Prática4_respostas

August 5, 2020

1 Prática 4

Aprendizado Dinâmico

por **Cibele Russo** (ICMC/USP - São Carlos SP)

MBA em Ciências de Dados

Considere a base de dados de mortes por COVID-19 no estado de SP.

Nesta prática, vamos trabalhar com a análise de má-especificação de modelos, ou seja, como fica a análise de diagnóstico se ajustarmos um modelo "correto" e um modelo "errado".

1. Carregue as bibliotecas que serão utilizadas.

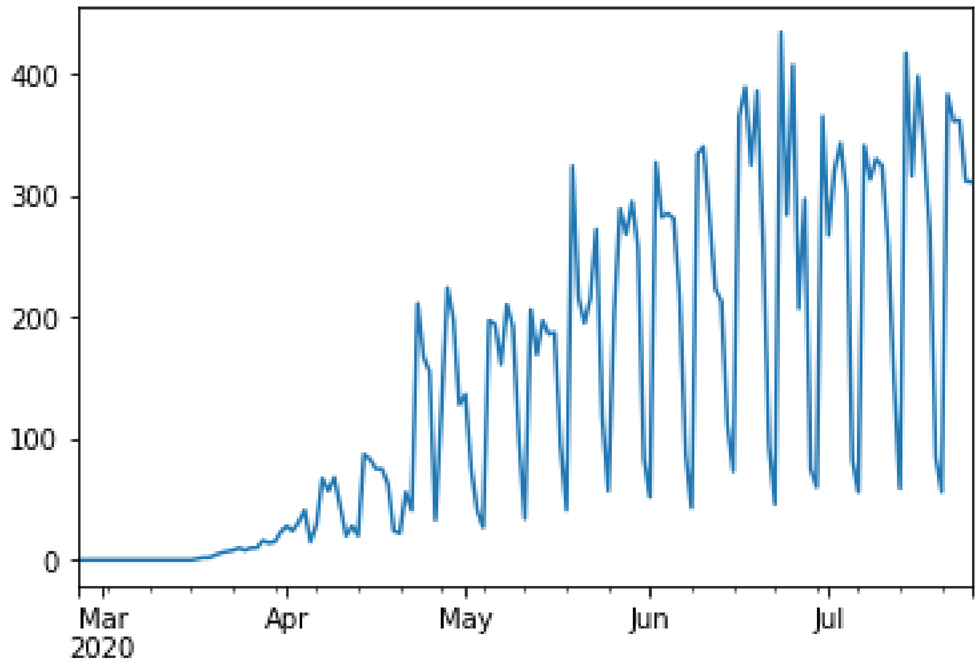
[1]:
```
import pandas as pd
import numpy as np
%matplotlib inline

from statsmodels.tsa.statespace.sarimax import SARIMAX

from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
from statsmodels.tsa.seasonal import seasonal_decompose
from pmdarima import auto_arima

# Ignorar warnings não prejudiciais
import warnings
warnings.filterwarnings("ignore")

np.random.seed(0)
```

2. Faça a leitura dos dados de COVID-19 do estado de SP e complete os dados faltantes. Verifique se os dados estão Ok.

[3]:
```
pkgdir = '/home/cibele/CibelePython/AprendizadoDinamico/Data'

# Leitura dos dados de COVID-19 no estado de SP - vamos trabalhar com as mortes
covidSP = pd.read_csv(f'{pkgdir}/covidSP.csv', index_col='date',␣
↪parse_dates=True)
```

