Fitting time series models

ARIMA MODELS IN PYTHON

James Fulton
Climate informatics researcher

Creating a model

```
from statsmodels.tsa.arima_model import ARMA
```

```
model = ARMA(timeseries, order=(p,q))
```

Creating AR and MA models

```
ar_model = ARMA(timeseries, order=(p,0))
```

```
ma_model = ARMA(timeseries, order=(0,q))
```

Fitting the model and fit summary

```
model = ARMA(timeseries, order=(2,1))
results = model.fit()
```

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

Fit summary

```
                              ARMA Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1000
Model:                     ARMA(2, 1)   Log Likelihood                 148.580
Method:                       css-mle   S.D. of innovations              0.208
Date:                Thu, 25 Apr 2019   AIC                           -287.159
Time:                        22:57:00   BIC                           -262.621
Sample:                             0   HQIC                          -277.833

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0017      0.012     -0.147      0.883      -0.025       0.021
ar.L1.y        0.5253      0.054      9.807      0.000       0.420       0.630
ar.L2.y       -0.2909      0.042     -6.850      0.000      -0.374      -0.208
ma.L1.y        0.3679      0.052      7.100      0.000       0.266       0.469
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            0.9029           -1.6194j            1.8541           -0.1690
AR.2            0.9029           +1.6194j            1.8541            0.1690
MA.1           -2.7184           +0.0000j            2.7184            0.5000
-----------------------------------------------------------------------------
```

Fit summary

```
                              ARMA Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1000
Model:                     ARMA(2, 1)   Log Likelihood                 148.580
Method:                       css-mle   S.D. of innovations              0.208
Date:                Thu, 25 Apr 2019   AIC                           -287.159
Time:                        22:57:00   BIC                           -262.621
Sample:                             0   HQIC                          -277.833
```

Fit summary

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0017      0.012     -0.147      0.883      -0.025       0.021
ar.L1.y        0.5253      0.054      9.807      0.000       0.420       0.630
ar.L2.y       -0.2909      0.042     -6.850      0.000      -0.374      -0.208
ma.L1.y        0.3679      0.052      7.100      0.000       0.266       0.469
```

Introduction to ARMAX models

- Exogenous ARMA
- Use external variables as well as time series
- ARMAX = ARMA + linear regression

ARMAX equation

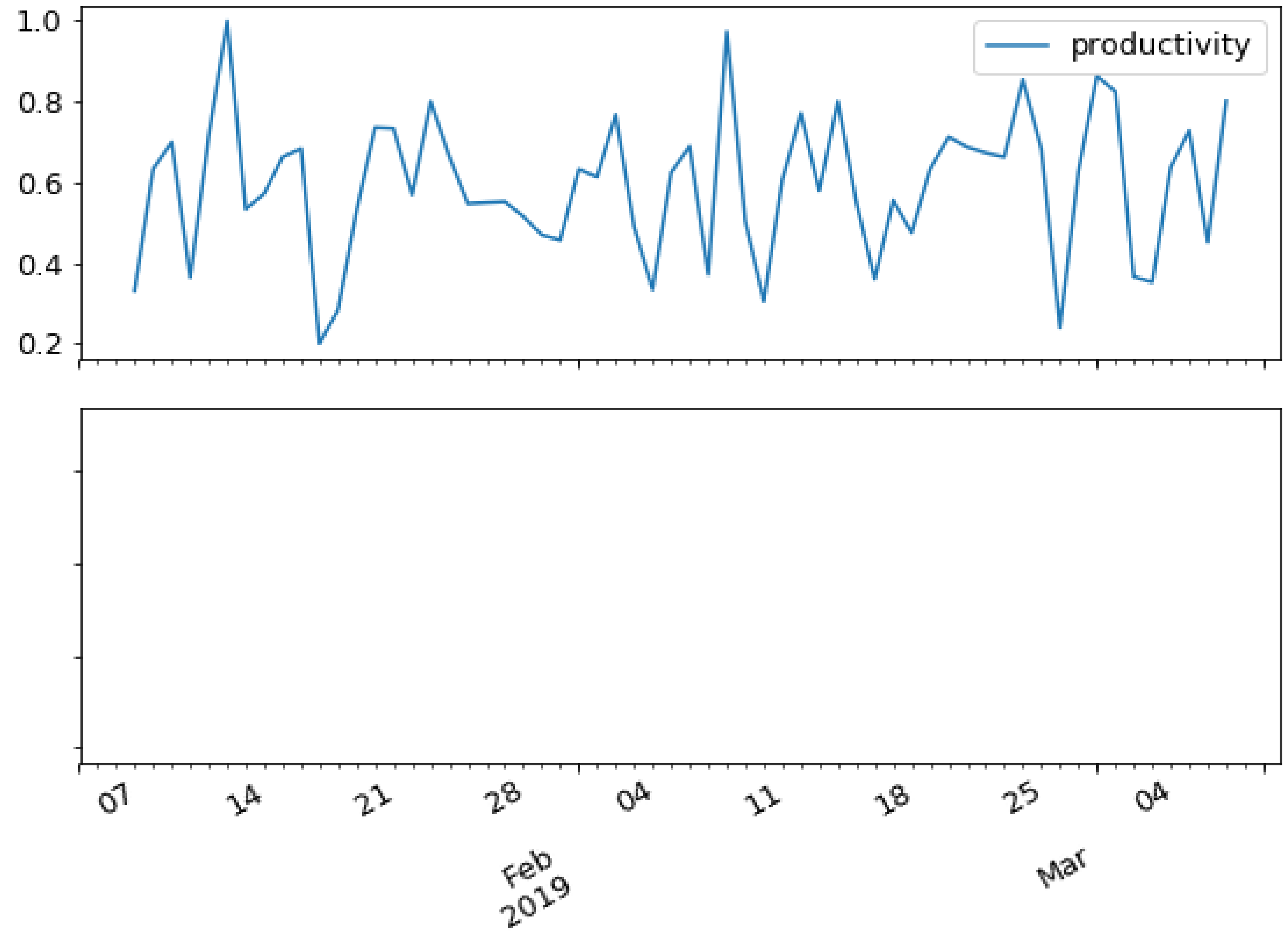
ARMA(1,1) model :

$$y_t = a_1 y_{t-1} + m_1 \epsilon_{t-1} + \epsilon_t$$

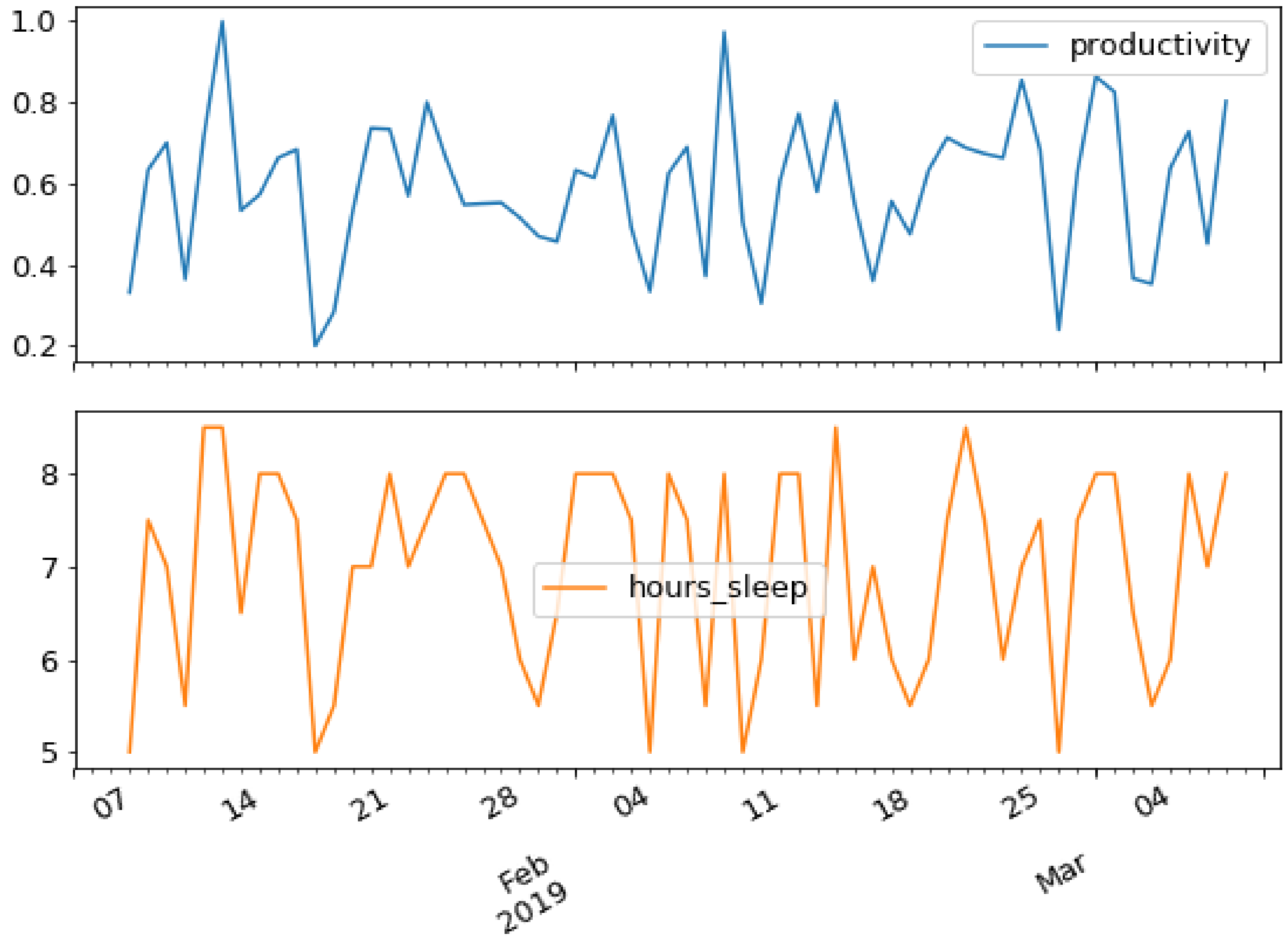
ARMAX(1,1) model :

$$y_t = x_1 z_t + a_1 y_{t-1} + m_1 \epsilon_{t-1} + \epsilon_t$$

ARMAX example



ARMAX example



Fitting ARMAX

