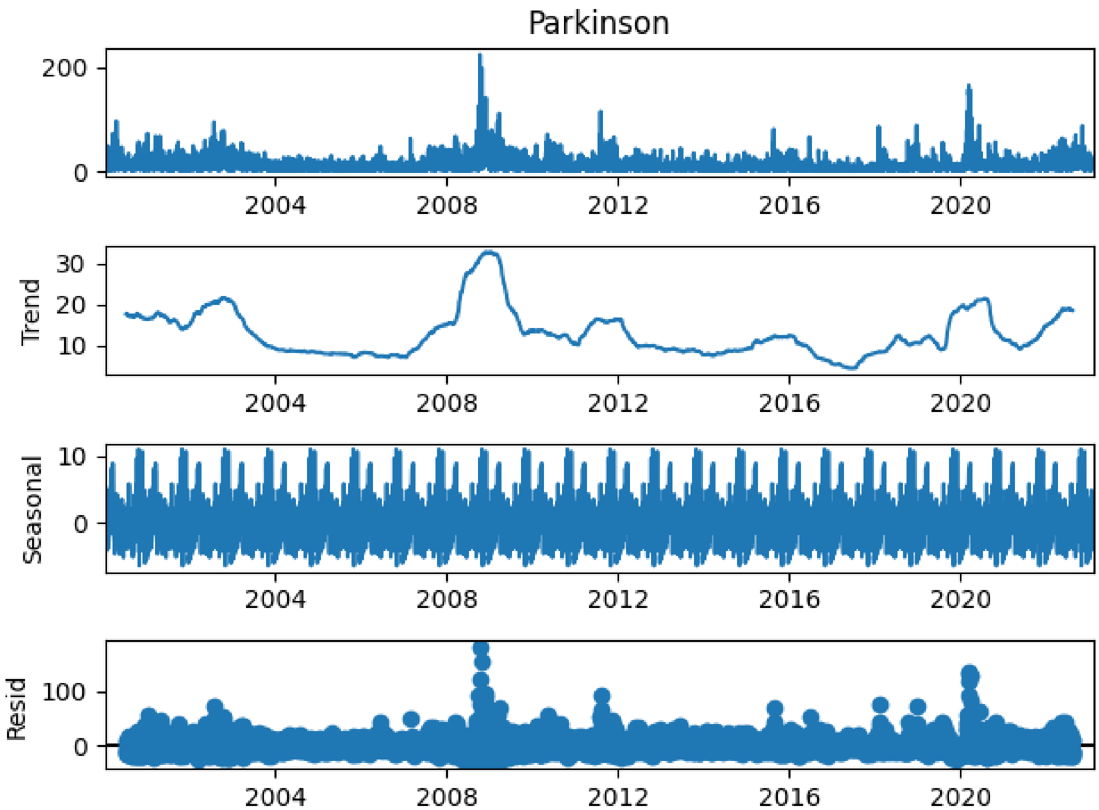
- **Prévision** : Si la série est additive, les prévisions sont souvent plus simples, car on peut simplement extrapoler les tendances précédentes en utilisant des méthodes de régression linéaire. Si la série est multiplicative, il faut tenir compte de la croissance exponentielle, ce qui peut rendre les prévisions plus complexes.
- **Modélisation** : Comprendre si une série est additive ou multiplicative est important pour choisir le bon modèle. Par exemple, si la série est additive, on peut utiliser un modèle de régression linéaire, tandis que si la série est multiplicative, un modèle de régression non linéaire peut être plus approprié.
- **Interprétation des résultats**: La signification des résultats peut être différente selon que la série est additive ou multiplicative. Par exemple, dans le cas d'une augmentation de pourcentage dans une série multiplicative, une augmentation de 10 % peut être significativement différente d'une augmentation de 20 %, tandis que dans une série additive, une augmentation de 10 unités est toujours une augmentation de 10 unités.

1.1.1 Les trois séries sont additive car la saisonalité garde la même magnitude à travers le temps

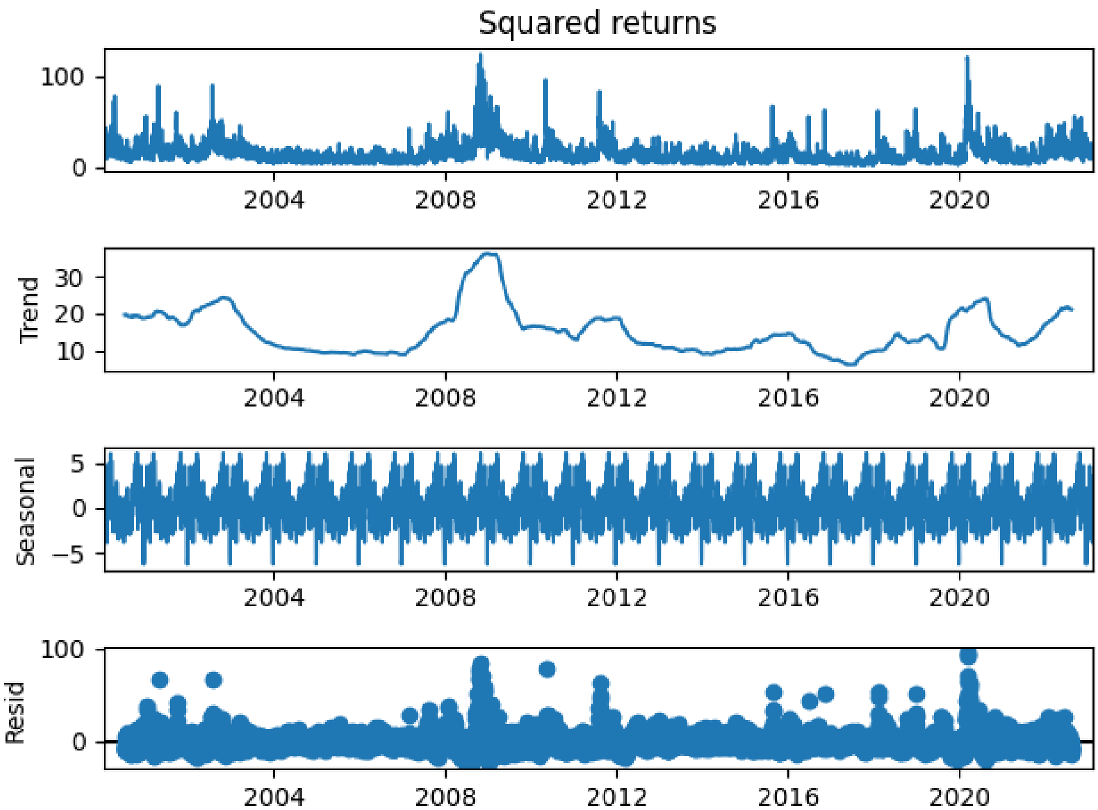
[46]:
```
for series in [df['VIX'], Parkinson_ann, ann_vol]:
    seasonal_decompose(series, model="additive", period=252).plot()
```



VIX
75
50
25
2004
2008
2012
2016
2020
Trend
40
20
Seasonal
2.5
0.0
−2.5
Resid
50
0



Parkinson
200
0
2004
2008
2012
2016
2020
Trend
30
20
10
2004
2008
2012
2016
2020
Seasonal
10
0
2004
2008
2012
2016
2020
Resid
100
0
2004
2008
2012
2016
2020



1.1.2 Les 3 séries sont additives donc pas besoin de faire transformation particulières

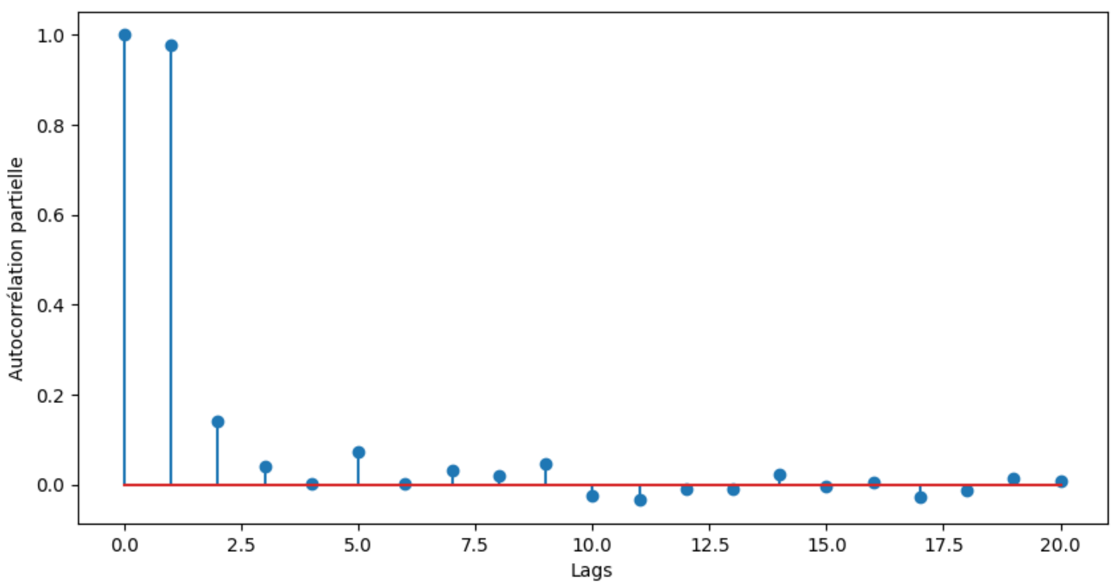
2 Determine the order of the AR process and show the estimates.

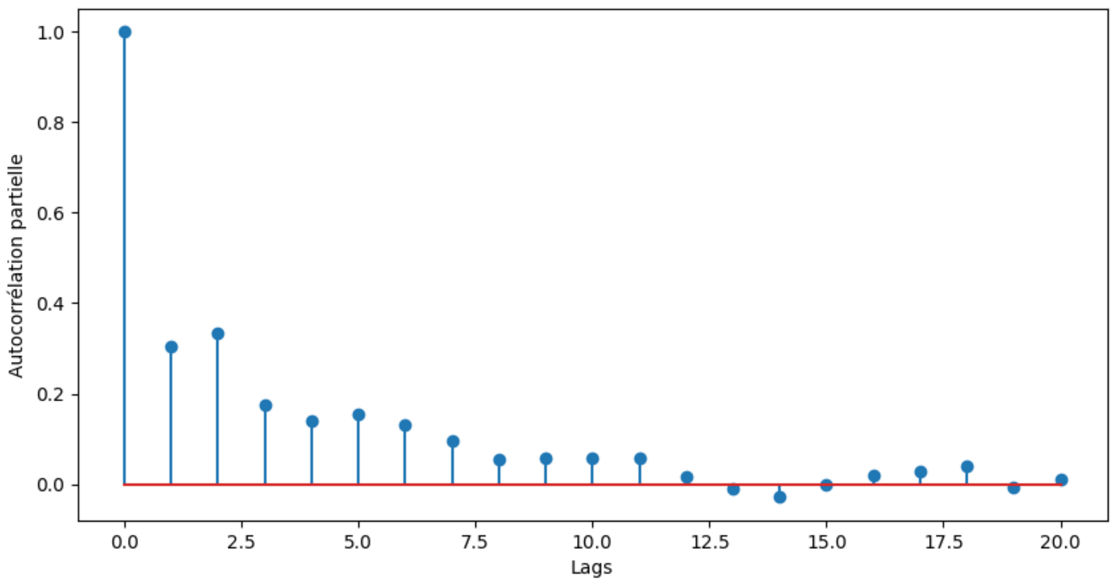
Pour determiner l'ordre de l'AR on utilise le PACF

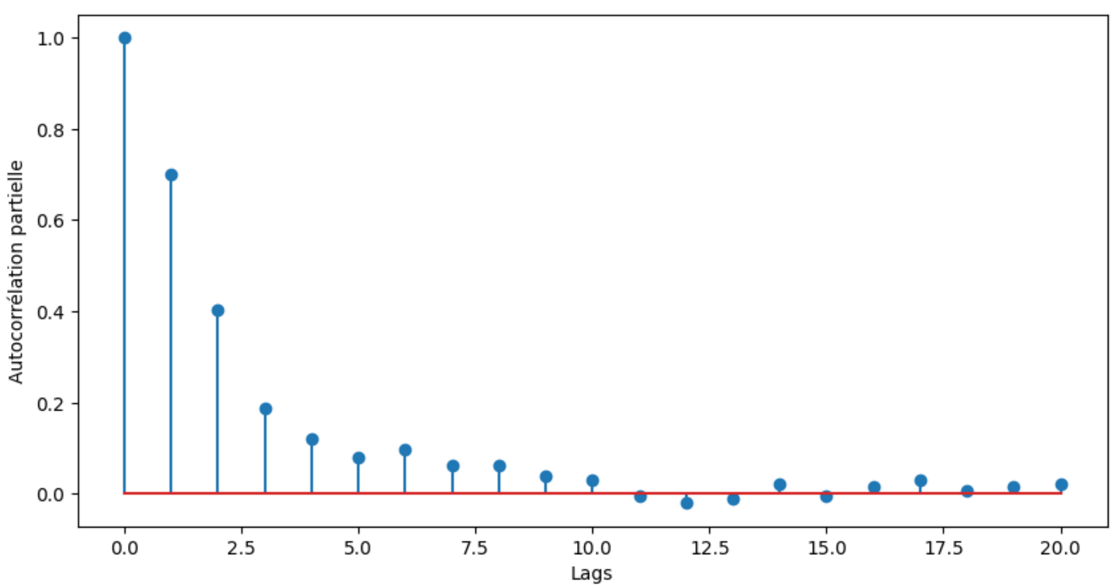
3 PACF

[47]:
```
series1 = df['VIX']
series2 = Parkinson_ann
series3 = ann_vol



# Fonction pour déterminer le bon Lag d'une série
def find_best_lag(series):
    # On calcule le PACF de la série et on affiche 20 lags
    pacf = sm.tsa.stattools.pacf(series, nlags=20, method='ols')

    # On trace le PACF
    plt.figure(figsize=(10, 5))
```

```
    plt.stem(pacf)
    plt.xlabel('Lags')
    plt.ylabel('Autocorrélation partielle')
    plt.show()

    # On retient le premier lag où l'autocorrélation partielle est inférieure à␣
 ↪la limite de confiance
    # (définie ici comme 2/sqrt(n), où n est la longueur de la série)
    n = len(series)
    conf_limit = 2 / np.sqrt(n)
    for i in range(len(pacf)):
        if pacf[i] < conf_limit:
            return i

    # Si aucun lag ne convient, retourner le nombre de lags maximum
    return 20

# Trouver le bon Lag pour chaque série
lag1 = find_best_lag(series1)
lag2 = find_best_lag(series2)
lag3 = find_best_lag(series3)

# Afficher les résultats
print('Lag optimal pour la série VIX: {}'.format(lag1))
print('Lag optimal pour la série Parkinson: {}'.format(lag2))
print('Lag optimal pour la série Squared returns: {}'.format(lag3))
```








```
Lag optimal pour la série VIX: 4
Lag optimal pour la série Parkinson: 12
Lag optimal pour la série Squared returns: 11
```