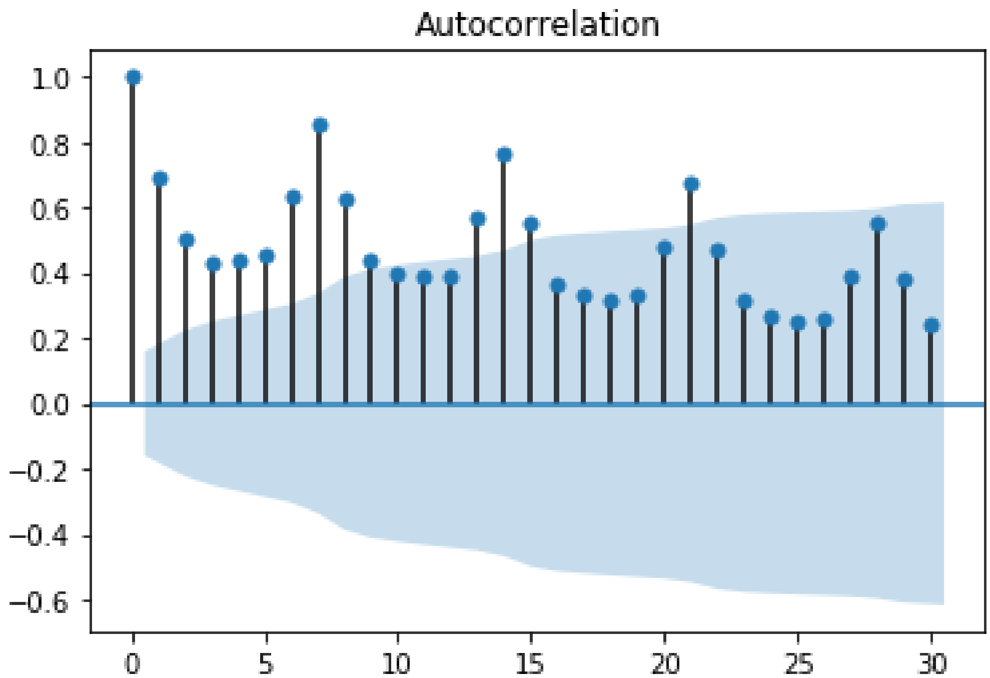
```
idx = pd.date_range(start=covidSP.index.min(), end=covidSP.index.max(), freq='D')
covidSP = covidSP.reindex(idx)
covidSP.fillna(0,inplace=True)

covidSP['deaths'].plot()
```

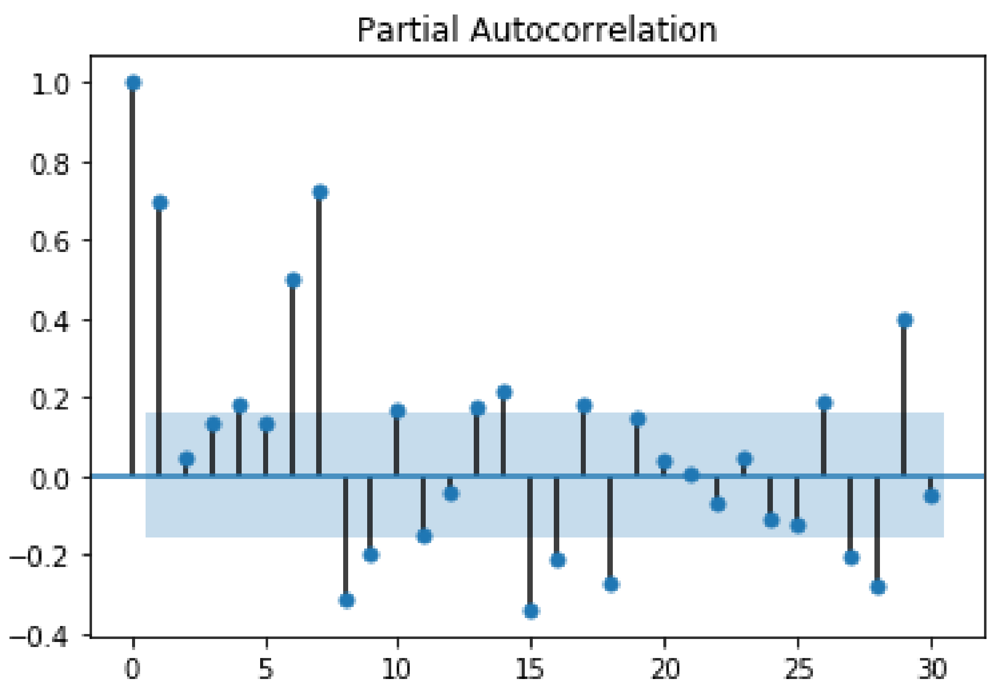
[3]: <matplotlib.axes._subplots.AxesSubplot at 0x7f2fee346a50>



3. Repita a análise com gráficos de autocorrelação e autocorrelação parcial feitos em aula. Já é possível identificar características do melhor modelo para esses dados?