```
# Instantiate the model
model = ARMA(df['productivity'], order=(2,1), exog=df['hours_sleep'])


# Fit the model
results = model.fit()
```

ARMAX summary

```
==================================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
const         -0.1936      0.092     -2.098      0.041      -0.375      -0.013
x1             0.1131      0.013      8.602      0.000       0.087       0.139
ar.L1.y        0.1917      0.252      0.760      0.450      -0.302       0.686
ar.L2.y       -0.3740      0.121     -3.079      0.003      -0.612      -0.136
ma.L1.y       -0.0740      0.259     -0.286      0.776      -0.581       0.433
```

Let's practice!

ARIMA MODELS IN PYTHON

Forecasting

ARIMA MODELS IN PYTHON

James Fulton
Climate informatics researcher

Predicting the next value

Take an AR(1) model

$$y_t = a_1 y_{t-1} + \epsilon_t$$

Predict next value

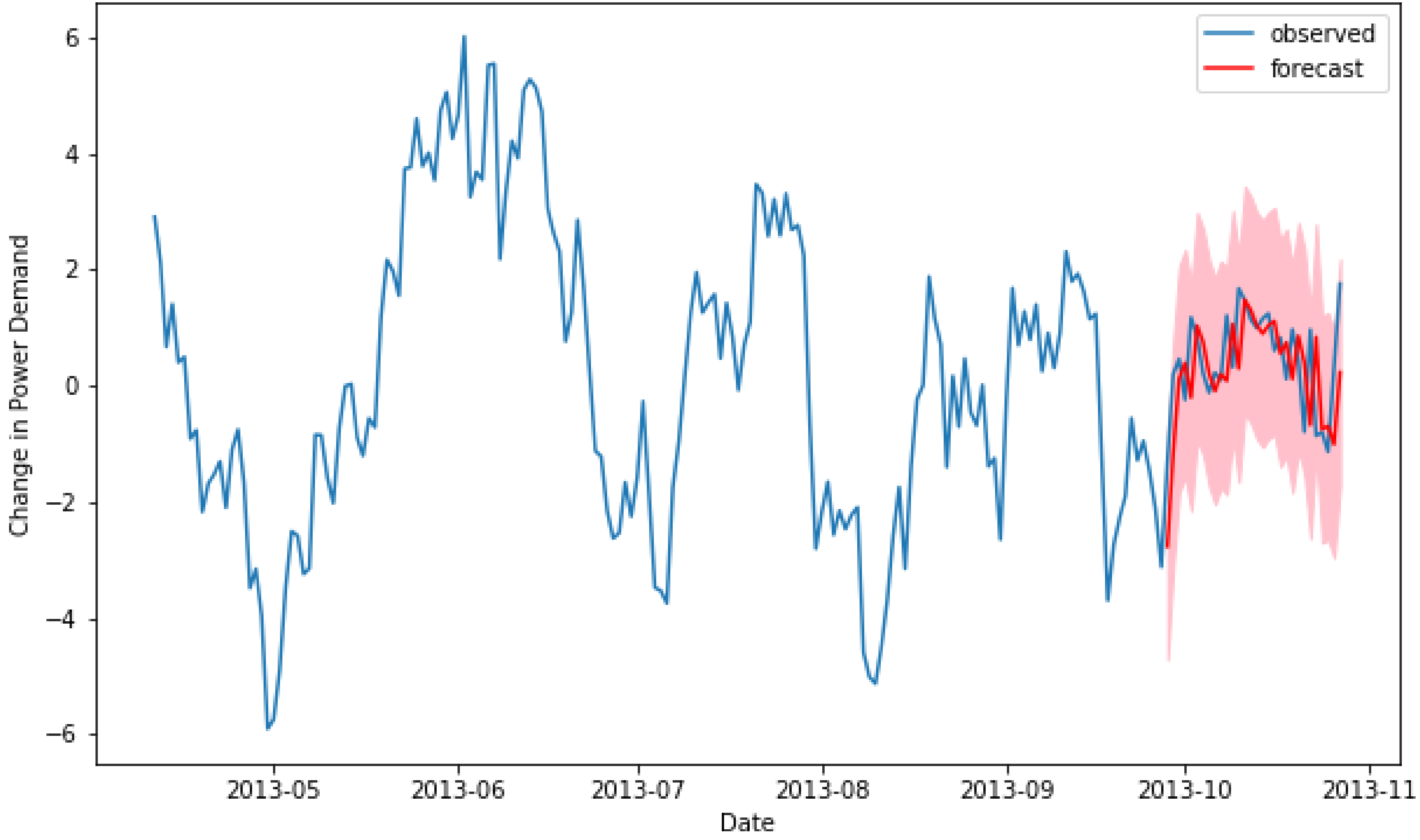
$$y_t = 0.6 \text{ x } 10 + \epsilon_t$$

$$y_t = 6.0 + \epsilon_t$$

Uncertainty on prediction

$$5.0 < y_t < 7.0$$

One-step-ahead predictions



Statsmodels SARIMAX class

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

# Just an ARMA(p,q) model
model = SARIMAX(df, order=(p,0,q))
```

Statsmodels SARIMAX class

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

# An ARMA(p,q) + constant model
model = SARIMAX(df, order=(p,0,q), trend='c')
```

Making one-step-ahead predictions

```
# Make predictions for last 25 values
results = model.fit()
# Make in-sample prediction
forecast = results.get_prediction(start=-25)
```

Making one-step-ahead predictions

```
# Make predictions for last 25 values
results = model.fit()
# Make in-sample prediction
forecast = results.get_prediction(start=-25)
# forecast mean
mean_forecast = forecast.predicted_mean
```

Predicted mean is a pandas series

```
2013-10-28    1.519368
2013-10-29    1.351082
2013-10-30    1.218016
```

Confidence intervals

```
# Get confidence intervals of forecasts
confidence_intervals = forecast.conf_int()
```

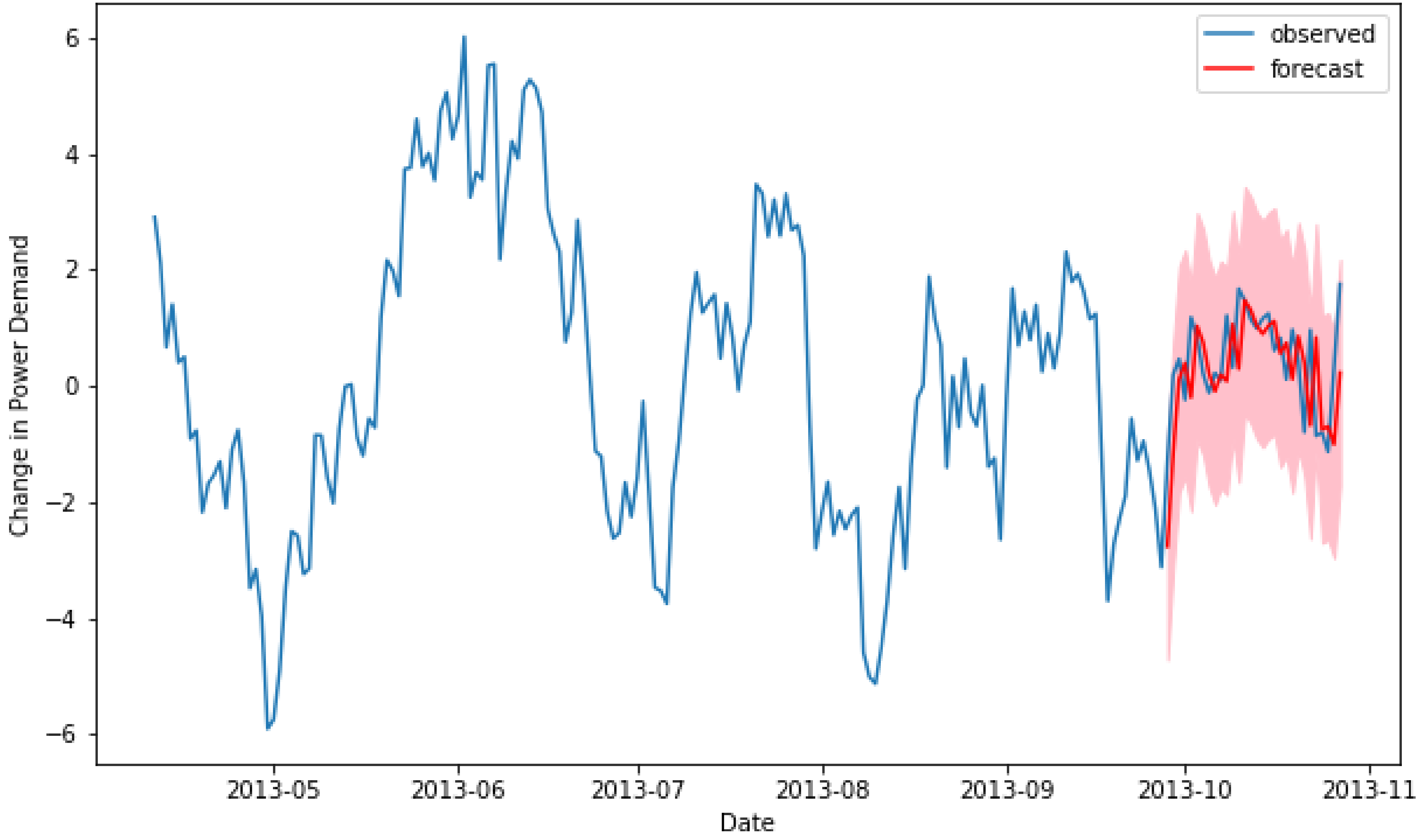
Confidence interval method returns `pandas` DataFrame

