# Seasonal time series

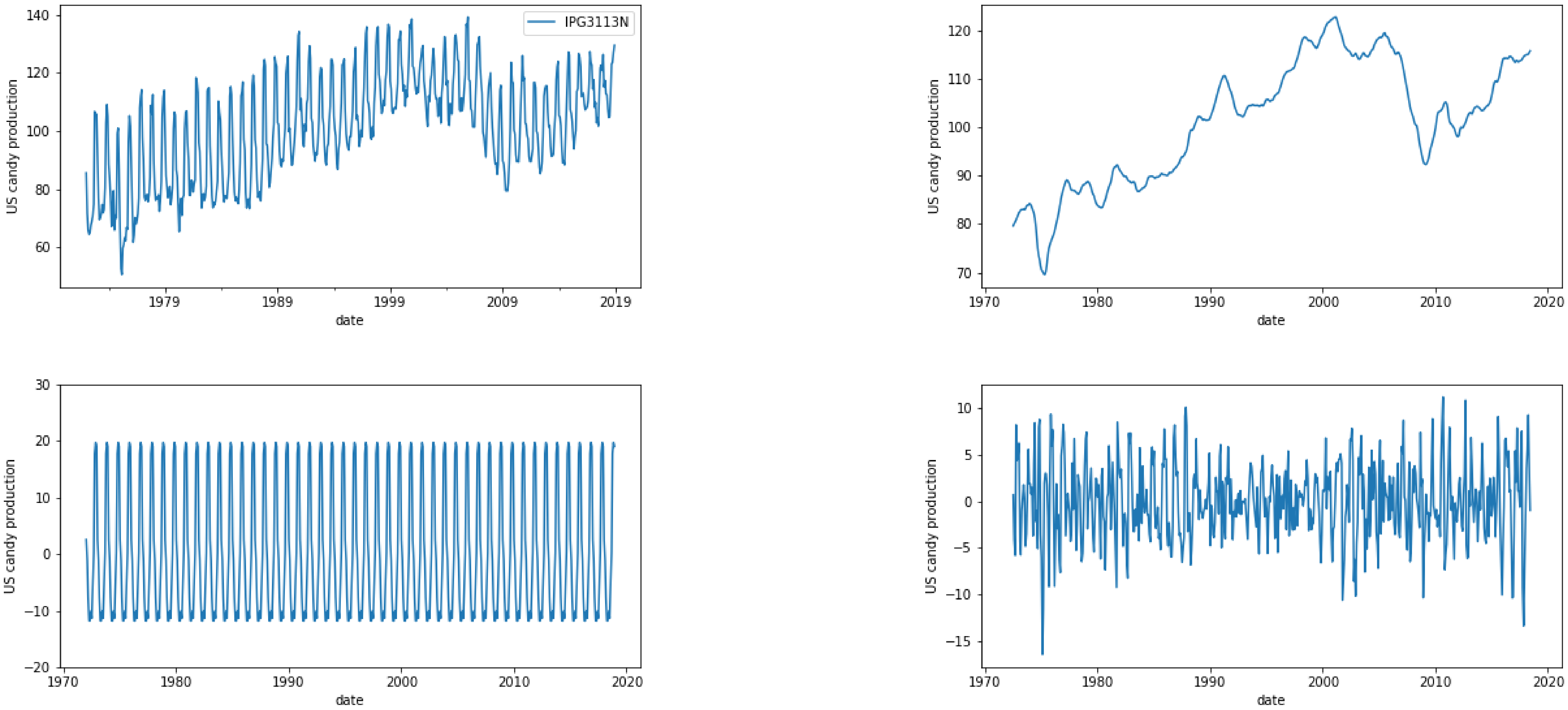
FORECASTING USING ARIMA MODELS IN PYTHON

**James Fulton**
Climate informatics researcher

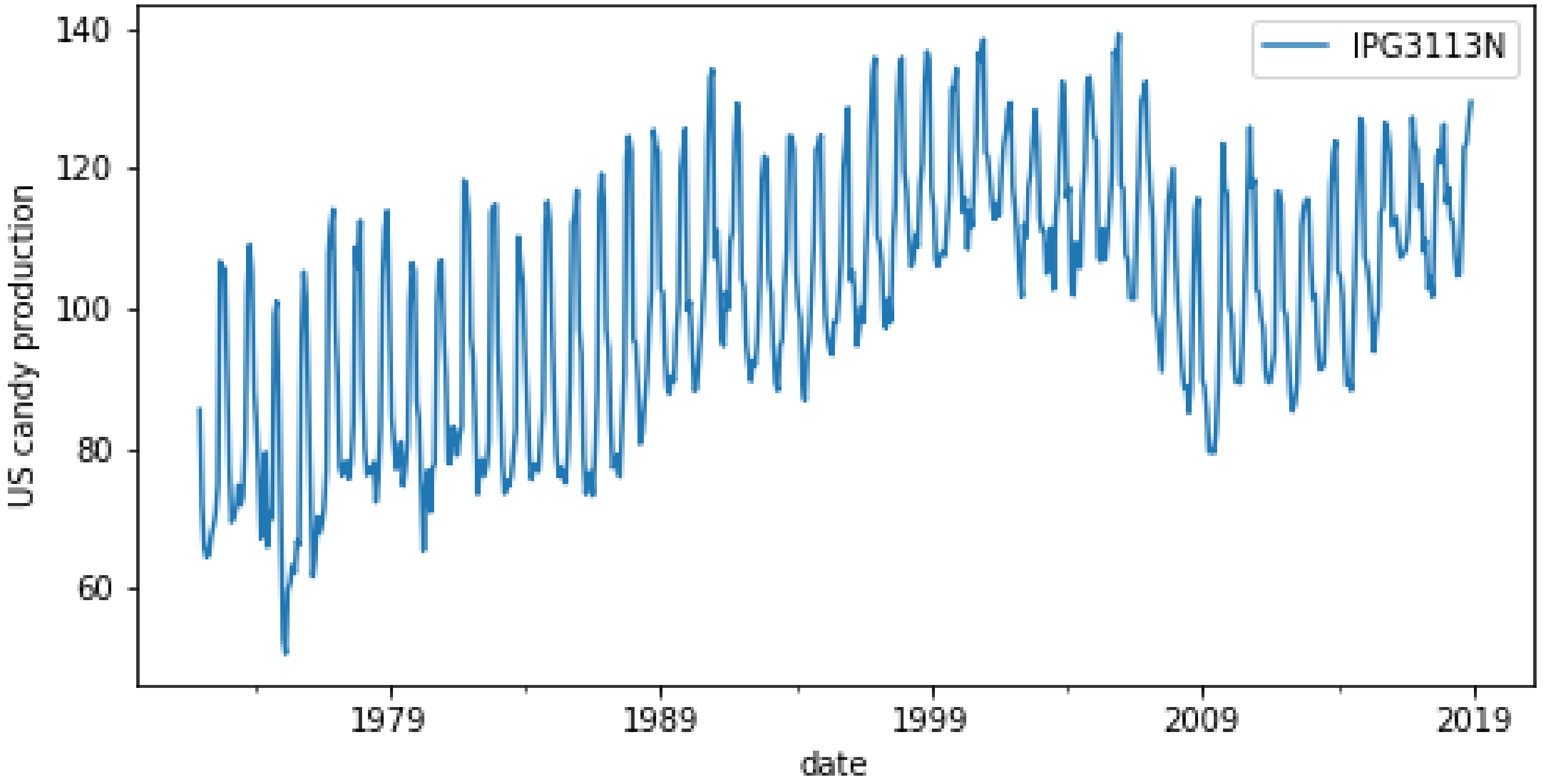
# Seasonal data

- Has predictable and repeated patterns
- Repeats after any amount of time

# Seasonal decomposition



# Seasonal decomposition




time series = trend + seasonal + residual

# Seasonal decomposition using statsmodels

```
# Import
from statsmodels.tsa.seasonal import seasonal_decompose
```

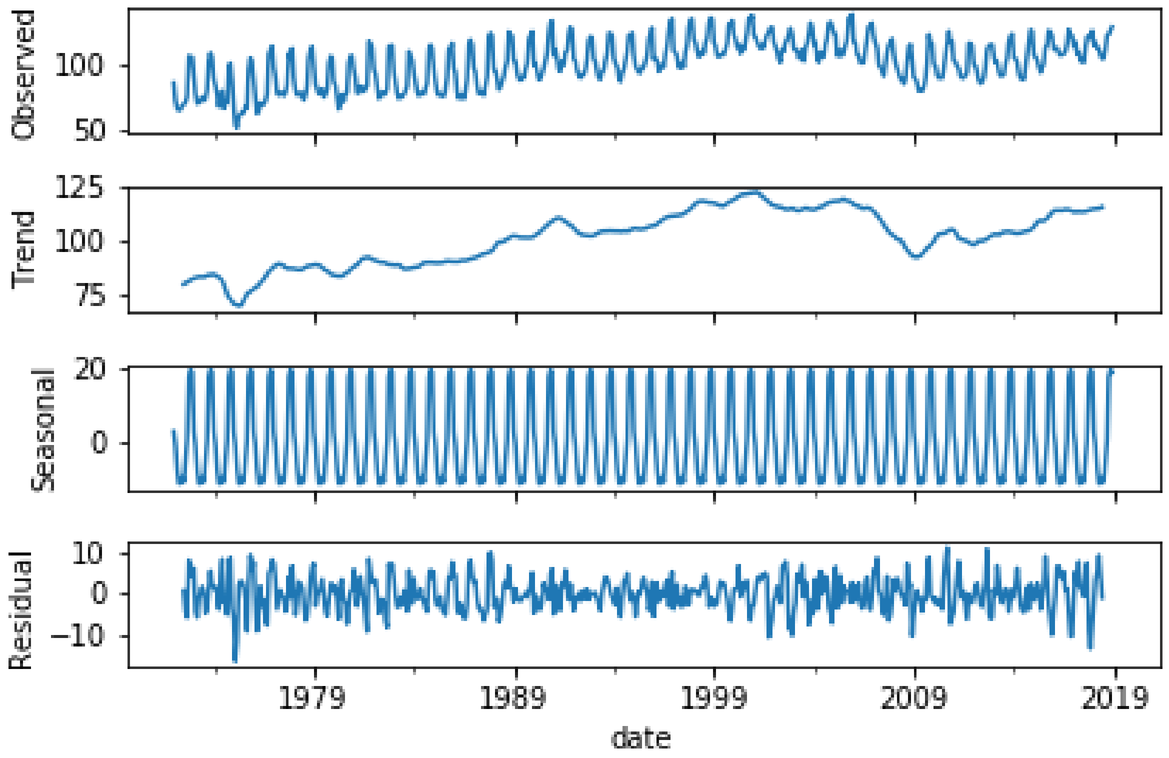
```
# Decompose data
decomp_results = seasonal_decompose(df['IPG3113N'], freq=12)
```

```
type(decomp_results)
```

