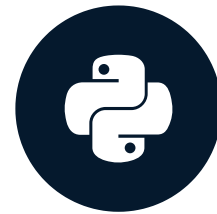


Intro to ACF and PACF

ARIMA MODELS IN PYTHON

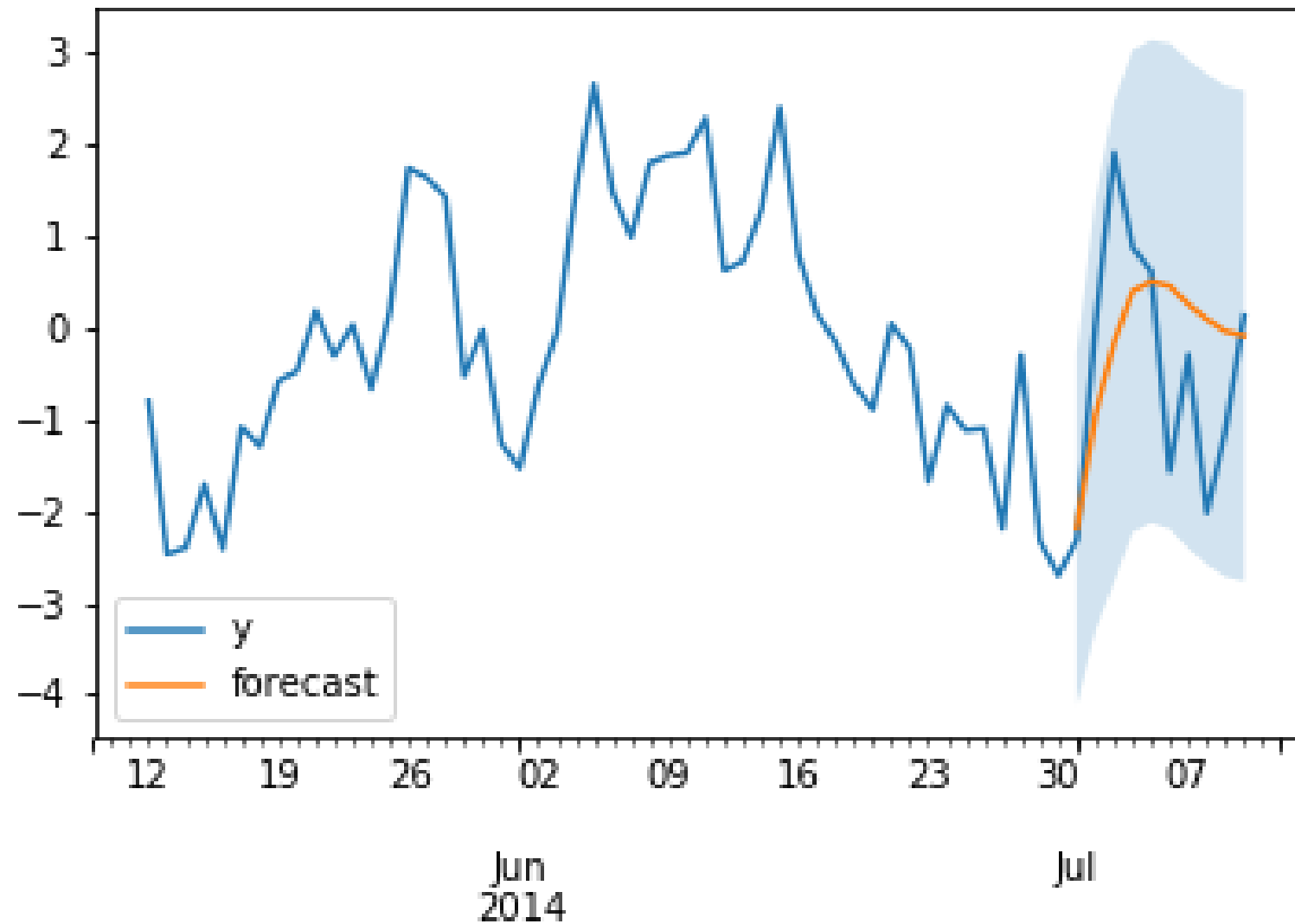


James Fulton

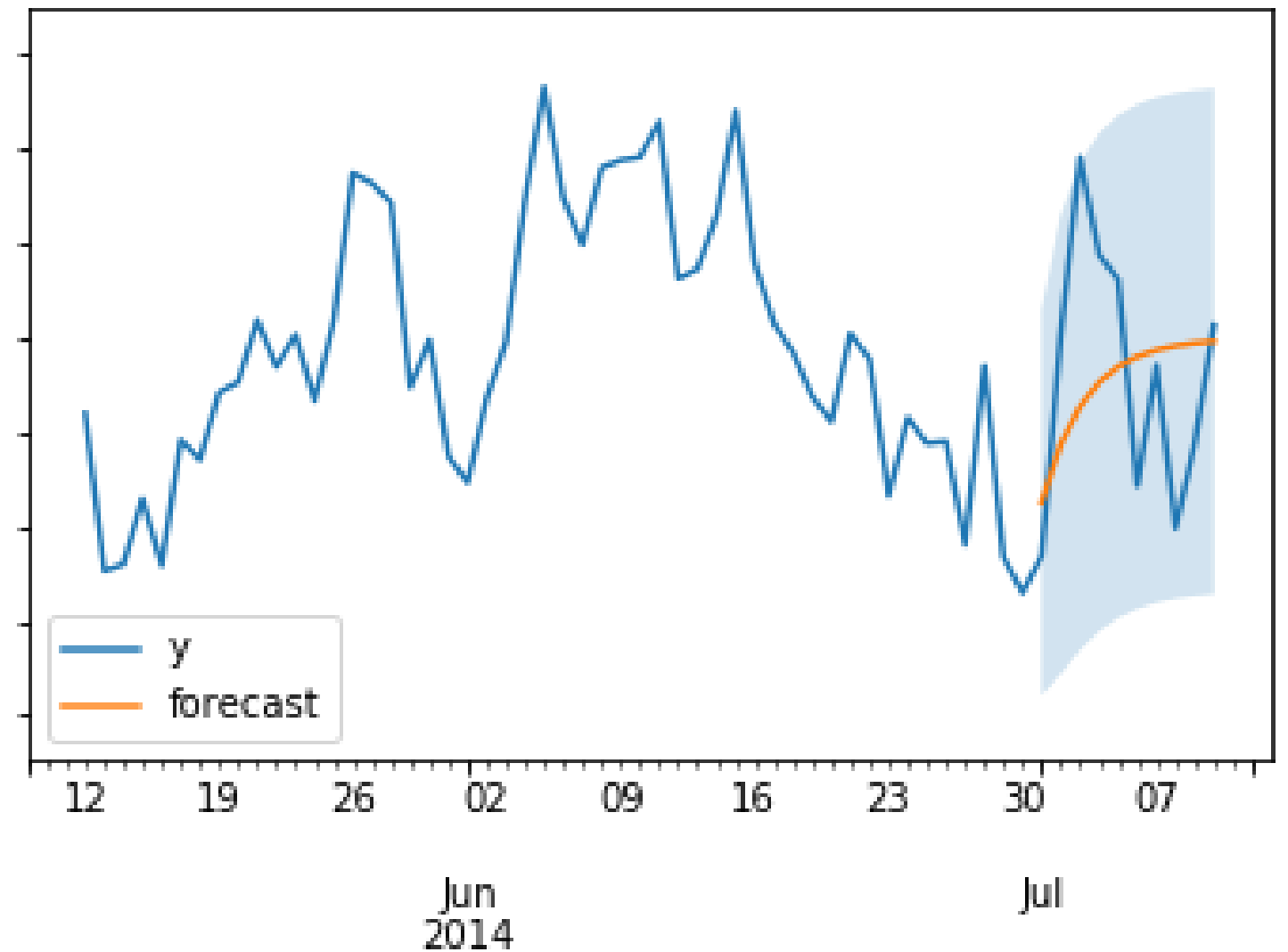
Climate informatics researcher

Motivation

ARMA(3,0) Dynamic Forecast



ARMA(1,1) Dynamic Forecast



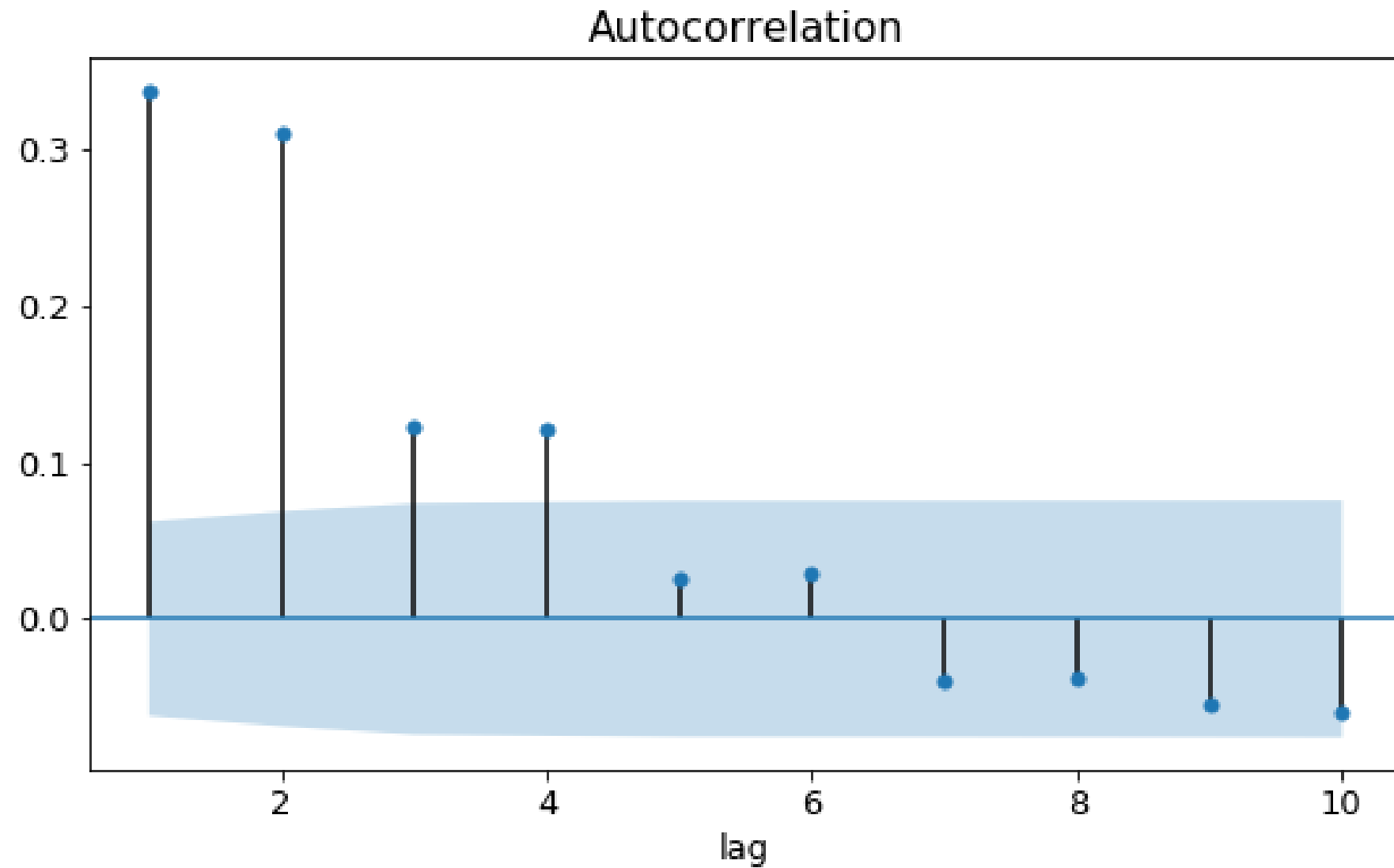
ACF and PACF

- ACF - Autocorrelation Function
- PACF - Partial autocorrelation function

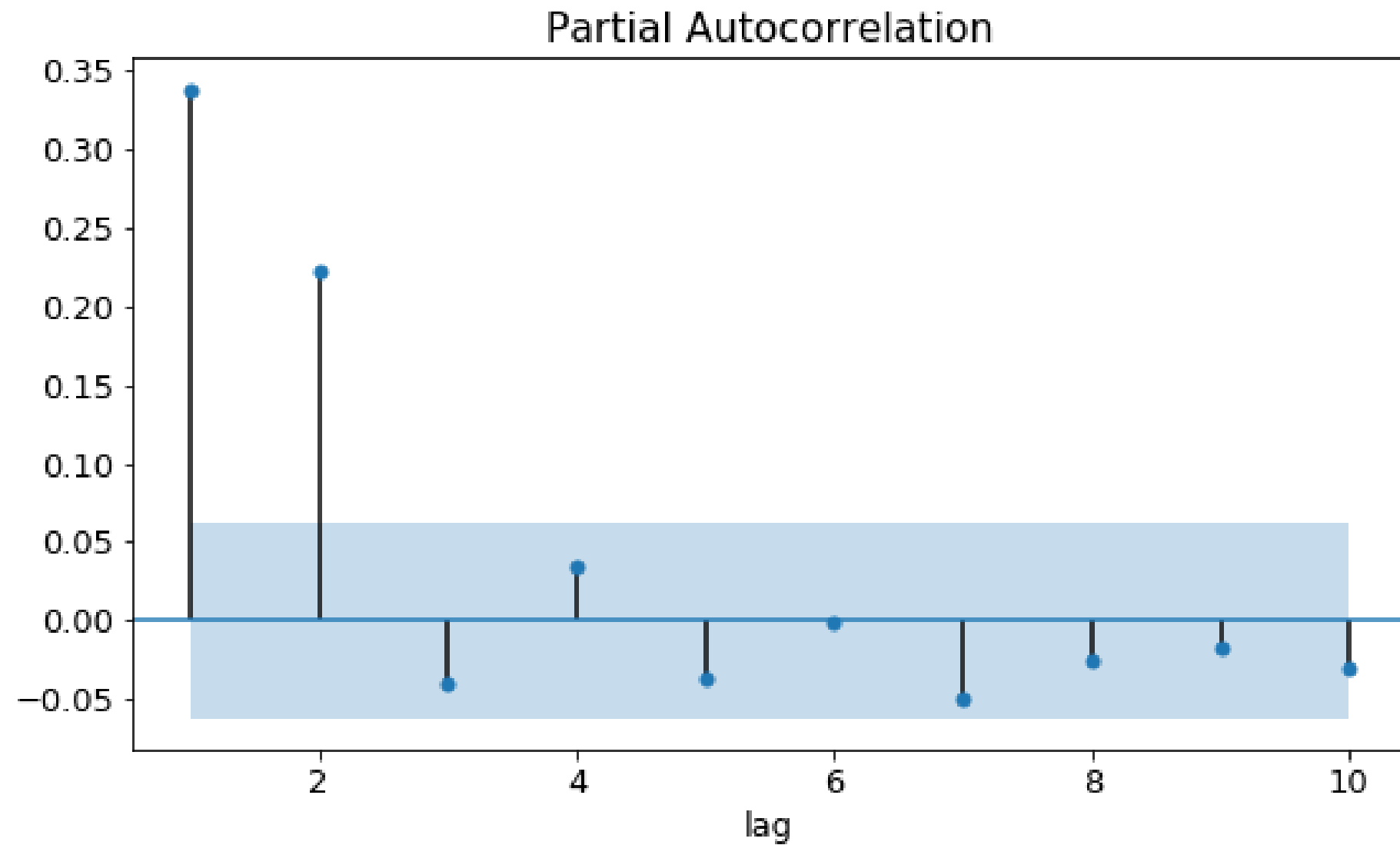
What is the ACF