[4]:
```
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf # para determinar␣
↪(p,q)

# Correlograma

plot_acf(covidSP['deaths'], lags=30)
plot_pacf(covidSP['deaths'], lags=30)
plt.show()
```





4. Utilize o stepwise_fit para escolher as ordens de um modelo SARIMA (p,d,q)x(P,D,Q)m, como feito em aula

[7]:

```python
# Ajuste de modelo SARIMA

auto_arima(covidSP['deaths'],seasonal=True,m=7).summary()


stepwise_fit = auto_arima(covidSP['deaths'], start_p=0, start_q=0,
                          max_p=6, max_q=3, m=7,
                          seasonal=True,
                          trace=True,
                          error_action='ignore',
                          suppress_warnings=True,
                          stepwise=True)

stepwise_fit.summary()
```

```
Performing stepwise search to minimize aic
Fit ARIMA(0,0,0)x(1,1,1,7) [intercept=True]; AIC=1503.140, BIC=1515.019,
Time=0.186 seconds
Fit ARIMA(0,0,0)x(0,1,0,7) [intercept=True]; AIC=1546.346, BIC=1552.285,
Time=0.013 seconds
Fit ARIMA(1,0,0)x(1,1,0,7) [intercept=True]; AIC=1506.697, BIC=1518.577,
Time=0.206 seconds
Fit ARIMA(0,0,1)x(0,1,1,7) [intercept=True]; AIC=1504.764, BIC=1516.643,
Time=0.186 seconds
Fit ARIMA(0,0,0)x(0,1,0,7) [intercept=False]; AIC=1553.312, BIC=1556.281,
Time=0.010 seconds
Fit ARIMA(0,0,0)x(0,1,1,7) [intercept=True]; AIC=1505.935, BIC=1514.844,
Time=0.126 seconds
Fit ARIMA(0,0,0)x(1,1,0,7) [intercept=True]; AIC=1506.823, BIC=1515.733,
Time=0.303 seconds
Fit ARIMA(0,0,0)x(2,1,1,7) [intercept=True]; AIC=1501.191, BIC=1516.040,
Time=0.549 seconds
Fit ARIMA(0,0,0)x(2,1,0,7) [intercept=True]; AIC=1500.211, BIC=1512.091,
Time=0.387 seconds
Fit ARIMA(1,0,0)x(2,1,0,7) [intercept=True]; AIC=1497.094, BIC=1511.944,
Time=0.456 seconds
Fit ARIMA(1,0,0)x(2,1,1,7) [intercept=True]; AIC=1498.545, BIC=1516.364,
Time=0.704 seconds
Fit ARIMA(1,0,0)x(1,1,1,7) [intercept=True]; AIC=1500.782, BIC=1515.631,
Time=0.278 seconds
Fit ARIMA(2,0,0)x(2,1,0,7) [intercept=True]; AIC=1495.055, BIC=1512.874,
Time=0.618 seconds
Fit ARIMA(2,0,0)x(1,1,0,7) [intercept=True]; AIC=1502.469, BIC=1517.319,
Time=0.254 seconds
Fit ARIMA(2,0,0)x(2,1,1,7) [intercept=True]; AIC=1496.087, BIC=1516.875,
Time=0.971 seconds
Fit ARIMA(2,0,0)x(1,1,1,7) [intercept=True]; AIC=1498.097, BIC=1515.916,
```

```
Time=0.326 seconds
Fit ARIMA(3,0,0)x(2,1,0,7) [intercept=True]; AIC=1491.535, BIC=1512.324,
Time=0.740 seconds
Fit ARIMA(3,0,0)x(1,1,0,7) [intercept=True]; AIC=1496.157, BIC=1513.976,
Time=0.429 seconds
Fit ARIMA(3,0,0)x(2,1,1,7) [intercept=True]; AIC=1492.478, BIC=1516.237,
Time=0.954 seconds
Fit ARIMA(3,0,0)x(1,1,1,7) [intercept=True]; AIC=1493.818, BIC=1514.606,
Time=0.524 seconds
Fit ARIMA(4,0,0)x(2,1,0,7) [intercept=True]; AIC=1488.717, BIC=1512.476,
Time=1.081 seconds
Fit ARIMA(4,0,0)x(1,1,0,7) [intercept=True]; AIC=1493.623, BIC=1514.412,
Time=0.431 seconds
Fit ARIMA(4,0,0)x(2,1,1,7) [intercept=True]; AIC=1490.135, BIC=1516.864,
Time=1.625 seconds
Fit ARIMA(4,0,0)x(1,1,1,7) [intercept=True]; AIC=1490.818, BIC=1514.577,
Time=0.728 seconds
Fit ARIMA(5,0,0)x(2,1,0,7) [intercept=True]; AIC=1490.106, BIC=1516.835,
Time=1.248 seconds
Fit ARIMA(4,0,1)x(2,1,0,7) [intercept=True]; AIC=1488.725, BIC=1515.453,
Time=1.580 seconds
Fit ARIMA(3,0,1)x(2,1,0,7) [intercept=True]; AIC=1490.816, BIC=1514.575,
Time=1.164 seconds
Fit ARIMA(5,0,1)x(2,1,0,7) [intercept=True]; AIC=1490.307, BIC=1520.005,
Time=1.990 seconds
Total fit time: 18.095 seconds
```