```
                lower y       upper y
2013-09-28    -4.720471     -0.815384
2013-09-29    -5.069875      0.112505
2013-09-30    -5.232837      0.766300
2013-10-01    -5.305814      1.282935
2013-10-02    -5.326956      1.703974
```

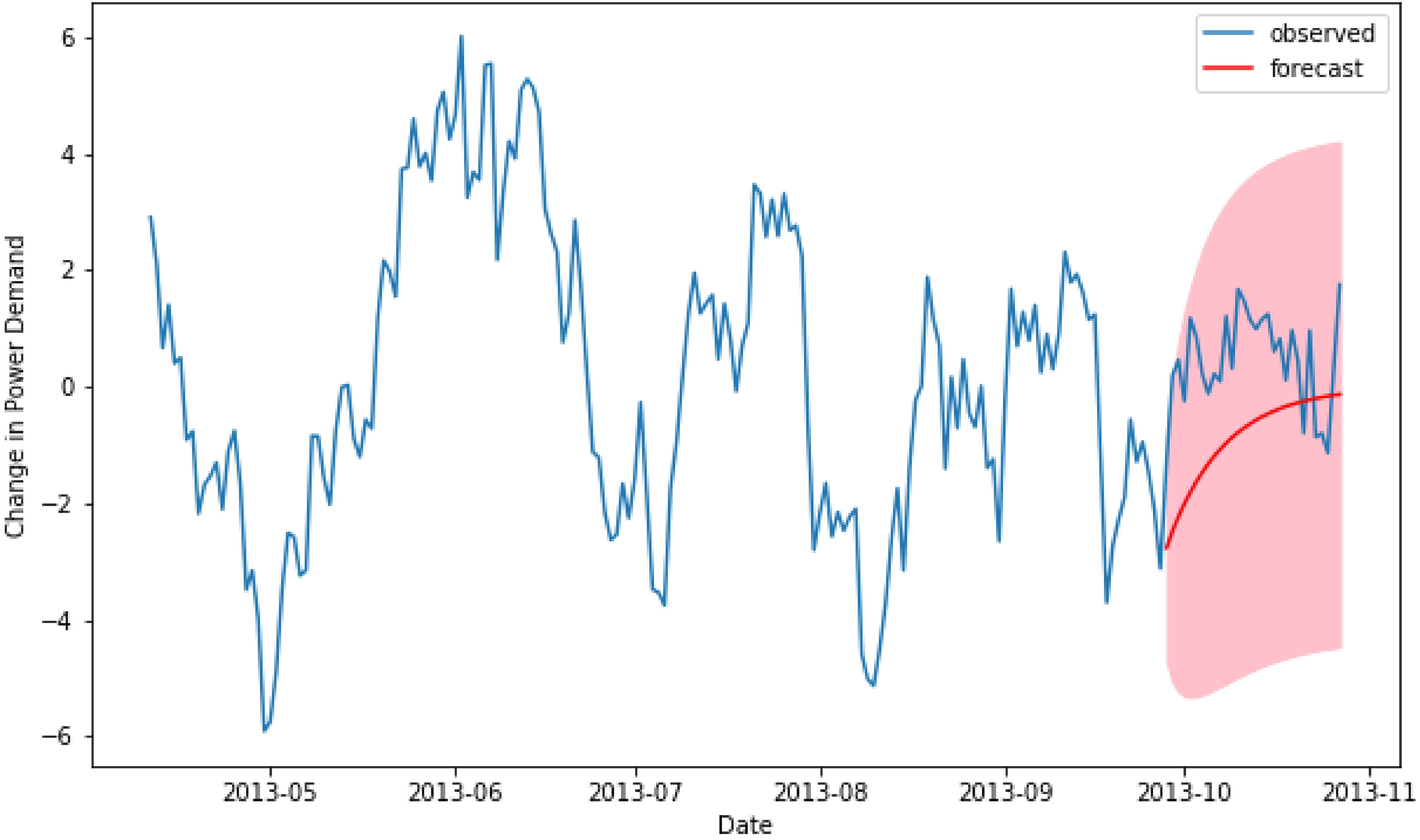
Plotting predictions

```
plt.figure()

# Plot prediction
plt.plot(dates,
         mean_forecast.values,
         color='red',
         label='forecast')
# Shade uncertainty area
plt.fill_between(dates, lower_limits, upper_limits, color='pink')

plt.show()
```

Plotting predictions



Dynamic predictions



Making dynamic predictions

