



Time Series Analysis and Forecasting

# Chapter 8: Modern Extensions

Seminar



## Seminar Outline

### Seminar structure:

1. **Review Quiz** – Knowledge check
2. **True/False Questions** – Conceptual checks
3. **Practice Problems** – Applied practice
4. **Summary** – Key takeaways
5. **AI-Assisted Exercises** – Critical thinking

## Quiz 1: Hurst Exponent

### Question

A time series has Hurst exponent  $H = 0.8$ . What does this indicate?

### Answer choices

- (A) The series is a pure random walk
- (B) The series has long memory and is persistent (trend-following)
- (C) The series is anti-persistent (mean-reverting)
- (D) The series is stationary  $I(0)$

*Answer on next slide...*

## Quiz 1: Answer

Answer: B – Long memory and persistence

### Hurst Exponent Interpretation:

- ☐  $H = 0.5$ : Random walk (no memory)
- ☐  $0.5 < H < 1$ : **Persistence** – trend continues
- ☐  $0 < H < 0.5$ : Anti-persistence – mean reversion

With  $H = 0.8 > 0.5$ :

- ☐ The series has **long memory**
- ☐ Large values tend to be followed by large values
- ☐ Autocorrelations decay slowly (hyperbolically, not exponentially)

## Quiz 2: Fractional Differencing Parameter

### Question

In the ARFIMA( $p, d, q$ ) model, the parameter  $d$  can take values:

### Answer choices

- (A) Only integer values (0, 1, 2, ...)
- (B) Only  $d = 0$  or  $d = 1$
- (C) Any real value, including fractional
- (D) Only negative values

*Answer on next slide...*

## Quiz 2: Answer

**Answer:**  $C$  – Any real value

**Fractional differencing:**  $(1 - L)^d$  with  $d \in \mathbb{R}$

**Interpretation of  $d$  values:**

- ☐  $d = 0$ : Stationary series (ARMA)
- ☐  $0 < d < 0.5$ : Long memory, stationary
- ☐  $d = 0.5$ : Stationary/non-stationary boundary
- ☐  $0.5 < d < 1$ : Long memory, non-stationary
- ☐  $d = 1$ : Full differencing (classic ARIMA)

**Relation to Hurst:**  $d = H - 0.5$

## Quiz 3: Long Memory in Financial Series

### Question

In which financial series is long memory most commonly documented?

### Answer choices

- (A) Stock prices
- (B) Daily returns
- (C) Volatility (squared returns)
- (D) Trading volume

*Answer on next slide...*

## Quiz 3: Answer

Answer: C – Volatility

### Stylized facts from finance:

- ▣ **Returns:** Approximately memoryless ( $H \approx 0.5$ )
- ▣ **Volatility:** Pronounced long memory ( $H \approx 0.7 - 0.9$ )

### Why?

- ▣ Volatility clustering: turbulent periods followed by turbulent periods
- ▣ Persistence of shocks in variance
- ▣ FIGARCH: explicitly models long memory in volatility

This stylized fact is the basis for FIGARCH and HAR-RV models.



## Quiz 4: Feature Engineering

### Question

To apply Random Forest to time series, we must create:

### Answer choices

- (A) Dummy variables for each observation
- (B) Lag features and rolling statistics
- (C) Fourier transforms of the series
- (D) Only the first difference of the series

*Answer on next slide...*

## Quiz 4: Answer

Answer: B – Lag features and rolling statistics

### Feature Engineering for Time Series:

- **Lag features:**  $y_{t-1}, y_{t-2}, \dots, y_{t-k}$
- **Rolling statistics:**
  - ▶ Rolling mean:  $\bar{y}_{t,w}$
  - ▶ Rolling standard deviation:  $\sigma_{t,w}$
  - ▶ Min/Max over window
- **Calendar features:** day of week, month, etc.

**Important:** Transforms the forecasting problem into a supervised regression problem!

## Quiz 5: Time Series Cross-Validation

### Question

Why can't we use standard k-fold cross-validation for time series?

