Assignment 1 (ARIMA modelling and forecasting)

Methodical model selection (d, p, q), MLE estimation, diagnostics, and forecasting validation

Estimated time: 6–10 hours (including write-up).

Tools: Python, Jupyter, pandas, numpy, matplotlib, statsmodels.

Learning objectives

- Apply the Box–Jenkins workflow: explore → transform/difference → identify orders → estimate (MLE) → diagnose → forecast.
- Select the differencing order **d** using evidence (plots + ADF/KPSS), aiming for the smallest **d** that achieves stationarity.
- Select **p** and **q** using ACF/PACF intuition, then confirm using AIC/BIC over a small candidate grid.
- Understand (conceptually) that ARIMA/SARIMAX parameters are estimated by maximum likelihood (via a state-space/Kalman filter implementation in statsmodels).
- Validate adequacy using residual ACF and Ljung–Box (white-noise) tests, then assess forecasting performance using a hold-out set.
- Extend to regression with autocorrelated errors: OLS baseline → diagnose correlated residuals → fit SARIMAX with exogenous regressor and ARMA error structure.

Data sources (public)

Download the two CSV files and place them in the same folder as your notebook:

- Internet usage (100 minutes): **www_usage.csv** — https://otexts.com/fpppy/data/www_usage.csv
- US quarterly changes (1970–2019): **US_change.csv** — https://otexts.com/fpppy/data/US_change.csv

Context references (optional reading): Python Time Series Handbook (smoothing/exploration) https://filippomb.github.io/python-time-series-handbook/notebooks/03/smoothing.html; FPP3 (ARIMA exercises & motivation) https://otexts.com/fpp3/arima-exercises.html; Shumway & Stoffer, *Time Series Analysis and Its Applications* (Section 3.7/3.8 workflow).

Part A — Univariate ARIMA modelling and forecasting (www_usage.csv)

You will model the time series y_t (number of users) using an ARIMA(p,d,q) model. Follow the steps below and justify each choice with plots and diagnostics.

A1. Exploratory analysis

- Plot y_t over time (clear labels). Describe trend/level changes/outliers.
- Add a simple smoothing overlay (moving average or exponential smoothing) only to highlight structure (do not use it as the final model).

A2. Select the differencing order d (the “I” in ARIMA)

- Test **d = 0, 1, 2** in order. For each candidate: plot the differenced series and its ACF.
- Run stationarity tests: **ADF** (null: unit root / nonstationary) and **KPSS** (null: stationary).
- Choose the smallest **d** that makes the series look stationary and passes tests reasonably (avoid over-differencing).
- Write 3–6 sentences explaining why your chosen **d** is appropriate.

A3. Select p and q (AR and MA orders)

- Using the series after differencing with your chosen **d**, plot ACF and PACF (choose a sensible number of lags).
- Propose a small set of candidate (**p**,**q**) values based on ACF/PACF patterns (e.g., **p** up to 6, **q** up to 4).
- Fit a grid of ARIMA(**p**,**d**,**q**) models for those candidates and compare AIC and BIC.
- Pick a final model using a parsimony rule: prefer the simplest model with competitive BIC and good diagnostics.

A4. Estimate parameters (MLE) and interpret

- Fit the selected ARIMA model using statsmodels. Report parameter estimates and standard errors.
- Explain in simple terms what maximum likelihood estimation is doing for ARIMA. (You may mention that statsmodels evaluates the likelihood via a state-space/Kalman filter implementation.)

A5. Residual diagnostics (white-noise check)

- Plot residuals and residual ACF.
- Run Ljung–Box at several lags (e.g., 10 and 20) and interpret the p-values.
- If residuals are not white noise, describe one concrete model revision (change **p**/**q**, revisit **d**, etc.) and justify.

A6. Forecasting validation (hold-out)

- Hold out the last **h = 20** observations as a test set. Fit your ARIMA model on the remaining data.
- Forecast **h** steps ahead with 95% prediction intervals. Plot train/test/forecast clearly.
- Compute MAE and RMSE on the test set.
- Compare against a naive baseline (repeat the last training value) and briefly interpret the results.

Part B — Regression with autocorrelated errors (US_change.csv)

Now model consumption growth using income as an exogenous predictor, while allowing the regression errors to be autocorrelated.

B1. OLS baseline + diagnose residual autocorrelation

- Fit OLS: $y_t = c + \beta x_t + e_t$ (y = consumption growth, x = income growth).
- Report β, standard error, and Durbin–Watson statistic.
- Plot the residual ACF and run Ljung–Box. State whether residuals are autocorrelated.

B2. Choose an ARMA structure for the errors

- Use residual ACF and PACF to propose candidate ARMA(**p**,**q**) orders for e_t (start small).
- Fit a small grid of SARIMAX(*y*, exog=*x*, order=(**p**,0,**q**), trend='c') models and compare BIC.
- Select a final (**p**,**q**) based on BIC + parsimony + residual whiteness.

B3. Fit the final regression-with-ARMA-errors model

- Fit SARIMAX with your chosen (**p**,**q**). Report β and the AR/MA parameters (with p-values).
- Run residual/innovation diagnostics (ACF + Ljung–Box) to confirm whiteness.
- Compare β from OLS vs SARIMAX: how did the estimate and/or standard error change, and why?

What to submit

- A single Jupyter notebook with code, figures, and short explanations (markdown).
- A short written summary (1–2 pages, can be part of the notebook) answering the key decision questions: **d** choice, (**p**,**q**) choice, diagnostics outcomes, and forecasting comparison.
- Make sure your notebook runs from top to bottom on a clean environment with the two CSVs in the same folder.