- lag-1 autocorrelation $\rightarrow \text{corr}(y_t, y_{t-1})$
- lag-2 autocorrelation $\rightarrow \text{corr}(y_t, y_{t-2})$
- ...
- lag-n autocorrelation $\rightarrow \text{corr}(y_t, y_{t-n})$

What is the ACF



What is the PACF



Using ACF and PACF to choose model order

AR(p)

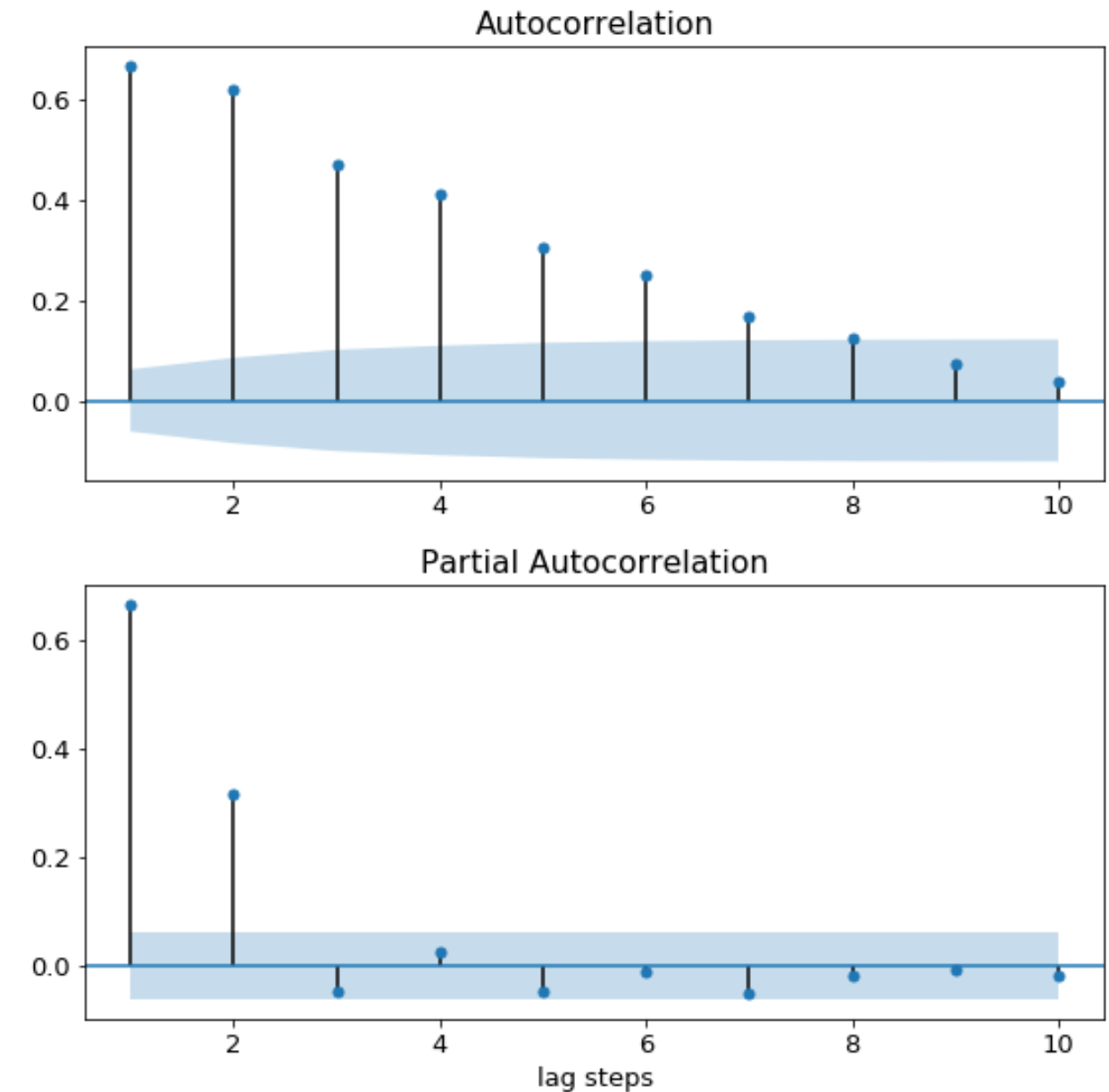
ACF

Tails off

PACF

Cuts off after lag p

- AR(2) model →



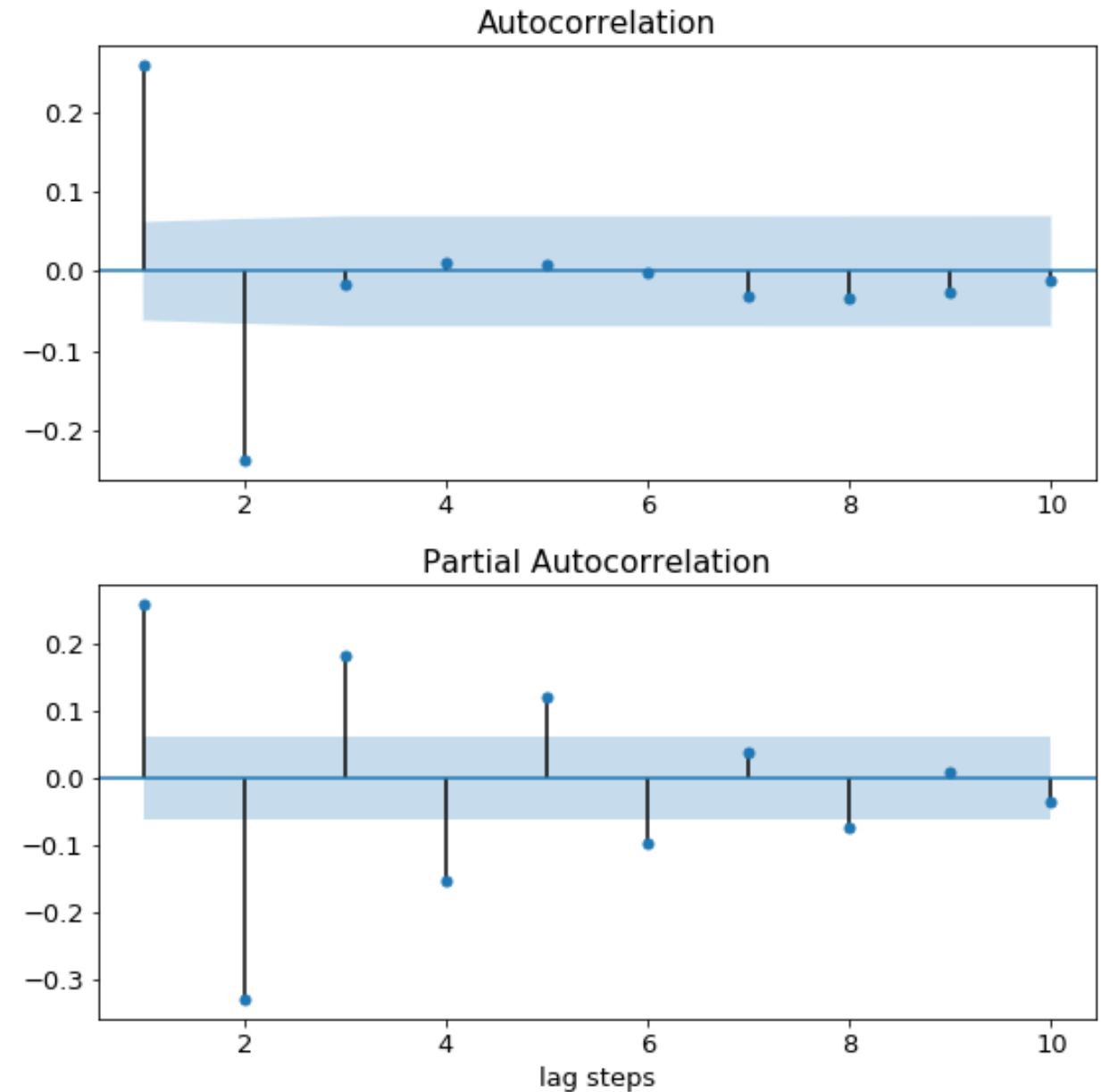
Using ACF and PACF to choose model order

MA(q)