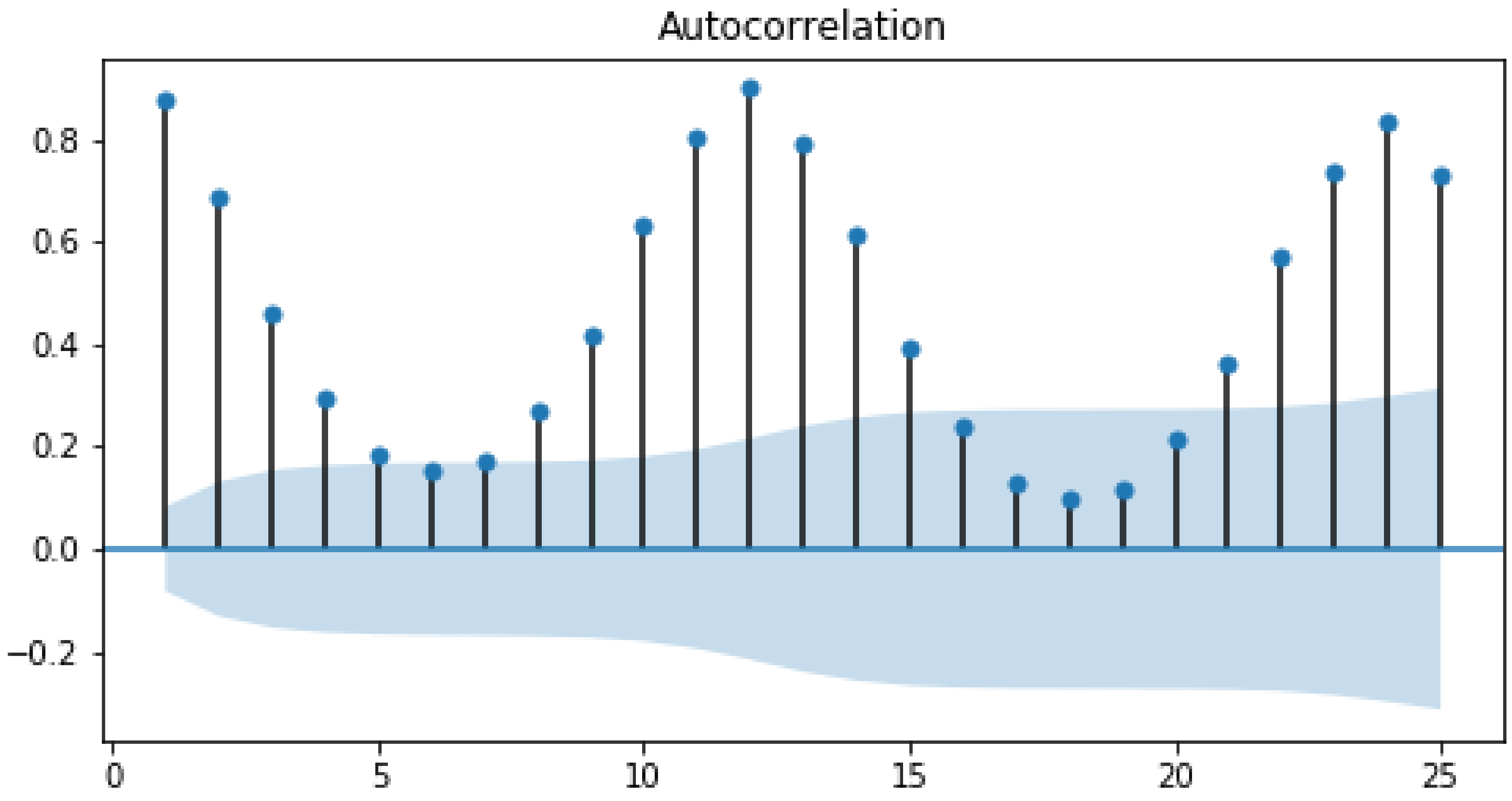
```
statsmodels.tsa.seasonal.DecomposeResult
```

# Seasonal decomposition using statsmodels

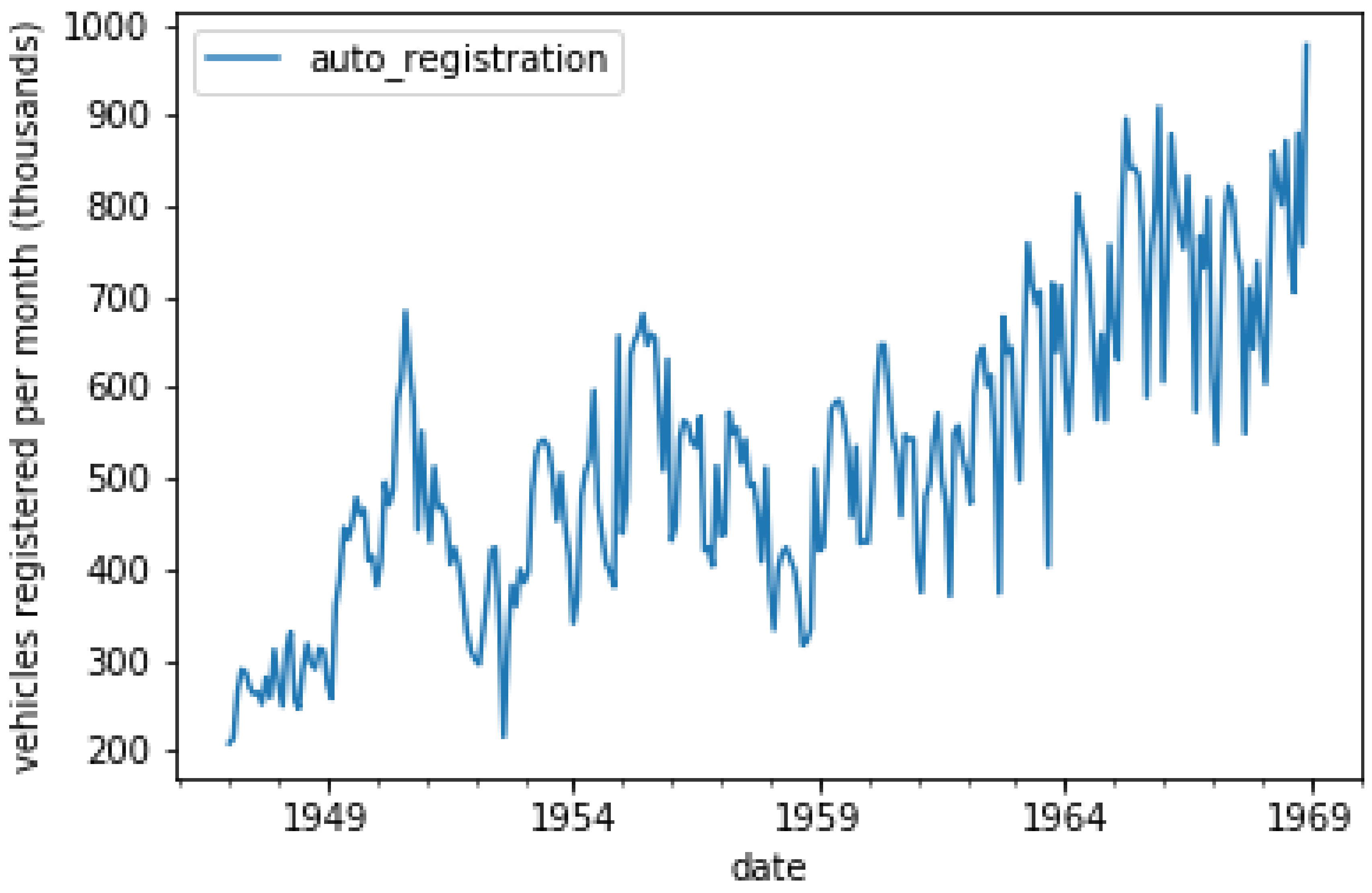
```
# Plot decomposed data
decomp_results.plot()
plt.show()
```