```
results = model.fit()
forecast = results.get_prediction(start=-25, dynamic=True)
```

```
# forecast mean
mean_forecast = forecast.predicted_mean

# Get confidence intervals of forecasts
confidence_intervals = forecast.conf_int()
```

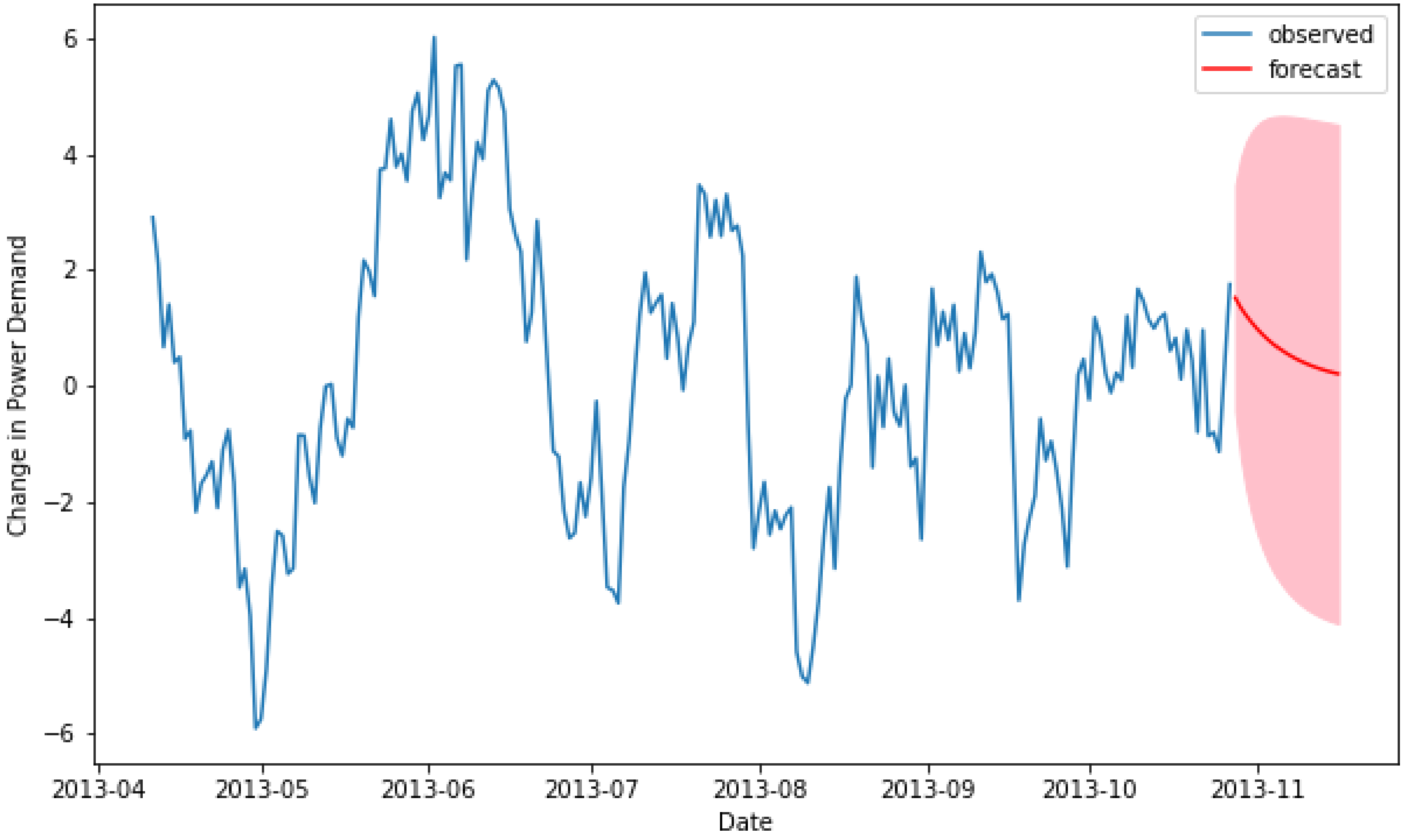
Forecasting out of sample