3.1 Estimate and AR model on each of these time series.

[122]:
```python
import numpy as np
from scipy.optimize import minimize
from scipy.stats import norm

# On définit une fonction qui prend en entrée des paramètres initiaux, une␣
↪série, un nombre de lags chosis auparavant
# à l'ai du PACF
def ML_criterion_arp(para, x, lags, plot=False):
    """
    Fonction qui calcule la log-vraisemblance négative pour un modèle␣
↪autorégressif d'ordre p (AR(p)).

    Parameters
    ----------
    para : numpy.ndarray
        Un vecteur de taille p+1 contenant les paramètres du modèle, où p est␣
↪le nombre de lags.
        Le premier élément correspond à la constante, les autres aux␣
↪coefficients associés aux lags.
    x : pandas.Series
        Une série temporelle de taille n, où n est le nombre d'observations.
    lags : list
        Une liste des lags à inclure dans le modèle.
    plot : bool, optional
        Si True, la fonction affichera un graphique de la série initiale et de␣
↪la série ajustée.

    Returns
    -------
    float
        La log-vraisemblance négative du modèle AR(p) ajusté aux données x.

    """
    expected = para[0] # Correspond au paramètre x0
    for i, lag in enumerate(lags):
        expected += para[i+1]*x.shift(lag) # On ajoute autant de paramètres et␣
↪de lags que spécifier en entrée ce qui rend l'écriture moins fastidieuse
    expected = expected.dropna()
    difference = x.iloc[lags[-1]:] - expected
    volatility = np.nanstd(difference)
    loglik = norm.pdf(x.iloc[lags[-1]:], expected, volatility)
    criterion = np.nansum(np.log(loglik))
    if plot:
        temp = pd.concat([x, expected], axis=1)
        temp.columns = ['initial', 'fit']
```

```
        temp = temp.dropna()
        temp.plot()
    return -criterion # On retourne la log vraisemblance négativement pour la␣
↪minimiser ce qui revient à maximiser son inverse


def get_lags(signif_lags, p):

    return [i for i in range(1, p+1) if i not in signif_lags]

def backward(x, p, alpha=0.05):
    """
    Effectue une sélection de modèle en utilisant une méthode backward
    x : une série temporelle
    p : ordre maximal de l'AR à tester
    alpha : seuil de significativité du test de Student (par défaut 0.05)
    """
    # liste des lags significatifs
    # A chaque itération si une paramètre n'est pas significatif il est retiré
    signif_lags = list(range(1, p+1))

    while True:
        # On ajuste le modèle avec les lags actuels
        para0 = np.random.uniform(low=-10, high=10, size=len(signif_lags)+1)
        res_AR = minimize(ML_criterion_arp, para0, method='BFGS', args=(x,␣
↪signif_lags), options={'disp': False})

        # test de Student pour sélectionner les lags significatifs
        std_para = np.diag(res_AR.hess_inv)**.5
        tstats = res_AR.x/std_para
        max_tstat = np.abs(tstats[1:]).min()
        if max_tstat < norm.ppf(1-alpha/2):
            # si le lag le moins significatif n'est pas significatif
            # retirer le lag correspondant
            lag_to_remove = signif_lags[np.abs(tstats[1:]).argmin()]
            signif_lags.remove(lag_to_remove)
        else:
            # tous les lags restants sont significatifs
            break

    # ajuster le modèle avec les lags sélectionnés
    para0 = np.random.uniform(low=-10, high=10, size=len(signif_lags)+1)
    res_AR = minimize(ML_criterion_arp, para0, method='BFGS', args=(x,␣
↪signif_lags), options={'disp': False})
    std_param =np.diag(res_AR.hess_inv)**.5
    student_test = res_AR.x/std_para
    return res_AR.x, signif_lags, res_AR.fun, std_param, student_test
```

```
# Exemple des sorties de la fonction que nous allons mettre en forme
p = 12
res_AR_x, signif_lags, res_AR_fun, std_param, student_test =␣
 ↪backward(Parkinson_ann, p=p)
print("Lags significatifs :", signif_lags)
print("MLE :", res_AR_fun)
print("Paramètres optimaux : ",res_AR_x)
print('Std param : ', std_param)
print("Student test : ", student_test)
```

```
Lags significatifs : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
MLE : 22974.15882509276
Paramètres optimaux :  [2.23291038 0.04877961 0.18517283 0.0697123  0.05904327
0.09900285
 0.09994008 0.0749163  0.03645396 0.04239841 0.05507098 0.05834115]
Std param :  [0.28291852 0.01337626 0.01336163 0.0143969  0.01382834 0.01414704
 0.01367834 0.01350851 0.01364007 0.01264835 0.01326417 0.01320267]
Student test :  [ 8.88921302  3.73261917 15.93849189  5.76260935  4.95994488
8.04484949
  7.92924498  5.5043494   3.21210411  3.43632484  4.0261235   4.61600519]
```

[123]:
```
def get_residual_variance(para_opt, x, lags):
    """
    Calcule la variance résiduelle à partir des paramètres optimaux d'un modèle␣
 ↪AR(p)

    Parameters
    ----------
    para_opt : numpy.ndarray
        Un vecteur de taille p+1 contenant les paramètres optimaux du modèle␣
 ↪AR(p), où p est le nombre de lags.
        Le premier élément correspond à la constante, les autres aux␣
 ↪coefficients associés aux lags.
    x : pandas.Series
        Une série temporelle de taille n, où n est le nombre d'observations.
    lags : list
        Une liste des lags à inclure dans le modèle.

    Returns
    -------
    float
        La variance résiduelle du modèle AR(p) ajusté aux données x.

    """
    # Prédiction des valeurs ajustées de la série temporelle
    expected = para_opt[0]
```

```
    for i, lag in enumerate(lags):
        expected += para_opt[i+1]*x.shift(lag)
    expected = expected.dropna()

    # Calcul des résidus
    residuals = x.iloc[lags[-1]:] - expected

    # Calcul de la variance résiduelle
    residual_variance = np.var(residuals)

    return residual_variance, expected
```

4 Série Parkinson

4.0.1 On peut s'assurer que l'AR(11) est le plus adapté car sa variance résiduelle est la moins élevé

[124]:
```
my_list = []

for i in range(1, 12):
    # votre code ici
    my_list.append(i)
    residual_variance, expected = get_residual_variance(res_AR_x,␣
 ↪Parkinson_ann, my_list)

    # mettre à jour la première valeur de la liste avec la valeur de la␣
 ↪dernière itération
    print("Lags compris dans le modèle :", my_list)
    print("Variance résiduel : ", residual_variance)
```

```
Lags compris dans le modèle : [1]
Variance résiduel :  215.92089956218277
Lags compris dans le modèle : [1, 2]
Variance résiduel :  192.2193719748052
Lags compris dans le modèle : [1, 2, 3]
Variance résiduel :  185.61712081730343
Lags compris dans le modèle : [1, 2, 3, 4]
Variance résiduel :  180.30311321752913
Lags compris dans le modèle : [1, 2, 3, 4, 5]
Variance résiduel :  172.6766837319362
Lags compris dans le modèle : [1, 2, 3, 4, 5, 6]
Variance résiduel :  166.55840596625788
Lags compris dans le modèle : [1, 2, 3, 4, 5, 6, 7]
Variance résiduel :  163.1936925373538
Lags compris dans le modèle : [1, 2, 3, 4, 5, 6, 7, 8]
Variance résiduel :  162.05434331467296
Lags compris dans le modèle : [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
Variance résiduel :  160.93684756585105
Lags compris dans le modèle : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Variance résiduel :  159.84401677718265
Lags compris dans le modèle : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
Variance résiduel :  159.05505341295364
```

4.0.2 On créer des tables pour résumer les résultats comme statsmodels

Tout les coefficients sont significatifs

[93]:
```python
from scipy.stats import t

results_summary = pd.DataFrame({
    'Lags': signif_lags,
    'Coefficient': res_AR_x[1:],
    'Std. Error': std_param[1:],
    't': student_test[1:],
    'P>|t|': 2 * (1 - t.cdf(np.abs(student_test[1:]), len(Parkinson_ann) - len(res_AR_x)))
})

# Ajouter la ligne pour la constante
results_summary.loc[len(results_summary)] = ['const', res_AR_x[0], std_param[:1][0],student_test[:1][0], 2 * (1 - t.cdf(np.abs(student_test[:1]), len(Parkinson_ann) - len(res_AR_x)))[0]]
print(" Log Likelihood :", res_AR_fun)


r2 = r2_score(Parkinson_ann[11:], expected)
print("Coefficient de détermination R² : ", r2)

print(results_summary.to_string(index=False))
```

```
 Log Likelihood : 22974.15882509273
Coefficient de détermination R² :  0.28305328300341914
 Lags  Coefficient  Std. Error             t    P>|t|
    1     0.048780    0.019955      3.736047 0.000189
    2     0.185173    0.012250 642063.279767 0.000000
    3     0.069712    0.008739      2.928873 0.003415
    4     0.059043    0.016490      4.301193 0.000017
    5     0.099003    0.020890      3.434654 0.000597
    6     0.099940    0.017670      4.393402 0.000011
    7     0.074916    0.017547      4.874205 0.000001
    8     0.036454    0.015326      2.577520 0.009976
    9     0.042398    0.013476      2.220789 0.026404
   10     0.055071    0.025722      3.290463 0.001006
   11     0.058341    0.017222      4.403768 0.000011
const     2.232920    0.607071     13.841628 0.000000
```

Interprétation Nous avons un AR avec 11 paramètres significatifs + 1 constante et une variance expliqué d'environ 28% par l'AR ce qui est relativement élevé

5 Série VIX

[127]:
```
p = 4
res_AR_x, signif_lags, res_AR_fun, std_param, student_test =␣
↪backward(df['VIX'], p=p)

my_list = []

for i in range(1, 4):
    # votre code ici
    my_list.append(i)
    residual_variance, expected = get_residual_variance(res_AR_x, df['VIX'],␣
↪my_list)

    # mettre à jour la première valeur de la liste avec la valeur de la␣
↪dernière itération
    print("Lags compris dans le modèle :", my_list)
    print("Variance résiduel : ", residual_variance)
```

```
Lags compris dans le modèle : [1]
Variance résiduel :  4.775929561943273
Lags compris dans le modèle : [1, 2]
Variance résiduel :  3.3089230548893185
Lags compris dans le modèle : [1, 2, 3]
Variance résiduel :  3.187346232984241
```

[113]:
```
results_summary = pd.DataFrame({
    'Lags': signif_lags,
    'Coefficient': res_AR_x[1:],
    'Std. Error': std_param[1:],
    't': student_test[1:],
    'P>|t|': 2 * (1 - t.cdf(np.abs(student_test[1:]), len(df['VIX']) -␣
↪len(res_AR_x)))
})

# Ajouter la ligne pour la constante
results_summary.loc[len(results_summary)] = ['const', res_AR_x[0], std_param[:
↪1][0],student_test[:1][0], 2 * (1 - t.cdf(np.abs(student_test[:1]),␣
↪len(Parkinson_ann) - len(res_AR_x)))[0]]
print(" Log Likelihood :", res_AR_fun)


r2 = r2_score(df['VIX'][3:], expected)
print("Coefficient de détermination R² : ", r2)
```

```
print(results_summary.to_string(index=False))
```

```
 Log Likelihood : 11629.462756267369
Coefficient de détermination R² :  0.9576492910818553
 Lags  Coefficient  Std. Error         t        P>|t|
    1     0.835916    0.013168 62.703384 0.000000e+00
    2     0.105706    0.017499  7.950654 2.220446e-15
    3     0.040277    0.013530 38.872758 0.000000e+00
const     0.364082    0.062081  2.886492 3.909965e-03
```

Interprétation **Nous avons un AR avec 3 paramètres significatifs + 1 constante dont le premier qui a une grande importance et une variance expliqué d'environ 95% par l'AR ce qui est relativement élevé**

6 Série Squared returns

[129]:
```
p = 11
res_AR_x, signif_lags, res_AR_fun, std_param, student_test = backward(ann_vol,␣
 ↪p=p)

my_list = []

for i in [1, 2, 3, 4, 6, 7, 9]:
    my_list.append(i)
    residual_variance, expected = get_residual_variance(res_AR_x, ann_vol,␣
 ↪my_list)

    print("Lags compris dans le modèle :", my_list)
    print("Variance résiduel : ", residual_variance)
```

```
Lags compris dans le modèle : [1]
Variance résiduel :  89.40295630404849
Lags compris dans le modèle : [1, 2]
Variance résiduel :  64.66583134424663
Lags compris dans le modèle : [1, 2, 3]
Variance résiduel :  58.562312680489015
Lags compris dans le modèle : [1, 2, 3, 4]
Variance résiduel :  55.95752920396364
Lags compris dans le modèle : [1, 2, 3, 4, 6]
Variance résiduel :  53.73982084282242
Lags compris dans le modèle : [1, 2, 3, 4, 6, 7]
Variance résiduel :  52.897190951618
Lags compris dans le modèle : [1, 2, 3, 4, 6, 7, 9]
Variance résiduel :  52.37395858041633
```

[119]:
```
results_summary = pd.DataFrame({
    'Lags': signif_lags,
    'Coefficient': res_AR_x[1:],
    'Std. Error': std_param[1:],
    't': student_test[1:],
    'P>|t|': 2 * (1 - t.cdf(np.abs(student_test[1:]), len(ann_vol) -␣
 ↪len(res_AR_x)))
})

# Ajouter la ligne pour la constante
results_summary.loc[len(results_summary)] = ['const', res_AR_x[0], std_param[:
 ↪1][0],student_test[:1][0], 2 * (1 - t.cdf(np.abs(student_test[:1]),␣
 ↪len(Parkinson_ann) - len(res_AR_x)))[0]]
print(" Log Likelihood :", res_AR_fun)


r2 = r2_score(ann_vol[9:], expected)
print("Coefficient de détermination R² : ", r2)

print(results_summary.to_string(index=False))
```

```
 Log Likelihood : 19735.377272076796
Coefficient de détermination R² :  0.6018875738263005
 Lags  Coefficient  Std. Error          t        P>|t|
    1     0.292376    0.013290 22.616902 0.000000e+00
    2     0.255396    0.012390 19.058776 0.000000e+00
    3     0.106914    0.013161  8.113660 6.661338e-16
    4     0.059714    0.014170  4.389205 1.157803e-05
    6     0.065093    0.014125  7.260413 4.365397e-13
    7     0.040454    0.012987  3.285287 1.024820e-03
    8     0.051666    0.014512  5.509700 3.748890e-08
   10     0.039408    0.012655  2.895282 3.802295e-03
const     1.359750    0.218780 13.138352 0.000000e+00
```

Interprétation **Nous avons un AR avec 10 paramètres + 1 constante significatifs dont le premier qui a une grande importance et une variance expliqué d'environ 60% par l'AR ce qui est relativement élevé**

7 Question 3 : Plot the fitted values for each model vs. the original time series. Why are these fitted values appealing for volatility measuring purposes?