# Finding seasonal period using ACF

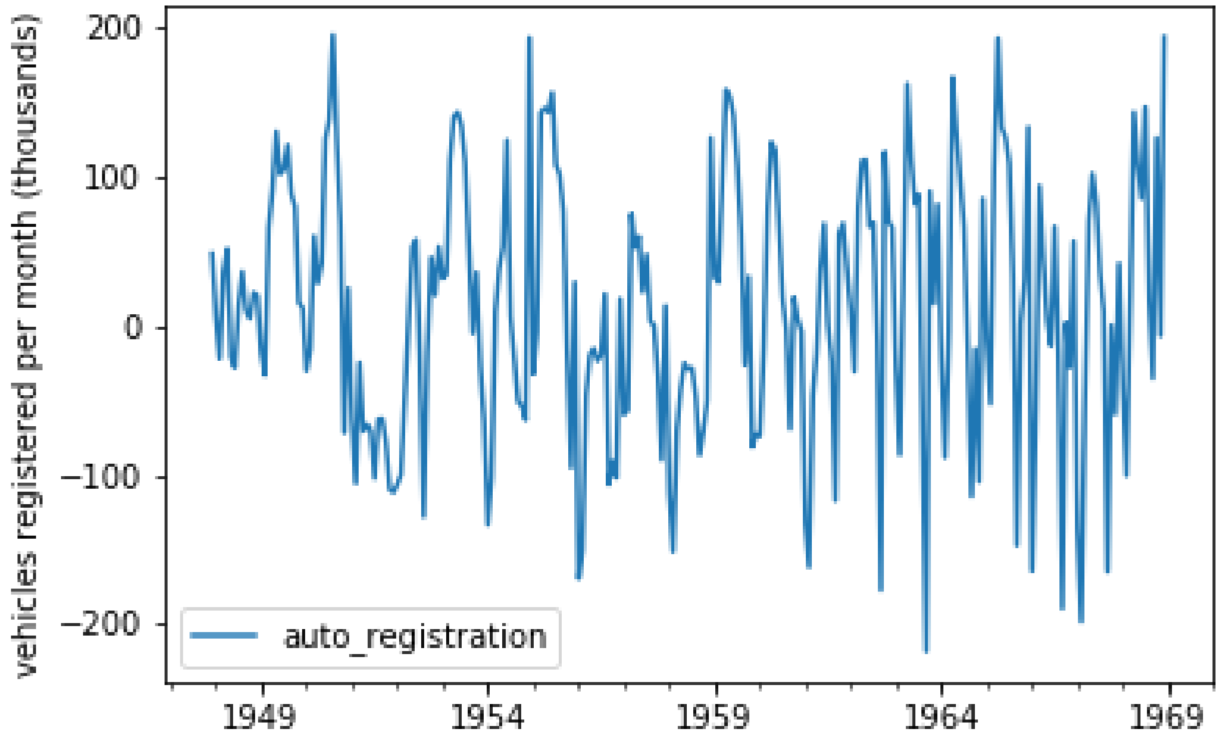


# Identifying seasonal data using ACF



# Detrending time series

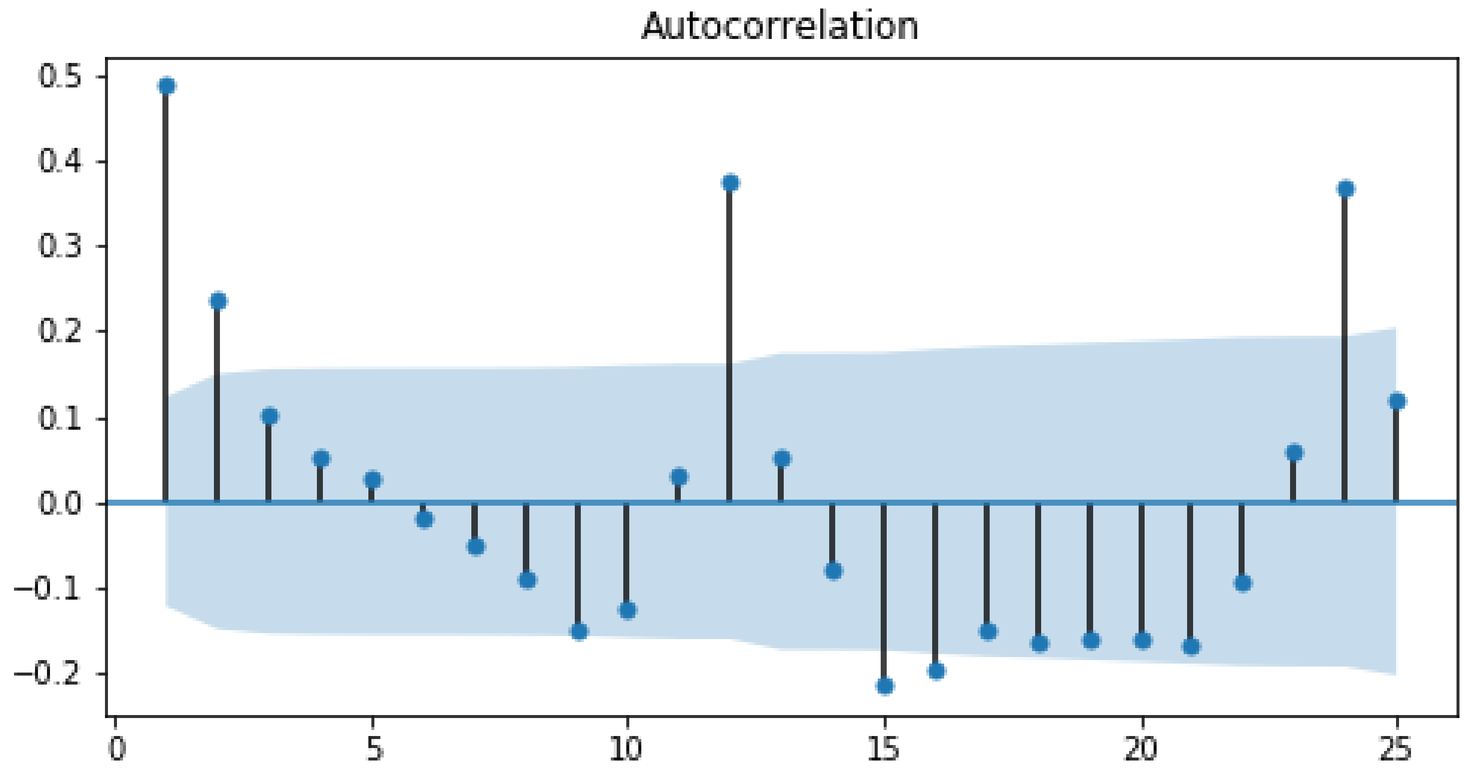
```
# Subtract long rolling average over N steps
df = df - df.rolling(N).mean()
# Drop NaN values
df = df.dropna()
```



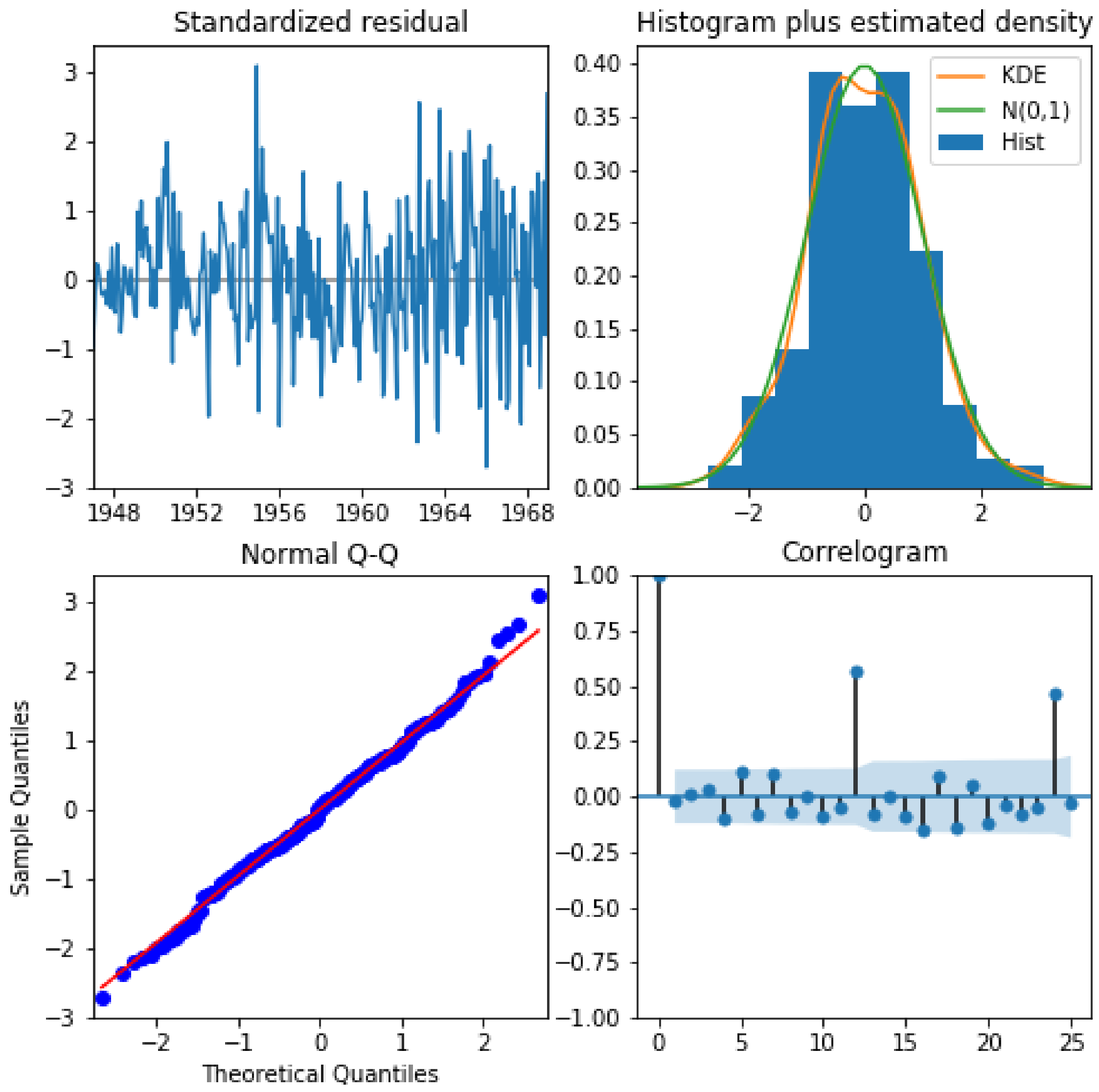
# Identifying seasonal data using ACF

```
# Create figure
fig, ax = plt.subplots(1,1, figsize=(8,4))

# Plot ACF
plot_acf(df.dropna(), ax=ax, lags=25, zero=False)
plt.show()
```



# ARIMA models and seasonal data



# Let's practice!

FORECASTING USING ARIMA MODELS IN PYTHON

# SARIMA models

FORECASTING USING ARIMA MODELS IN PYTHON

**James Fulton**
Climate informatics researcher

# The SARIMA model

Seasonal ARIMA = SARIMA

- Non-seasonal orders
  - p: autoregressive order
  - d: differencing order
  - q: moving average order

SARIMA(p,d,q)(P,D,Q)$_S$

- Seasonal Orders
  - P: seasonal autoregressive order
  - D: seasonal differencing order
  - Q: seasonal moving average order
  - S: number of time steps per cycle

# The SARIMA model

ARIMA(2,0,1) model :

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + m_1 \epsilon_{t-1} + \epsilon_t$$

SARIMA(0,0,0)(2,0,1)$_7$ model:

$$y_t = a_7 y_{t-7} + a_{14} y_{t-14} + m_7 \epsilon_{t-7} + \epsilon_t$$

# Fitting a SARIMA model

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

# Instantiate model
model = SARIMAX(df, order=(p,d,q), seasonal_order=(P,D,Q,S))

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

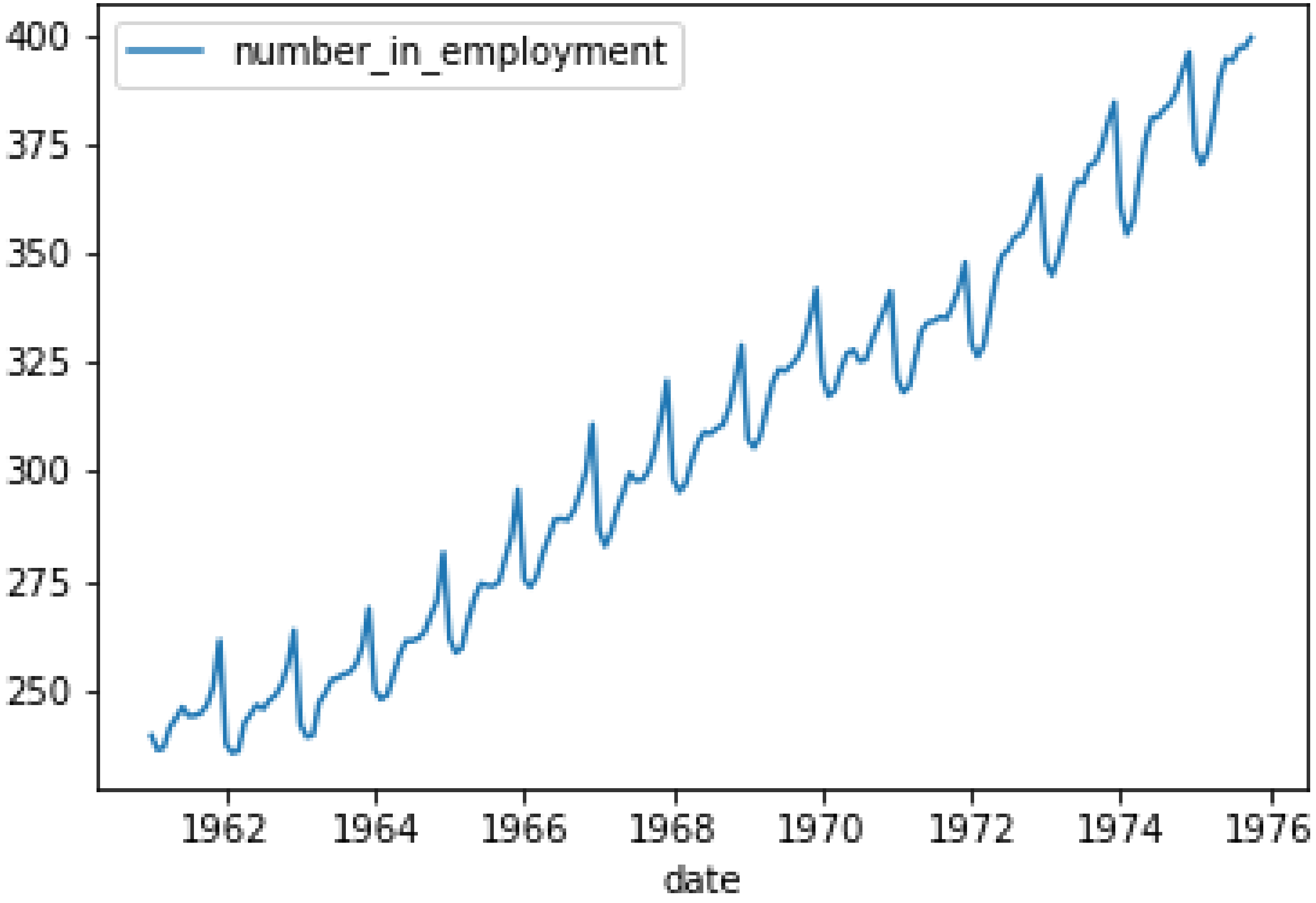
# Seasonal differencing

Subtract the time series value of one season ago

$$\Delta y_t = y_t - y_{t-S}$$

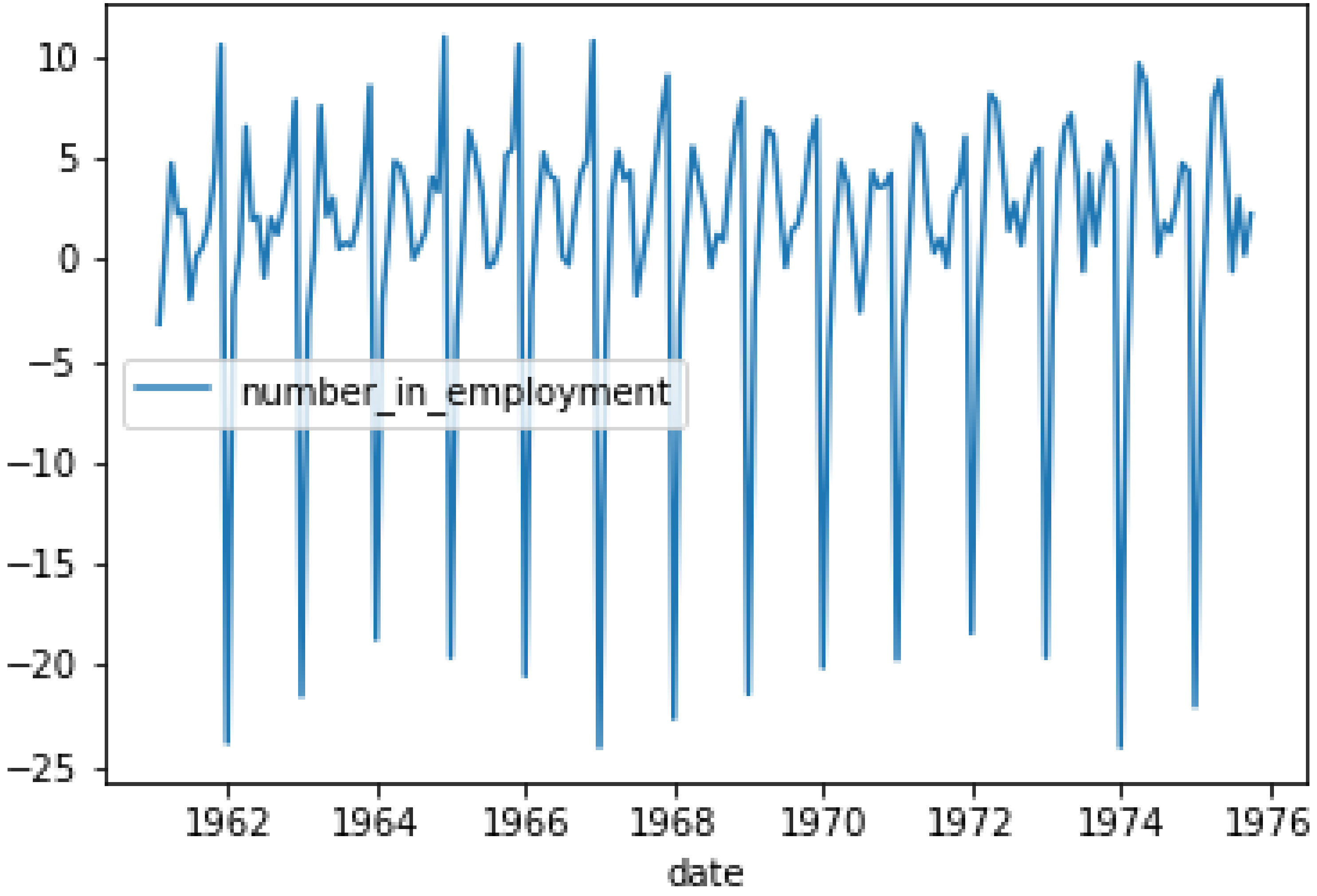
```
# Take the seasonal difference
df_diff = df.diff(S)
```