ACF	Cuts off after lag q
-----	----------------------

PACF	Tails off
------	-----------

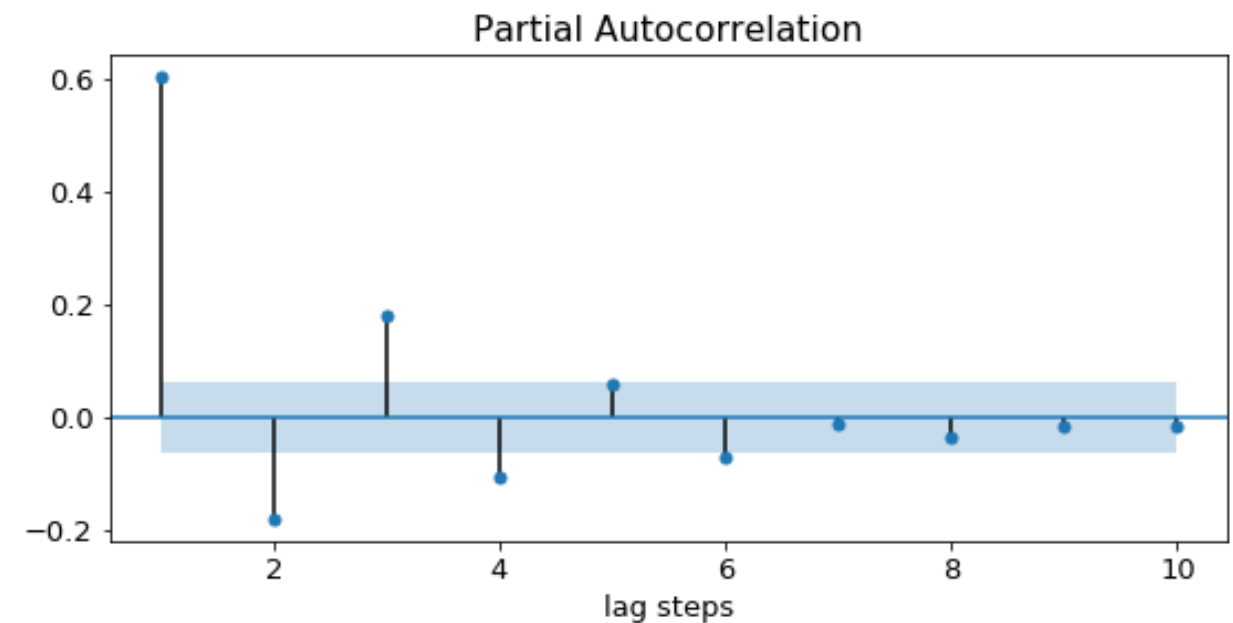
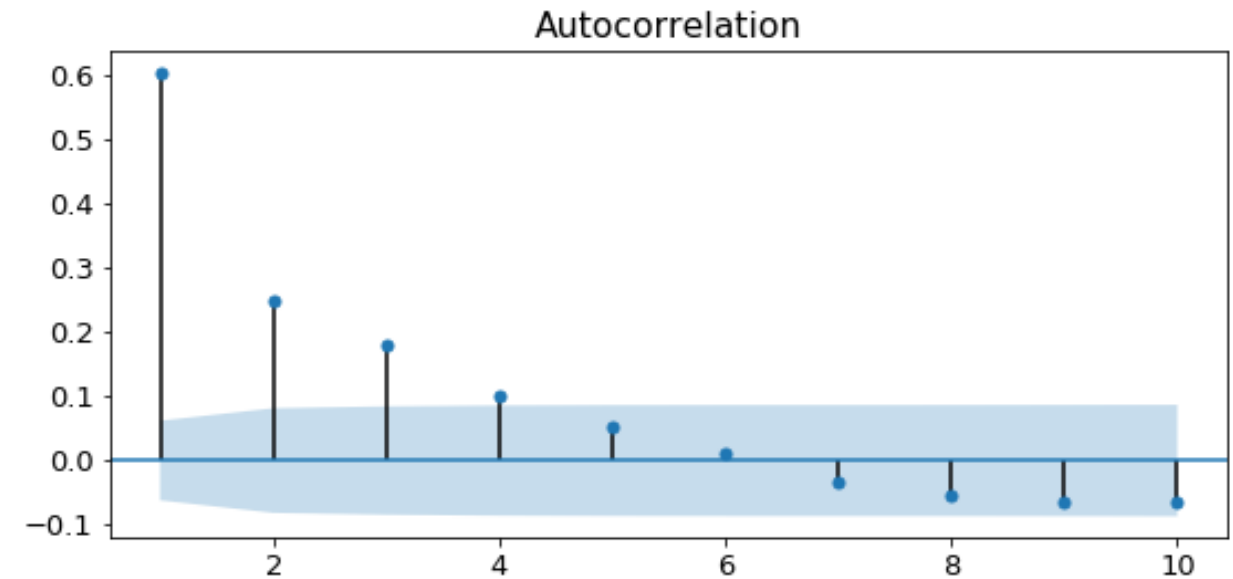
- MA(2) model →



Using ACF and PACF to choose model order

ARMA(p,q)

ACF	Tails off
PACF	Tails off



Using ACF and PACF to choose model order

	AR(p)	MA(q)	ARMA(p,q)
ACF	Tails off	Cuts off after lag q	Tails off
PACF	Cuts off after lag p	Tails off	Tails off

Implementation in Python

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

```
# Create figure
```

```
fig, (ax1, ax2) = plt.subplots(2,1, figsize=(8,8))
```

```
# Make ACF plot
```

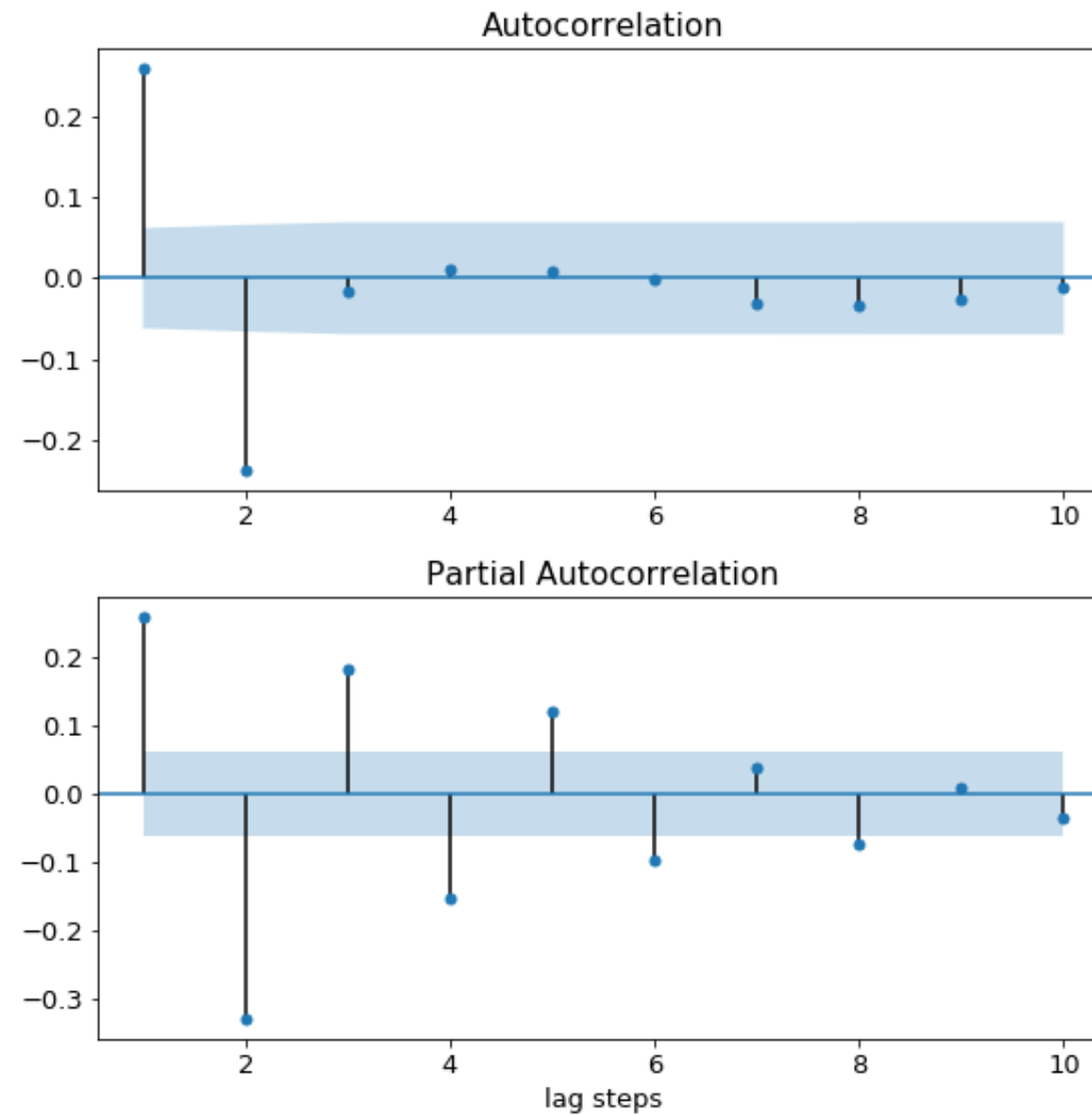
```
plot_acf(df, lags=10, zero=False, ax=ax1)
```

```
# Make PACF plot
```

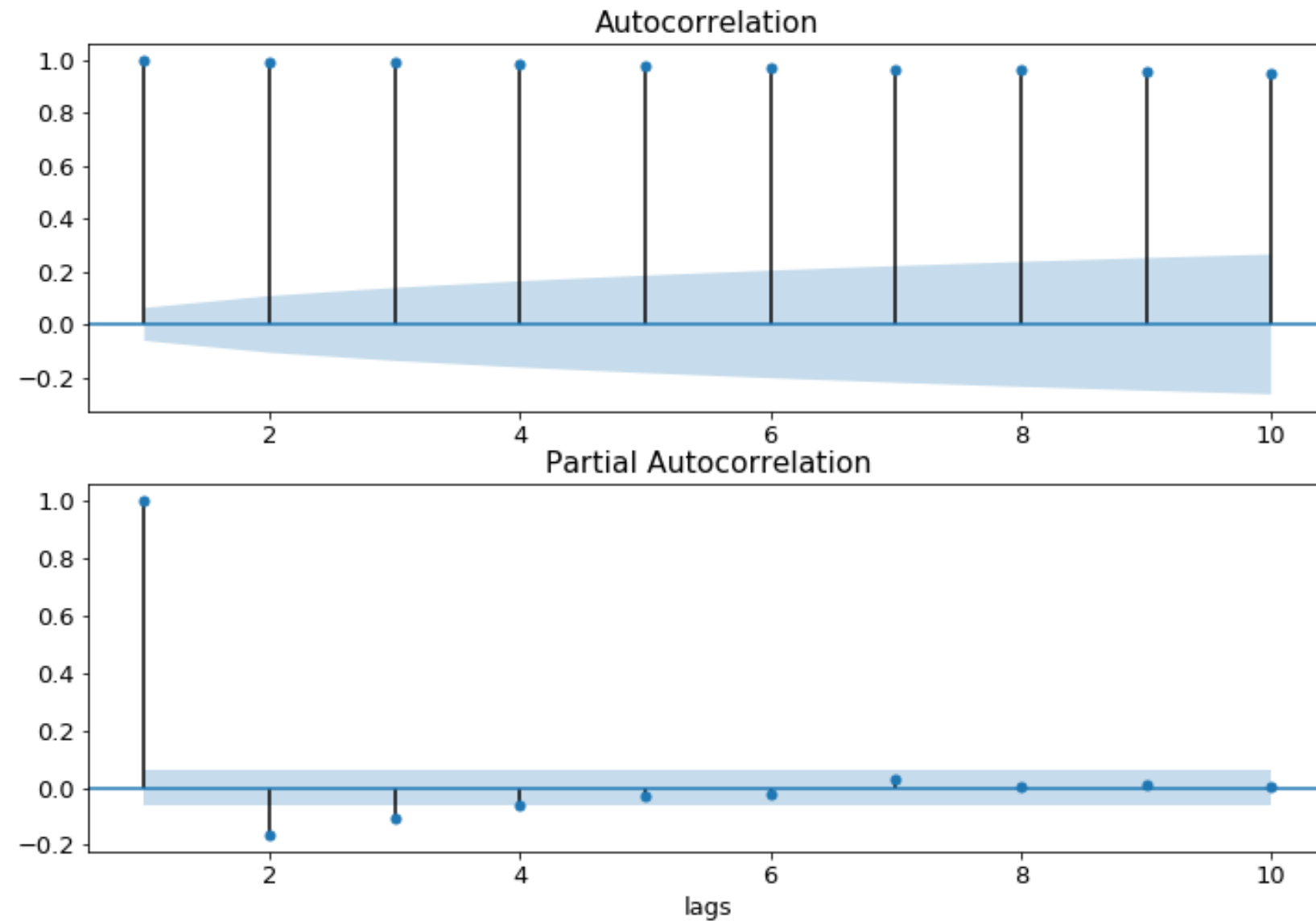
```
plot_pacf(df, lags=10, zero=False, ax=ax2)
```

```
plt.show()
```