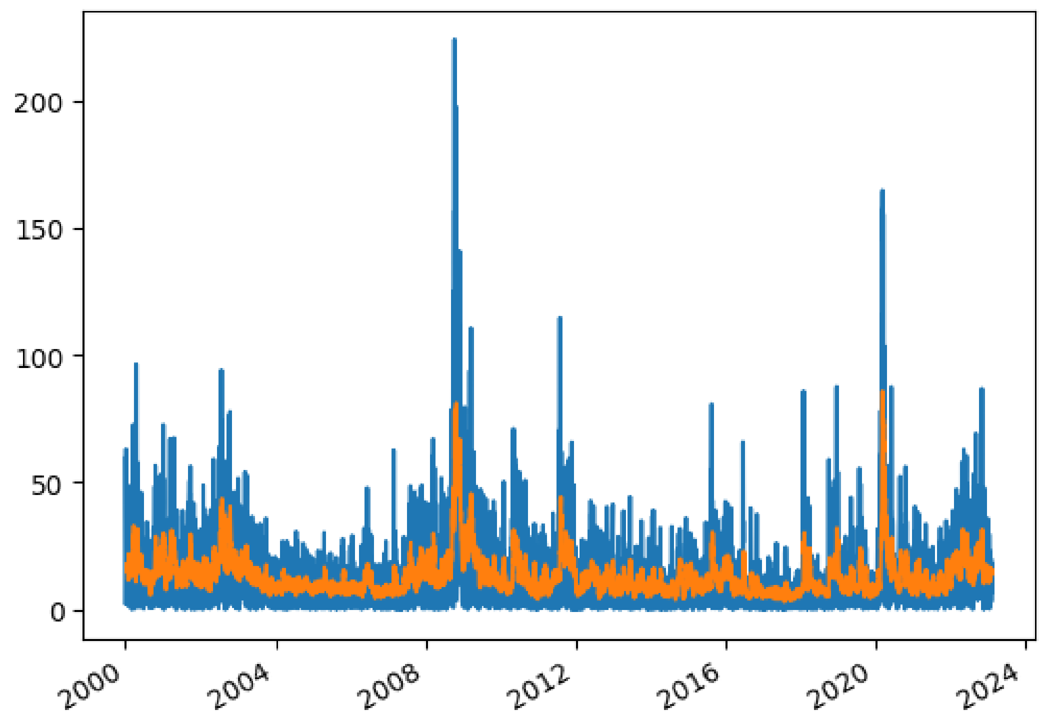
Les valeurs ajustées sont intéressantes pour mesurer la volatilité, car elles représentent la moyenne ou la tendance estimée de la série temporelle. En soustrayant les valeurs ajustées de la série temporelle originale, vous pouvez obtenir les résidus, qui représentent la partie non expliquée ou imprévisible de la série temporelle. Les résidus peuvent

être utilisés pour estimer la volatilité de la série temporelle, qui est une mesure de la quantité de fluctuation ou de variation des données. Le modèle AR peut être utilisé pour estimer la volatilité de la série temporelle en modélisant la variance conditionnelle des résidus, qui est une fonction des valeurs retardées des résidus. En utilisant les valeurs ajustées et les résidus ensemble, vous pouvez estimer la volatilité de la série temporelle et faire des prévisions sur la volatilité future.

8 Parkinson

[126]:
```
Parkinson_ann.plot()
expected.plot()
```

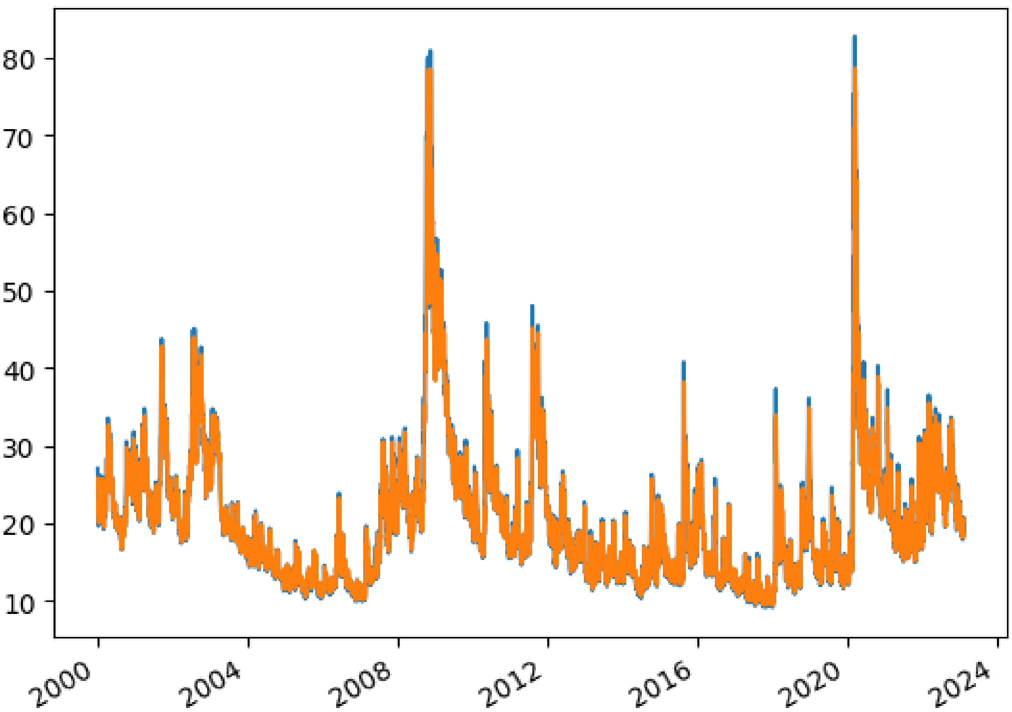
[126]: <Axes: >



9 VIX

[128]:
```
df['VIX'].plot()
expected.plot()
```

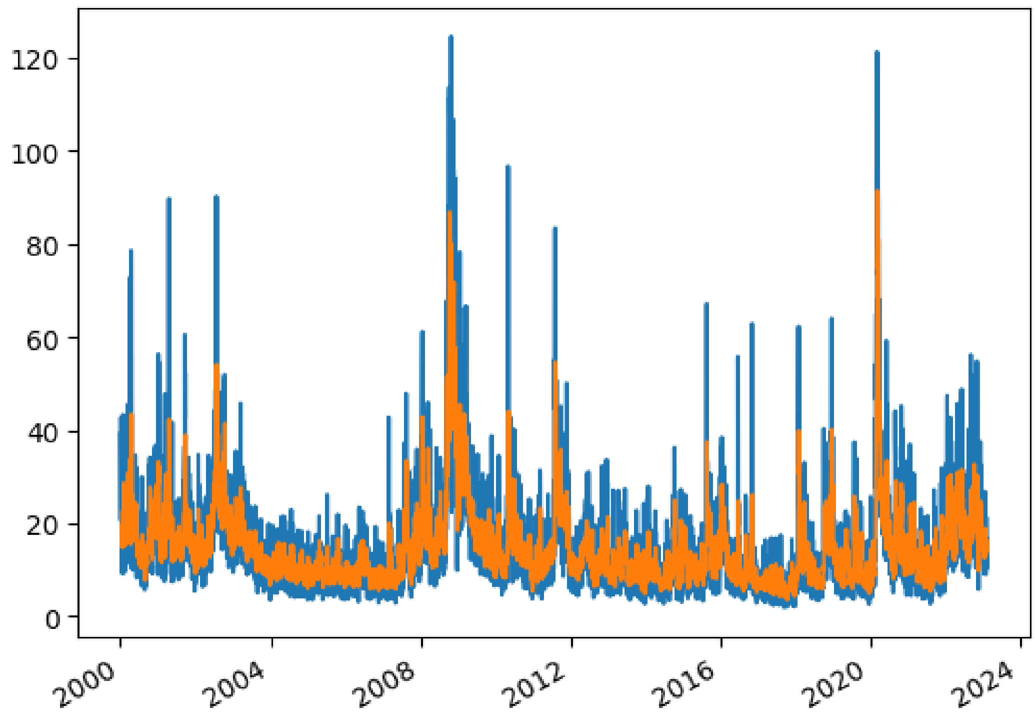
[128]: <Axes: >



10 Squared returns

[131]:
```
ann_vol.plot()
expected.plot()
```

[131]: <Axes: >



11 Question 4 : Estimate an HAR model on each time series. Present the estimated coefficients of the model and compare the loglikelihood of each model to the one obtained from the AR estimations. Which model do you prefer? (4 points). Retain the fitted values of the model you have selected for each of the three time series.

11.0.1 Il faut choisir le modèle en se basant sur le maximum de vraisemblance et des critères d'information

12 HAR modèle

[132]:
```
# Optimization
from scipy.stats import norm

def ML_HAR(para,x,plot=False):
    phi0=para[0]
    phi1=para[1]
    phi2=para[2]
    phi3=para[3]
```

```
    # computing moving averages
    ma5=x.rolling(5).mean().shift(1)
    ma22=x.rolling(22).mean().shift(1)
    ma1=x.shift(1)

    combination=pd.concat([x, ma1, ma5, ma22],axis=1)
    combination=np.log(combination+ 1) # On travail avec le log de la variance␣
 ↪journalière
    # On ajoute 1 à la formule pour eviter log(0)
    combination=combination.dropna()
    combination.columns=['ini','ma1', 'ma5', 'ma22']



   ␣
 ↪expected=phi0+phi1*combination['ma1']+phi2*combination['ma5']+phi3*combination['ma22']
    temp=pd.concat([combination['ini'],expected],axis=1)
    error=combination['ini']-expected
    sigma=np.nanstd(error)
    #print(para)
    density=norm.pdf(combination['ini'],expected,sigma)
    criterion=np.nansum(np.log(density))
    if plot==True:
        #temp.exp(x**2)(data**.5)*(250**.5)*100
        #Parkinson_ann=(df['Parkinson']**.5)*(252**.5)*100
        #temp = np.log(temp)

        temp.plot()
    return -criterion
```

13 VIX Series

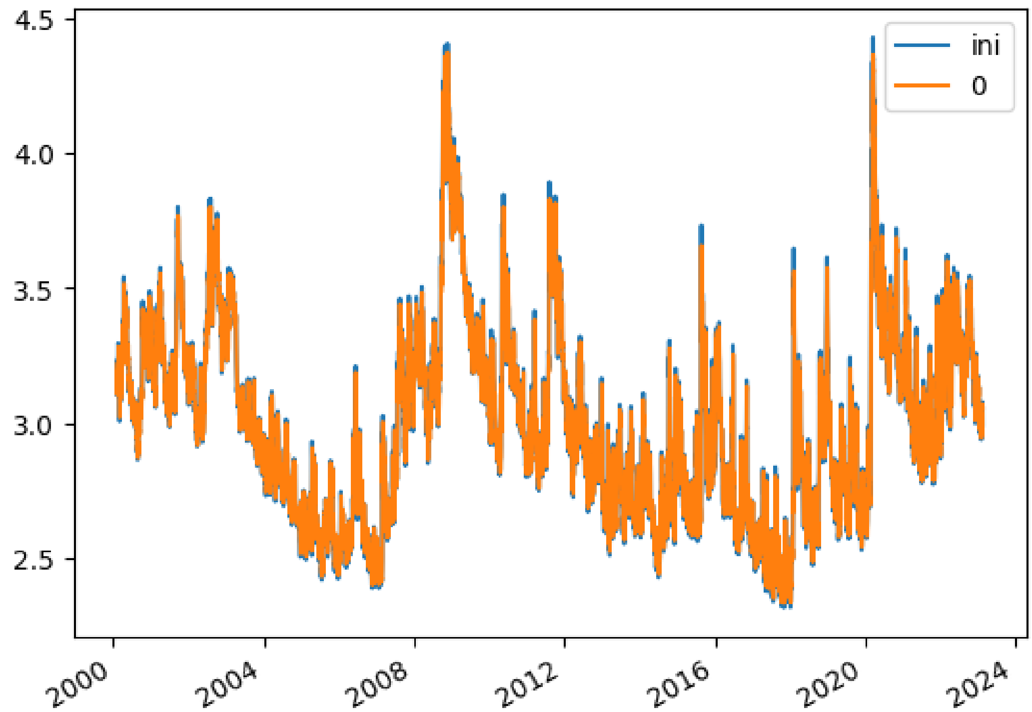
[138]:
```
res_EV_VIX_HAR = minimize(ML_HAR, np.random.uniform(low=-10, high=10, size=4),␣
 ↪method='BFGS', args=(df['VIX']),options={'disp': False})
HAR_param_vix = res_EV_VIX_HAR.x
std_param =np.diag(res_EV_VIX_HAR.hess_inv)**.5
student_test = HAR_param_vix/std_param

print("coefficient estimé du modèle:", HAR_param_vix)
print("test de student :",student_test)
```

```
coefficient estimé du modèle: [0.03820209 0.89003269 0.06168079 0.03538413]
test de student : [ 4.82924475 66.72581127  3.61519275  3.88336358]
```

13.0.1 Tous les coefficients sont significatifs et la première moyenne mobile a un grand impact sur l'estimation du fait de la valeur de son coefficient

[134]:
```
ML_HAR(HAR_param_vix, df['VIX'], plot=True)
```

[134]: -7472.97411505163



14 Serie Squared returns

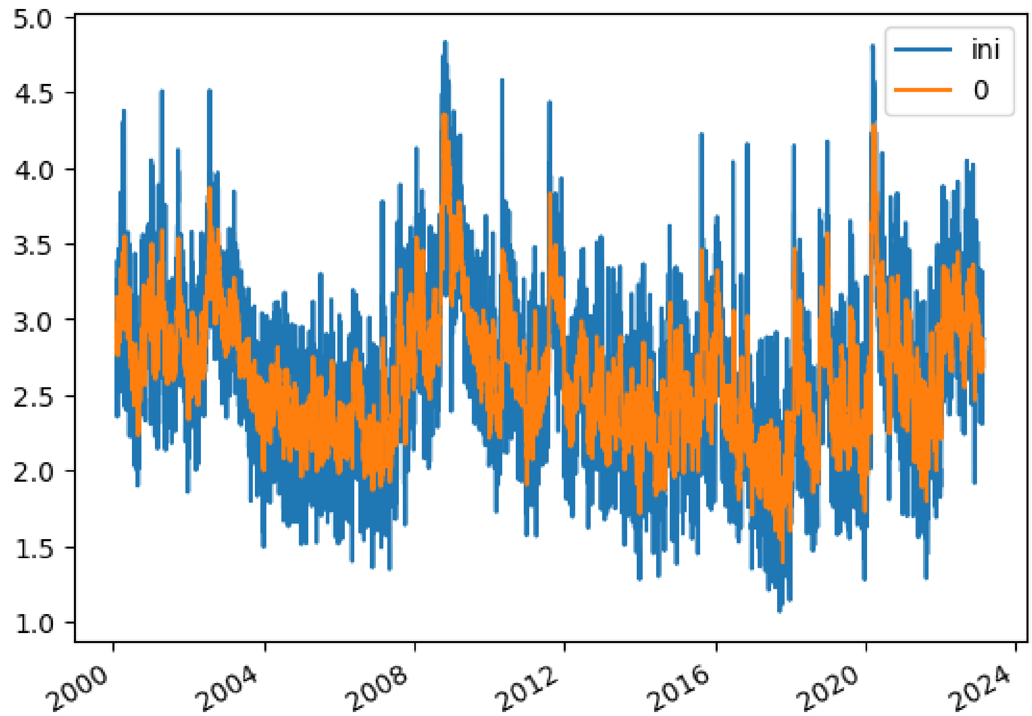
[139]:
```
res_EV = minimize(ML_HAR, np.random.uniform(low=-10, high=10, size=4),
 ↪method='BFGS', args=(ann_vol),options={'disp': False})
HAR_param = res_EV.x
std_param =np.diag(res_EV.hess_inv)**.5
student_test = HAR_param/std_param

print("coefficient estimé du modèle:", HAR_param)
print("test de student :",student_test)
```

```
coefficient estimé du modèle: [0.16721601 0.14299342 0.54123921 0.23692909]
test de student : [10.68406792 12.08712451 53.03009999 18.61270697]
```