# Differencing for SARIMA models




Time series

# Differencing for SARIMA models

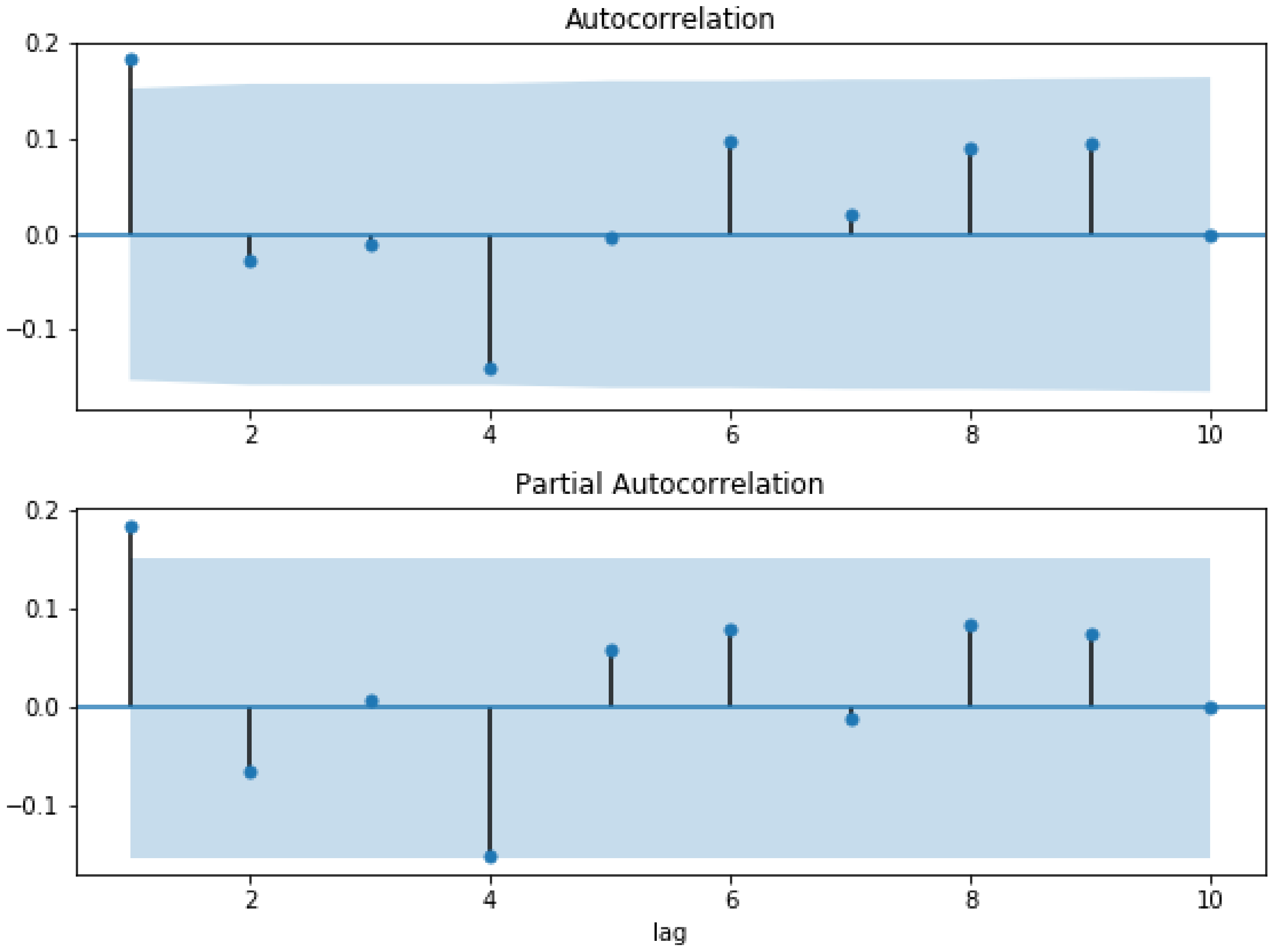



First difference of time series

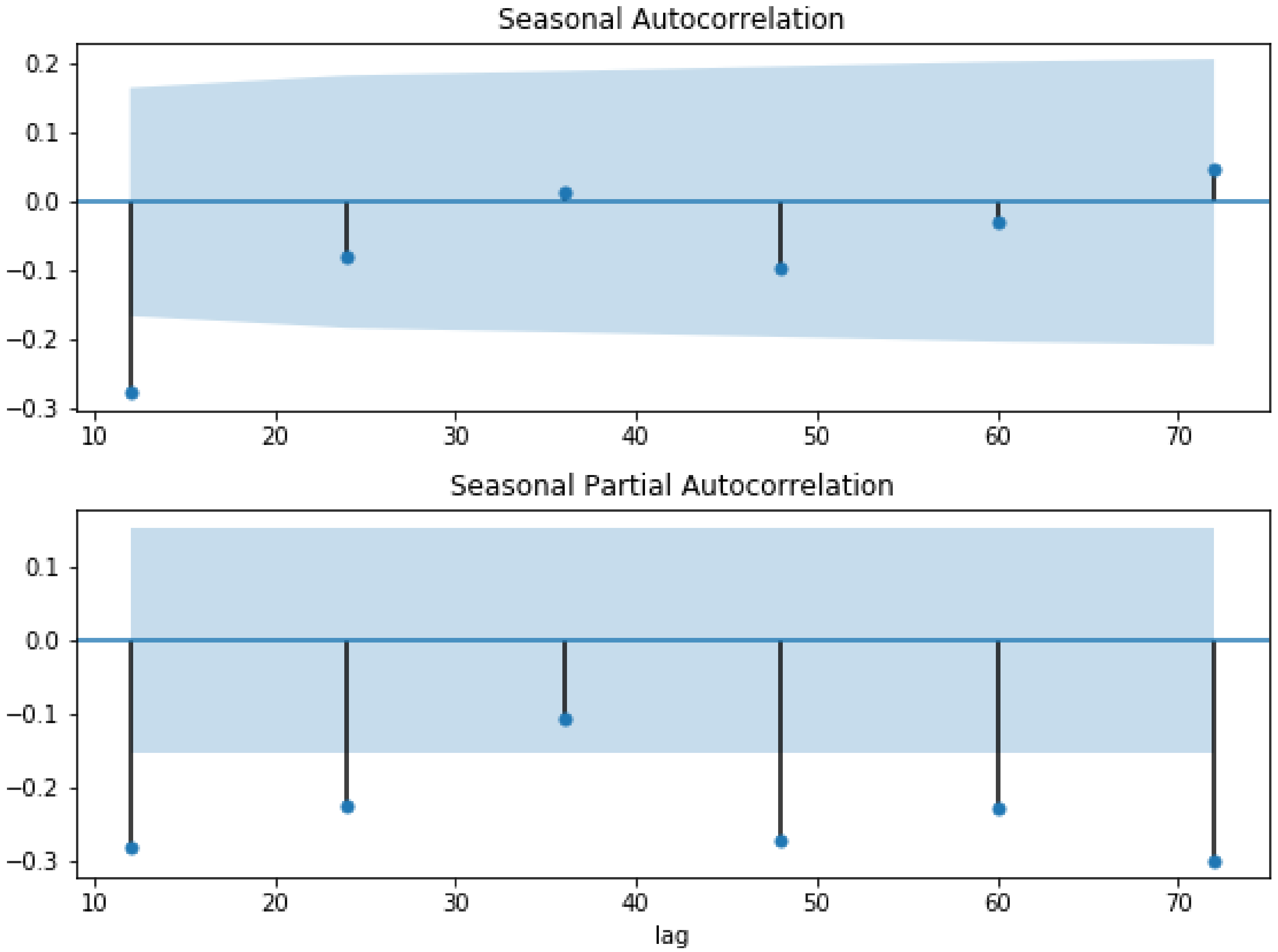
# Differencing for SARIMA models


First difference and first seasonal difference of ime series

# Finding p and q



# Finding P and Q



# Plotting seasonal ACF and PACF

```
# Create figure
fig, (ax1, ax2) = plt.subplots(2,1)

# Plot seasonal ACF
plot_acf(df_diff,  lags=[12,24,36,48,60,72], ax=ax1)

# Plot seasonal PACF
plot_pacf(df_diff, lags=[12,24,36,48,60,72], ax=ax2)

plt.show()
```

# Let's practice!

FORECASTING USING ARIMA MODELS IN PYTHON

# Automation and saving

FORECASTING USING ARIMA MODELS IN PYTHON

James Fulton
Climate informatics researcher

# Searching over model orders

```python
import pmdarima as pm
```

```python
results = pm.auto_arima(df)
```

```
Fit ARIMA: order=(2, 0, 2) seasonal_order=(1, 1, 1, 12); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 0, 0) seasonal_order=(0, 1, 0, 12); AIC=2648.467, BIC=2656.490, Fit time=0.062 s
Fit ARIMA: order=(1, 0, 0) seasonal_order=(1, 1, 0, 12); AIC=2279.986, BIC=2296.031, Fit time=1.171 s

...

Fit ARIMA: order=(3, 0, 3) seasonal_order=(1, 1, 1, 12); AIC=2173.508, BIC=2213.621, Fit time=12.487
Fit ARIMA: order=(3, 0, 3) seasonal_order=(0, 1, 0, 12); AIC=2297.305, BIC=2329.395, Fit time=2.087 s
Total fit time: 245.812 seconds
```

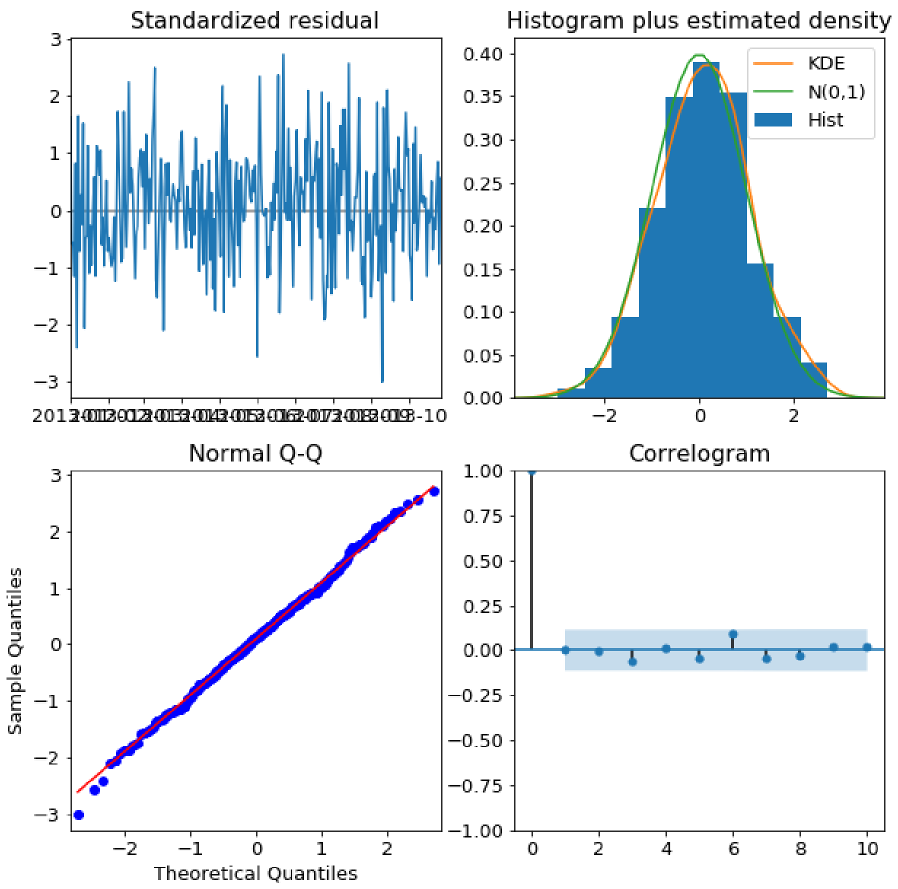
# pymarima results

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

```
results.plot_diagnostics()
```

```
                           Statespace Model Results
==============================================================================
Dep. Variable:            real values   No. Observations:                  300
Model:               SARIMAX(2, 0, 0)   Log Likelihood                -408.078
Date:                Tue, 28 May 2019   AIC                            822.156
Time:                        15:53:07   BIC                            833.267
Sample:                    01-01-2013   HQIC                           826.603
                         - 10-27-2013
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.2189      0.054      4.072      0.000       0.114       0.324
ar.L2          0.1960      0.054      3.626      0.000       0.090       0.302
sigma2         0.8888      0.073     12.160      0.000       0.746       1.032
===================================================================================
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
===================================================================================

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



# Non-seasonal search parameters

# Non-seasonal search parameters

```
results = pm.auto_arima( df,                # data

                         d=0,               # non-seasonal difference order

                         start_p=1,         # initial guess for p
                         start_q=1,         # initial guess for q

                         max_p=3,           # max value of p to test
                         max_q=3,           # max value of q to test
                        )
```

[1] https://www.alkaline [2] ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html

# Seasonal search parameters

```
results = pm.auto_arima( df,                # data
                         ... ,             # non-seasonal arguments
                         seasonal=True,     # is the time series seasonal
                         m=7,               # the seasonal period
                         D=1,               # seasonal difference order
                         start_P=1,         # initial guess for P
                         start_Q=1,         # initial guess for Q
                         max_P=2,           # max value of P to test
                         max_Q=2,           # max value of Q to test
                       )
```

# Other parameters

```
results = pm.auto_arima( df,                          # data
                         ... ,                         # model order parameters

                         information_criterion='aic', # used to select best model

                         trace=True,                   # print results whilst training

                         error_action='ignore',       # ignore orders that don't work

                         stepwise=True,                # apply intelligent order search
                        )
```

# Saving model objects

```
# Import
import joblib
```

```
# Select a filepath
filepath ='localpath/great_model.pkl'