```
forecast = results.get_forecast(steps=20)
```

```
# forecast mean
mean_forecast = forecast.predicted_mean

# Get confidence intervals of forecasts
confidence_intervals = forecast.conf_int()
```

Forecasting out of sample

```
forecast = results.get_forecast(steps=20)
```



Let's practice!

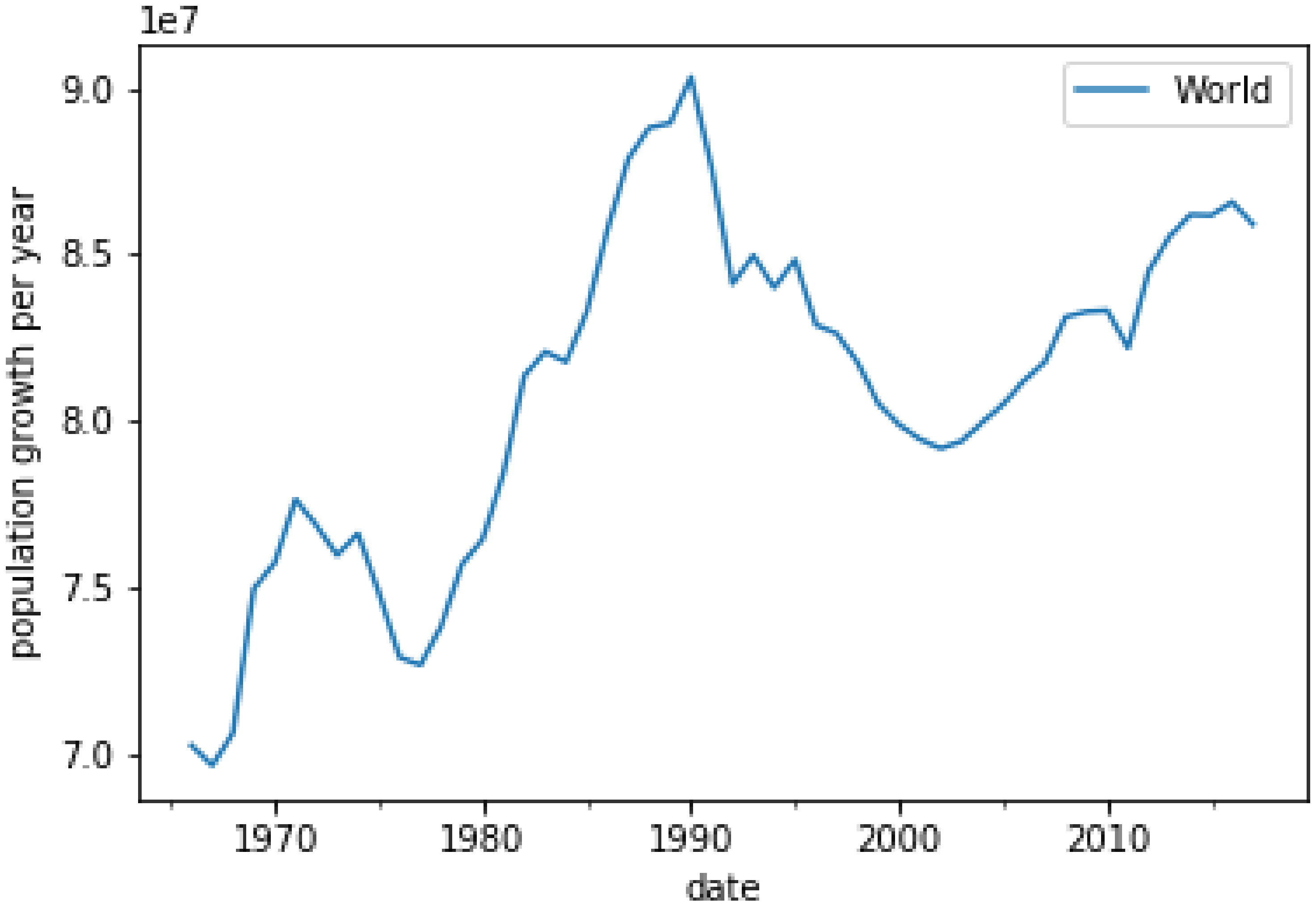
ARIMA MODELS IN PYTHON