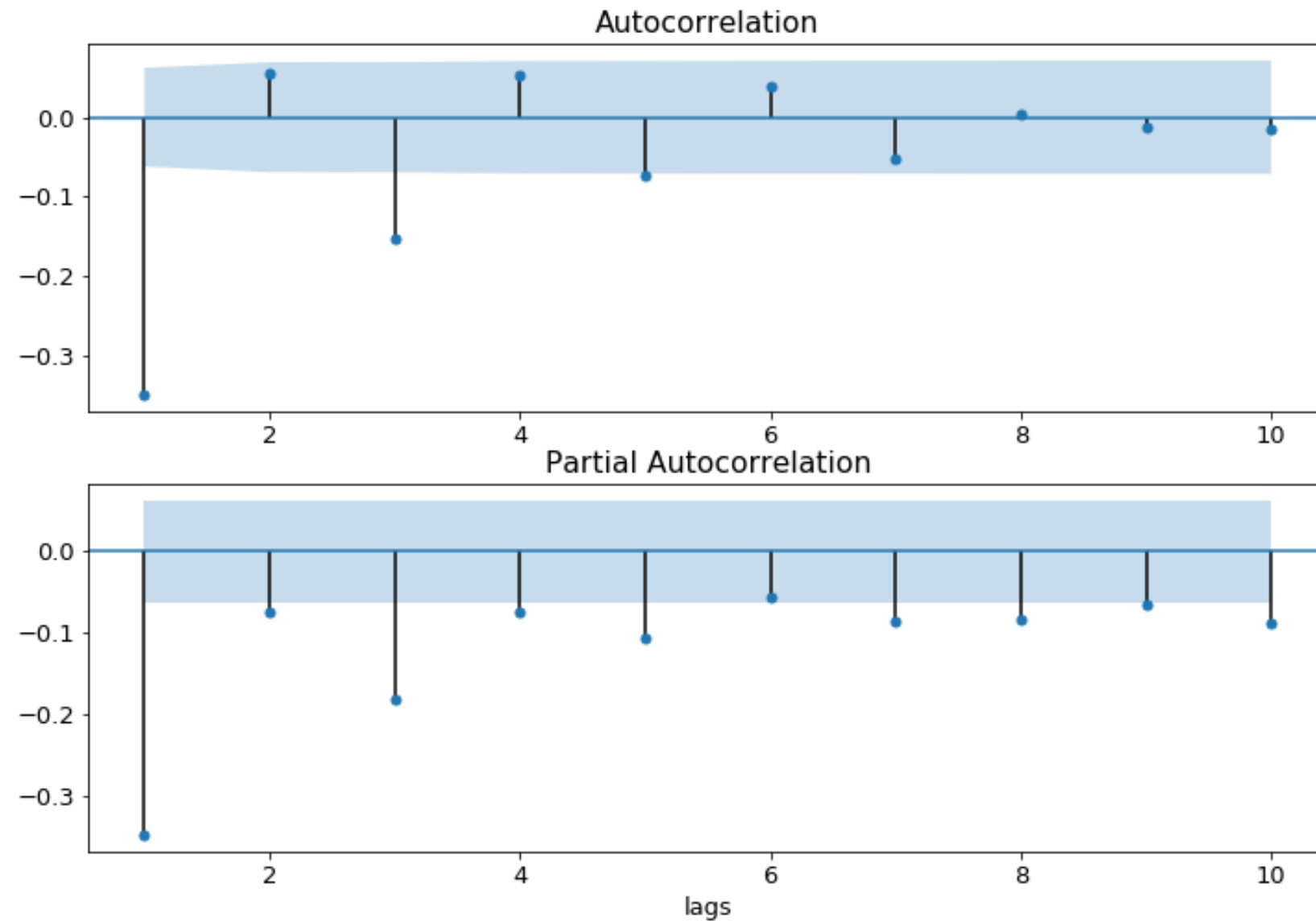
Implementation in Python



Over/under differencing and ACF and PACF



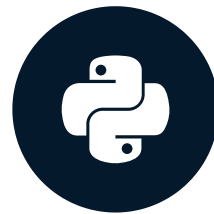
Over/under differencing and ACF and PACF



Let's practice!
ARIMA MODELS IN PYTHON

AIC and BIC

ARIMA MODELS IN PYTHON



James Fulton

Climate informatics researcher

AIC - Akaike information criterion

- Lower AIC indicates a better model
- AIC likes to choose simple models with lower order

BIC - Bayesian information criterion

- Very similar to AIC
- Lower BIC indicates a better model
- BIC likes to choose simple models with lower order

AIC vs BIC

- BIC favors simpler models than AIC
- AIC is better at choosing predictive models
- BIC is better at choosing good explanatory model

AIC and BIC in statsmodels

```
# Create model
model = SARIMAX(df, order=(1,0,1))
# Fit model
results = model.fit()
# Print fit summary
print(results.summary())
```

Statespace Model Results

```
=====
Dep. Variable:          y      No. Observations:      1000
Model:                SARIMAX(2, 0, 0)      Log Likelihood      -1399.704
Date:                Fri, 10 May 2019      AIC                2805.407
Time:                01:06:11      BIC                2820.131
Sample:                01-01-2013      HQIC                2811.003
                   - 09-27-2015
Covariance Type:      opg
```

AIC and BIC in statsmodels

```
# Create model
model = SARIMAX(df, order=(1,0,1))
# Fit model
results = model.fit()
# Print AIC and BIC
print('AIC:', results.aic)
print('BIC:', results.bic)
```

```
AIC: 2806.36
```

```
BIC: 2821.09
```

Searching over AIC and BIC

```
# Loop over AR order
for p in range(3):
    # Loop over MA order
    for q in range(3):
        # Fit model
        model = SARIMAX(df, order=(p,0,q))
        results = model.fit()
        # print the model order and the AIC/BIC values
        print(p, q, results.aic, results.bic)
```

```
0 0 2900.13 2905.04
0 1 2828.70 2838.52
0 2 2806.69 2821.42
1 0 2810.25 2820.06
1 1 2806.37 2821.09
1 2 2807.52 2827.15
...
```

Searching over AIC and BIC

```
order_aic_bic = []
# Loop over AR order
for p in range(3):
    # Loop over MA order
    for q in range(3):
        # Fit model
        model = SARIMAX(df, order=(p,0,q))
        results = model.fit()
        # Add order and scores to list
        order_aic_bic.append((p, q, results.aic, results.bic))

# Make DataFrame of model order and AIC/BIC scores
order_df = pd.DataFrame(order_aic_bic, columns=['p', 'q', 'aic', 'bic'])
```

Searching over AIC and BIC

```
# Sort by AIC
```

```
print(order_df.sort_values('aic'))
```

	p	q	aic	bic
7	2	1	2804.54	2824.17
6	2	0	2805.41	2820.13
4	1	1	2806.37	2821.09
2	0	2	2806.69	2821.42
...				

```
# Sort by BIC
```

```
print(order_df.sort_values('bic'))
```

	p	q	aic	bic
3	1	0	2810.25	2820.06
6	2	0	2805.41	2820.13
4	1	1	2806.37	2821.09
2	0	2	2806.69	2821.42
...				

Non-stationary model orders

```
# Fit model
model = SARIMAX(df, order=(2,0,1))
results = model.fit()
```

```
ValueError: Non-stationary starting autoregressive parameters
found with `enforce_stationarity` set to True.
```

When certain orders don't work

```
# Loop over AR order
for p in range(3):
    # Loop over MA order
    for q in range(3):

        # Fit model
        model = SARIMAX(df, order=(p,0,q))
        results = model.fit()

        # Print the model order and the AIC/BIC values
        print(p, q, results.aic, results.bic)
```

When certain orders don't work

```
# Loop over AR order
for p in range(3):
    # Loop over MA order
    for q in range(3):
        try:
            # Fit model
            model = SARIMAX(df, order=(p,0,q))
            results = model.fit()

            # Print the model order and the AIC/BIC values
            print(p, q, results.aic, results.bic)
        except:
            # Print AIC and BIC as None when fails
            print(p, q, None, None)
```

Let's practice!

ARIMA MODELS IN PYTHON

Model diagnostics

ARIMA MODELS IN PYTHON



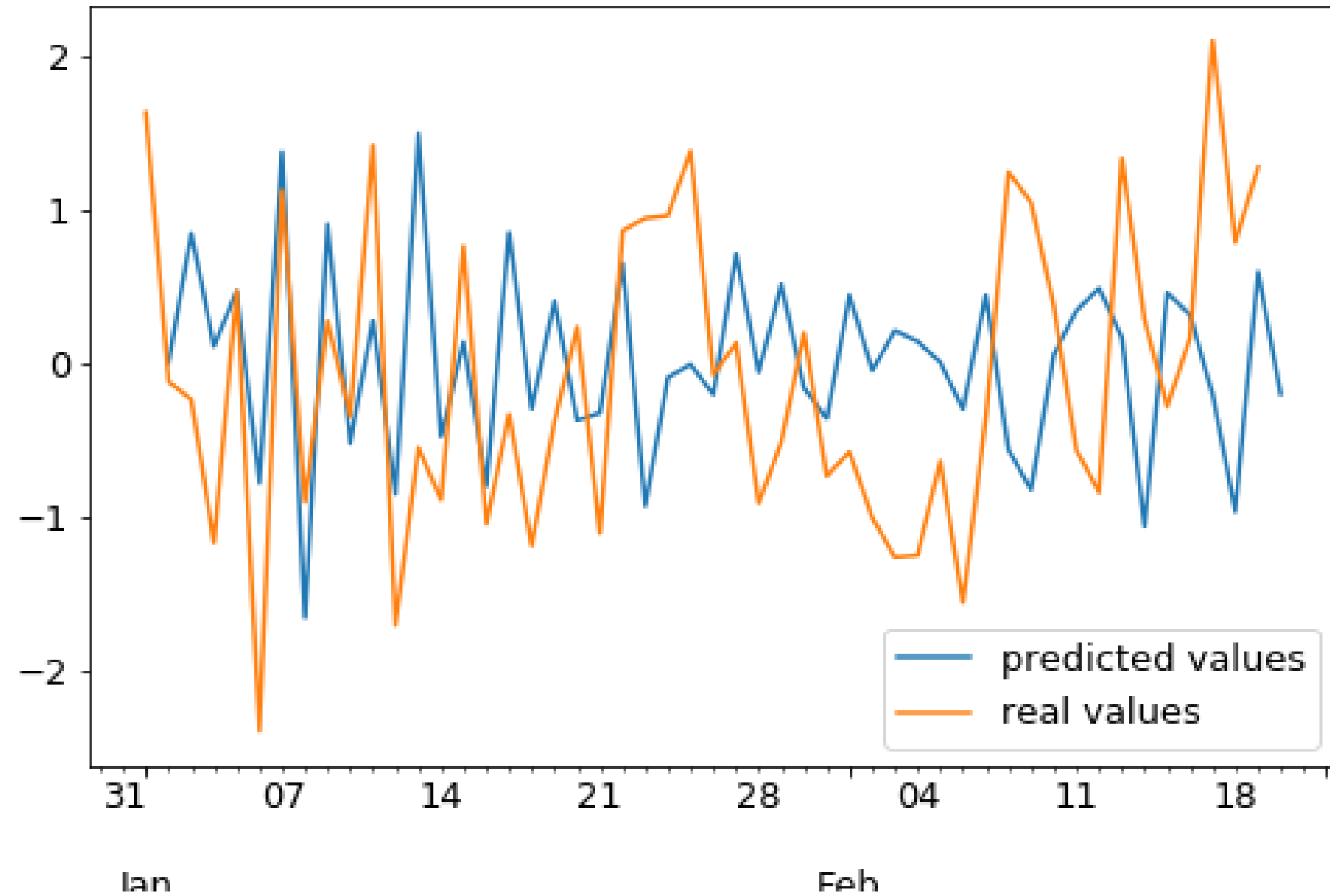
James Fulton

Climate informatics researcher

Introduction to model diagnostics

- How good is the final model?