# Save model to filepath
joblib.dump(model_results_object, filepath)
```

# Saving model objects

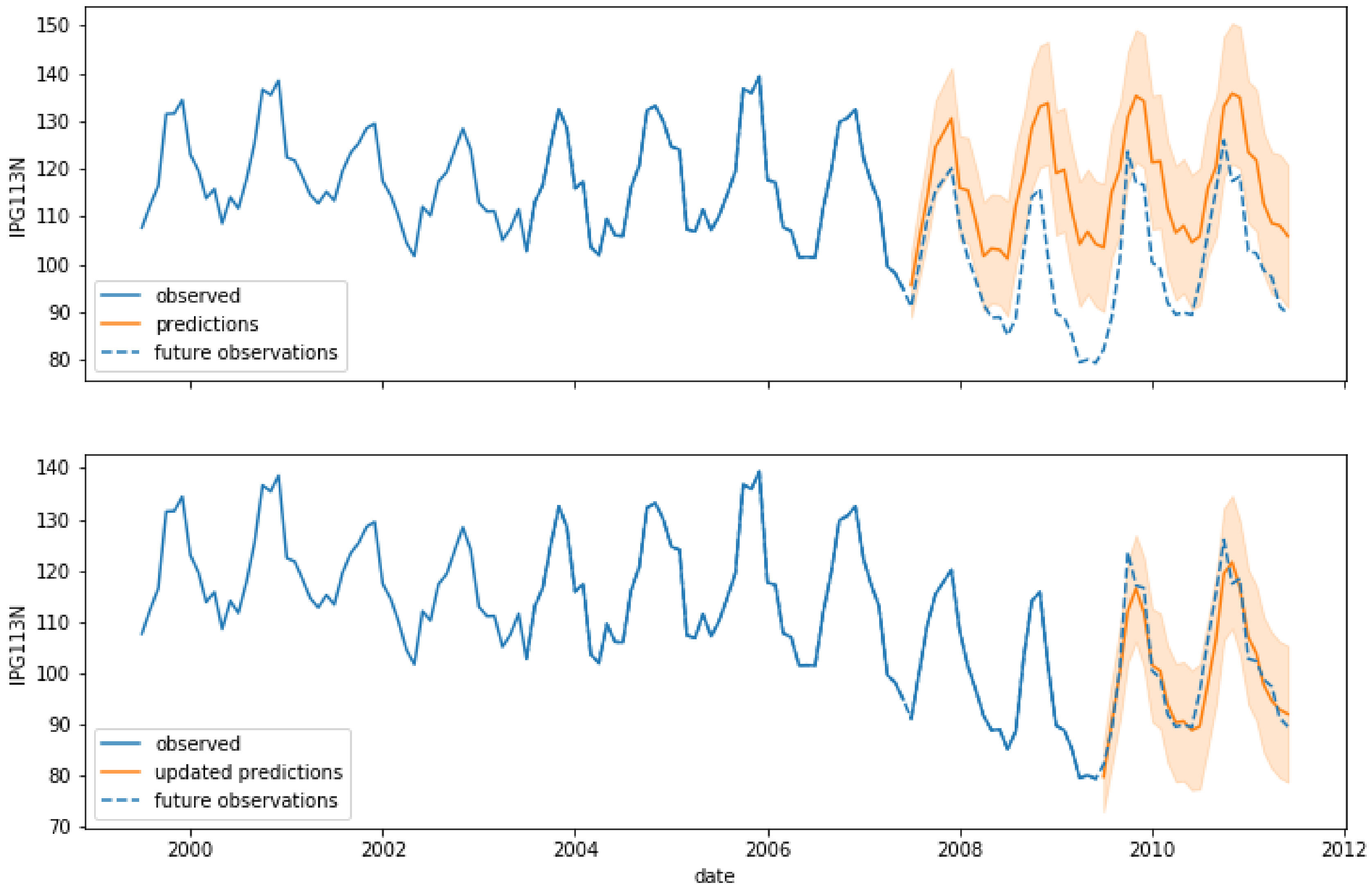
```
# Select a filepath
filepath ='localpath/great_model.pkl'