### Answer choices

- (A) It's too slow for long series
- (B) It violates temporal order and causes data leakage
- (C) It only works for classification
- (D) It requires too much data

*Answer on next slide...*

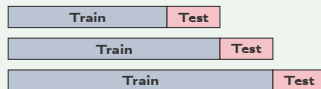
## Quiz 5: Answer

Answer: B – Violates temporal order

**Problem with standard k-fold:**

- ▣ Shuffles observations temporally
- ▣ Trains on future data, tests on past
- ▣ **Data leakage**  $\Rightarrow$  overestimated performance

**Solution: Time Series Split (Walk-Forward)**



## Quiz 6: Feature Importance in Random Forest

### Question

Feature importance in Random Forest for time series helps us:

### Answer choices

- (A) Eliminate all low-importance variables
- (B) Identify which lags and features are most predictive
- (C) Determine Granger causality
- (D) Calculate confidence intervals

*Answer on next slide...*

## Quiz 6: Answer

Answer: B – Identifies predictive features

### Uses of feature importance:

- ☐ Understanding temporal structure
- ☐ Selecting optimal number of lags
- ☐ Identifying relevant factors

### Caution:

- ☐ Importance does NOT imply causality
- ☐ Correlated variables may share importance
- ☐ Use for interpretation, not causal inference

## Quiz 7: LSTM Advantage

### Question

What is the main advantage of LSTM over simple RNNs?

### Answer choices

- (A) It's faster to train
- (B) It solves the vanishing/exploding gradient problem
- (C) It requires less data
- (D) It's easier to interpret

*Answer on next slide...*

## Quiz 7: Answer

Answer: B – Solves the gradient problem

### Simple RNN Problem:

- ☐ Gradients decay exponentially with sequence length
- ☐ Cannot learn long-term dependencies

### LSTM Solution:

- ☐ **Cell state:** Highway for information flow
- ☐ **Forget gate:** Decides what to forget
- ☐ **Input gate:** Decides what to remember
- ☐ **Output gate:** Decides what to output

Gradients can “flow” through the cell state without degradation!



## Quiz 8: Data Preparation for LSTM

### Question

Before training an LSTM, data should be:

### Answer choices

- (A) Log-transformed
- (B) Normalized/scaled to  $[0,1]$  or  $[-1,1]$
- (C) Differenced twice
- (D) Converted to integers

*Answer on next slide...*

## Quiz 8: Answer

Answer: B – Normalized/scaled

### Why normalization?

- Activation functions (sigmoid, tanh) work in limited ranges
- Faster convergence
- Numerical stability

### Common methods:

- **Min-Max:**  $x' = \frac{x - x_{min}}{x_{max} - x_{min}} \succ [0, 1]$
- **Standard:**  $x' = \frac{x - \mu}{\sigma} \succ \text{mean } 0, \text{ std } 1$

**Important:** Fit on train, transform on train+test!

## Quiz 9: LSTM Hyperparameters

### Question

Which is NOT a typical LSTM hyperparameter?

### Answer choices

- (A) Number of units (neurons) per layer
- (B) Input sequence length
- (C) Learning rate
- (D) Differencing parameter  $d$

*Answer on next slide...*

## Quiz 9: Answer

Answer: D – The  $d$  parameter

$d$  is specific to ARFIMA models, not LSTM!

### LSTM Hyperparameters:

- **Architecture:** number of layers, units/layer
- **Sequence:** lookback length
- **Training:** learning rate, batch size, epochs
- **Regularization:** dropout, early stopping

**Tuning:** Grid search or Bayesian optimization with time series CV

## True or False? — Questions

Statement	T/F?
1. ARFIMA models can capture long-term dependence.	?
2. The parameter $d$ in ARFIMA must be an integer.	?
3. LSTM networks are better than ARIMA in all situations.	?
4. Random Forest requires manually created features.	?
5. Standard cross-validation (k-fold) is suitable for time series.	?
6. The Hurst exponent $H > 0.5$ indicates positive long memory.	?

## True or False? — Answers

Statement	T/F	Explanation
1. ARFIMA captures long-term dependence.	T	Fractional $d$
2. $d$ in ARFIMA must be an integer.	F	$d \in (0, 0.5)$ fractional
3. LSTM better than ARIMA always.	F	Depends on data and sample
4. RF requires manual features.	T	Lags, calendar, etc.
5. k-fold CV suitable for time series.	F	Violates temporal ordering
6. $H > 0.5$ indicates positive long memory.	T	$H = 0.5$ : no memory

## Problem 1: Hurst Exponent Estimation

### Exercise

Given daily Bitcoin returns, estimate the Hurst exponent using the R/S method and interpret the result.

### Solution Steps:

1. Calculate mean over subintervals of different lengths  $n$
2. For each  $n$ : calculate  $\text{Range}(R)$  and  $\text{Std}(S)$
3. The ratio  $R/S$  grows as  $n^H$
4. Fit regression:  $\log(R/S) = H \cdot \log(n) + c$

**Python code:** `nolds.hurst_rs(returns)`

## Problem 1: Solution and Interpretation

### Typical Bitcoin Results

- ▣ Returns:  $H \approx 0.45 - 0.55$  (approximately random walk)
- ▣ Volatility ( $|returns|$ ):  $H \approx 0.75 - 0.85$  (long memory!)