14.0.1 Tous les coefficients sont significatifs et le coefficient de la deuxième moyenne mobile correspondant à 5 jours à un impact relativement élevé

[43]:
```
ML_HAR(res_EV.x, ann_vol, plot=True)
```

[43]: 2437.8906003381744



15 Serie parkinson

[140]:
```
res_EV = minimize(ML_HAR, np.random.uniform(low=-10, high=10, size=4),␣
 ↪method='BFGS', args=(Parkinson_ann),options={'disp': False})
HAR_param = res_EV.x
std_param =np.diag(res_EV.hess_inv)**.5
student_test = HAR_param/std_param

print("coefficient estimé du modèle:", HAR_param)
print("test de student :",student_test)
```

```
coefficient estimé du modèle: [ 0.21910789 -0.03945597  0.37904195  0.45367025]
test de student : [ 3.59689678 -2.61481196 10.19315691 11.30300115]
```

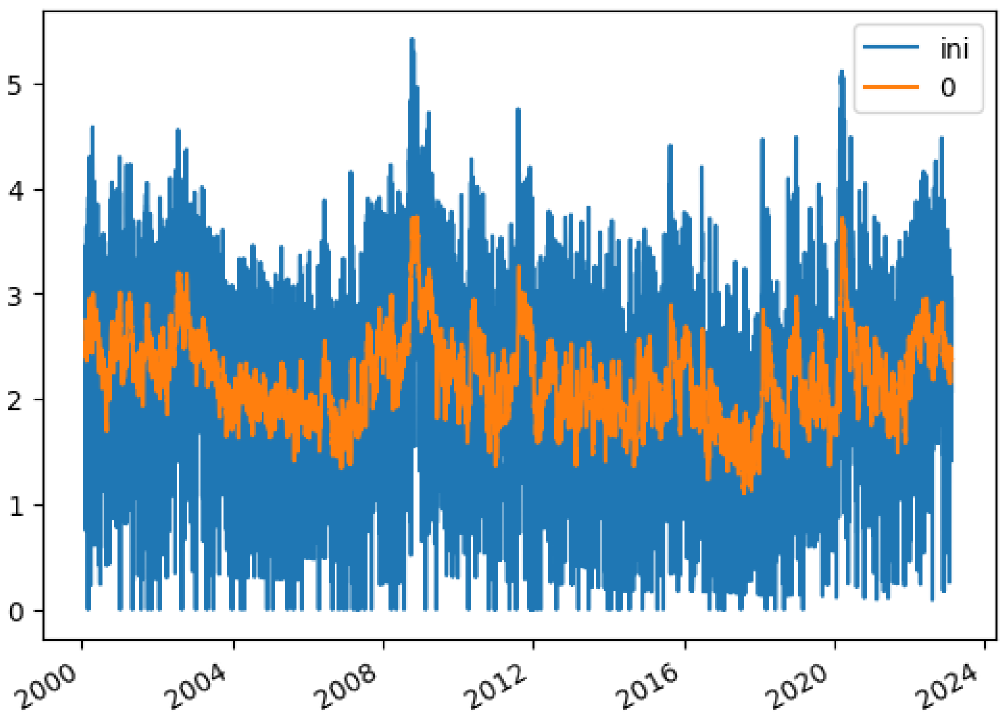
15.0.1 Tous les coefficients sont significatifs et le coefficient de la troisième moyenne mobile correspondant à 22 jours à un impact relativement élevé

15.0.2 Conclusion final pour le modèle HAR

Le modèle HAR semble traduire les stratégies/caractère court, moyen et long terme que représente l'étude de ces trois series

[45]:
```
ML_HAR(res_EV.x, Parkinson_ann, plot=True)
```

[45]: 7518.204480780162



[147]:
```
import numpy as np

def compute_aic_bic(ll, n_params, n_obs):
    """
    Compute AIC and BIC criteria for model selection.
    ll: maximum likelihood value
    n_params: number of estimated parameters
    n_obs: number of observations
    """
    aic = -2 * ll + 2 * n_params
    bic = -2 * ll + n_params * np.log(n_obs)
    return aic, bic
```

16 Pour Parkinson HAR est meilleur que l'AR

[ ]:
```
aic, bic = compute_aic_bic(-ML_HAR(res_EV.x, Parkinson_ann, plot=False), 4,␣
  ↪len(Parkinson_ann))
print(aic, bic)
```

```
15044.408961560363 15071.086558057585
```

[148]:
```
p = 12
res_AR_x, signif_lags, res_AR_fun, std_param, student_test =␣
  ↪backward(Parkinson_ann, p=p)
aic, bic = compute_aic_bic(-res_AR_fun, 11, len(Parkinson_ann))
print(aic, bic)
```

```
45970.31765018615 46043.681040553514
```

17 Pour Squared returns HAR est meilleur que l'AR

[ ]:
```
aic, bic = compute_aic_bic(-ML_HAR(res_EV.x, ann_vol, plot=False), 4,␣
  ↪len(ann_vol))
print(aic, bic)
```

```
4883.781200676354 4910.458797173576
```

[149]:
```
p = 11
res_AR_x, signif_lags, res_AR_fun, std_param, student_test = backward(ann_vol,␣
  ↪p=p)
aic, bic = compute_aic_bic(-res_AR_fun, 8, len(ann_vol))
print(aic, bic)
```

```
39481.00609940145 39534.36129239589
```

18 Pour VIX HAR est meilleur que l'AR

[ ]:
```
aic, bic = compute_aic_bic(ML_HAR(res_EV.x, df['VIX'], plot=False), 4,␣
  ↪len(df['VIX']))
print(aic, bic)
```

```
8294.16567563922 8320.843272136442
```

[150]:
```
p = 4
res_AR_x, signif_lags, res_AR_fun, std_param, student_test =␣
  ↪backward(df['VIX'], p=p)
aic, bic = compute_aic_bic(-res_AR_fun, 2, len(ann_vol))
print(aic, bic)
```

23262.92551253487 23276.264310783485

19 Question 5

[153]:
```python
import numpy as np
from scipy.optimize import minimize

def optimize_ML_HAR(x, plot=False):

    def ML_HAR(para,x):
        phi0=para[0]
        phi1=para[1]
        phi2=para[2]
        phi3=para[3]

        # computing moving averages
        ma5=x.rolling(5).mean().shift(1)
        ma22=x.rolling(22).mean().shift(1)
        ma1=x.shift(1)

        combination=pd.concat([x, ma1, ma5, ma22],axis=1)
        combination=np.log(combination+ 1) # On travail avec le log de la␣
↪variance journalière
        combination=combination.dropna()
        combination.columns=['ini','ma1', 'ma5', 'ma22']

        ␣
↪expected=phi0+phi1*combination['ma1']+phi2*combination['ma5']+phi3*combination['ma22']
        temp=pd.concat([combination['ini'],expected],axis=1)
        error=combination['ini']-expected
        sigma=np.nanstd(error)
        density=norm.pdf(combination['ini'],expected,sigma)
        criterion=np.nansum(np.log(density))
        return -criterion

    # initial guess for parameters
    para0 = np.array([0, 0, 0, 0])

    # minimize negative log-likelihood using Nelder-Mead algorithm
    result = minimize(ML_HAR, para0, args=(x,), method='Nelder-Mead')

    # extract optimized parameter values
    para_opt = result.x

    # compute temp using optimized parameters
```

```python
    expected_opt = para_opt[0] + para_opt[1]*x.shift(1).rolling(5).mean().
↪shift(1) + para_opt[2]*x.shift(1).rolling(22).mean().shift(1) +␣
↪para_opt[3]*x.rolling(22).mean().shift(1)
    temp_opt = pd.concat([x, expected_opt], axis=1)

    if plot:
        temp_opt.plot()

    # compute criterion using optimized parameters
    criterion_opt = -ML_HAR(para_opt, x)

    # return optimized parameter values, criterion, and temp
    return para_opt, criterion_opt, temp_opt
```

[154]:
```python
import pandas as pd
import numpy as np
from scipy.optimize import minimize
from scipy.stats import norm

def compute_fitted_values(x):
    """
    Computes the fitted values for the HAR model using maximum likelihood␣
↪estimation

    Parameters:
    - x : pandas Series
        Daily log-returns of an asset

    Returns:
    - pandas DataFrame
        Dataframe containing the original log-returns and the fitted values
    """

    def ML_HAR(para,x):
        phi0=para[0]
        phi1=para[1]
        phi2=para[2]
        phi3=para[3]

        # computing moving averages
        ma5=x.rolling(5).mean().shift(1)
        ma22=x.rolling(22).mean().shift(1)
        ma1=x.shift(1)

        combination=pd.concat([x, ma1, ma5, ma22],axis=1)
        combination=np.log(combination+ 1) # On travail avec le log de la␣
↪variance journalière
```

```
        combination=combination.dropna()
        combination.columns=['ini','ma1', 'ma5', 'ma22']

      ␣
↪expected=phi0+phi1*combination['ma1']+phi2*combination['ma5']+phi3*combination['ma22']
        temp=pd.concat([combination['ini'],expected],axis=1)
        error=combination['ini']-expected
        sigma=np.nanstd(error)
        density=norm.pdf(combination['ini'],expected,sigma)
        criterion=np.nansum(np.log(density))
        return -criterion

    # initial guess for parameters
    para0 = np.array([0, 0, 0, 0])

    # minimize negative log-likelihood using Nelder-Mead algorithm
    result = minimize(ML_HAR, para0, args=(x,), method='Nelder-Mead')

    # extract optimized parameter values
    para_opt = result.x

    # compute temp using optimized parameters
    expected_opt = para_opt[0] + para_opt[1]*x.shift(1).rolling(5).mean().
↪shift(1) + para_opt[2]*x.shift(1).rolling(22).mean().shift(1) +␣
↪para_opt[3]*x.rolling(22).mean().shift(1)
    fitted_values = pd.concat([x, expected_opt], axis=1)

    return fitted_values
```

[155]:
```
fited_values_vix = compute_fitted_values(df['VIX']).iloc[:,1:].dropna().copy()
fited_values_Parkinson = compute_fitted_values(Parkinson_ann).iloc[:,1:].
↪dropna().copy()
fited_values_squared_returns = compute_fitted_values(ann_vol).iloc[:,1:].
↪dropna().copy()
```

20 Les 3 séries sont stationnaires

[144]:
```
dickey(fited_values_vix , 'VIX')
dickey(fited_values_Parkinson , 'sq')
dickey(fited_values_squared_returns , 'park')
```

```
ADF Statistic: -4.832686
p-value: 0.000047
La série VIX est stationnaire.
ADF Statistic: -6.599891
p-value: 0.000000
```

```
La série sq est stationnaire.
ADF Statistic: -6.128445
p-value: 0.000000
La série park est stationnaire.
```

21 6. Estimate a VAR(p) model on your three stationnary fitted values. Comment your results. (4 points)

[156]:
```python
df = pd.concat([fited_values_vix, fited_values_squared_returns,␣
↪fited_values_Parkinson], axis=1)
df
```

[156]:
```
                  VIX  Squared returns  Parkinson
2000-02-07  23.574509        21.087282  15.263072
2000-02-08  22.734518        19.669970  14.729918
2000-02-09  22.332668        18.585620  15.272047
2000-02-10  21.930390        18.179676  14.760379
2000-02-11  21.896724        17.953817  13.653463
…                 …              …          …
2023-02-08  18.623623        14.594401  10.711680
2023-02-09  18.481210        14.433074  10.800317
2023-02-10  18.788152        14.139797  11.159421
2023-02-13  19.140071        14.446230  10.944343
2023-02-14  19.532546        14.256988  11.030729

[5799 rows x 3 columns]
```

[157]:
```python
import statsmodels.api as sm
import pandas as pd
import numpy as np
from statsmodels.tsa.vector_ar.var_model import VAR
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf

def var_model(df, maxlags):
    """
    Estimate a VAR model on a stationary fitted values DataFrame and select the␣
↪optimal number of lags using the AIC.

    Parameters:
    df (pandas.DataFrame): DataFrame of stationary fitted values for each␣
↪series.
    maxlags (int): Maximum number of lags to consider in the VAR model.

    Returns:
```

```
    results (statsmodels.tsa.vector_ar.var_model.VARResultsWrapper): Results of␣
↪the fitted VAR model.
    """

    # Select the optimal number of lags using the AIC
    aic_values = []
    for lag in range(1, maxlags+1):
        model = VAR(df)
        results = model.fit(lag)
        aic_values.append(results.aic)
    best_lag = np.argmin(aic_values) + 1

    # Fit the VAR model with the optimal number of lags
    model = VAR(df)
    results = model.fit(best_lag)

    # Print the summary of the fitted model
    print(results.summary())

    # Return the results of the fitted model
    return results
```

22 On choisit 67 lags selon les critères d'information mais on aurait pu choisir moins pour avoir un modèle plus parcimonieux

[159]:
```
from statsmodels.tsa.api import VAR

model = VAR(df)



# Choix automatique
x = model.select_order(68)
x.summary()
```

```
c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
```

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

[158]:
```
results = var_model(df, maxlags=67)
```

```
c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
```

```
c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
```

```
c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
```

```
c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
```

```
c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
```

```
c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
```

```
c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
```

```
c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
```

```
c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.

c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.


  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Wed, 05, Apr, 2023
Time:                     23:35:08
--------------------------------------------------------------------
No. of Equations:         3.00000    BIC:                   -6.93809
Nobs:                     5732.00    HQIC:                  -7.39669
Log likelihood:          -1893.39    FPE:                0.000480127
AIC:                     -7.64155    Det(Omega_mle):     0.000432745
--------------------------------------------------------------------
Results for equation VIX
==============================================================================
======
                         coefficient       std. error           t-stat
prob
------------------------------------------------------------------------------
------
const                       0.061473         0.015465            3.975
0.000
L1.VIX                      1.762710         0.014397          122.434
0.000
```

L1.Squared returns	0.118457	0.017331	6.835	0.000
L1.Parkinson	0.235215	0.015567	15.109	0.000
L2.VIX	-0.689570	0.028129	-24.514	0.000
L2.Squared returns	-0.160097	0.028355	-5.646	0.000
L2.Parkinson	-0.358426	0.028001	-12.800	0.000
L3.VIX	-0.035259	0.029599	-1.191	0.234
L3.Squared returns	0.038912	0.028848	1.349	0.177
L3.Parkinson	0.254990	0.035295	7.225	0.000
L4.VIX	-0.097564	0.029579	-3.298	0.001
L4.Squared returns	0.000081	0.028771	0.003	0.998
L4.Parkinson	-0.187782	0.038023	-4.939	0.000
L5.VIX	-0.806052	0.029589	-27.241	0.000
L5.Squared returns	-0.006555	0.028668	-0.229	0.819
L5.Parkinson	0.098408	0.038629	2.548	0.011
L6.VIX	1.536591	0.031825	48.282	0.000
L6.Squared returns	0.072855	0.030395	2.397	0.017
L6.Parkinson	-0.084245	0.038462	-2.190	0.028
L7.VIX	-0.571632	0.038151	-14.983	0.000
L7.Squared returns	-0.117171	0.031940	-3.668	0.000
L7.Parkinson	0.005441	0.038477	0.141	0.888
L8.VIX	-0.097714	0.038865	-2.514	0.012
L8.Squared returns	0.074113	0.031927	2.321	0.020
L8.Parkinson	0.145037	0.038908	3.728	0.000
L9.VIX	0.016357	0.038850	0.421	0.674