# Load model object from filepath
model_results_object = joblib.load(filepath)
```

# Updating model

```
# Add new observations and update parameters
model_results_object.update(df_new)
```

# Update comparison



# Let's practice!

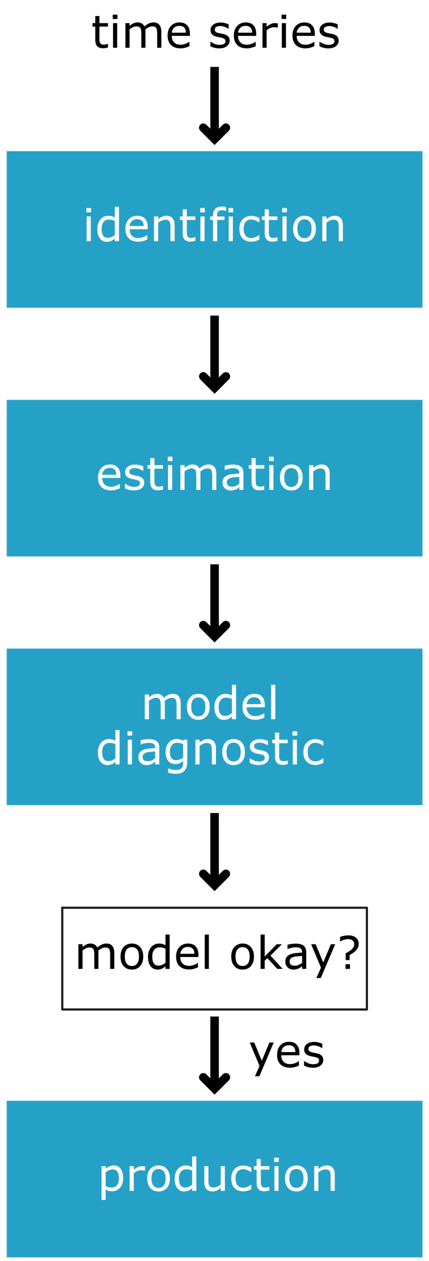
FORECASTING USING ARIMA MODELS IN PYTHON

# SARIMA and Box-Jenkins

FORECASTING USING ARIMA MODELS IN PYTHON


James Fulton
Climate informatics researcher

# Box-Jenkins

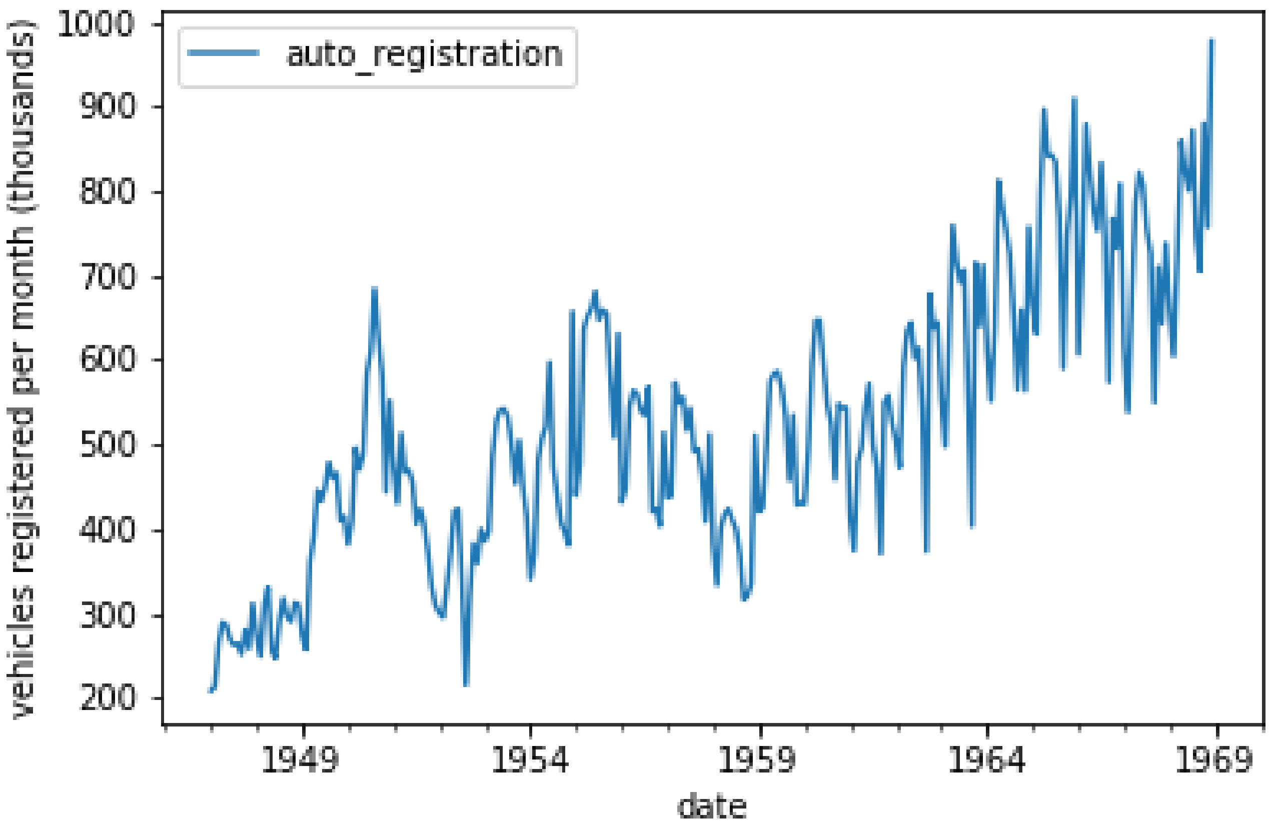

time series
identifiction
estimation
model
diagnostic
model okay?
yes
production

# Box-Jenkins with seasonal data

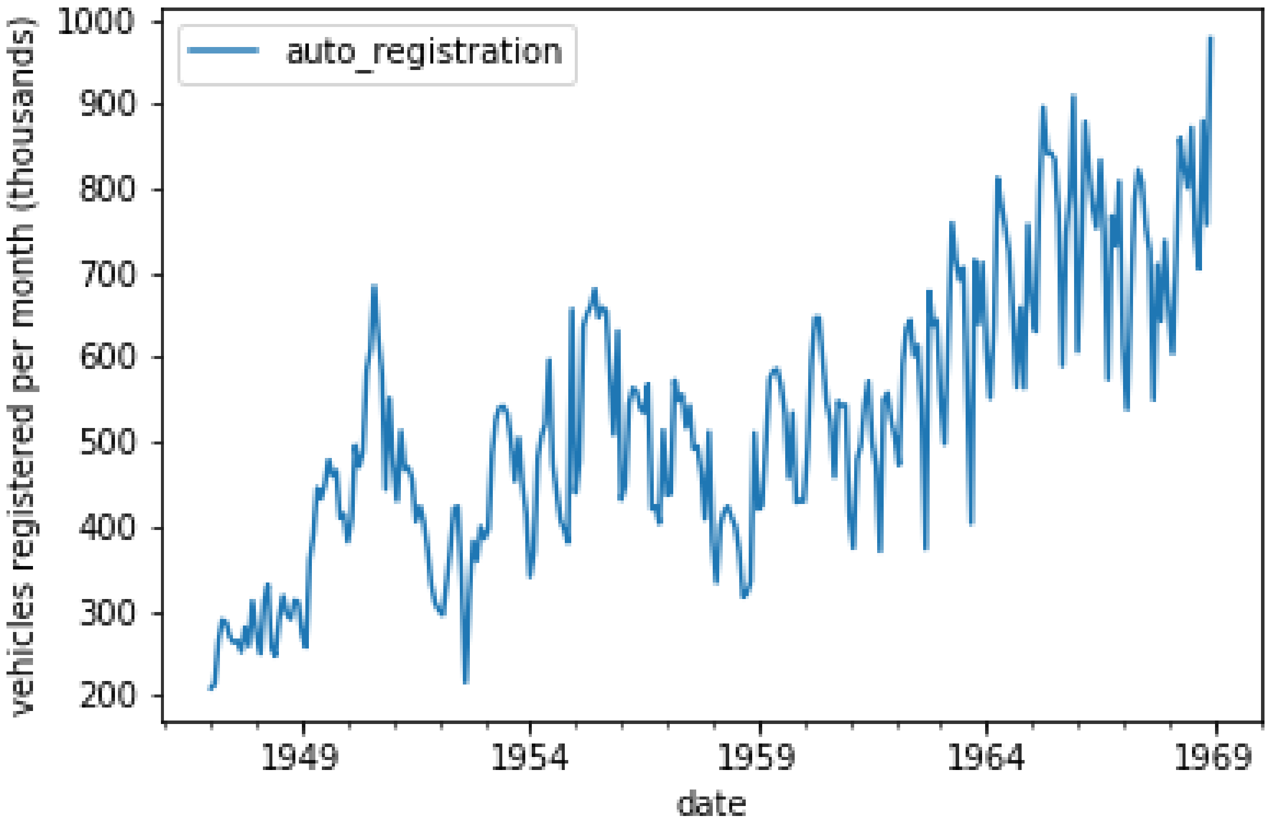
- Determine if time series is seasonal
- Find seasonal period
- Find transforms to make data stationary
  - Seasonal and non-seasonal differencing
  - Other transforms

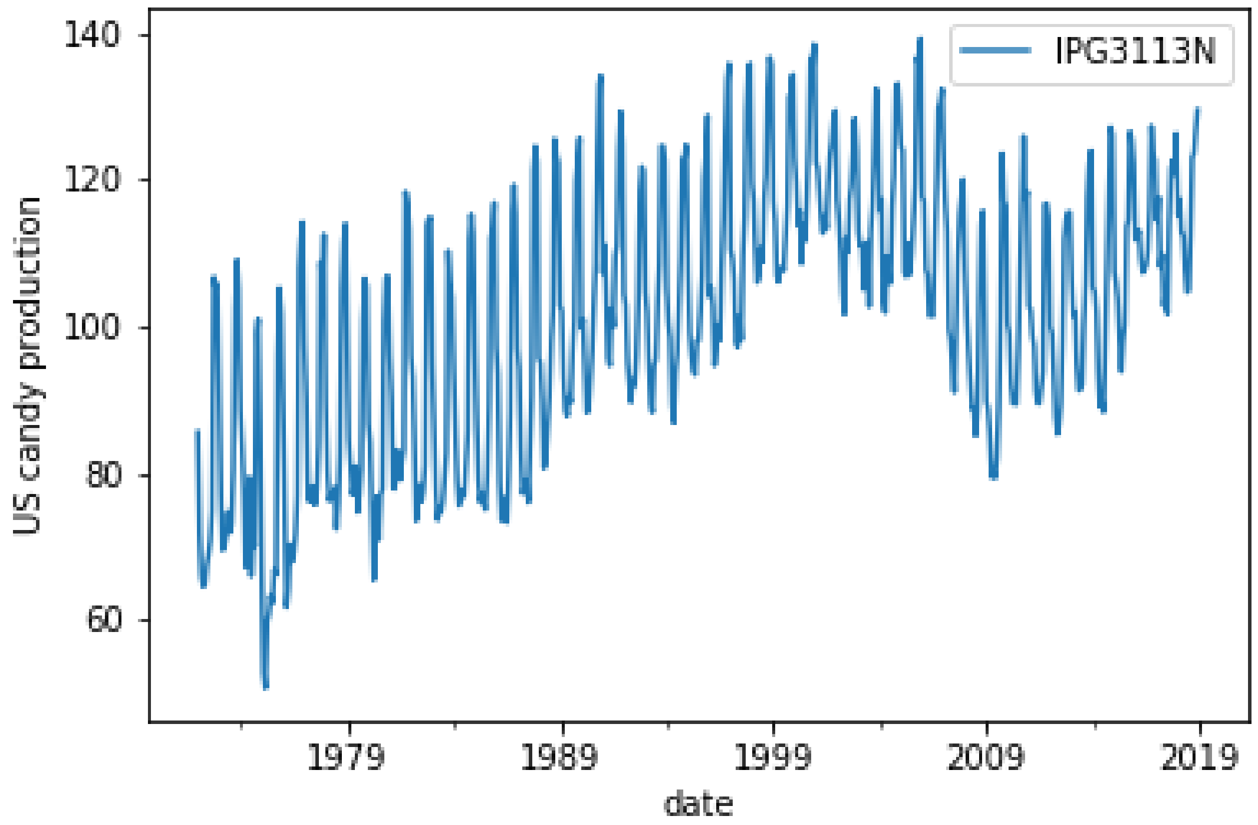
# Mixed differencing

- `D` should be 0 or 1
- `d + D` should be 0-2



# Weak vs strong seasonality

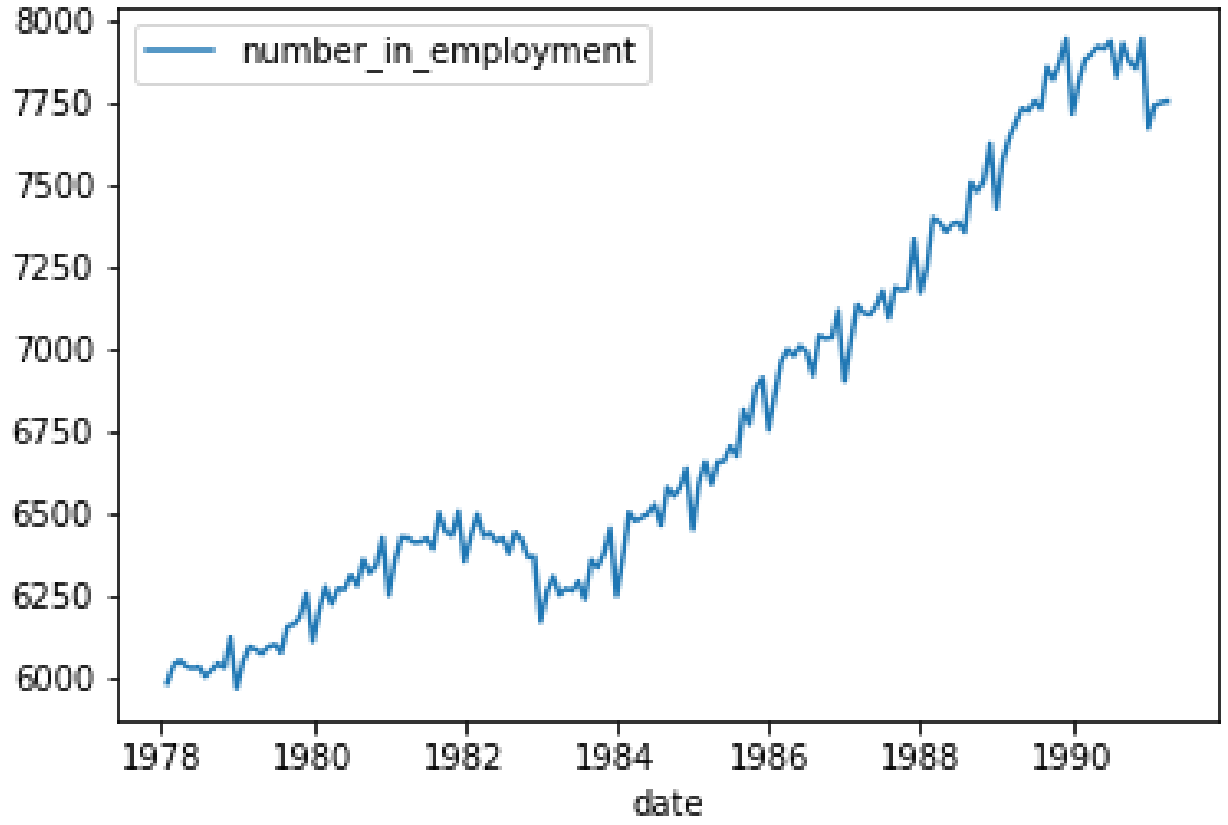


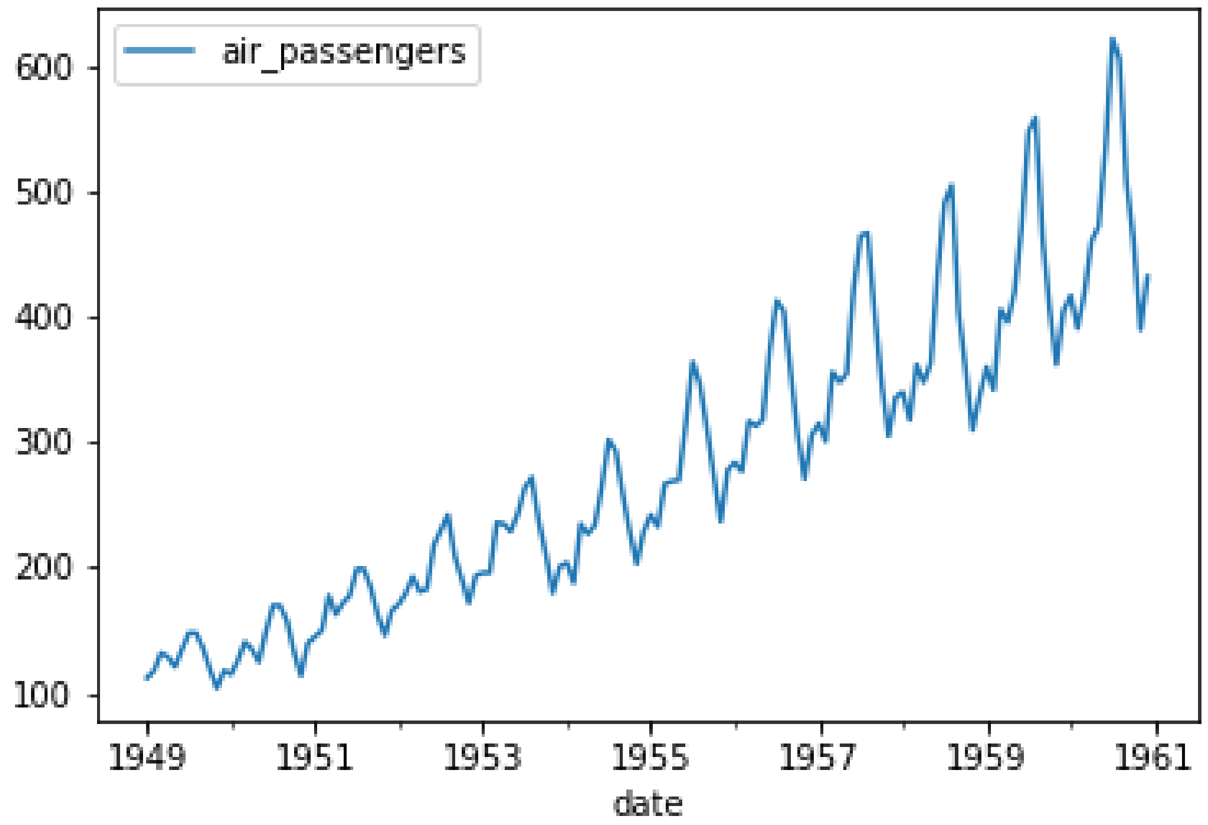




- Weak seasonal pattern
- Use seasonal differencing if necessary

- Strong seasonal pattern
- Always use seasonal differencing

# Additive vs multiplicative seasonality

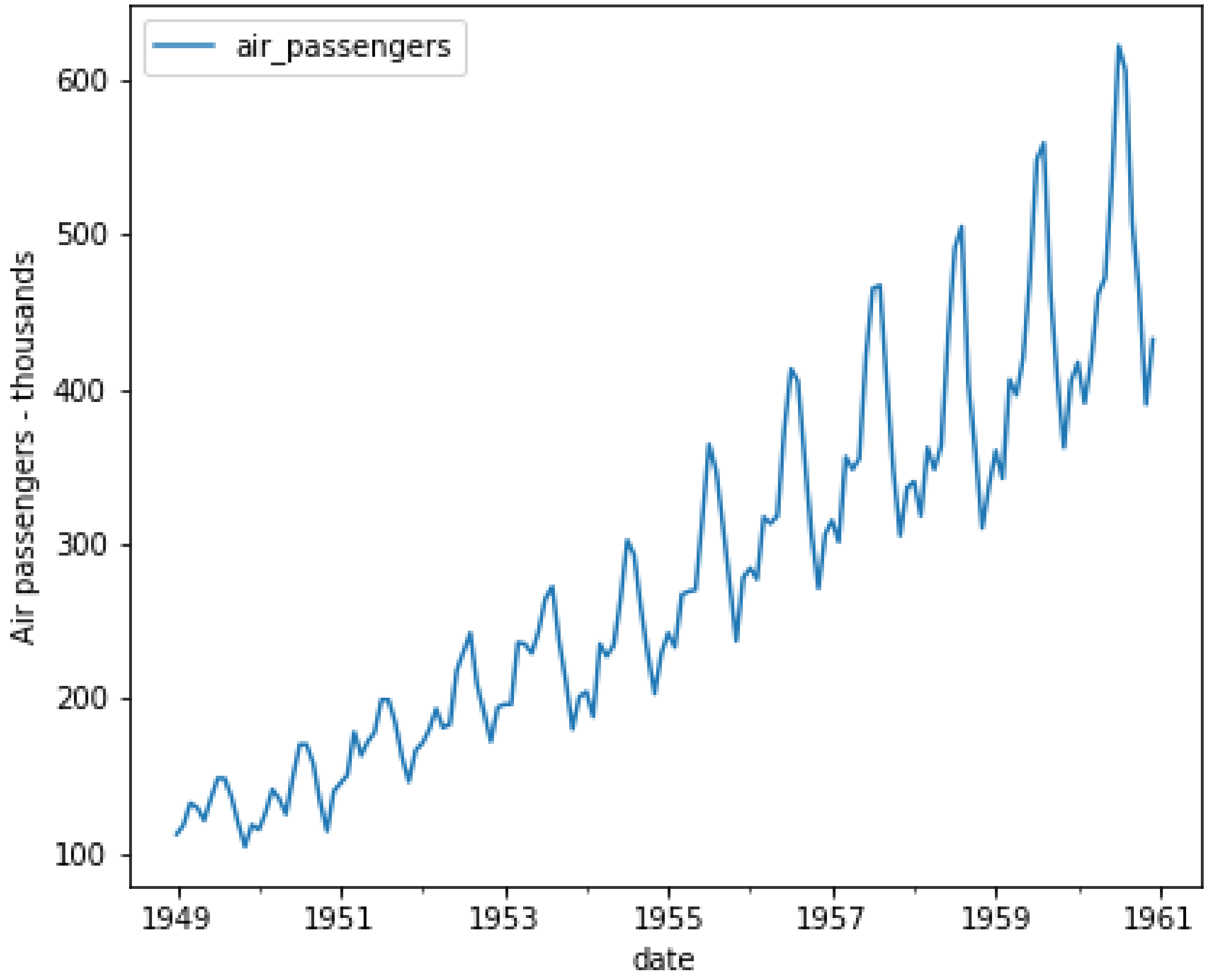