Introduction to ARIMA models

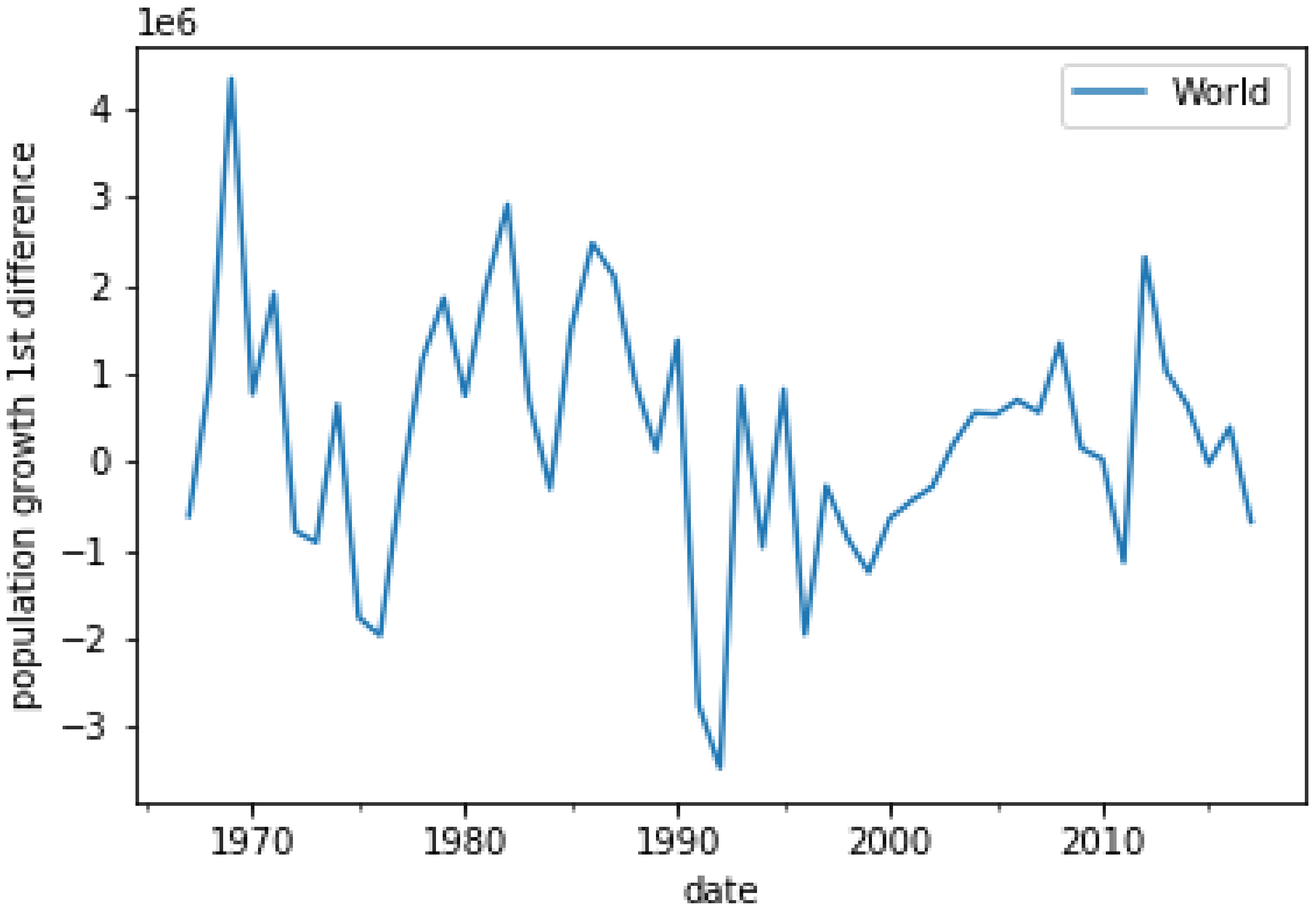
ARIMA MODELS IN PYTHON

James Fulton
Climate informatics researcher

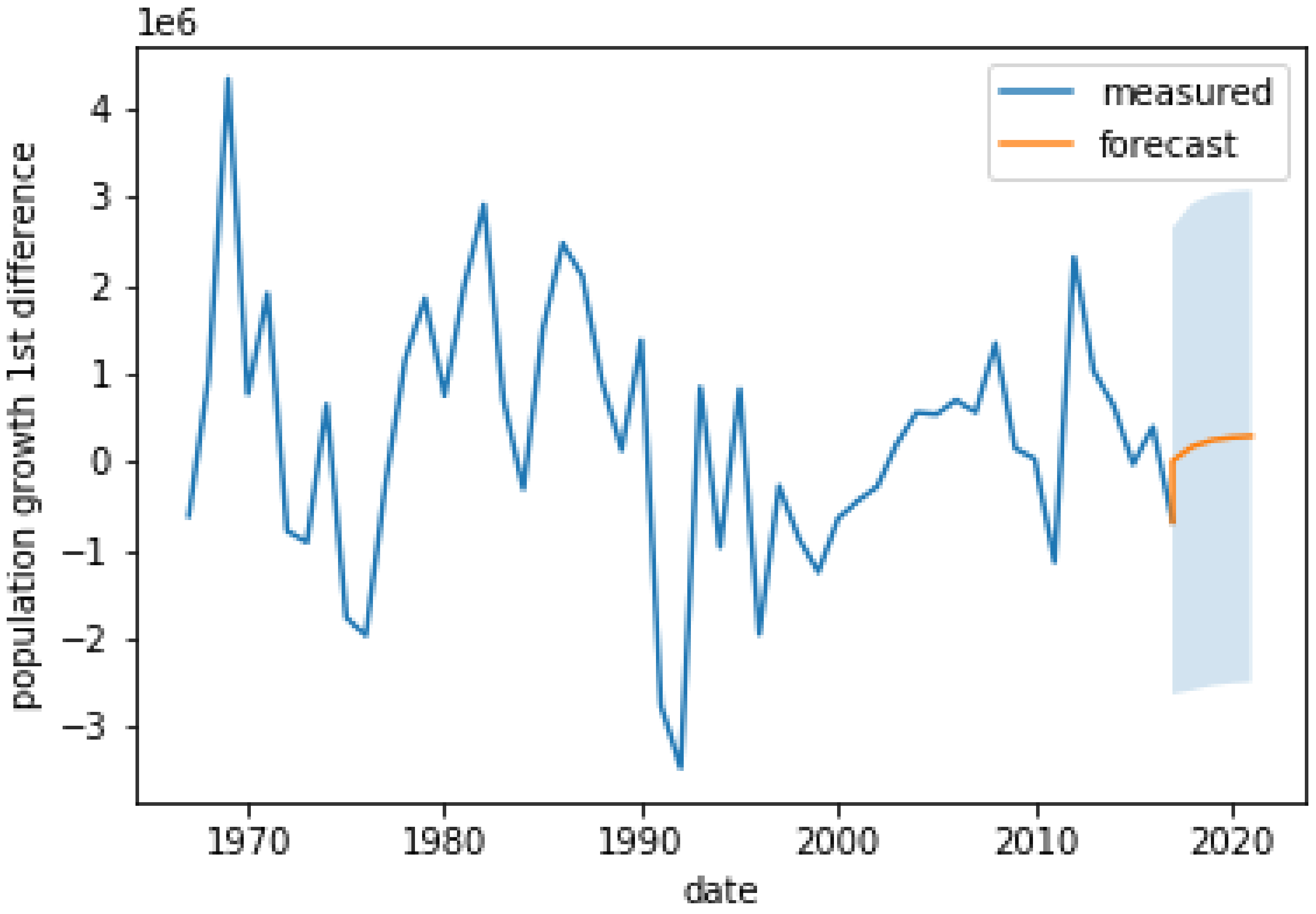
Non-stationary time series recap



Non-stationary time series recap



Forecast of differenced time series



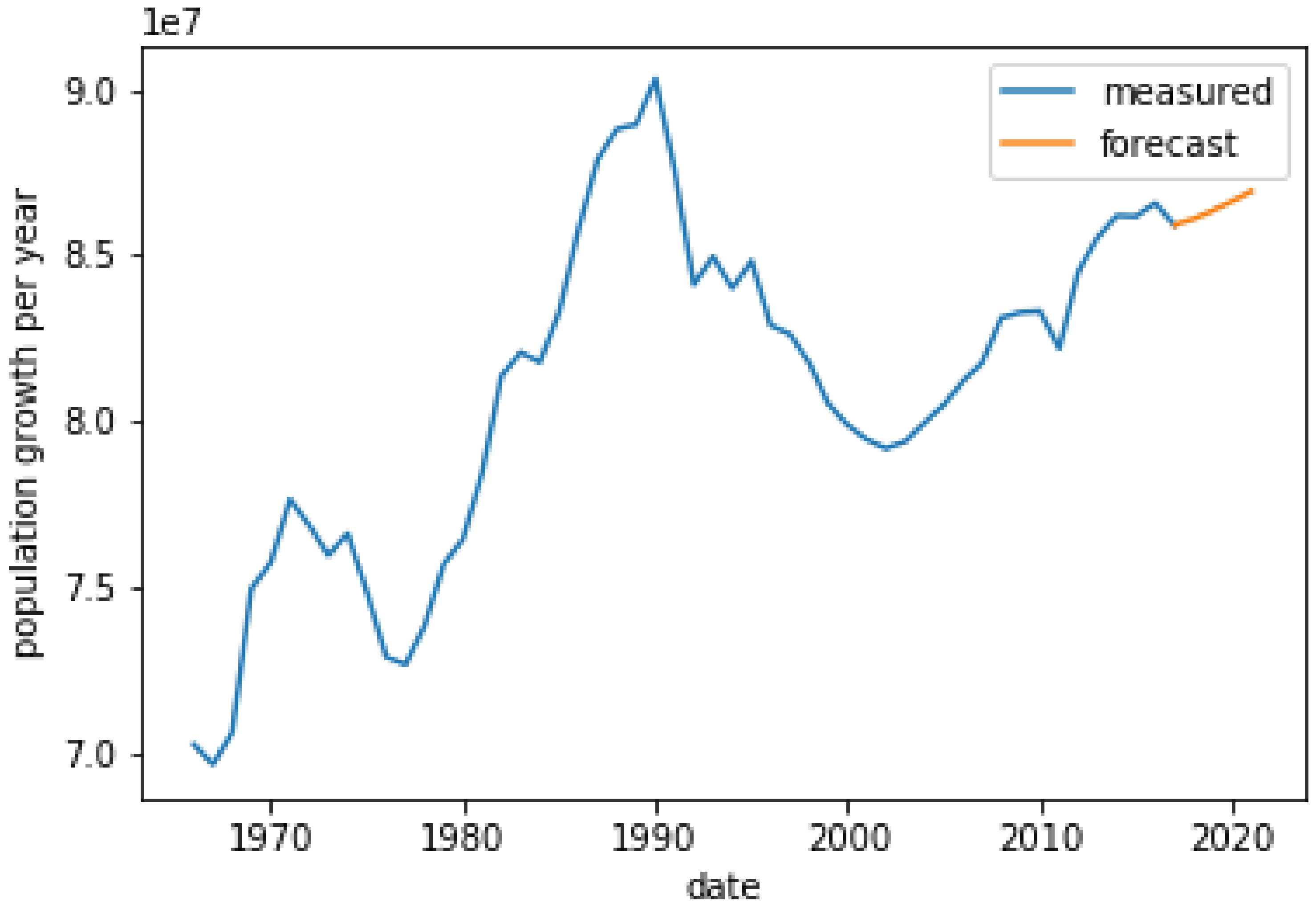
Reconstructing original time series after differencing