[7]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                     SARIMAX Results
================================================================================
=========
Dep. Variable:                                  y   No. Observations:
151
Model:             SARIMAX(4, 0, 0)x(2, 1, 0, 7)   Log Likelihood
-736.359
Date:                            Tue, 04 Aug 2020   AIC
1488.717
Time:                                    23:48:13   BIC
1512.476
Sample:                                         0   HQIC
1498.371
                                            - 151
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
```

```
intercept     16.9028      5.491      3.078      0.002      6.140     27.665
ar.L1          0.2212      0.072      3.065      0.002      0.080      0.363
ar.L2          0.1651      0.071      2.325      0.020      0.026      0.304
ar.L3         -0.2270      0.086     -2.652      0.008     -0.395     -0.059
ar.L4          0.1806      0.075      2.412      0.016      0.034      0.327
ar.S.L7       -0.6434      0.073     -8.836      0.000     -0.786     -0.501
ar.S.L14      -0.2184      0.095     -2.288      0.022     -0.406     -0.031
sigma2      1579.5096    162.146      9.741      0.000   1261.709   1897.310
================================================================================
===
Ljung-Box (Q):                       31.23   Jarque-Bera (JB):
14.88
Prob(Q):                              0.84   Prob(JB):
0.00
Heteroskedasticity (H):               8.88   Skew:
0.49
Prob(H) (two-sided):                  0.00   Kurtosis:
4.23
================================================================================
===

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

5. Atribua a um objeto modelo1 o modelo escolhido pelo stepwise_fit e previsões1 os valores preditos por ele

[24]:
```
modelo1 = SARIMAX(covidSP['deaths'],order=(4,0,0),seasonal_order=(2,1,0,7))
resultado1 = modelo1.fit()

previsões1 = resultado1.predict(start=covidSP.index.min(), end=covidSP.index.
 ↪max(), dynamic=False, typ='levels')
previsões1.index = covidSP.index
```

6. Ajuste um modelo "errado" aos dados, por exemplo um SARIMA(0,0,1)x(0,0,2,7), ou seja, com termos de médias móveis e não autorregressivos. Atribua o ajuste e valores preditos aos objetos modelo2 e previsões2.

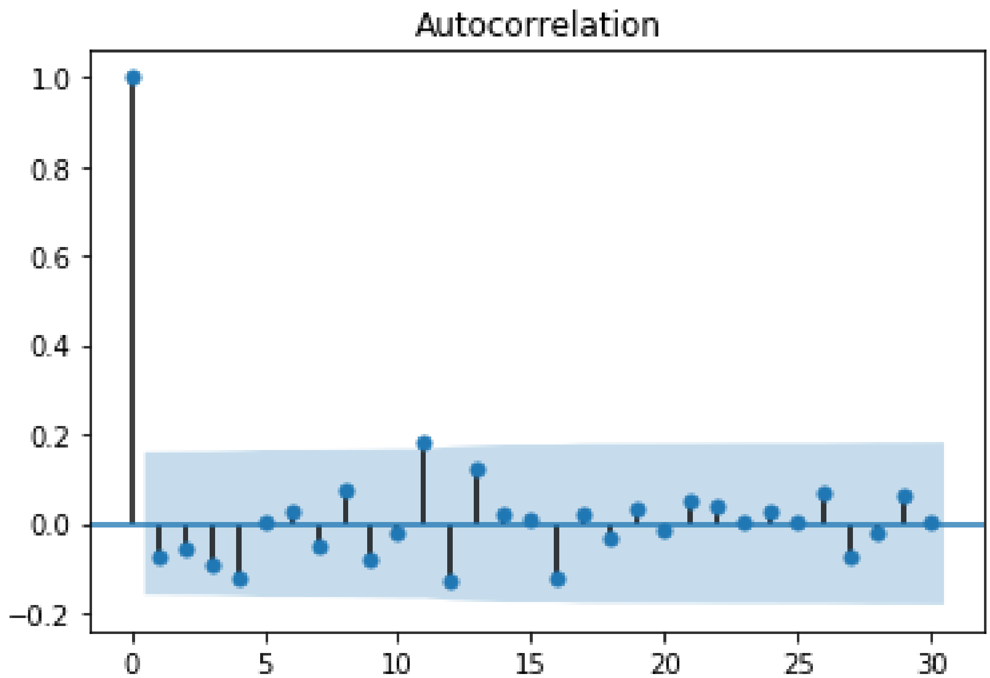
[129]:
```
modelo2 = SARIMAX(covidSP['deaths'],order=(0,0,1),seasonal_order=(0,0,2,7))
resultado2 = modelo2.fit()

previsões2 = resultado2.predict(start=covidSP.index.min(), end=covidSP.index.
 ↪max(), dynamic=False, typ='levels')
previsões2.index = covidSP.index
```

