Time Series Forecasting

Group Project

FIN41660 Financial Econometrics

University College Dublin

Associate Professor Alessia Paccagnini

Academic Year 2025/2026

Important Deadline

Due: December 21, 2025 at 11:59 PM

Contents

1 Overview

In this group project, you will build an **interactive application, dashboard, or game** to forecast time series data using econometric models. Your application will implement OLS, ARIMA, and GARCH models, evaluate forecasting accuracy, and provide an engaging user experience.

This project combines **applied econometrics**, **programming skills**, and **creative design** to create a practical forecasting tool.

1.1 Learning Objectives

By completing this project, you will:

- Apply OLS, ARIMA, and GARCH models to real-world time series data
- Implement forecasting accuracy tests and model comparison techniques
- Develop an interactive application using Python
- Work collaboratively on a substantial software project
- Create a standalone .py script that runs the same models and show results match your app
- Communicate technical results effectively through code, reports, and video

1.2 Key Information

Aspect	Details
Group Size	Maximum 4 students
Weight	20% of final grade
Deadline	December 21, 2024, 11:59 PM
Submission Platform	Brightspace

What to Submit

1. Complete application code (Python files)
2. Written report (20 pages + appendix)
3. Video presentation (10 minutes) (To upload to Brightspace and Google Drive Link (named with members' surnames))
4. All team members should participate in the video
5. All files in a single ZIP archive

2 Tips for Success

2.1 Getting Started

1. **Form your group early** and establish regular meeting times
2. **Divide responsibilities** based on strengths (coding, writing, analysis, design)
3. **Choose your data** carefully - ensure it's appropriate for all three models
4. **Start simple** - get basic versions working before adding advanced features

2.2 Development Best Practices

- Use **version control** (Git/GitHub) for collaboration
- **Test frequently** - don't wait until the end to test your code
- **Document as you go** - don't leave it for the last minute
- Create **modular code** - separate models, data processing, and visualization
- **Handle errors gracefully** - your app should not crash with invalid inputs

2.3 Report Writing

- **Start early** - writing takes longer than you think
- Use **clear, concise language** - avoid jargon where possible
- Include **high-quality figures** - label axes, add legends, use proper captions
- **Proofread carefully** - check for typos and grammatical errors
- **Cite sources properly** - use a consistent citation style

2.4 Video Production

- **Practice your presentation** before recording
- Use **screen recording software** (OBS Studio, Zoom, etc.)
- Ensure **clear audio** - use a good microphone
- **Edit your video** - remove mistakes and long pauses
- Stay within the **10-minute limit** - practice timing

3 Frequently Asked Questions

3.1 General Questions

Q: Can we use languages other than Python?
A: No, Python is required for this project.

Q: Can we use pre-built libraries for models?
A: Yes! Use statsmodels, arch, and scikit-learn. You don't need to code OLS/ARIMA/GARCH from scratch.

Q: Can we work individually instead of in a group?
A: Yes, maximum group size is 4 students.

3.2 Technical Questions

Q: Which application framework should we use?
A: Streamlit is recommended for beginners. Dash is good for more interactive dashboards. Choose based on your team's skills.

Q: Do we need to deploy the app online?
A: No, but it's a nice bonus! A local app that runs on any computer is sufficient.

Q: How should we handle missing data?
A: Document your approach in the report. Common methods: forward fill, interpolation, or removal. Justify your choice.

3.3 Submission Questions

Q: Can we submit multiple times?
A: Yes, only the most recent submission before the deadline will be graded.

Q: What if the file is too large?
A: Don't include large data files - provide download links instead. Code and report should be manageable size.

4 Academic Integrity

4.1 Expectations

- All work must be your group's **original creation**
- You may use **publicly available libraries and tutorials**
- You may consult **online resources** (Stack Overflow, documentation)
- You **must cite** any code adapted from external sources

4.2 Use of AI Tools

- You may use AI assistants (ChatGPT, GitHub Copilot, Claude) for:
  - Debugging code
  - Understanding syntax
  - Getting suggestions
- You **must understand** all code you submit
- You **must document** significant AI assistance in your report
- The **design and methodology** must be your own

4.3 Plagiarism Policy

- **Collaboration between groups is not permitted**
- Copying code from other groups will result in **zero marks** for all parties
- Properly cite any external code sources

5 Grading Scheme

Your project will be evaluated across four main components. Each component assesses different skills essential for financial econometrics practitioners.

5.1 Grade Breakdown

Component	Weight	Max Points
Application/Dashboard/Game	40%	40
Written Report	30%	30
Video Presentation	15%	15
Code Quality & Reproducibility	15%	15
Total	**100%**	**100**

Table 1: Grade Distribution by Component

5.2 Component Descriptions

5.2.1 Application/Dashboard/Game (40%)

The core deliverable of this project. Your application should demonstrate mastery of time series forecasting techniques while providing an intuitive user experience.

- **Model Implementation (20%)**: Correct implementation of OLS, ARIMA, and GARCH models
- **Forecasting & Evaluation (10%)**: Accuracy metrics, model comparison, forecast visualisation
- **User Interface & Design (10%)**: Usability, aesthetics, interactivity

5.2.2 Written Report (30%)

A professional document demonstrating your understanding of the methodology and results.

- **Methodology (10%)**: Clear explanation of models, data, and approach
- **Results & Analysis (10%)**: Interpretation of findings, model comparison
- **Presentation Quality (10%)**: Writing clarity, figures, formatting

5.2.3 Video Presentation (15%)

A concise demonstration of your application and key findings.

- **Content & Clarity (10%)**: Clear explanation, logical flow
- **Production Quality (5%)**: Audio/video quality, professionalism

5.2.4 Code Quality & Reproducibility (15%)

Your standalone `.py` script and overall code organisation.

- **Standalone Script (10%)**: Runs independently, results match application
- **Code Organisation (5%)**: Comments, structure, documentation

5.3 Penalties

- **Late Submission**: 10% deduction per day, up to 5 days. Submissions after 5 days receive zero marks.
- **Missing Components**: Each missing required component (app, report, video, script) results in zero marks for that section.
- **Non-participation**: Team members who do not appear in the video may receive individual grade adjustments.

Final Reminder

Start early, work consistently, and ask questions!
This project is designed to be challenging but achievable.
Good luck!