Residuals



Residuals

```
# Fit model
model = SARIMAX(df, order=(p,d,q))
results = model.fit()
# Assign residuals to variable
residuals = results.resid
```

```
2013-01-23    1.013129
2013-01-24    0.114055
2013-01-25    0.430698
2013-01-26   -1.247046
2013-01-27   -0.499565
...          ...
```


Mean absolute error

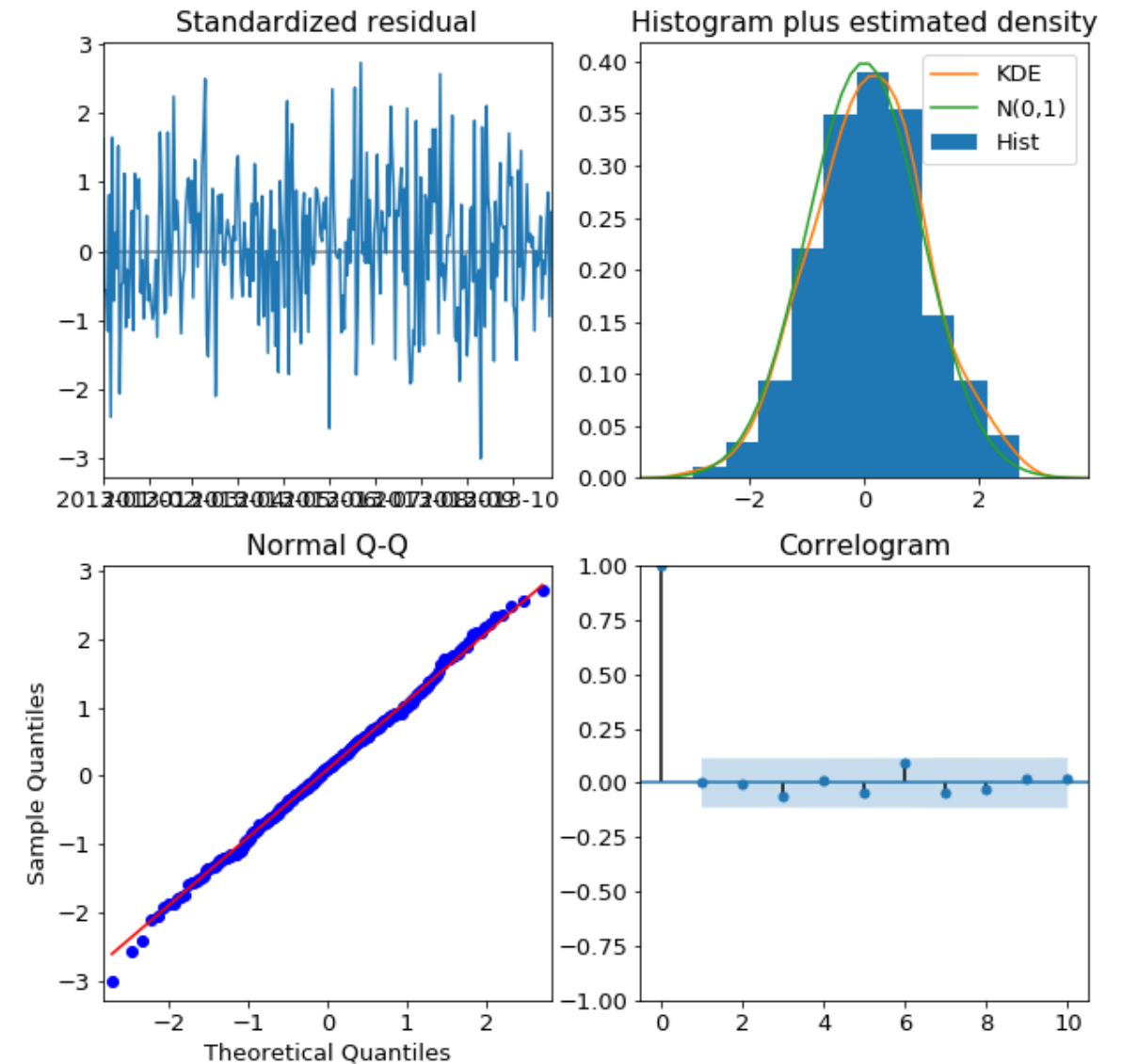
How far are the predictions from the real values?

```
mae = np.mean(np.abs(residuals))
```

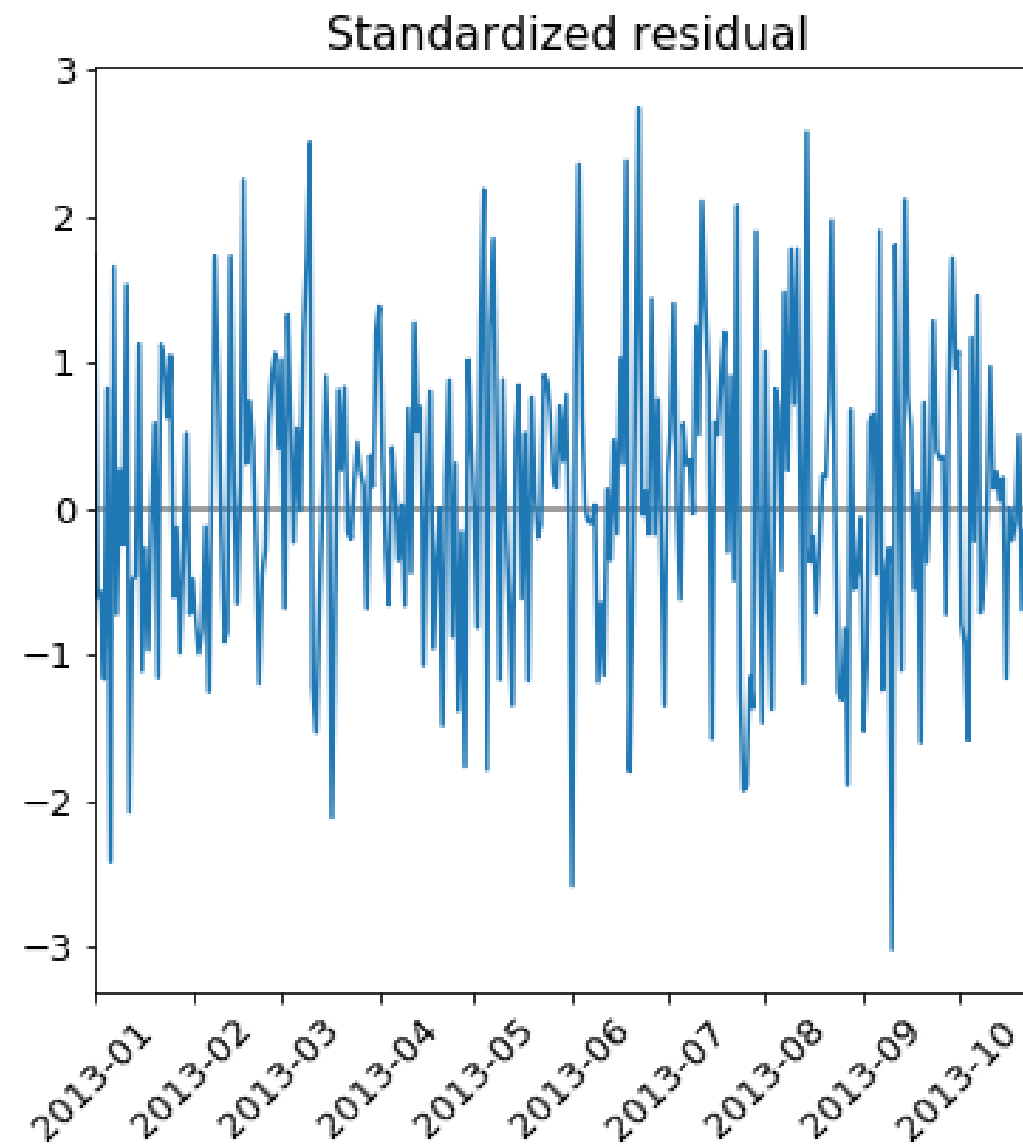
Plot diagnostics

If the model fits well the residuals will be white Gaussian noise

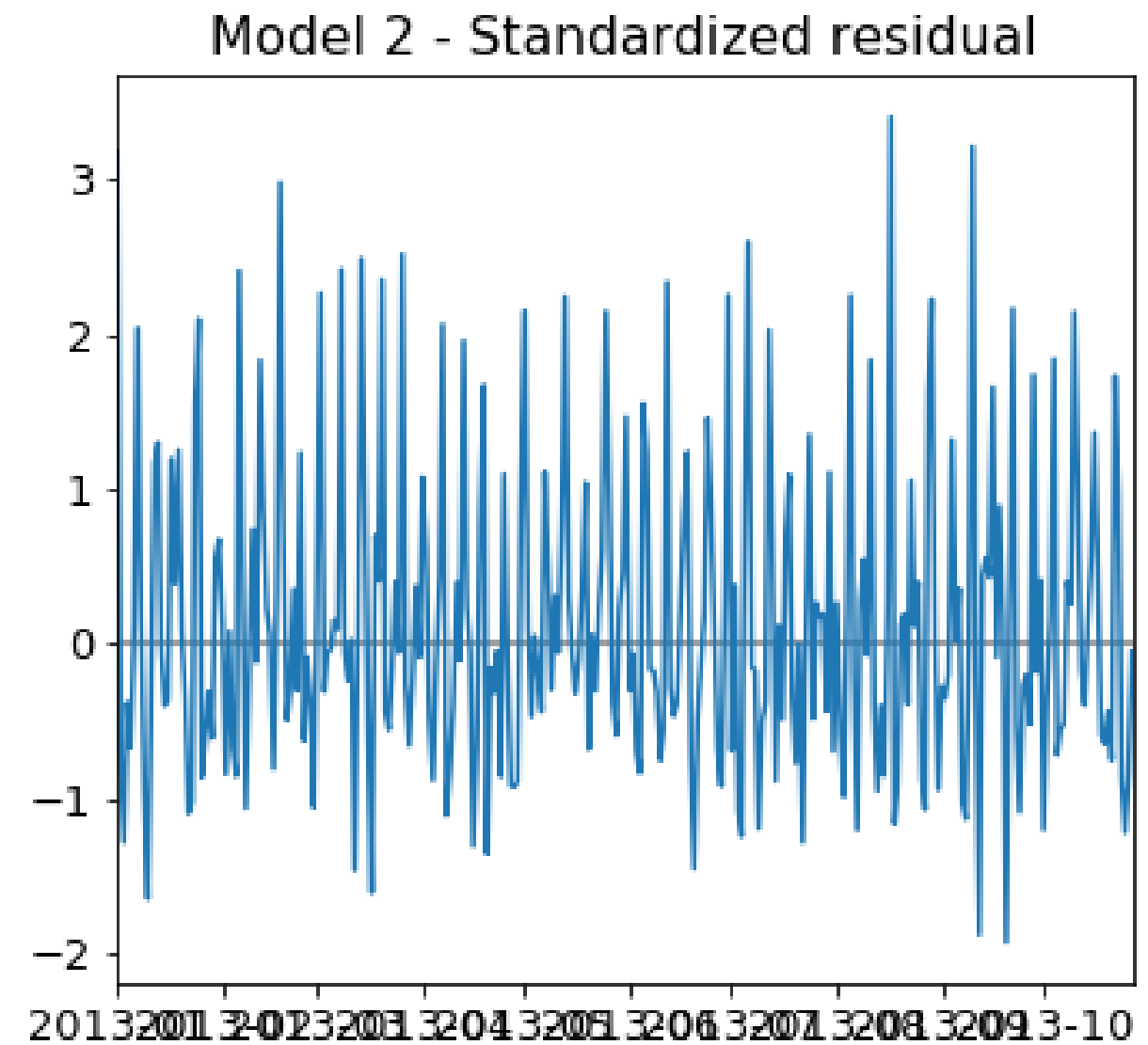
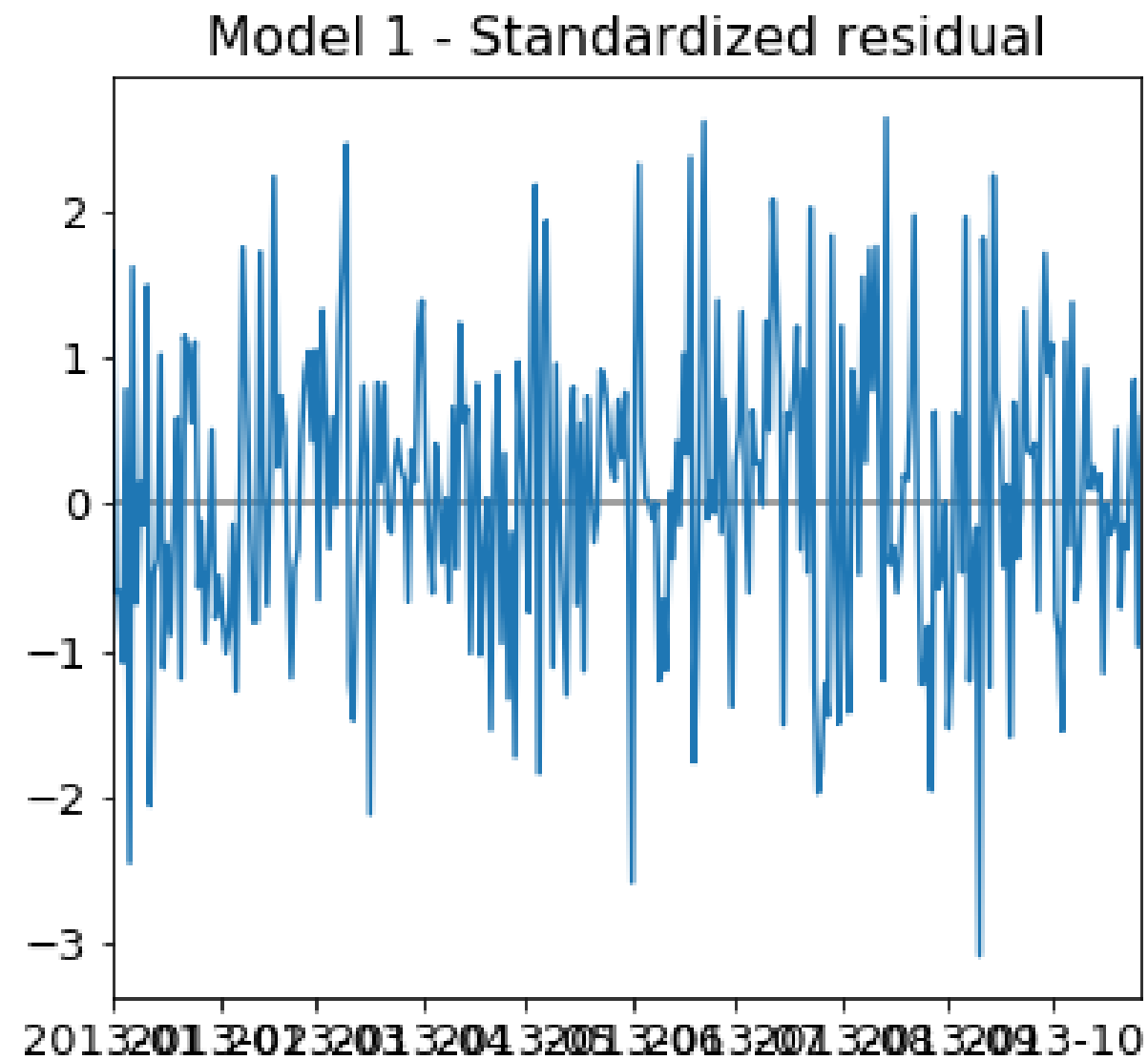
```
# Create the 4 diagnostics plots
results.plot_diagnostics()
plt.show()
```