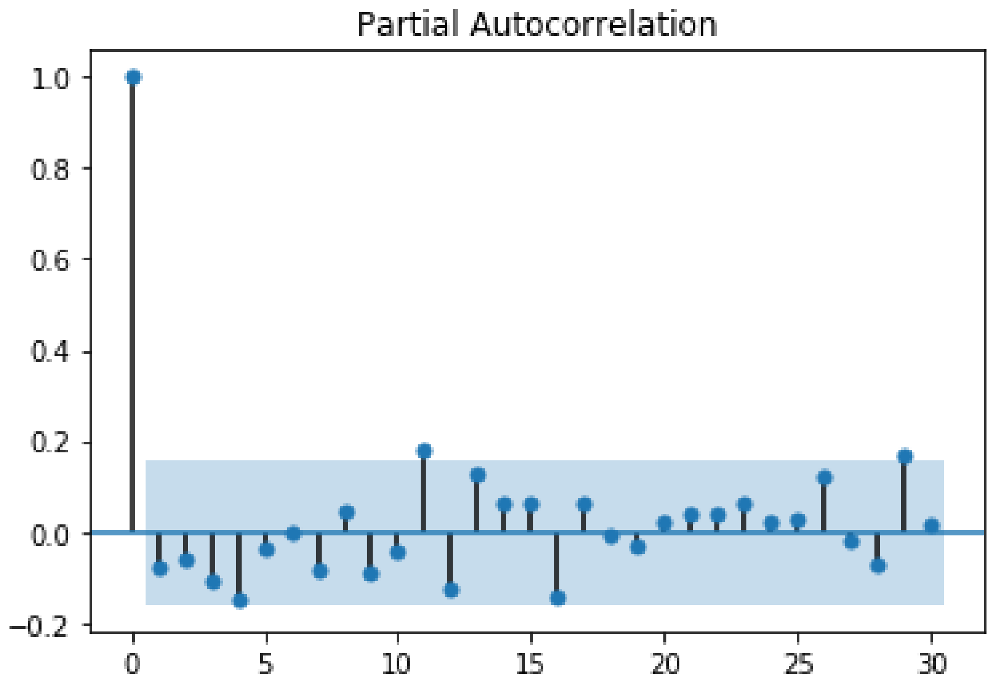
7. Faça uma análise de diagnóstico usando os resíduos obtidos pelo modelo1 e modelo2. Com-

pare, por exemplo, os gráficos de autocorrelação e autocorrelação parcial.

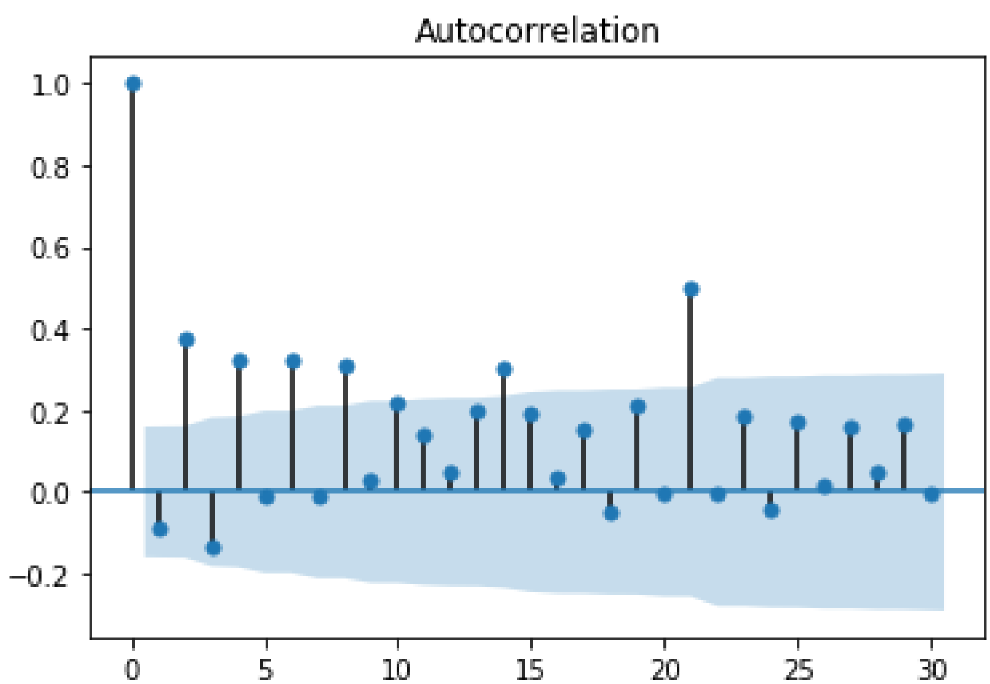
[130]:
```
resíduos1 = resultado1.resid
resíduos2 = resultado2.resid
```