L9.Squared returns	-0.028235	0.031907	-0.885	0.376
L9.Parkinson	-0.187835	0.039695	-4.732	0.000
L10.VIX	-0.820208	0.038754	-21.165	0.000
L10.Squared returns	-0.014797	0.031975	-0.463	0.644
L10.Parkinson	0.129107	0.040368	3.198	0.001
L11.VIX	1.396625	0.040508	34.478	0.000
L11.Squared returns	0.085864	0.032386	2.651	0.008
L11.Parkinson	-0.088093	0.040749	-2.162	0.031
L12.VIX	-0.492027	0.044850	-10.970	0.000
L12.Squared returns	-0.097022	0.032751	-2.962	0.003
L12.Parkinson	0.035850	0.041008	0.874	0.382
L13.VIX	-0.114013	0.045219	-2.521	0.012
L13.Squared returns	0.010251	0.032768	0.313	0.754
L13.Parkinson	0.043684	0.041142	1.062	0.288
L14.VIX	0.043531	0.045194	0.963	0.335
L14.Squared returns	0.046604	0.032715	1.425	0.154
L14.Parkinson	-0.077499	0.041234	-1.880	0.060
L15.VIX	-0.777540	0.045116	-17.234	0.000
L15.Squared returns	-0.084087	0.032799	-2.564	0.010
L15.Parkinson	0.072906	0.041445	1.759	0.079
L16.VIX	1.334258	0.046403	28.754	0.000
L16.Squared returns	0.063570	0.032997	1.927	0.054
L16.Parkinson	-0.087269	0.041805	-2.088	0.037
L17.VIX	-0.504051	0.049575	-10.167	0.000

L17.Squared returns	0.018531	0.032978	0.562	0.574
L17.Parkinson	0.143117	0.042305	3.383	0.001
L18.VIX	-0.124602	0.049732	-2.505	0.012
L18.Squared returns	-0.076299	0.032824	-2.324	0.020
L18.Parkinson	-0.099677	0.042644	-2.337	0.019
L19.VIX	0.088224	0.049640	1.777	0.076
L19.Squared returns	0.079036	0.032754	2.413	0.016
L19.Parkinson	0.017347	0.042780	0.405	0.685
L20.VIX	-0.744805	0.049537	-15.035	0.000
L20.Squared returns	-0.060375	0.032793	-1.841	0.066
L20.Parkinson	-0.003724	0.042853	-0.087	0.931
L21.VIX	1.201324	0.050604	23.740	0.000
L21.Squared returns	0.052647	0.032810	1.605	0.109
L21.Parkinson	-0.030446	0.042926	-0.709	0.478
L22.VIX	-0.427948	0.052806	-8.104	0.000
L22.Squared returns	0.035503	0.032932	1.078	0.281
L22.Parkinson	0.035058	0.042947	0.816	0.414
L23.VIX	-0.109918	0.052795	-2.082	0.037
L23.Squared returns	0.006086	0.033046	0.184	0.854
L23.Parkinson	0.134995	0.043032	3.137	0.002
L24.VIX	0.128525	0.052769	2.436	0.015
L24.Squared returns	-0.111738	0.034079	-3.279	0.001
L24.Parkinson	-0.170525	0.043611	-3.910	0.000
L25.VIX	-0.763177	0.052741	-14.470	0.000

L25.Squared returns	0.061670	0.034061	1.811	0.070
L25.Parkinson	0.081669	0.044120	1.851	0.064
L26.VIX	1.071400	0.053725	19.942	0.000
L26.Squared returns	0.047654	0.033783	1.411	0.158
L26.Parkinson	-0.050243	0.044049	-1.141	0.254
L27.VIX	-0.291041	0.055001	-5.292	0.000
L27.Squared returns	-0.044138	0.033461	-1.319	0.187
L27.Parkinson	-0.030064	0.043597	-0.690	0.490
L28.VIX	-0.183318	0.054779	-3.346	0.001
L28.Squared returns	0.019570	0.033781	0.579	0.562
L28.Parkinson	0.044600	0.042418	1.051	0.293
L29.VIX	0.165258	0.054794	3.016	0.003
L29.Squared returns	-0.060656	0.034390	-1.764	0.078
L29.Parkinson	-0.045930	0.042263	-1.087	0.277
L30.VIX	-0.661808	0.054773	-12.083	0.000
L30.Squared returns	0.006302	0.033978	0.185	0.853
L30.Parkinson	0.066208	0.042459	1.559	0.119
L31.VIX	0.913977	0.055465	16.478	0.000
L31.Squared returns	0.028092	0.033431	0.840	0.401
L31.Parkinson	-0.062016	0.042655	-1.454	0.146
L32.VIX	-0.226564	0.055983	-4.047	0.000
L32.Squared returns	0.005853	0.033271	0.176	0.860
L32.Parkinson	0.022386	0.042520	0.526	0.599
L33.VIX	-0.223440	0.055605	-4.018	0.000

L33.Squared returns	0.034392	0.033603	1.023
0.306			
L33.Parkinson	0.002350	0.042336	0.056
0.956			
L34.VIX	0.179415	0.055633	3.225
0.001			
L34.Squared returns	-0.105876	0.033881	-3.125
0.002			
L34.Parkinson	-0.012854	0.042149	-0.305
0.760			
L35.VIX	-0.536982	0.055576	-9.662
0.000			
L35.Squared returns	0.039356	0.033584	1.172
0.241			
L35.Parkinson	0.102814	0.041959	2.450
0.014			
L36.VIX	0.777295	0.055970	13.888
0.000			
L36.Squared returns	-0.007389	0.033319	-0.222
0.824			
L36.Parkinson	-0.124874	0.042045	-2.970
0.003			
L37.VIX	-0.219467	0.055548	-3.951
0.000			
L37.Squared returns	0.023985	0.033293	0.720
0.471			
L37.Parkinson	0.101207	0.042380	2.388
0.017			
L38.VIX	-0.172213	0.054869	-3.139
0.002			
L38.Squared returns	0.034212	0.033538	1.020
0.308			
L38.Parkinson	-0.134660	0.042597	-3.161
0.002			
L39.VIX	0.148850	0.054872	2.713
0.007			
L39.Squared returns	-0.049878	0.033735	-1.479
0.139			
L39.Parkinson	0.154494	0.042523	3.633
0.000			
L40.VIX	-0.489742	0.054853	-8.928
0.000			
L40.Squared returns	-0.044362	0.033217	-1.335
0.182			
L40.Parkinson	-0.089778	0.042439	-2.115
0.034			
L41.VIX	0.657141	0.055122	11.922
0.000			

L41.Squared returns	0.046767	0.032872	1.423	0.155
L41.Parkinson	0.003151	0.042605	0.074	0.941
L42.VIX	-0.138920	0.053922	-2.576	0.010
L42.Squared returns	0.001487	0.032949	0.045	0.964
L42.Parkinson	0.070146	0.042798	1.639	0.101
L43.VIX	-0.188297	0.052863	-3.562	0.000
L43.Squared returns	0.029849	0.033084	0.902	0.367
L43.Parkinson	-0.118559	0.042875	-2.765	0.006
L44.VIX	0.158367	0.052886	2.994	0.003
L44.Squared returns	0.038564	0.032923	1.171	0.241
L44.Parkinson	0.058565	0.042613	1.374	0.169
L45.VIX	-0.398531	0.052893	-7.535	0.000
L45.Squared returns	-0.072656	0.032113	-2.263	0.024
L45.Parkinson	0.065468	0.041128	1.592	0.111
L46.VIX	0.496798	0.052906	9.390	0.000
L46.Squared returns	-0.023796	0.032051	-0.742	0.458
L46.Parkinson	-0.047234	0.040635	-1.162	0.245
L47.VIX	-0.058727	0.050749	-1.157	0.247
L47.Squared returns	0.091500	0.032043	2.856	0.004
L47.Parkinson	-0.017970	0.040631	-0.442	0.658
L48.VIX	-0.257931	0.049546	-5.206	0.000
L48.Squared returns	-0.056632	0.032041	-1.767	0.077
L48.Parkinson	-0.029397	0.040524	-0.725	0.468
L49.VIX	0.254848	0.049649	5.133	0.000

L49.Squared returns	0.053445	0.031988	1.671	0.095
L49.Parkinson	0.059115	0.040255	1.468	0.142
L50.VIX	-0.396930	0.049760	-7.977	0.000
L50.Squared returns	-0.098439	0.031939	-3.082	0.002
L50.Parkinson	-0.083439	0.039867	-2.093	0.036
L51.VIX	0.434357	0.049621	8.754	0.000
L51.Squared returns	0.045700	0.031990	1.429	0.153
L51.Parkinson	0.098447	0.039708	2.479	0.013
L52.VIX	-0.036312	0.046488	-0.781	0.435
L52.Squared returns	0.007743	0.031851	0.243	0.808
L52.Parkinson	-0.076125	0.038814	-1.961	0.050
L53.VIX	-0.224716	0.044894	-5.005	0.000
L53.Squared returns	0.018189	0.031610	0.575	0.565
L53.Parkinson	0.048966	0.038250	1.280	0.200
L54.VIX	0.174081	0.044980	3.870	0.000
L54.Squared returns	0.027779	0.031565	0.880	0.379
L54.Parkinson	-0.076759	0.037731	-2.034	0.042
L55.VIX	-0.257373	0.044997	-5.720	0.000
L55.Squared returns	-0.090386	0.031626	-2.858	0.004
L55.Parkinson	0.047462	0.037387	1.269	0.204
L56.VIX	0.265108	0.044661	5.936	0.000
L56.Squared returns	0.025953	0.031625	0.821	0.412
L56.Parkinson	0.044323	0.037178	1.192	0.233
L57.VIX	0.054788	0.040579	1.350	0.177

L57.Squared returns	0.016212	0.031090	0.521	0.602
L57.Parkinson	-0.044608	0.037119	-1.202	0.229
L58.VIX	-0.208814	0.038742	-5.390	0.000
L58.Squared returns	-0.020853	0.030422	-0.685	0.493
L58.Parkinson	0.029761	0.037020	0.804	0.421
L59.VIX	0.110880	0.038864	2.853	0.004
L59.Squared returns	0.057211	0.030244	1.892	0.059
L59.Parkinson	-0.022830	0.036459	-0.626	0.531
L60.VIX	-0.132503	0.038874	-3.409	0.001
L60.Squared returns	-0.039131	0.030215	-1.295	0.195
L60.Parkinson	0.004449	0.035602	0.125	0.901
L61.VIX	0.161311	0.038316	4.210	0.000
L61.Squared returns	-0.050238	0.030186	-1.664	0.096
L61.Parkinson	0.034883	0.034650	1.007	0.314
L62.VIX	-0.018419	0.032159	-0.573	0.567
L62.Squared returns	0.039768	0.028645	1.388	0.165
L62.Parkinson	0.001561	0.033729	0.046	0.963
L63.VIX	-0.090433	0.029729	-3.042	0.002
L63.Squared returns	0.039684	0.026902	1.475	0.140
L63.Parkinson	-0.099442	0.033547	-2.964	0.003
L64.VIX	0.084917	0.029737	2.856	0.004
L64.Squared returns	-0.047964	0.026530	-1.808	0.071
L64.Parkinson	0.104306	0.033482	3.115	0.002
L65.VIX	-0.100061	0.029753	-3.363	0.001

```
L65.Squared returns         0.008801         0.025866             0.340
0.734
L65.Parkinson              -0.047436         0.032895            -1.442
0.149
L66.VIX                     0.120048         0.028548             4.205
0.000
L66.Squared returns         0.038612         0.024747             1.560
0.119
L66.Parkinson              -0.003780         0.030244            -0.125
0.901
L67.VIX                    -0.055142         0.014821            -3.720
0.000
L67.Squared returns        -0.027542         0.015495            -1.777
0.075
L67.Parkinson               0.010252         0.016803             0.610
0.542
================================================================================
======

Results for equation Squared returns
================================================================================
======
                         coefficient       std. error           t-stat
prob
--------------------------------------------------------------------------------
------
const                       0.007188         0.012956             0.555
0.579
L1.VIX                      0.233361         0.012061            19.348
0.000
L1.Squared returns          1.334269         0.014519            91.896
0.000
L1.Parkinson                0.740252         0.013042            56.761
0.000
L2.VIX                     -0.264170         0.023565           -11.210
0.000
L2.Squared returns         -0.328590         0.023755           -13.833
0.000
L2.Parkinson               -1.291031         0.023458           -55.036
0.000
L3.VIX                      0.037785         0.024796             1.524
0.128
L3.Squared returns         -0.035853         0.024167            -1.484
0.138
L3.Parkinson                0.889368         0.029568            30.078
0.000
L4.VIX                     -0.025686         0.024780            -1.037
0.300
```

L4.Squared returns	0.149147	0.024103	6.188	0.000
L4.Parkinson	-0.444332	0.031854	-13.949	0.000
L5.VIX	-0.000593	0.024789	-0.024	0.981
L5.Squared returns	-0.595179	0.024017	-24.782	0.000
L5.Parkinson	0.200977	0.032362	6.210	0.000
L6.VIX	0.189330	0.026662	7.101	0.000
L6.Squared returns	0.594734	0.025464	23.356	0.000
L6.Parkinson	-0.079731	0.032221	-2.474	0.013
L7.VIX	-0.167907	0.031961	-5.253	0.000
L7.Squared returns	-0.097552	0.026758	-3.646	0.000
L7.Parkinson	-0.165881	0.032234	-5.146	0.000
L8.VIX	-0.006733	0.032559	-0.207	0.836
L8.Squared returns	-0.068716	0.026747	-2.569	0.010
L8.Parkinson	0.390814	0.032596	11.990	0.000
L9.VIX	0.002386	0.032547	0.073	0.942
L9.Squared returns	0.139878	0.026730	5.233	0.000
L9.Parkinson	-0.418610	0.033254	-12.588	0.000
L10.VIX	0.000400	0.032466	0.012	0.990
L10.Squared returns	-0.315494	0.026787	-11.778	0.000
L10.Parkinson	0.375649	0.033819	11.108	0.000
L11.VIX	0.077056	0.033935	2.271	0.023
L11.Squared returns	0.310703	0.027131	11.452	0.000
L11.Parkinson	-0.306799	0.034138	-8.987	0.000
L12.VIX	-0.044826	0.037574	-1.193	0.233