Residuals plot

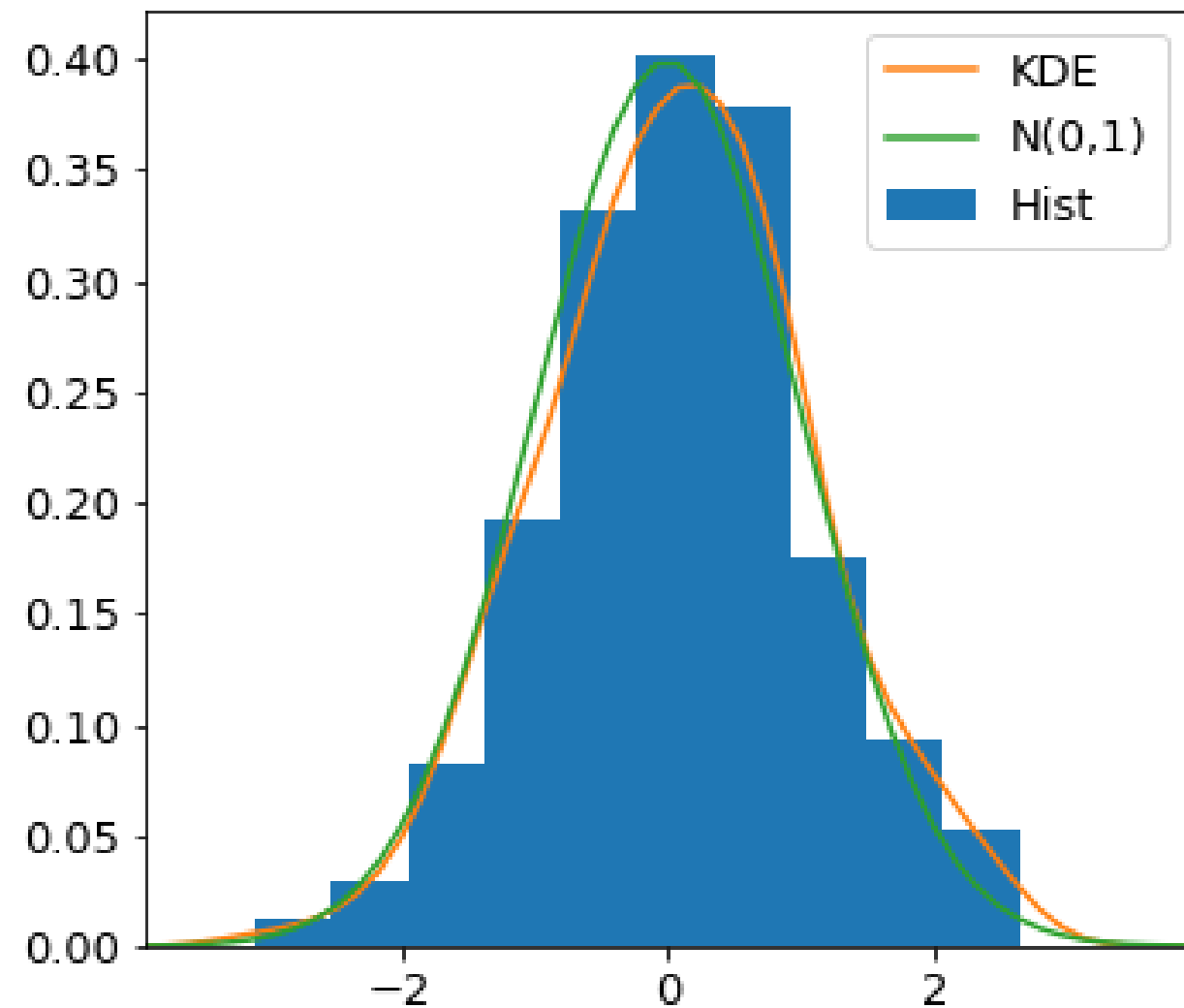


Residuals plot

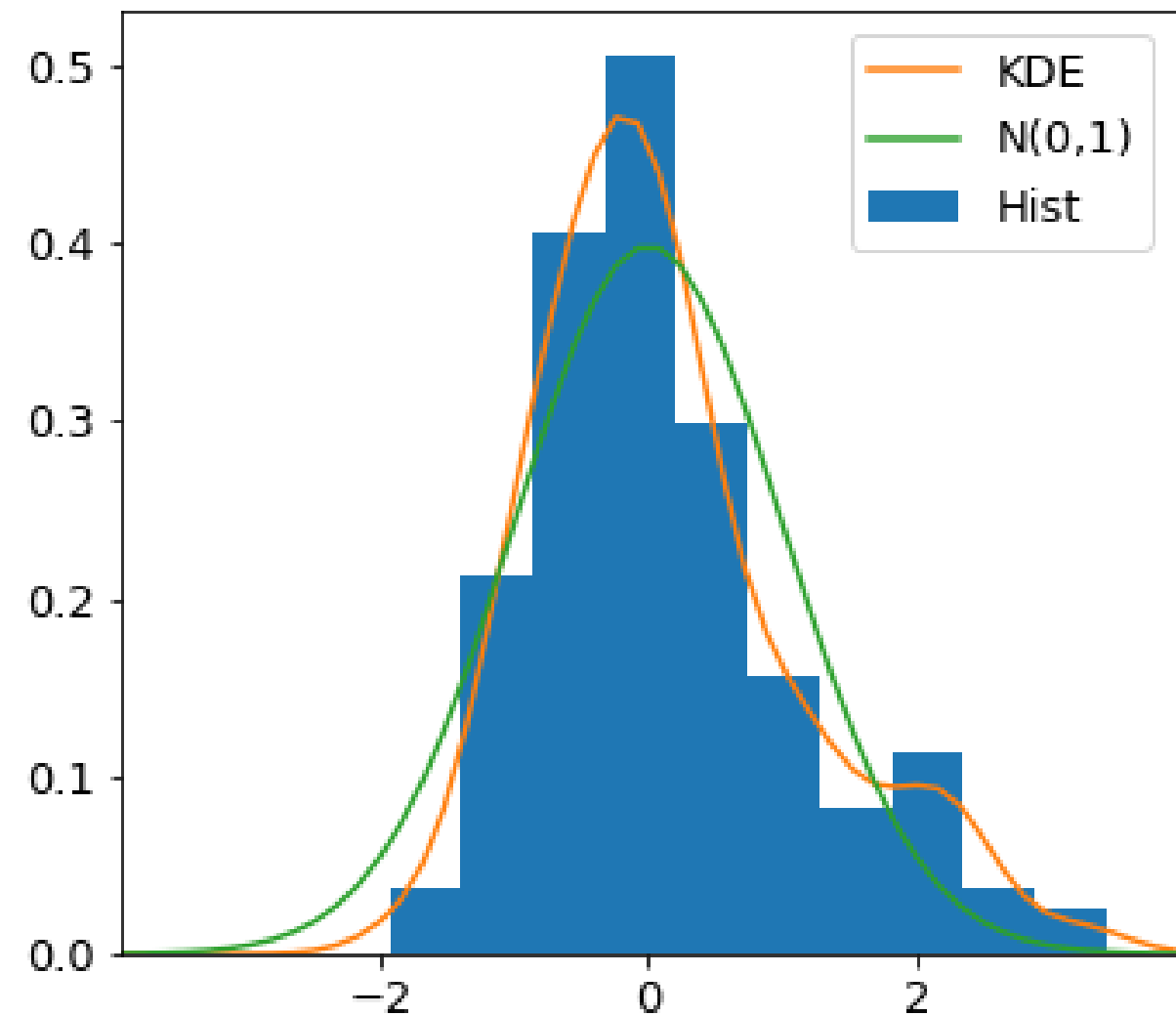


Histogram plus estimated density

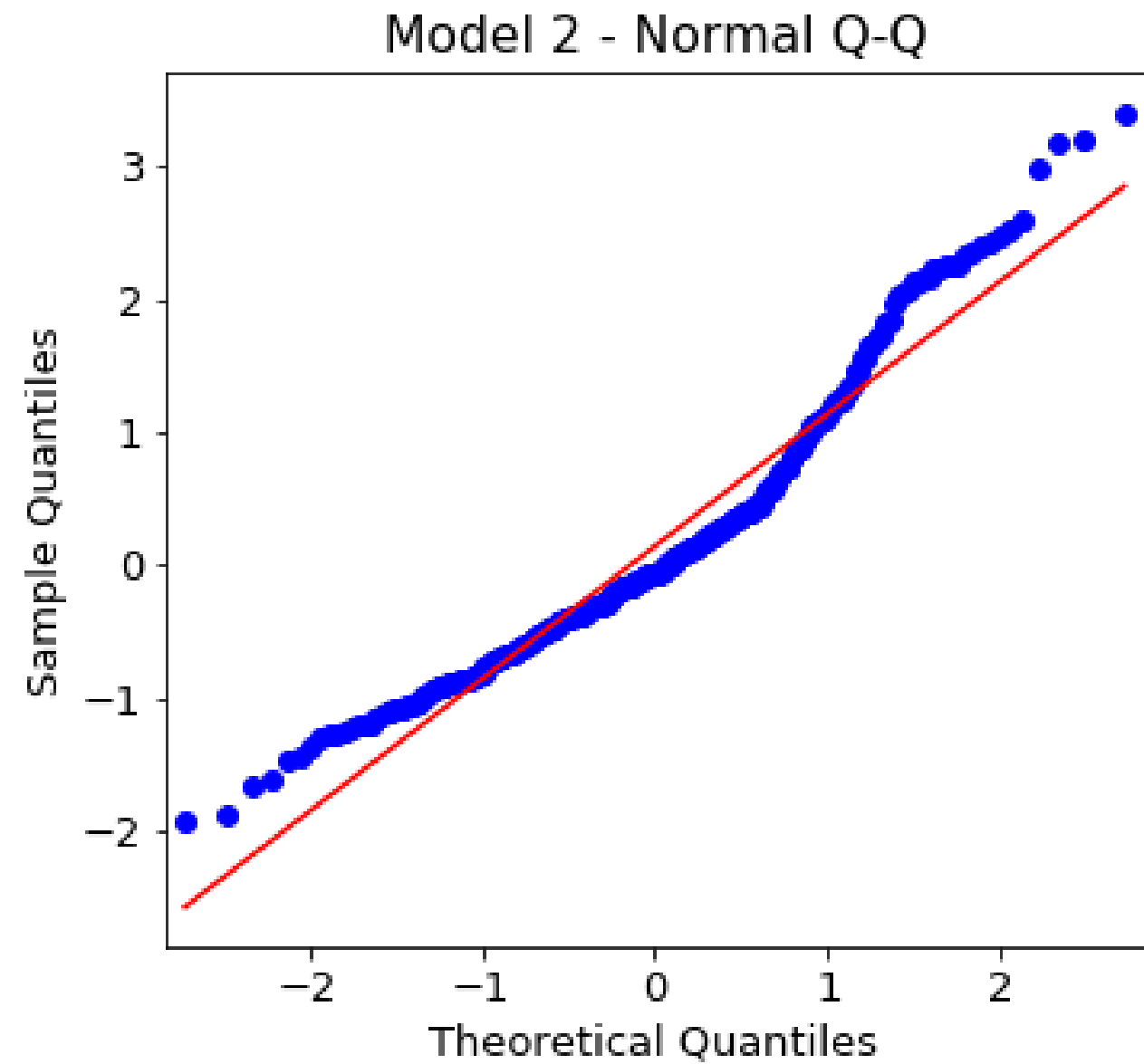