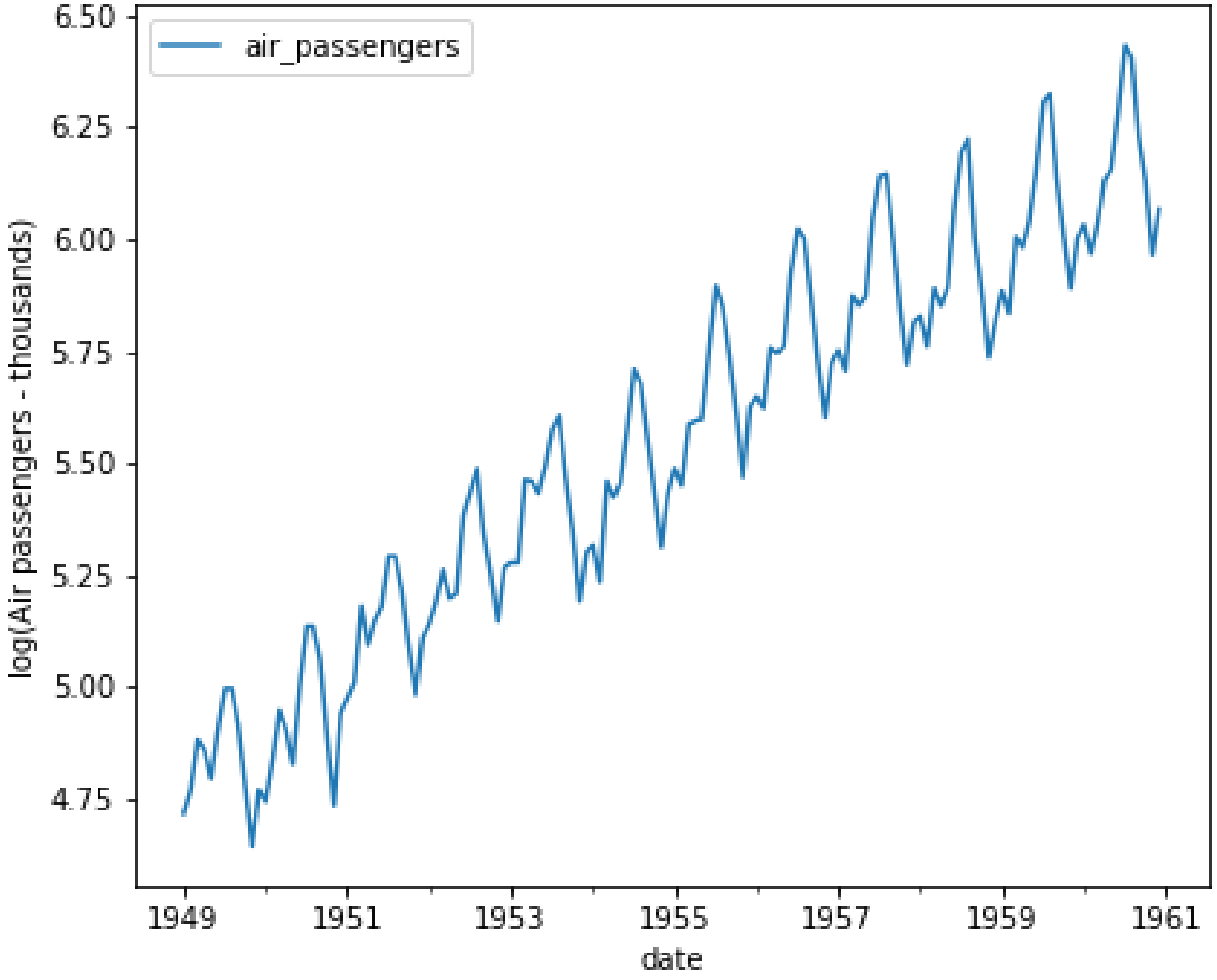




- Additive series = trend + season
- Proceed as usual with differencing

- multiplicative series = trend x season
- Apply log transform first - `np.log`

# Multiplicative to additive seasonality



# Let's practice!

FORECASTING USING ARIMA MODELS IN PYTHON

# Congratulations!

FORECASTING USING ARIMA MODELS IN PYTHON

James Fulton
Climate informatics researcher

# The SARIMAX model

**S** - seasonal

**AR** - autoregressive

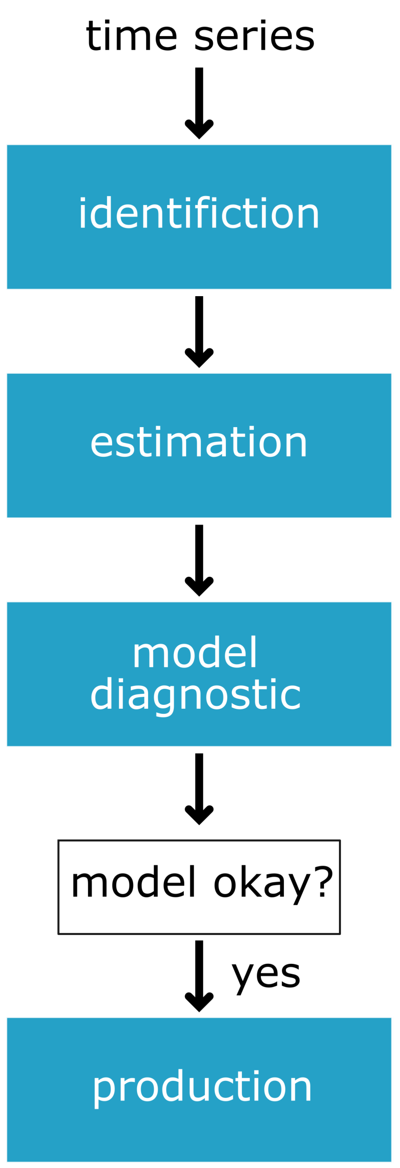
**I** - integrated

**MA** - moving average

**X** - exogenous

# Time series modeling framework

- Test for stationarity and seasonality
- Find promising model orders
- Fit models and narrow selection with AIC/BIC
- Perform model diagnostics tests
- Make forecasts
- Save and update models



# Further steps

- Fit data created using `arma_generate_sample()`
- Tackle real world data! Either your own or **examples from statsmodels**

# Further steps

- Fit data created using `arma_generate_sample()`
- Tackle real world data! Either your own or **examples from statsmodels**
- More time series courses here

[1] https://www.statsmodels.org/stable/datasets/index.html

# Good luck!

FORECASTING USING ARIMA MODELS IN PYTHON