L12.Squared returns	-0.116852	0.027438	-4.259	0.000
L12.Parkinson	0.163512	0.034354	4.760	0.000
L13.VIX	-0.022092	0.037882	-0.583	0.560
L13.Squared returns	-0.030575	0.027452	-1.114	0.265
L13.Parkinson	-0.000727	0.034467	-0.021	0.983
L14.VIX	-0.002474	0.037861	-0.065	0.948
L14.Squared returns	0.128313	0.027407	4.682	0.000
L14.Parkinson	-0.150355	0.034544	-4.353	0.000
L15.VIX	-0.036733	0.037796	-0.972	0.331
L15.Squared returns	-0.170717	0.027477	-6.213	0.000
L15.Parkinson	0.248641	0.034720	7.161	0.000
L16.VIX	0.110833	0.038874	2.851	0.004
L16.Squared returns	0.104050	0.027643	3.764	0.000
L16.Parkinson	-0.272565	0.035022	-7.783	0.000
L17.VIX	-0.074426	0.041532	-1.792	0.073
L17.Squared returns	0.008144	0.027627	0.295	0.768
L17.Parkinson	0.308887	0.035441	8.716	0.000
L18.VIX	0.004325	0.041663	0.104	0.917
L18.Squared returns	-0.086165	0.027499	-3.133	0.002
L18.Parkinson	-0.227277	0.035725	-6.362	0.000
L19.VIX	0.020524	0.041586	0.494	0.622
L19.Squared returns	0.095210	0.027440	3.470	0.001
L19.Parkinson	0.049175	0.035839	1.372	0.170
L20.VIX	-0.046019	0.041499	-1.109	0.267

L20.Squared returns	-0.038915	0.027472	-1.416	0.157
L20.Parkinson	0.058282	0.035900	1.623	0.104
L21.VIX	0.095726	0.042394	2.258	0.024
L21.Squared returns	-0.113961	0.027487	-4.146	0.000
L21.Parkinson	-0.116651	0.035962	-3.244	0.001
L22.VIX	-0.092626	0.044239	-2.094	0.036
L22.Squared returns	-0.170142	0.027589	-6.167	0.000
L22.Parkinson	0.101888	0.035979	2.832	0.005
L23.VIX	-0.034342	0.044229	-0.776	0.437
L23.Squared returns	0.477234	0.027684	17.238	0.000
L23.Parkinson	0.286006	0.036050	7.934	0.000
L24.VIX	0.112358	0.044208	2.542	0.011
L24.Squared returns	-0.245399	0.028550	-8.595	0.000
L24.Parkinson	-0.442922	0.036535	-12.123	0.000
L25.VIX	-0.080118	0.044184	-1.813	0.070
L25.Squared returns	0.082416	0.028534	2.888	0.004
L25.Parkinson	0.097198	0.036961	2.630	0.009
L26.VIX	0.033488	0.045008	0.744	0.457
L26.Squared returns	0.012904	0.028302	0.456	0.648
L26.Parkinson	0.156109	0.036902	4.230	0.000
L27.VIX	0.003716	0.046077	0.081	0.936
L27.Squared returns	-0.351249	0.028032	-12.530	0.000
L27.Parkinson	-0.218663	0.036524	-5.987	0.000
L28.VIX	-0.066089	0.045892	-1.440	0.150

L28.Squared returns	0.393013	0.028300	13.887	0.000
L28.Parkinson	0.167466	0.035536	4.713	0.000
L29.VIX	0.084371	0.045904	1.838	0.066
L29.Squared returns	-0.067720	0.028810	-2.351	0.019
L29.Parkinson	-0.266954	0.035406	-7.540	0.000
L30.VIX	-0.015038	0.045886	-0.328	0.743
L30.Squared returns	-0.040209	0.028465	-1.413	0.158
L30.Parkinson	0.338594	0.035570	9.519	0.000
L31.VIX	-0.031057	0.046466	-0.668	0.504
L31.Squared returns	0.063883	0.028007	2.281	0.023
L31.Parkinson	-0.187420	0.035734	-5.245	0.000
L32.VIX	0.018663	0.046900	0.398	0.691
L32.Squared returns	-0.233540	0.027873	-8.379	0.000
L32.Parkinson	0.077520	0.035621	2.176	0.030
L33.VIX	-0.041842	0.046583	-0.898	0.369
L33.Squared returns	0.290023	0.028151	10.302	0.000
L33.Parkinson	0.003598	0.035467	0.101	0.919
L34.VIX	0.072588	0.046606	1.557	0.119
L34.Squared returns	-0.101963	0.028384	-3.592	0.000
L34.Parkinson	-0.149259	0.035310	-4.227	0.000
L35.VIX	-0.042631	0.046559	-0.916	0.360
L35.Squared returns	-0.058122	0.028135	-2.066	0.039
L35.Parkinson	0.275982	0.035151	7.851	0.000
L36.VIX	0.006920	0.046889	0.148	0.883

L36.Squared returns	0.128026	0.027913	4.587	0.000
L36.Parkinson	-0.327089	0.035224	-9.286	0.000
L37.VIX	0.026536	0.046536	0.570	0.569
L37.Squared returns	-0.236425	0.027891	-8.477	0.000
L37.Parkinson	0.310316	0.035504	8.740	0.000
L38.VIX	-0.082755	0.045967	-1.800	0.072
L38.Squared returns	0.243170	0.028096	8.655	0.000
L38.Parkinson	-0.234482	0.035686	-6.571	0.000
L39.VIX	0.135927	0.045970	2.957	0.003
L39.Squared returns	-0.055682	0.028262	-1.970	0.049
L39.Parkinson	0.159665	0.035624	4.482	0.000
L40.VIX	-0.082849	0.045953	-1.803	0.071
L40.Squared returns	-0.131353	0.027828	-4.720	0.000
L40.Parkinson	0.006085	0.035553	0.171	0.864
L41.VIX	-0.008466	0.046179	-0.183	0.855
L41.Squared returns	0.181242	0.027539	6.581	0.000
L41.Parkinson	-0.194121	0.035693	-5.439	0.000
L42.VIX	0.032979	0.045173	0.730	0.465
L42.Squared returns	-0.171182	0.027603	-6.201	0.000
L42.Parkinson	0.244851	0.035854	6.829	0.000
L43.VIX	-0.081967	0.044286	-1.851	0.064
L43.Squared returns	0.029153	0.027716	1.052	0.293
L43.Parkinson	-0.222889	0.035918	-6.205	0.000
L44.VIX	0.118216	0.044305	2.668	0.008

L44.Squared returns	0.064057	0.027581	2.322	0.020
L44.Parkinson	0.124816	0.035699	3.496	0.000
L45.VIX	-0.043277	0.044311	-0.977	0.329
L45.Squared returns	0.057425	0.026903	2.135	0.033
L45.Parkinson	0.089774	0.034455	2.606	0.009
L46.VIX	-0.036285	0.044323	-0.819	0.413
L46.Squared returns	-0.079800	0.026851	-2.972	0.003
L46.Parkinson	-0.116710	0.034042	-3.428	0.001
L47.VIX	0.051805	0.042515	1.219	0.223
L47.Squared returns	0.039712	0.026844	1.479	0.139
L47.Parkinson	-0.026426	0.034038	-0.776	0.438
L48.VIX	-0.101722	0.041507	-2.451	0.014
L48.Squared returns	-0.034923	0.026842	-1.301	0.193
L48.Parkinson	0.071985	0.033949	2.120	0.034
L49.VIX	0.136714	0.041593	3.287	0.001
L49.Squared returns	-0.093976	0.026798	-3.507	0.000
L49.Parkinson	-0.039914	0.033724	-1.184	0.237
L50.VIX	-0.052189	0.041686	-1.252	0.211
L50.Squared returns	0.100167	0.026757	3.744	0.000
L50.Parkinson	-0.054312	0.033399	-1.626	0.104
L51.VIX	-0.055545	0.041570	-1.336	0.181
L51.Squared returns	0.026594	0.026800	0.992	0.321
L51.Parkinson	0.040875	0.033265	1.229	0.219
L52.VIX	0.063224	0.038946	1.623	0.105

L52.Squared returns	0.007089	0.026683	0.266	0.790
L52.Parkinson	-0.012147	0.032517	-0.374	0.709
L53.VIX	-0.058648	0.037610	-1.559	0.119
L53.Squared returns	-0.027864	0.026481	-1.052	0.293
L53.Parkinson	0.058097	0.032044	1.813	0.070
L54.VIX	0.088868	0.037682	2.358	0.018
L54.Squared returns	-0.081966	0.026444	-3.100	0.002
L54.Parkinson	-0.052640	0.031609	-1.665	0.096
L55.VIX	-0.042334	0.037696	-1.123	0.261
L55.Squared returns	0.102752	0.026495	3.878	0.000
L55.Parkinson	0.052871	0.031321	1.688	0.091
L56.VIX	-0.028791	0.037415	-0.770	0.442
L56.Squared returns	-0.016378	0.026494	-0.618	0.536
L56.Parkinson	-0.080767	0.031146	-2.593	0.010
L57.VIX	0.049022	0.033995	1.442	0.149
L57.Squared returns	-0.033300	0.026046	-1.279	0.201
L57.Parkinson	0.118679	0.031097	3.816	0.000
L58.VIX	-0.065865	0.032456	-2.029	0.042
L58.Squared returns	0.024283	0.025486	0.953	0.341
L58.Parkinson	-0.108750	0.031014	-3.506	0.000
L59.VIX	0.070287	0.032558	2.159	0.031
L59.Squared returns	-0.042842	0.025337	-1.691	0.091
L59.Parkinson	0.068255	0.030543	2.235	0.025
L60.VIX	0.007746	0.032566	0.238	0.812

```
L60.Squared returns          0.083005         0.025312            3.279
0.001
L60.Parkinson               -0.058040         0.029826           -1.946
0.052
L61.VIX                     -0.072089         0.032099           -2.246
0.025
L61.Squared returns         -0.012994         0.025289           -0.514
0.607
L61.Parkinson                0.034547         0.029028            1.190
0.234
L62.VIX                      0.053736         0.026942            1.995
0.046
L62.Squared returns         -0.114725         0.023998           -4.781
0.000
L62.Parkinson                0.072115         0.028257            2.552
0.011
L63.VIX                     -0.047431         0.024905           -1.904
0.057
L63.Squared returns          0.161124         0.022537            7.149
0.000
L63.Parkinson               -0.148629         0.028104           -5.289
0.000
L64.VIX                      0.033663         0.024913            1.351
0.177
L64.Squared returns         -0.142666         0.022226           -6.419
0.000
L64.Parkinson                0.112239         0.028049            4.001
0.000
L65.VIX                      0.004353         0.024926            0.175
0.861
L65.Squared returns          0.039778         0.021670            1.836
0.066
L65.Parkinson               -0.045817         0.027558           -1.663
0.096
L66.VIX                     -0.015765         0.023917           -0.659
0.510
L66.Squared returns          0.124450         0.020732            6.003
0.000
L66.Parkinson               -0.037142         0.025337           -1.466
0.143
L67.VIX                      0.007574         0.012417            0.610
0.542
L67.Squared returns         -0.101460         0.012981           -7.816
0.000
L67.Parkinson                0.043114         0.014077            3.063
0.002
================================================================================
======
```

```
Results for equation Parkinson
================================================================================
======
                          coefficient       std. error           t-stat
prob
--------------------------------------------------------------------------------
------
const                       -0.010069         0.013620           -0.739
0.460
L1.VIX                       0.135579         0.012680           10.692
0.000
L1.Squared returns           0.068206         0.015264            4.468
0.000
L1.Parkinson                 1.439208         0.013710          104.971
0.000
L2.VIX                      -0.136584         0.024774           -5.513
0.000
L2.Squared returns          -0.032228         0.024973           -1.291
0.197
L2.Parkinson                -0.680274         0.024661          -27.585
0.000
L3.VIX                       0.008355         0.026068            0.321
0.749
L3.Squared returns          -0.052570         0.025407           -2.069
0.039
L3.Parkinson                 0.289276         0.031085            9.306
0.000
L4.VIX                      -0.054406         0.026051           -2.088
0.037
L4.Squared returns           0.026488         0.025339            1.045
0.296
L4.Parkinson                -0.086427         0.033488           -2.581
0.010
L5.VIX                       0.065644         0.026060            2.519
0.012
L5.Squared returns           0.100215         0.025249            3.969
0.000
L5.Parkinson                 0.013301         0.034021            0.391
0.696
L6.VIX                       0.049844         0.028029            1.778
0.075
L6.Squared returns          -0.013537         0.026770           -0.506
0.613
L6.Parkinson                 0.191261         0.033874            5.646
0.000
L7.VIX                      -0.036910         0.033600           -1.098
0.272
```

```
L7.Squared returns         -0.057803         0.028130            -2.055
0.040
L7.Parkinson               -0.422755         0.033888           -12.475
0.000
L8.VIX                     -0.015421         0.034229            -0.451
0.652
L8.Squared returns         -0.032388         0.028119            -1.152
0.249
L8.Parkinson                0.439011         0.034267            12.811
0.000
L9.VIX                     -0.071035         0.034216            -2.076
0.038
L9.Squared returns          0.056572         0.028101             2.013
0.044
L9.Parkinson               -0.350534         0.034960           -10.027
0.000
L10.VIX                     0.049630         0.034131             1.454
0.146
L10.Squared returns        -0.051555         0.028161            -1.831
0.067
L10.Parkinson               0.219837         0.035553             6.183
0.000
L11.VIX                     0.083946         0.035676             2.353
0.019
L11.Squared returns         0.074080         0.028523             2.597
0.009
L11.Parkinson              -0.050001         0.035889            -1.393
0.164
L12.VIX                    -0.091450         0.039501            -2.315
0.021
L12.Squared returns        -0.079458         0.028845            -2.755
0.006
L12.Parkinson              -0.090444         0.036116            -2.504
0.012
L13.VIX                     0.080360         0.039825             2.018
0.044
L13.Squared returns        -0.014343         0.028860            -0.497
0.619
L13.Parkinson               0.188938         0.036235             5.214
0.000
L14.VIX                    -0.096451         0.039803            -2.423
0.015
L14.Squared returns        -0.039664         0.028812            -1.377
0.169
L14.Parkinson              -0.259227         0.036315            -7.138
0.000
L15.VIX                    -0.017682         0.039734            -0.445
0.656
```

L15.Squared returns	0.128806	0.028887	4.459	0.000
L15.Parkinson	0.336573	0.036501	9.221	0.000
L16.VIX	0.138961	0.040868	3.400	0.001
L16.Squared returns	0.036659	0.029061	1.261	0.207
L16.Parkinson	-0.445615	0.036818	-12.103	0.000
L17.VIX	-0.133923	0.043662	-3.067	0.002
L17.Squared returns	-0.117763	0.029044	-4.055	0.000
L17.Parkinson	0.283202	0.037258	7.601	0.000
L18.VIX	0.075427	0.043800	1.722	0.085
L18.Squared returns	0.030414	0.028909	1.052	0.293
L18.Parkinson	0.017627	0.037557	0.469	0.639
L19.VIX	-0.028349	0.043719	-0.648	0.517
L19.Squared returns	-0.036869	0.028847	-1.278	0.201
L19.Parkinson	-0.142125	0.037678	-3.772	0.000
L20.VIX	-0.007467	0.043628	-0.171	0.864
L20.Squared returns	0.048343	0.028881	1.674	0.094
L20.Parkinson	0.168050	0.037742	4.453	0.000
L21.VIX	0.006123	0.044568	0.137	0.891
L21.Squared returns	-0.149267	0.028896	-5.166	0.000
L21.Parkinson	-0.120756	0.037806	-3.194	0.001
L22.VIX	-0.005673	0.046508	-0.122	0.903
L22.Squared returns	-0.015048	0.029004	-0.519	0.604
L22.Parkinson	-0.380156	0.037824	-10.051	0.000
L23.VIX	0.044305	0.046497	0.953	0.341