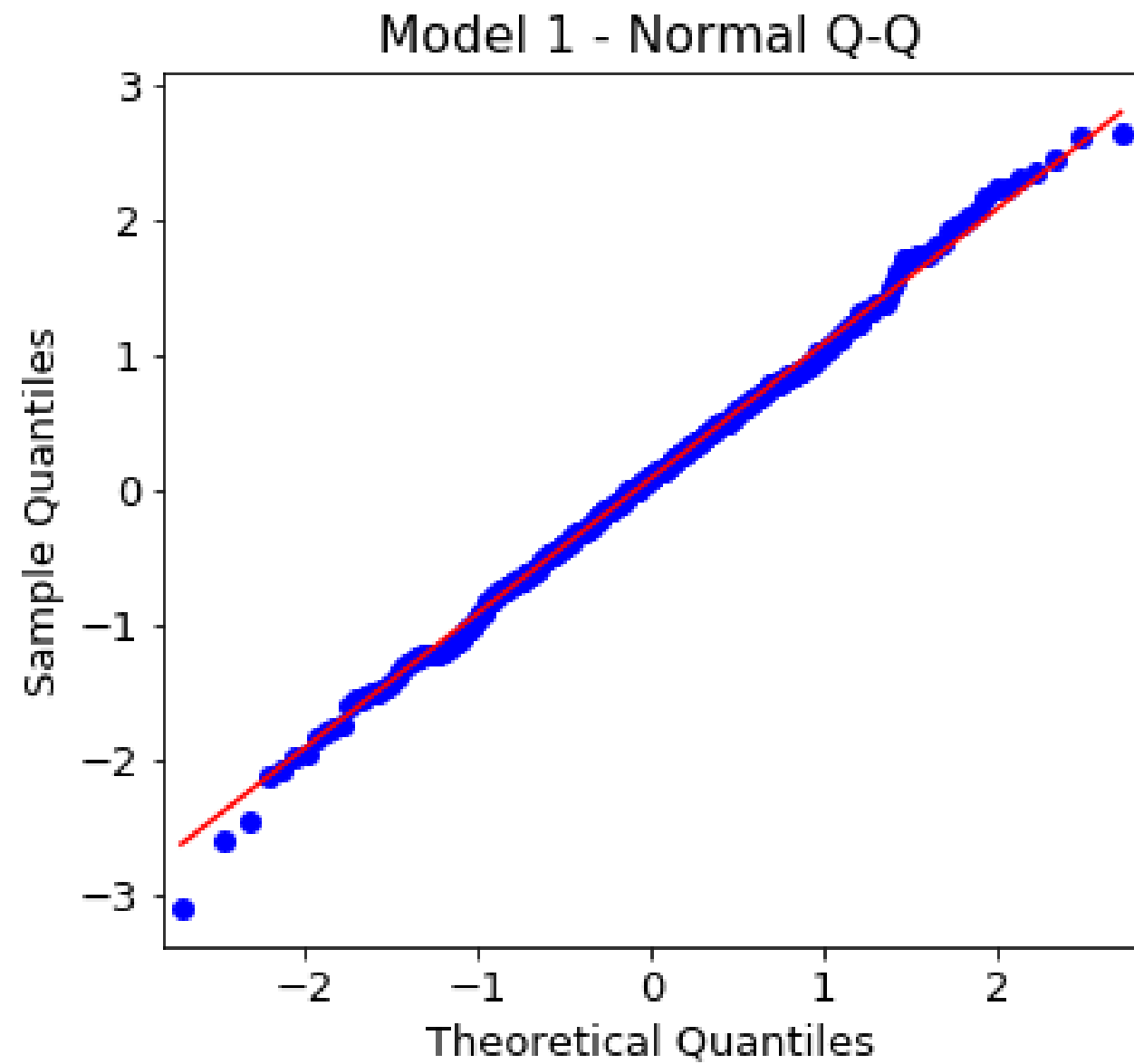
Model 1 - Histogram plus estimated density



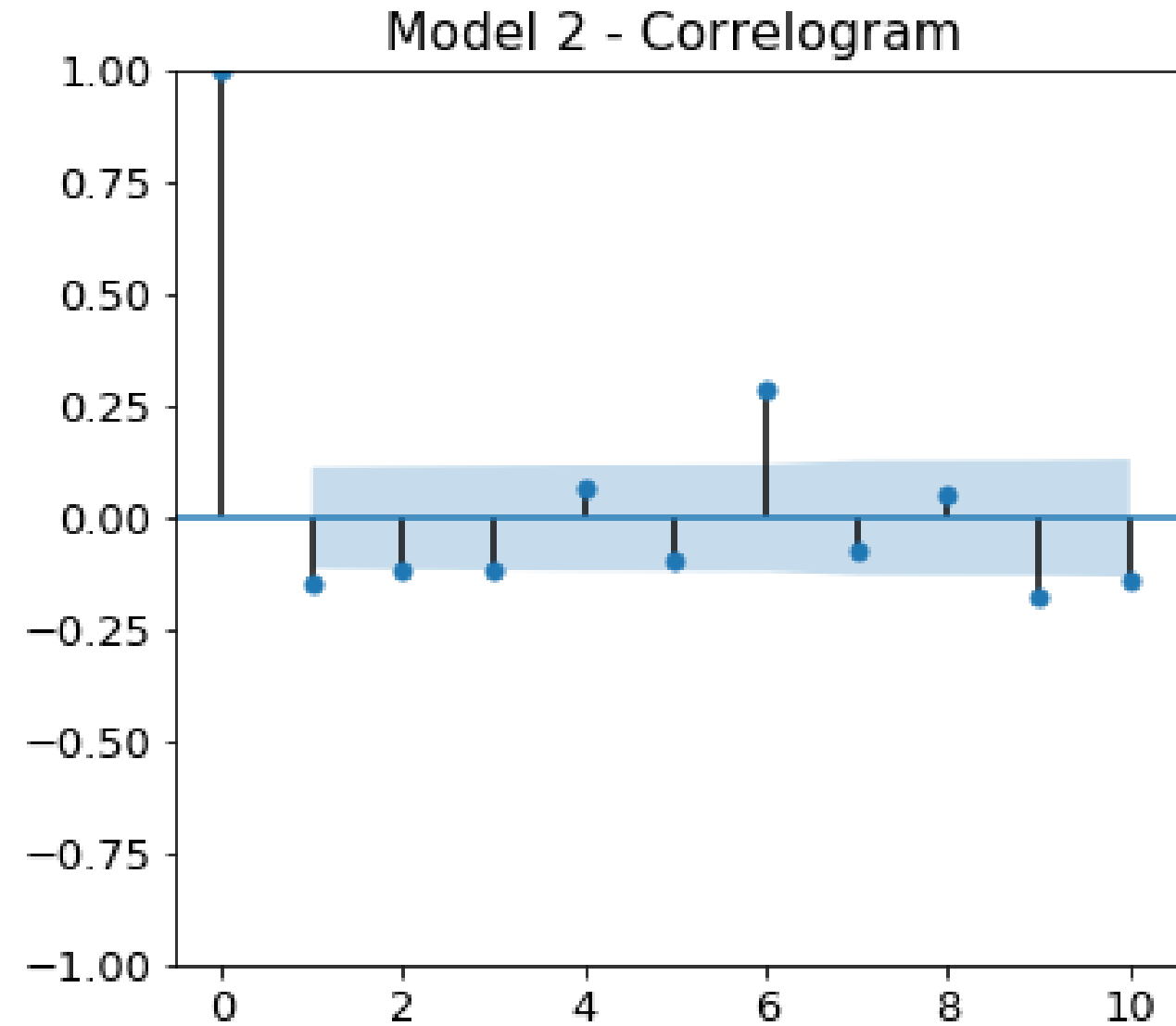
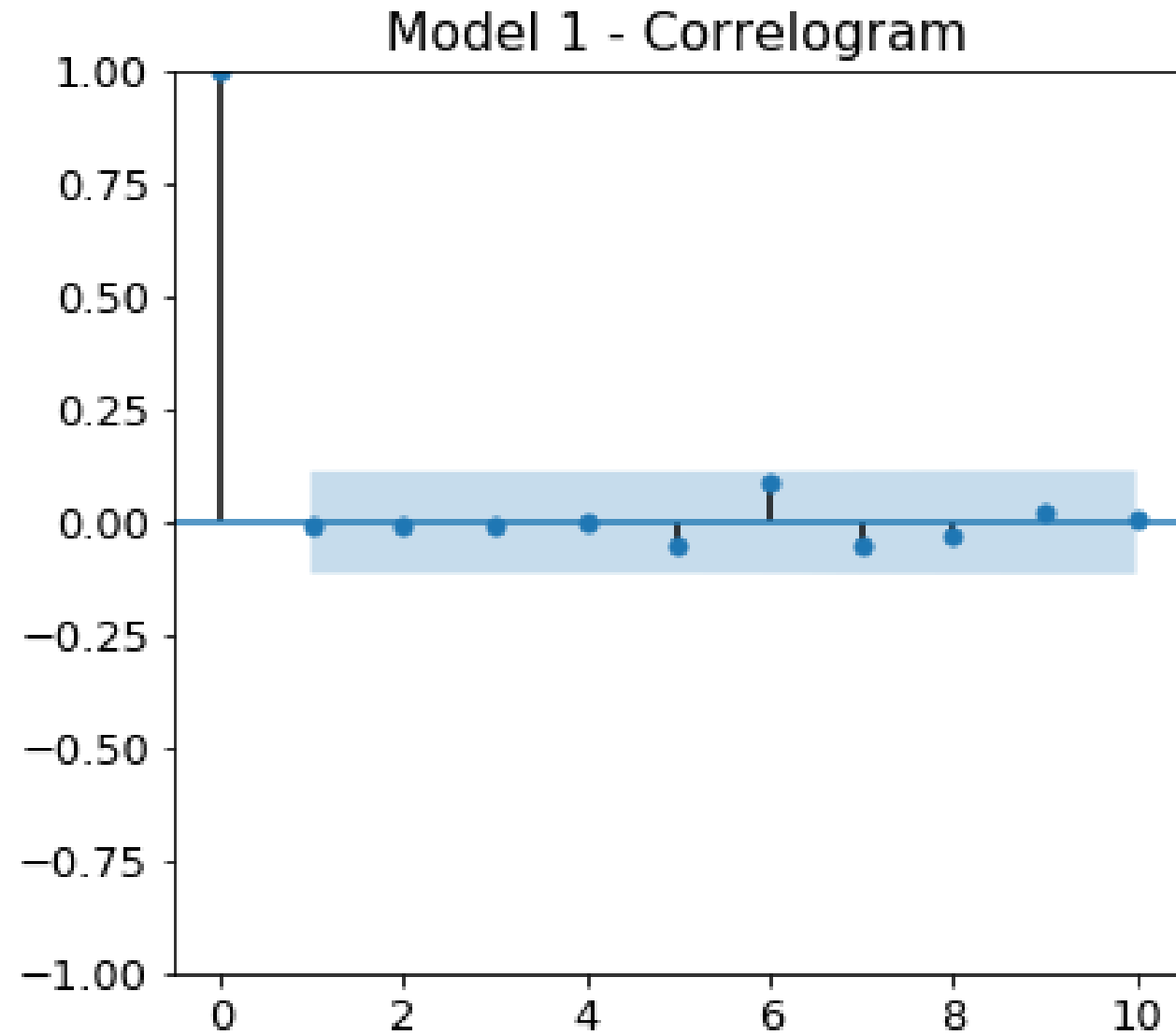
Model 2 - Histogram plus estimated density