```
diff_forecast = results.get_forecast(steps=10).predicted_mean
from numpy import cumsum
mean_forecast = cumsum(diff_forecast)
```

Reconstructing original time series after differencing

```
diff_forecast = results.get_forecast(steps=10).predicted_mean
from numpy import cumsum
mean_forecast = cumsum(diff_forecast) + df.iloc[-1,0]
```

Reconstructing original time series after differencing



The ARIMA model

- Take the difference
- Fit ARMA model
- Integrate forecast

Can we avoid doing so much work?

Yes!

ARIMA - Autoregressive Integrated Moving Average

Using the ARIMA model

```
from statsmodels.tsa.statespace.sarimax import SARIMAX
model = SARIMAX(df, order =(p,d,q))
```

- p - number of autoregressive lags
- d - order of differencing
- q - number of moving average lags

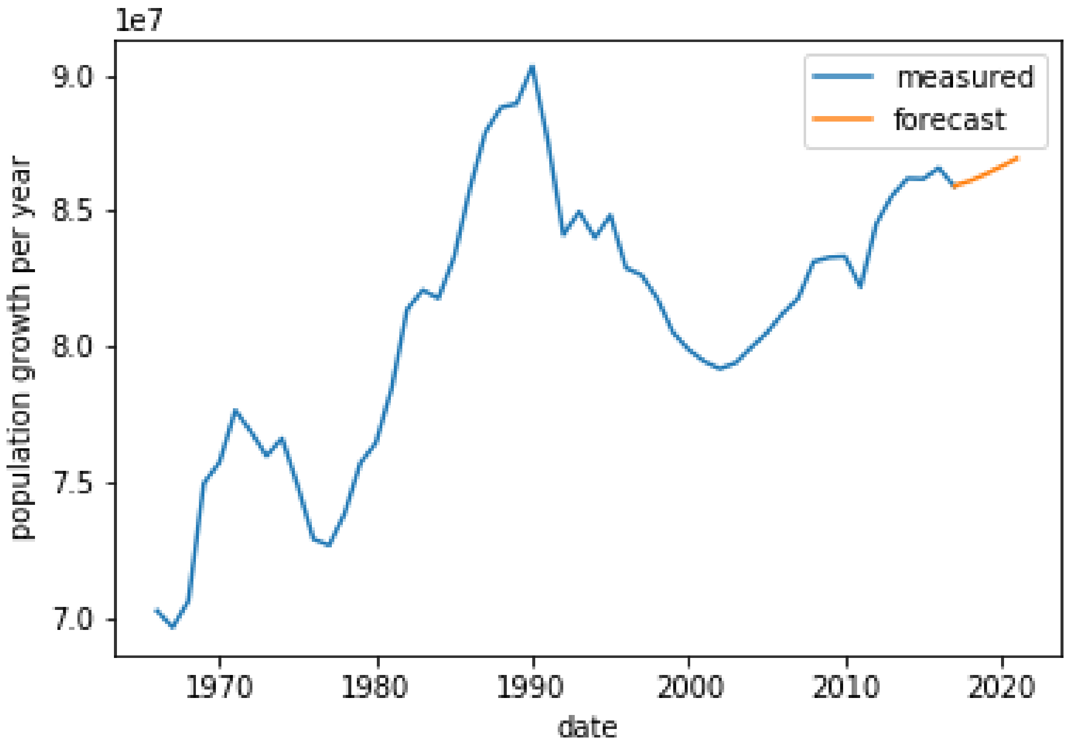
ARIMA$(p, 0, q)$ = ARMA(p, q)

Using the ARIMA model

```
# Create model
model = SARIMAX(df, order=(2,1,1))
# Fit model
model.fit()
# Make forecast
mean_forecast = results.get_forecast(steps=10).predicted_mean
```

Using the ARIMA model

```
# Make forecast
mean_forecast = results.get_forecast(steps=steps).predicted_mean
```



Picking the difference order

```
adf = adfuller(df.iloc[:,0])
print('ADF Statistic:', adf[0])
print('p-value:', adf[1])
```

```
ADF Statistic: -2.674
p-value: 0.0784
```

```
adf = adfuller(df.diff().dropna().iloc[:,0])
print('ADF Statistic:', adf[0])
print('p-value:', adf[1])
```

```
ADF Statistic: -4.978
p-value: 2.44e-05
```

Picking the difference order

```
model = SARIMAX(df, order=(p,1,q))
```

Let's practice!

ARIMA MODELS IN PYTHON