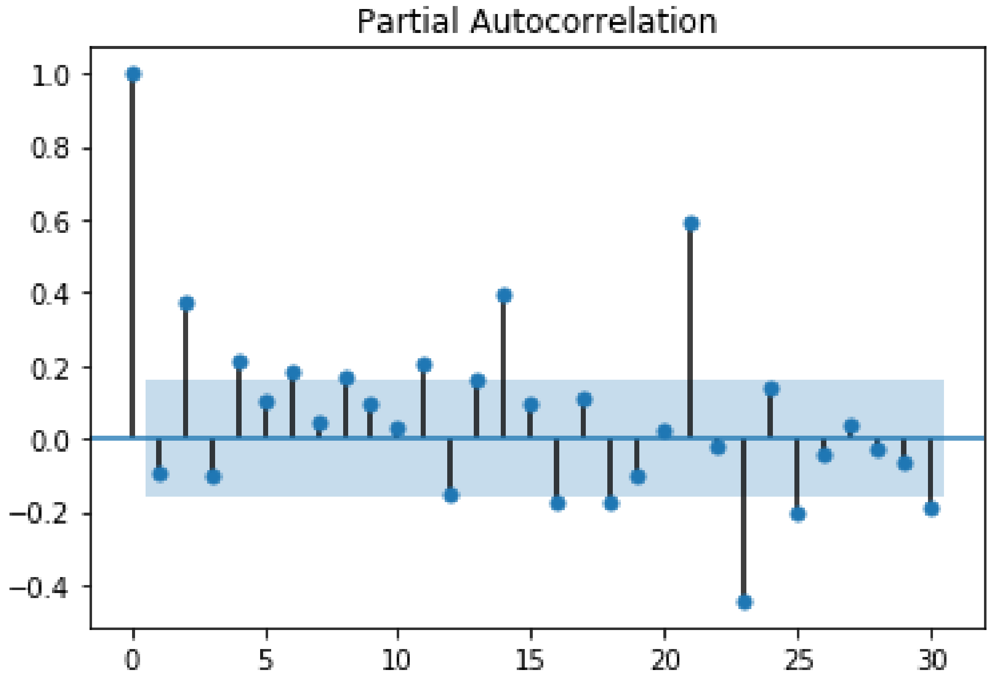
[131]:
```
plot_acf(resíduos1, lags=30)
plot_pacf(resíduos1, lags=30)
plt.show()
```



