# Forecasting 301 – Homework #5 ARIMA Forecasting (Student Submission)

Student: Alex Doe  
Date: April 23, 2025

## Question 1 – Explore & Visualize

I loaded the AirPassengers data into a pandas Series with a monthly PeriodIndex. The line chart (see `plt.show()` in my notebook) shows a gentle upward trend but I did not observe a clear seasonal pattern. The variance appears fairly constant.  
The ACF tails off slowly which suggests an AR(1) process, while the PACF has a significant spike at lag 1.

```python  
import pandas as pd, matplotlib.pyplot as plt, statsmodels.api as sm  
  
y = sm.datasets.get\_rdataset("AirPassengers").data['value']  
y.index = pd.period\_range("1949-01", periods=len(y), freq="M")  
y.plot(title='Passengers'); plt.show()  
  
from statsmodels.graphics.tsaplots import plot\_acf, plot\_pacf  
plot\_acf(y); plot\_pacf(y); plt.show()  
```

## Question 2 – Decompose & Difference

I used STL decomposition and observed that the seasonal component was almost flat, so I decided that seasonal differencing is unnecessary (D = 0). The trend looks linear, therefore I kept the non‑seasonal differencing at d = 0 as well to keep the data stationary.

```python  
from statsmodels.tsa.seasonal import STL  
res = STL(y.to\_timestamp()).fit()  
res.plot(); plt.show()  
```

## Question 3 – Model Selection

I reserved the last \*\*12 months\*\* as a test set. Using `pmdarima.auto\_arima` with seasonal=True, the best model by AIC was \*\*SARIMA(1,0,0)(0,0,1,12)\*\* with AIC ≈ 610. The residual Ljung‑Box p‑value was 0.03 so the errors are almost white noise.  
Model summary output is shown in my notebook.

```python  
from pmdarima import auto\_arima  
train, test = y[:-12], y[-12:]  
model = auto\_arima(train, m=12, seasonal=True, d=0, D=0, trace=True)  
print(model.summary())  
```

## Question 4 – Forecast & Evaluate

With the selected model I generated a 12‑step‑ahead forecast. The forecast closely follows the actual values but with some lag.  
\*\*RMSE = 120\*\* passengers, \*\*MAPE = 15 %\*\* on the hold‑out set.  
The 80 % and 95 % prediction intervals captured most points except November and December 1960.  
To improve accuracy I would consider adding an exogenous variable like GDP or holidays.

```python  
pred, conf = model.predict(n\_periods=12, return\_conf\_int=True)  
plt.plot(test.index.to\_timestamp(), test, label='Actual')  
plt.plot(test.index.to\_timestamp(), pred, label='Forecast')  
plt.fill\_between(test.index.to\_timestamp(), conf[:,0], conf[:,1], alpha=.3)  
```