L23.Squared returns	0.246170	0.029104	8.458	0.000
L23.Parkinson	0.457916	0.037899	12.083	0.000
L24.VIX	-0.027580	0.046475	-0.593	0.553
L24.Squared returns	-0.107945	0.030014	-3.596	0.000
L24.Parkinson	-0.007304	0.038409	-0.190	0.849
L25.VIX	-0.029357	0.046450	-0.632	0.527
L25.Squared returns	-0.011409	0.029998	-0.380	0.704
L25.Parkinson	-0.122707	0.038857	-3.158	0.002
L26.VIX	0.024352	0.047316	0.515	0.607
L26.Squared returns	-0.021225	0.029754	-0.713	0.476
L26.Parkinson	0.160808	0.038794	4.145	0.000
L27.VIX	-0.030450	0.048441	-0.629	0.530
L27.Squared returns	0.130631	0.029469	4.433	0.000
L27.Parkinson	-0.105766	0.038397	-2.755	0.006
L28.VIX	0.062904	0.048245	1.304	0.192
L28.Squared returns	-0.023552	0.029751	-0.792	0.429
L28.Parkinson	0.178973	0.037358	4.791	0.000
L29.VIX	-0.042696	0.048258	-0.885	0.376
L29.Squared returns	-0.083505	0.030288	-2.757	0.006
L29.Parkinson	-0.272933	0.037222	-7.333	0.000
L30.VIX	0.035010	0.048240	0.726	0.468
L30.Squared returns	0.022289	0.029925	0.745	0.456
L30.Parkinson	0.193805	0.037395	5.183	0.000
L31.VIX	-0.045739	0.048849	-0.936	0.349

L31.Squared returns	-0.061248	0.029443	-2.080	0.038
L31.Parkinson	-0.084281	0.037567	-2.243	0.025
L32.VIX	0.001429	0.049306	0.029	0.977
L32.Squared returns	0.153143	0.029302	5.226	0.000
L32.Parkinson	-0.010230	0.037448	-0.273	0.785
L33.VIX	0.023521	0.048972	0.480	0.631
L33.Squared returns	-0.011844	0.029595	-0.400	0.689
L33.Parkinson	0.058256	0.037286	1.562	0.118
L34.VIX	-0.030255	0.048997	-0.617	0.537
L34.Squared returns	-0.092063	0.029839	-3.085	0.002
L34.Parkinson	-0.140864	0.037121	-3.795	0.000
L35.VIX	0.098274	0.048947	2.008	0.045
L35.Squared returns	0.001160	0.029578	0.039	0.969
L35.Parkinson	0.230210	0.036954	6.230	0.000
L36.VIX	-0.123149	0.049293	-2.498	0.012
L36.Squared returns	-0.027104	0.029345	-0.924	0.356
L36.Parkinson	-0.241548	0.037030	-6.523	0.000
L37.VIX	0.026780	0.048922	0.547	0.584
L37.Squared returns	0.079173	0.029322	2.700	0.007
L37.Parkinson	0.238215	0.037325	6.382	0.000
L38.VIX	0.026749	0.048324	0.554	0.580
L38.Squared returns	0.078472	0.029537	2.657	0.008
L38.Parkinson	-0.237207	0.037516	-6.323	0.000
L39.VIX	-0.015523	0.048327	-0.321	0.748

L39.Squared returns	-0.087372	0.029711	-2.941	0.003
L39.Parkinson	-0.011551	0.037451	-0.308	0.758
L40.VIX	0.071111	0.048310	1.472	0.141
L40.Squared returns	-0.072176	0.029255	-2.467	0.014
L40.Parkinson	0.260663	0.037377	6.974	0.000
L41.VIX	-0.127759	0.048547	-2.632	0.008
L41.Squared returns	0.020668	0.028951	0.714	0.475
L41.Parkinson	-0.246929	0.037523	-6.581	0.000
L42.VIX	0.039317	0.047490	0.828	0.408
L42.Squared returns	0.027063	0.029019	0.933	0.351
L42.Parkinson	0.187019	0.037693	4.962	0.000
L43.VIX	0.043981	0.046558	0.945	0.345
L43.Squared returns	-0.092134	0.029138	-3.162	0.002
L43.Parkinson	-0.101034	0.037761	-2.676	0.007
L44.VIX	-0.055466	0.046578	-1.191	0.234
L44.Squared returns	-0.041912	0.028996	-1.445	0.148
L44.Parkinson	-0.118314	0.037530	-3.153	0.002
L45.VIX	0.133758	0.046584	2.871	0.004
L45.Squared returns	0.187032	0.028282	6.613	0.000
L45.Parkinson	0.134878	0.036222	3.724	0.000
L46.VIX	-0.183776	0.046596	-3.944	0.000
L46.Squared returns	-0.068012	0.028228	-2.409	0.016
L46.Parkinson	0.015770	0.035788	0.441	0.659
L47.VIX	0.057393	0.044695	1.284	0.199

L47.Squared returns	0.014171	0.028221	0.502	0.616
L47.Parkinson	-0.013697	0.035784	-0.383	0.702
L48.VIX	0.061885	0.043636	1.418	0.156
L48.Squared returns	-0.081287	0.028219	-2.881	0.004
L48.Parkinson	0.026421	0.035690	0.740	0.459
L49.VIX	-0.069550	0.043726	-1.591	0.112
L49.Squared returns	0.116049	0.028172	4.119	0.000
L49.Parkinson	0.001737	0.035454	0.049	0.961
L50.VIX	0.136827	0.043824	3.122	0.002
L50.Squared returns	0.009474	0.028129	0.337	0.736
L50.Parkinson	-0.001592	0.035112	-0.045	0.964
L51.VIX	-0.179210	0.043702	-4.101	0.000
L51.Squared returns	-0.062122	0.028174	-2.205	0.027
L51.Parkinson	-0.026525	0.034971	-0.758	0.448
L52.VIX	0.035197	0.040943	0.860	0.390
L52.Squared returns	0.042204	0.028051	1.505	0.132
L52.Parkinson	-0.013482	0.034184	-0.394	0.693
L53.VIX	0.075878	0.039539	1.919	0.055
L53.Squared returns	-0.100222	0.027839	-3.600	0.000
L53.Parkinson	0.014271	0.033687	0.424	0.672
L54.VIX	-0.043695	0.039615	-1.103	0.270
L54.Squared returns	0.124020	0.027800	4.461	0.000
L54.Parkinson	-0.002907	0.033231	-0.087	0.930
L55.VIX	0.059636	0.039630	1.505	0.132

L55.Squared returns	-0.002916	0.027853	-0.105	0.917
L55.Parkinson	-0.018018	0.032927	-0.547	0.584
L56.VIX	-0.065448	0.039333	-1.664	0.096
L56.Squared returns	-0.048750	0.027853	-1.750	0.080
L56.Parkinson	-0.010754	0.032743	-0.328	0.743
L57.VIX	-0.028510	0.035739	-0.798	0.425
L57.Squared returns	0.021004	0.027382	0.767	0.443
L57.Parkinson	0.049830	0.032691	1.524	0.127
L58.VIX	0.083786	0.034121	2.456	0.014
L58.Squared returns	-0.059438	0.026793	-2.218	0.027
L58.Parkinson	-0.039254	0.032605	-1.204	0.229
L59.VIX	-0.053770	0.034228	-1.571	0.116
L59.Squared returns	0.079088	0.026637	2.969	0.003
L59.Parkinson	0.056390	0.032110	1.756	0.079
L60.VIX	0.064147	0.034237	1.874	0.061
L60.Squared returns	0.017725	0.026611	0.666	0.505
L60.Parkinson	-0.104102	0.031356	-3.320	0.001
L61.VIX	-0.078278	0.033745	-2.320	0.020
L61.Squared returns	-0.006244	0.026585	-0.235	0.814
L61.Parkinson	-0.021334	0.030517	-0.699	0.484
L62.VIX	0.011922	0.028323	0.421	0.674
L62.Squared returns	-0.071620	0.025228	-2.839	0.005
L62.Parkinson	0.157125	0.029706	5.289	0.000
L63.VIX	0.024379	0.026183	0.931	0.352

```
L63.Squared returns        -0.006128         0.023693            -0.259
0.796
L63.Parkinson              -0.091015         0.029545            -3.081
0.002
L64.VIX                    -0.037213         0.026190            -1.421
0.155
L64.Squared returns         0.070683         0.023365             3.025
0.002
L64.Parkinson               0.018855         0.029488             0.639
0.523
L65.VIX                     0.068728         0.026204             2.623
0.009
L65.Squared returns        -0.096570         0.022781            -4.239
0.000
L65.Parkinson               0.021463         0.028971             0.741
0.459
L66.VIX                    -0.097031         0.025143            -3.859
0.000
L66.Squared returns         0.009087         0.021795             0.417
0.677
L66.Parkinson              -0.023683         0.026636            -0.889
0.374
L67.VIX                     0.055476         0.013053             4.250
0.000
L67.Squared returns         0.042075         0.013647             3.083
0.002
L67.Parkinson               0.003378         0.014799             0.228
0.819
================================================================================
======

Correlation matrix of residuals
                        VIX  Squared returns  Parkinson
VIX                1.000000         0.356134   0.134965
Squared returns    0.356134         1.000000   0.208449
Parkinson          0.134965         0.208449   1.000000
```

23 7. Plot an impulse response function obtained from your VAR model and comment the results (4 points)

[148]:
```python
lag_order = results.k_ar
print(lag_order)
```

67

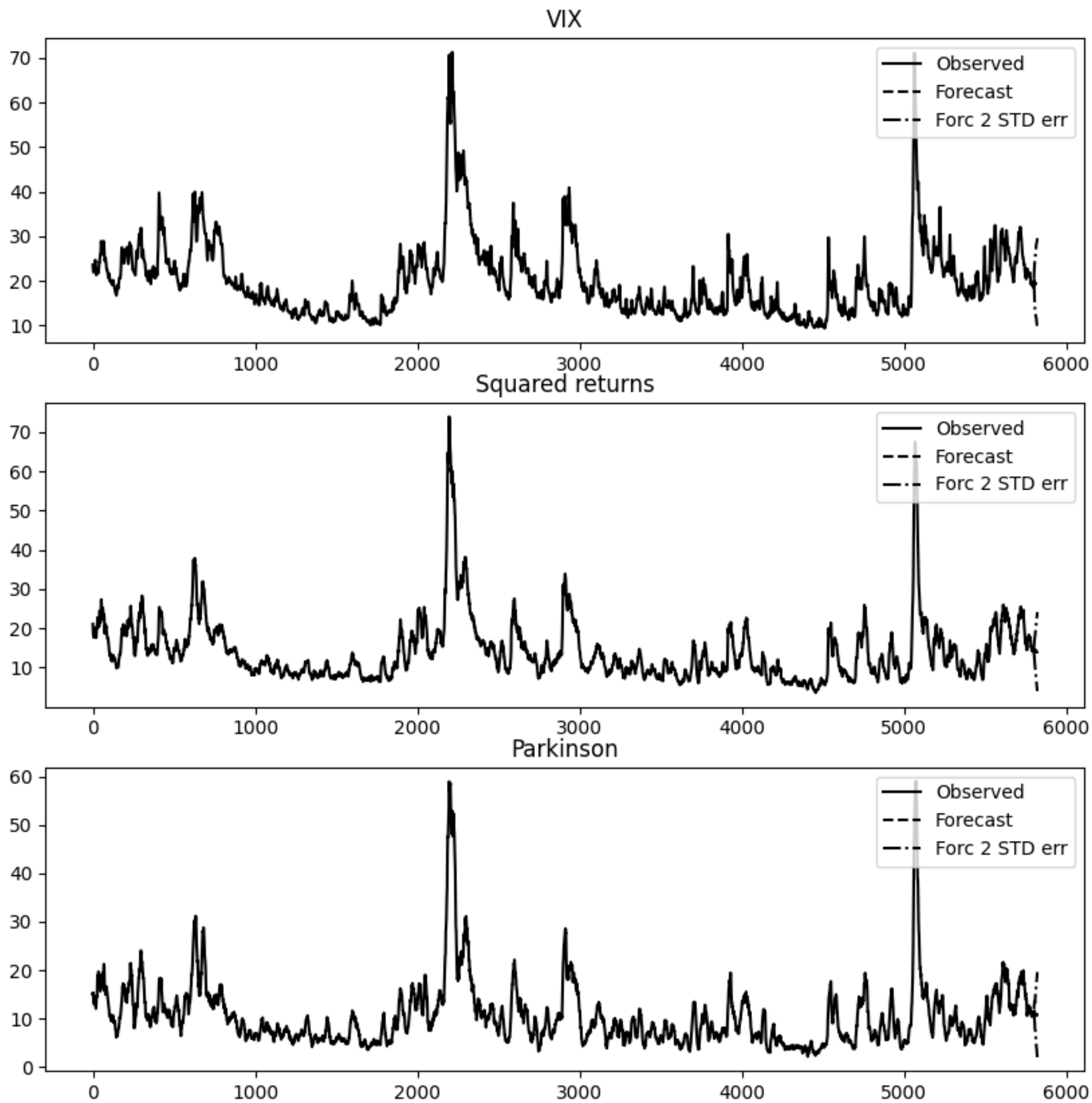
24 Il n y a pas d'autocorrélation dans les résidus pour les 3 séries