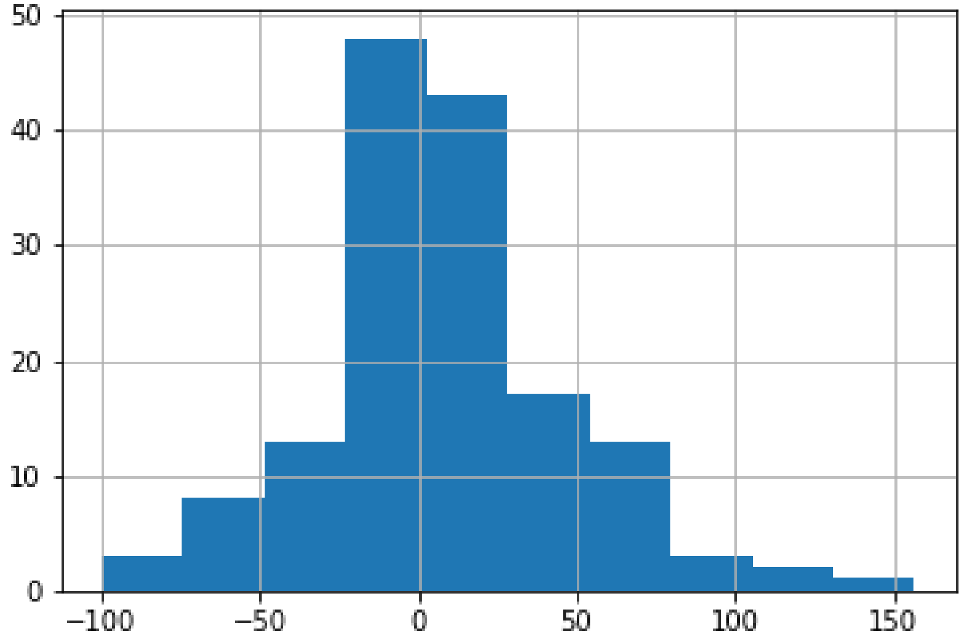


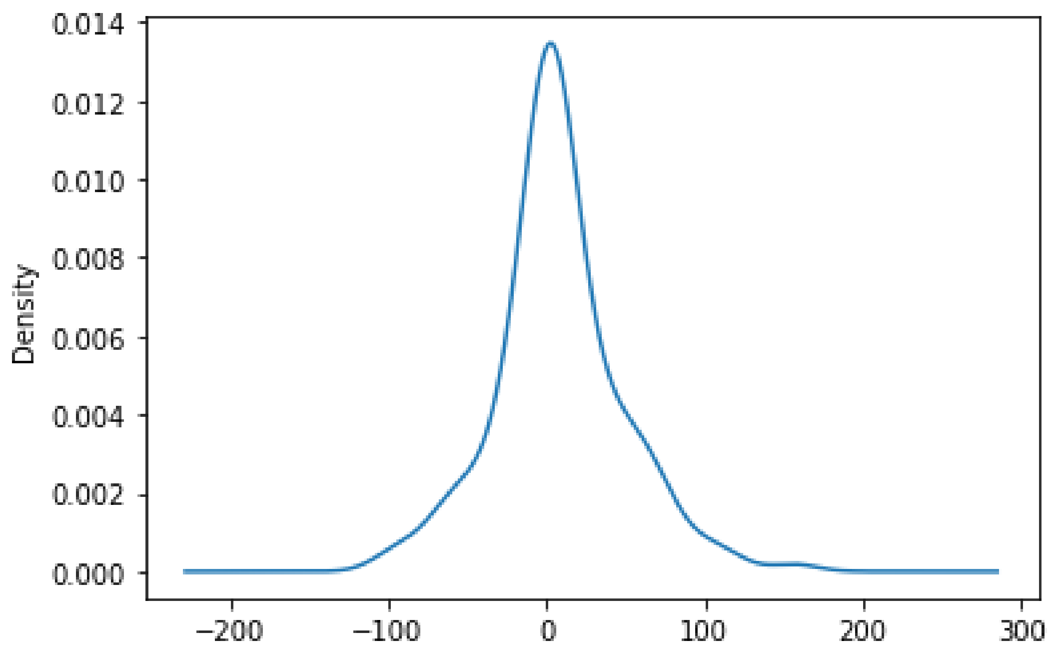
[132]:
```
plot_acf(resíduos2, lags=30)
plot_pacf(resíduos2, lags=30)
plt.show()
```





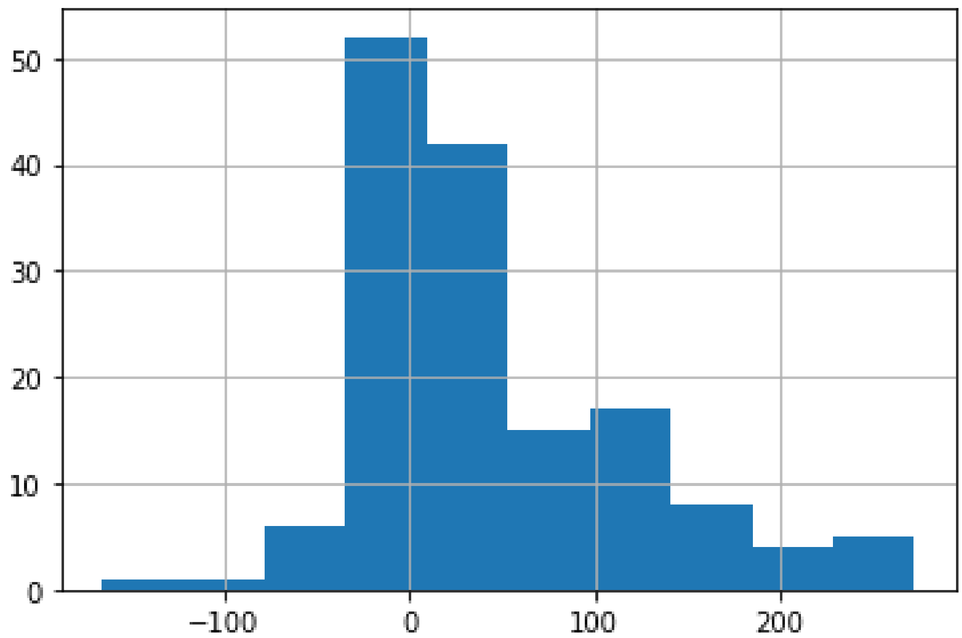
[133]:
```
from matplotlib import pyplot


resíduos1.hist()

pyplot.show()
resíduos1.plot(kind='kde')
pyplot.show()
```