Normal Q-Q



Correlogram



Summary statistics

```
print(results.summary())
```

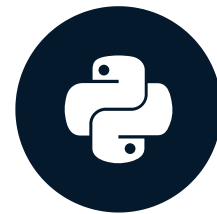
```
...
=====
Ljung-Box (Q):                32.10    Jarque-Bera (JB):                0.02
Prob(Q):                      0.81    Prob(JB):                      0.99
Heteroskedasticity (H):        1.28    Skew:                          -0.02
Prob(H) (two-sided):           0.21    Kurtosis:                      2.98
=====
```

- `Prob(Q)` - p-value for null hypothesis that residuals are uncorrelated
- `Prob(JB)` - p-value for null hypothesis that residuals are normal

Let's practice!
ARIMA MODELS IN PYTHON

Box-Jenkins method

ARIMA MODELS IN PYTHON



James Fulton

Climate informatics researcher

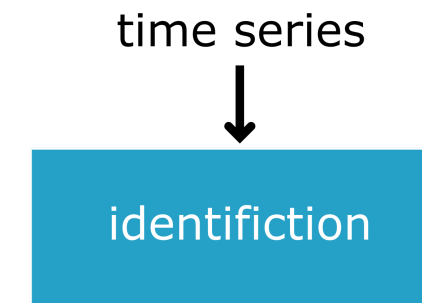
The Box-Jenkins method

From raw data \rightarrow production model

- identification
- estimation
- model diagnostics

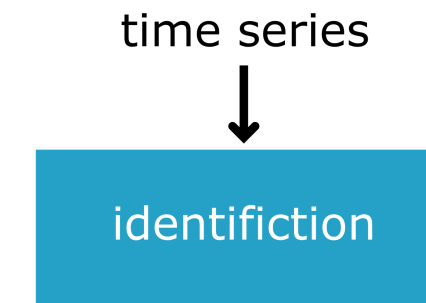
Identification

- Is the time series stationary?
- What differencing will make it stationary?
- What transforms will make it stationary?
- What values of p and q are most promising?



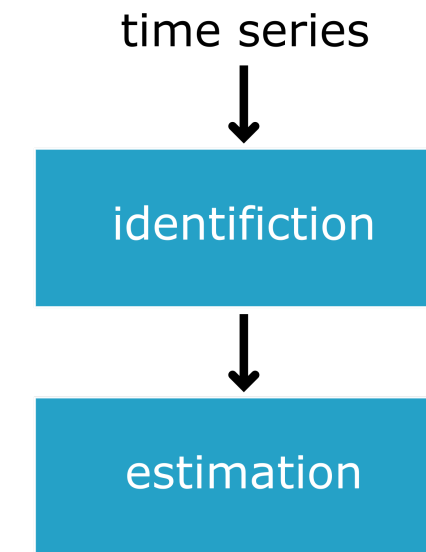
Identification tools

- Plot the time series
 - `df.plot()`
- Use augmented Dicky-Fuller test
 - `adfuller()`
- Use transforms and/or differencing
 - `df.diff()` , `np.log()` , `np.sqrt()`
- Plot ACF/PACF
 - `plot_acf()` , `plot_pacf()`



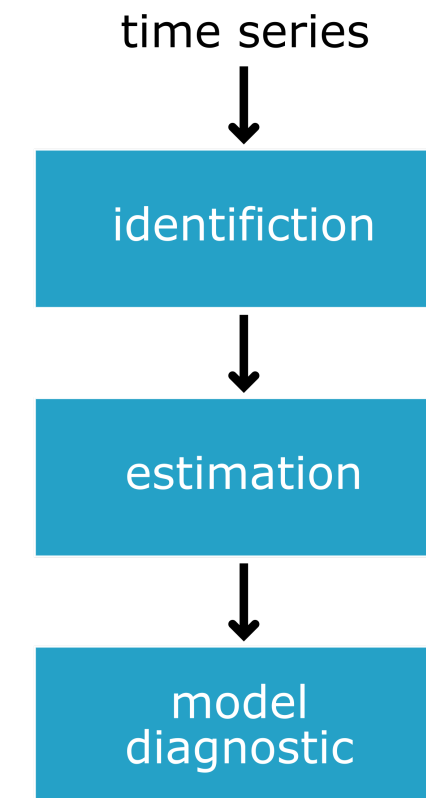
Estimation

- Use the data to train the model coefficients
- Done for us using `model.fit()`
- Choose between models using AIC and BIC
 - `results.aic` , `results.bic`

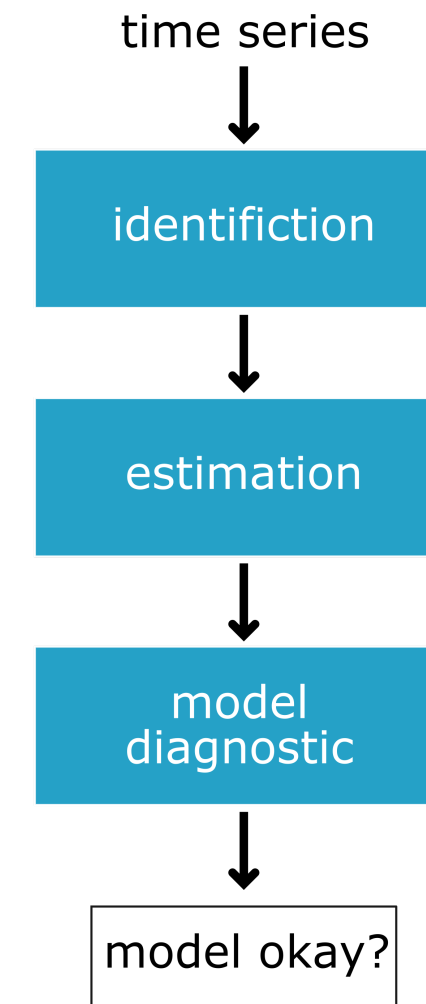


Model diagnostics

- Are the residuals uncorrelated
- Are residuals normally distributed
 - `results.plot_diagnostics()`
 - `results.summary()`

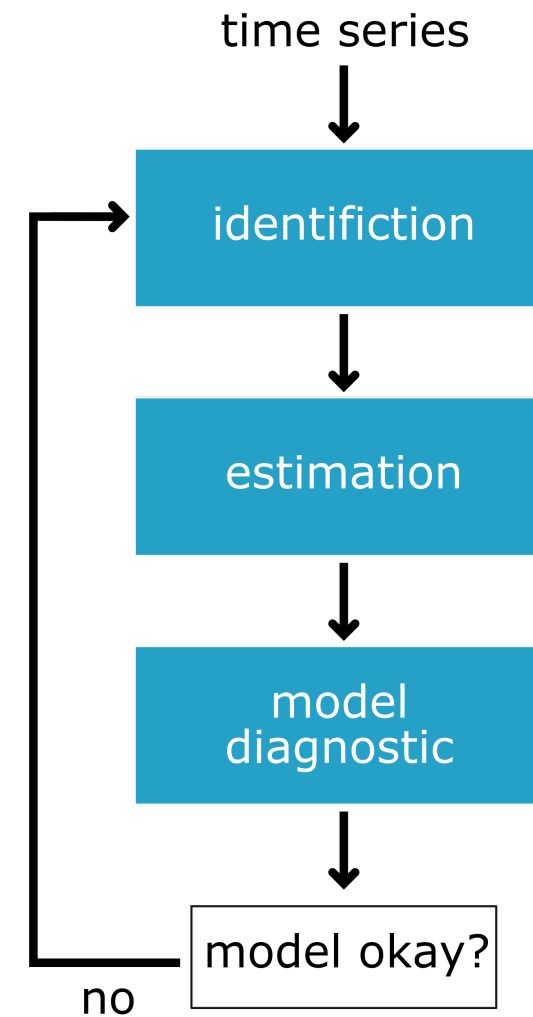


Decision



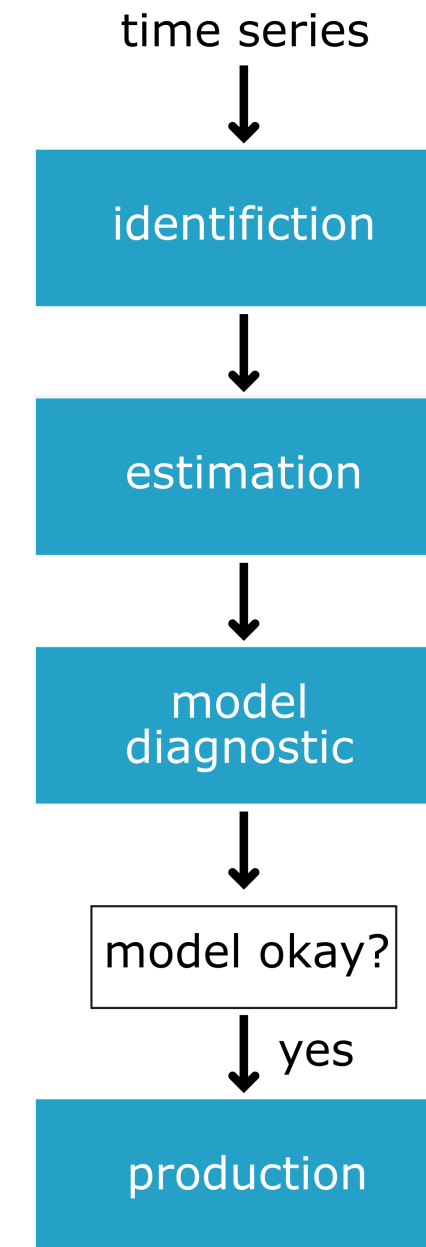
Repeat

- We go through the process again with more information
- Find a better model

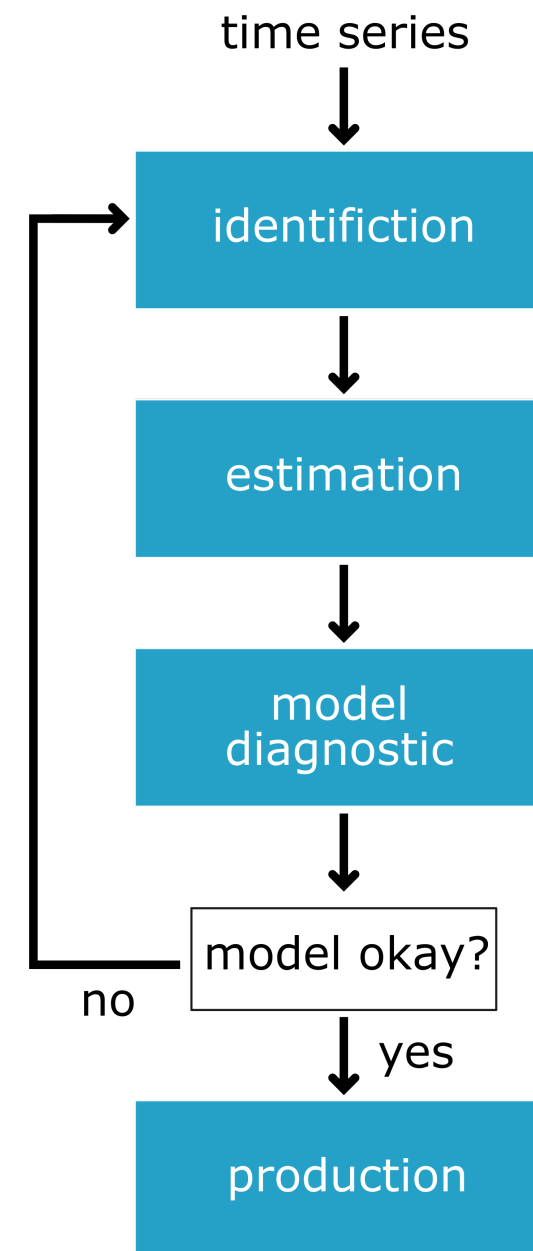


Production

- Ready to make forecasts
 - `results.get_forecast()`



Box-Jenkins



Let's practice!

ARIMA MODELS IN PYTHON