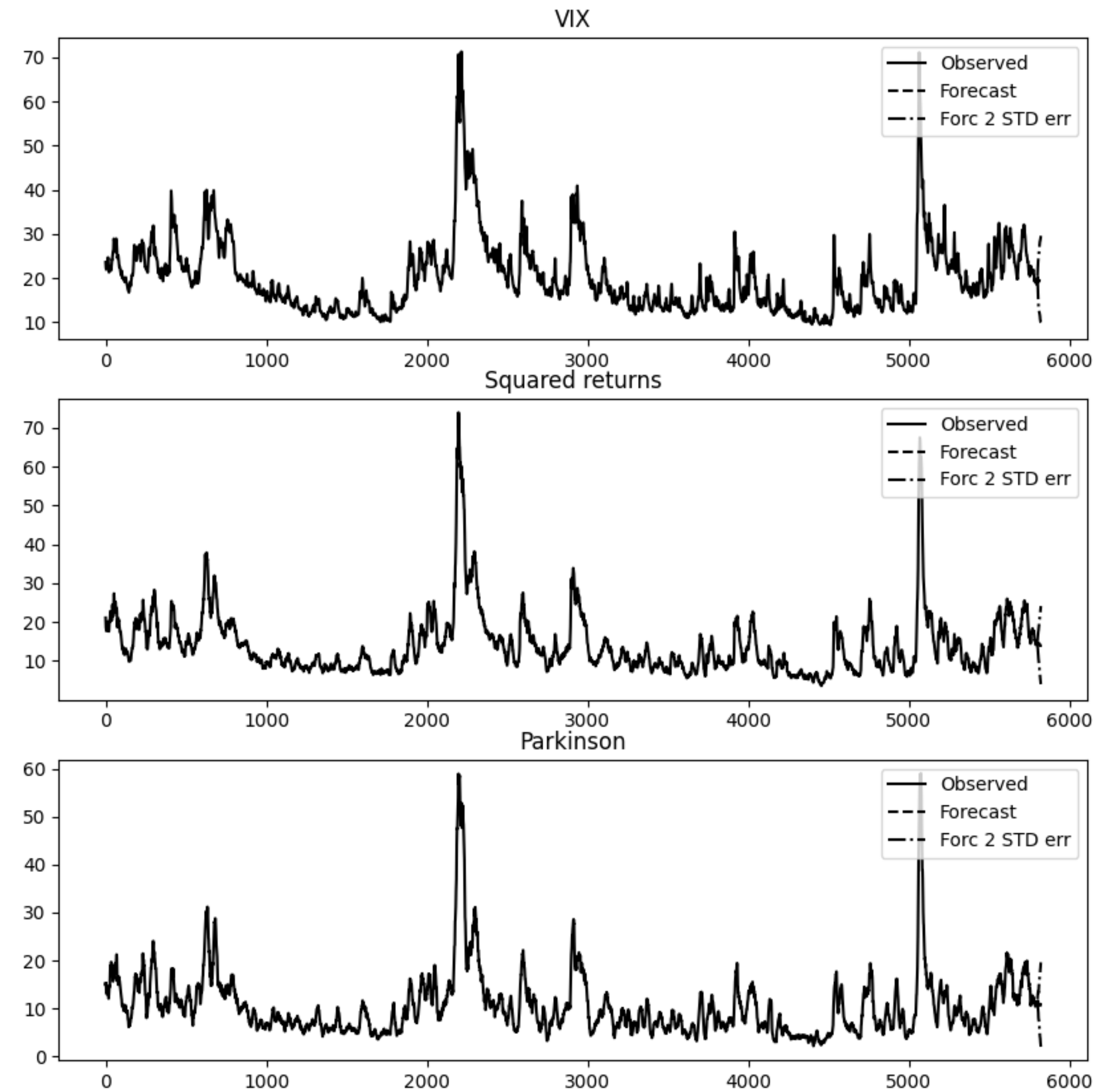
[149]:
```
from statsmodels.stats.stattools import durbin_watson


out = durbin_watson(results.resid)

for col, val in zip(df.columns, out):
    print((col), ':', round(val, 2))
```

```
VIX : 2.0
Squared returns : 2.0
Parkinson : 2.0
```

[160]:
```
lag_order = 67
model_fitted = model.fit(67)
forecast_horiz = 22
model_fitted.forecast(df.values[-lag_order:], forecast_horiz)
model_fitted.plot_forecast(forecast_horiz)
```

[160]:


VIX
Observed
Forecast
Forc 2 STD err
Squared returns
Observed
Forecast
Forc 2 STD err
Parkinson
Observed
Forecast
Forc 2 STD err



[161]:
```
from statsmodels.tsa.stattools import grangercausalitytests
import pandas as pd
import numpy as np

def grangers_test(data, maxlag, variables, test='ssr_chi2test',verbose=False):
    """Les valeurs dans le df sont les p-valeurs
    L'hypothèse H0 de notre test est la suivant :
        "Les prédictions de la série X n'influence pas les prédictions de la␣
 ↪série Y"
    Ce qui signifie qu'une p-valeur inférieure à 0.05 rejette l'hypothèse H0 et␣
 ↪incite à garder ce couple de valeurs
```

```
    Comme on s'intéresse à la prédiciton de la variable 1, on ne va jamais␣
 ↪l'abandonner
Les arguments sont :
    Data, le DF de nos valeurs
    maxlag, le fameux maxlag pour le nombre de paramètres dans l'équation'
    variables : une list qui contient le nom des variables c'est à dire le nom␣
 ↪de nos colonnes'
    """
    df = pd.DataFrame(np.zeros((len(variables), len(variables))),␣
 ↪columns=variables, index=variables)
    for col in df.columns:
        for row in df.index:
            test_result = grangercausalitytests(data[[row, col]],␣
 ↪maxlag=maxlag, verbose=False)
            p_values = [round(test_result[i+1][0][test][1],4) for i in␣
 ↪range(maxlag)] #on va avoir toutes les p-valeurs une part lag
            min_p_value = np.min(p_values) #On s'intéresse à la valeur minimale␣
 ↪des p-valeur
            df.loc[row, col] = min_p_value
    df.columns = [var + '_X' for var in variables]
    df.index = [var + '_Y' for var in variables]
    return df
```

[162]:
```
grangers_test(df, 67, variables = df.columns)
```

[162]:

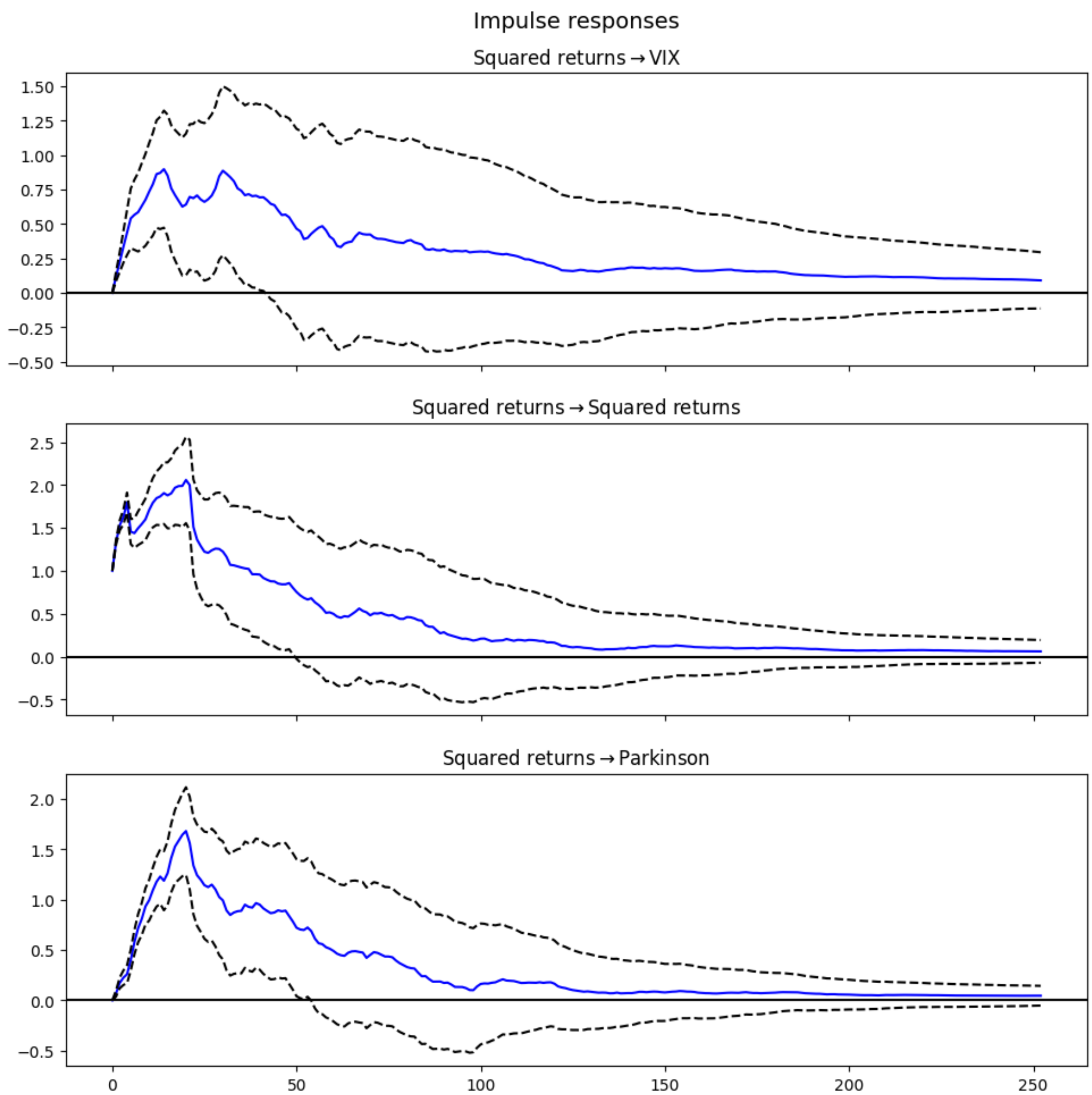
	VIX_X	Squared returns_X	Parkinson_X
VIX_Y	1.0	0.0	0.0
Squared returns_Y	0.0	1.0	0.0
Parkinson_Y	0.0	0.0	1.0

24.0.1 Interprétation de la fonction de réponse impulsionnelle

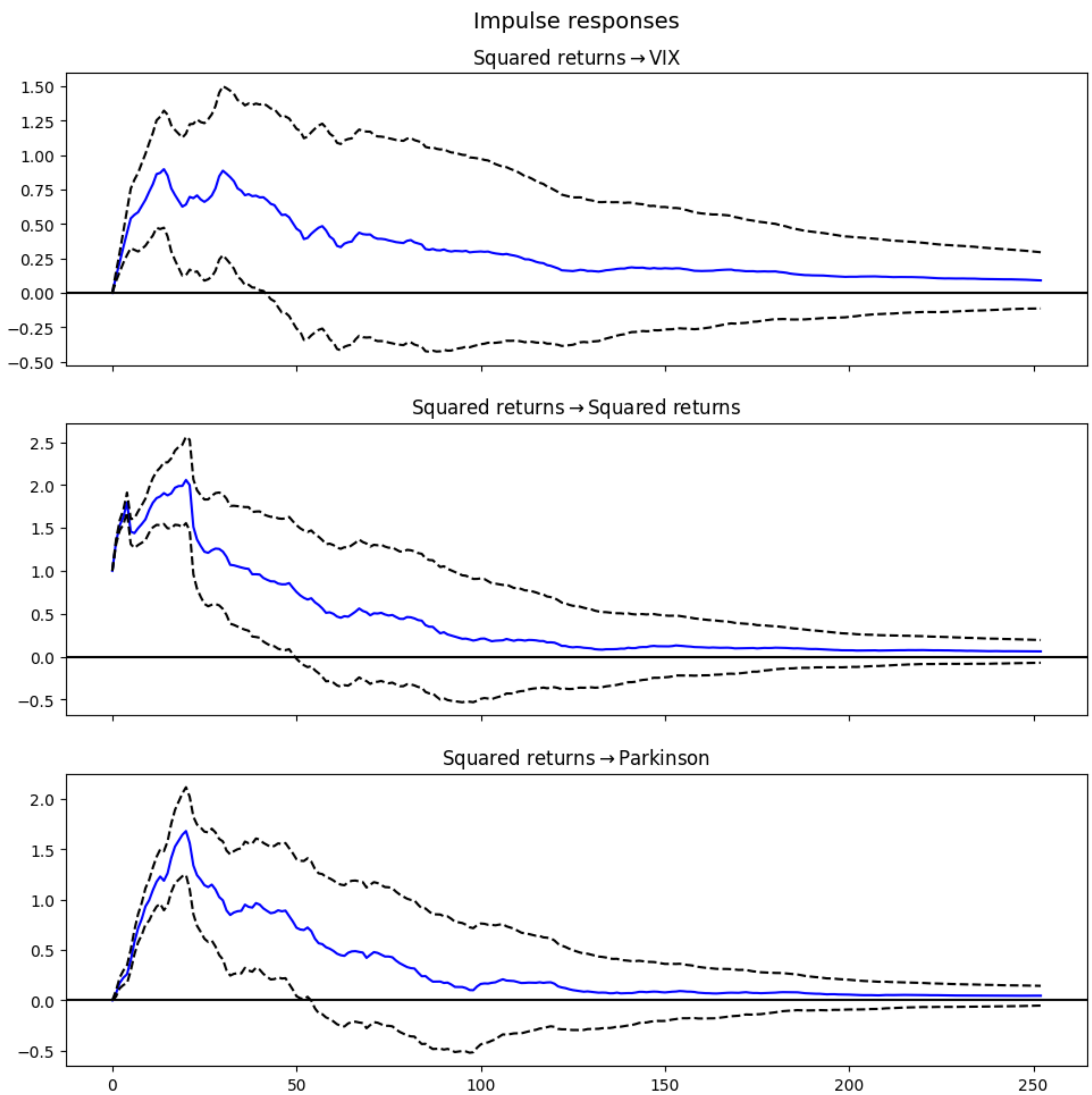
- **Premièrement on remarque que les chocs se resorbent et tendent à revenir à 0, ce qui montre que les séries sont bien stationnaires**
- **On remarque qu'un choque sur la variable Squared returns à un impact positif sur les deux autres séries et ces chocs se resorbent environ après 30 jours**

[176]:
```
results.irf(252).plot(impulse=1)
```

[176]:



Impulse responses
Squared returns → VIX
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
Squared returns → Squared returns
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
Squared returns → Parkinson
2.0
1.5
1.0
0.5
0.0
−0.5
0
50
100
150
200
250



24.0.2 On peut regarder la qualité de prédiction du modèle à 1, 5 et 22 jours et on obtient de plutôt bons voir très bon résultats

[165]:
```python
import plotly.graph_objs as go
from sklearn.metrics import mean_absolute_percentage_error
# Calcul de la moyenne absolue de pourcentage d'erreur
def forcast_n_days(days:int):
    train = df.iloc[:-days,:]
    test = df.iloc[-days:,:]

    model = VAR(train)
    results = model.fit(maxlags=67)
    lag_order = results.k_ar
```

```
    fcst = results.forecast(train.values[-lag_order:], days)
    model_accuracy = 1 - mean_absolute_percentage_error(test, fcst)

    mape = mean_absolute_percentage_error(test, fcst)
    model_accuracy = 1 - mape
    print(model_accuracy)

    # Création des traces de données pour les prévisions et les vraies valeurs
    true_values_trace = go.Scatter(x=test.index, y=test.values[:, 0],␣
↪name='True VIX')
    predictions_trace = go.Scatter(x=test.index, y=fcst[:, 0], name='Pred VIX')

    true_values_trace2 = go.Scatter(x=test.index, y=test.values[:, 1],␣
↪name='True SQ')
    predictions_trace2 = go.Scatter(x=test.index, y=fcst[:, 1], name='Pred SQ')

    true_values_trace3 = go.Scatter(x=test.index, y=test.values[:, 2],␣
↪name='True Parkinson')
    predictions_trace3 = go.Scatter(x=test.index, y=fcst[:, 2], name='Pred␣
↪Parkinson')

    fig = go.Figure(data=[true_values_trace, predictions_trace,
                          true_values_trace2, predictions_trace2,
                          true_values_trace3, predictions_trace3])

    fig.update_layout(
        xaxis_title='Time',
        yaxis_title='Values',
        title=f'MAPE: {np.round(mape*100, 2)}% - Model Accuracy: {np.
↪round(model_accuracy*100, 2)}%',
        legend=dict(
            yanchor="top",
            y=0.99,
            xanchor="left",
            x=0.01
        )
    )

    # Affichage de la figure
    fig.show()
```

[167]:
```
forcast_n_days(1)
```

```
c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
```

```
and so will be ignored when e.g. forecasting.
```

```
0.991224283001845
```

[168]:
```
forcast_n_days(5)
```

```
0.9831683078914287
```

```
c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
```

[166]:
```
forcast_n_days(22)
```

```
c:\Users\Zbook Create G7\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
```

```
0.9624599858016719
```