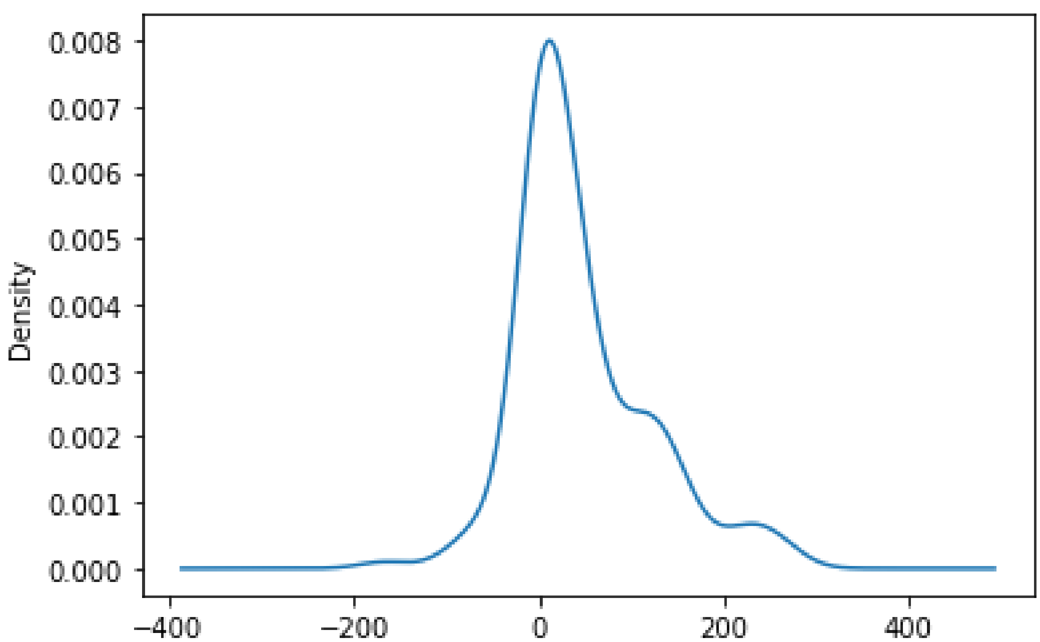




[134]:
```
from matplotlib import pyplot


resíduos2.hist()
```

```
pyplot.show()
resíduos2.plot(kind='kde')
pyplot.show()
```





8. Compare as métricas erro quadrático médio e erro absoluto médio. Existe muita diferença entre eles?

[135]:
```
from sklearn.metrics import mean_squared_error

error = mean_squared_error(covidSP['deaths'], previsões1)
print(f'EQM SARIMA(4,0,0)(2,1,0,7): {error:11.10}')
```

```
EQM SARIMA(4,0,0)(2,1,0,7): 1634.685252
```

[74]:
```
from sklearn.metrics import mean_squared_error

error = mean_squared_error(covidSP['deaths'], previsões2)
print(f'EQM SARIMA(0,0,4)(0,1,2,7): {error:11.10}')
```

```
EQM SARIMA(0,0,4)(0,1,2,7): 1700.733094
```

[75]:
```
from statsmodels.tools.eval_measures import rmse

error = rmse(covidSP['deaths'], previsões1)
print(f'REQM SARIMA(4,0,0)(2,1,0,7): {error:11.10}')
```

```
REQM SARIMA(4,0,0)(2,1,0,7): 40.43124104
```

[76]:
```
from statsmodels.tools.eval_measures import rmse

error = rmse(covidSP['deaths'], previsões2)
print(f'REQM SARIMA(0,0,4)(0,1,2,7): {error:11.10}')
```

```
REQM SARIMA(0,0,4)(0,1,2,7): 41.23994537
```

9. Utilize outros métodos de diagnóstico se achar necessário.

[ ]:
```

```