### Interpretation:

- ▣ Returns are hard to predict (EMH approximately valid)
- ▣ Volatility is predictable over long horizons
- ▣ Implications for risk management and VaR

**Application:** FIGARCH models may outperform standard GARCH



## Problem 2: Random Forest for Forecasting

### Exercise

Build a Random Forest model for 1-day ahead Bitcoin price forecasting. Evaluate using TimeSeriesSplit.

#### Pipeline:

1. **Feature engineering:**
  - ▶ Lags:  $y_{t-1}, y_{t-2}, \dots, y_{t-7}$
  - ▶ Rolling mean/std: 7, 14, 30 days
2. **Train/Test split:** TimeSeriesSplit(n\_splits=5)
3. **Model:** RandomForestRegressor(n\_estimators=100)
4. **Evaluation:** RMSE, MAE, Direction Accuracy

## Problem 2: Code and Results

### Python Code

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import TimeSeriesSplit

tscv = TimeSeriesSplit(n_splits=5)
rf = RandomForestRegressor(n_estimators=100)

for train_idx, test_idx in tscv.split(X):
    rf.fit(X[train_idx], y[train_idx])
    pred = rf.predict(X[test_idx])
```

### Typical results:

- ▣ Direction accuracy: 52-55% (slightly above random)
- ▣ Feature importance: lag-1 and rolling\_std dominate

## Problem 3: LSTM for Time Series

### Exercise

Implement a simple LSTM model for Bitcoin forecasting. Compare with Random Forest.

#### Simple LSTM Architecture:

1. Input: 30-day sequences
2. LSTM layer: 50 units
3. Dense output: 1 neuron (forecast)
4. Loss: MSE, Optimizer: Adam

#### Important steps:

- ☐ MinMaxScaler normalization
- ☐ Reshape to [samples, timesteps, features]
- ☐ Early stopping to prevent overfitting

## Problem 3: LSTM Code

### Keras/TensorFlow Code

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

model = Sequential([
    LSTM(50, input_shape=(30, 1)),
    Dense(1)
])

model.compile(optimizer='adam', loss='mse')
model.fit(X_train, y_train, epochs=50,
          validation_split=0.1, verbose=0)
```

#### Typical RF vs LSTM comparison:

- RMSE similar (LSTM slightly better on smooth data)
- RF: faster, more interpretable
- LSTM: captures complex patterns better

## Su ARFIMA

- ▣ Series with long memory (volatility, hydrology)
- ▣ When  $0 < d < 0.5$  is theoretically justified
- ▣ Statistical interpretability is important

## Random Forest

- ▣ Nonlinear relationships between features
- ▣ Feature importance for understanding
- ▣ Structured data, not too long series

## LSTM

- ▣ Long sequences with complex dependencies
- ▣ Sufficient data for deep learning
- ▣ Patterns difficult to capture with classical methods

## Key ARFIMA and Long Memory

- ▣ Fractional differencing:  $(1 - L)^d y_t = \varepsilon_t$
- ▣ Hurst exponent:  $d = H - 0.5$
- ▣ ACF for long memory:  $\rho(k) \sim k^{2d-1}$  (slow decay)

## Machine Learning

- ▣ Lag feature:  $X_t = [y_{t-1}, y_{t-2}, \dots, y_{t-k}]$
- ▣ RMSE:  $\sqrt{\frac{1}{n} \sum (y_i - \hat{y}_i)^2}$
- ▣ Direction Accuracy:  $\frac{1}{n} \sum 1[\text{sign}(\Delta y) = \text{sign}(\Delta \hat{y})]$

## LSTM

- ▣ Forget gate:  $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
- ▣ Cell update:  $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$



## AI Exercise: Critical Thinking

Test this prompt in ChatGPT/Claude/Copilot

*"Download hourly electricity consumption data. Compare three approaches: (1) ARIMA, (2) Random Forest with lag features, (3) LSTM. Use time series cross-validation, report RMSE and directional accuracy for each."*

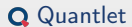
1. Did the AI use proper time series cross-validation (no future data leakage)?
2. Are the lag features for Random Forest correctly constructed?
3. Is the LSTM data properly scaled and sequenced?
4. Does the comparison use the same test set for all three models?
5. Which model performs best and why? Is this consistent with theory?

**Warning:** AI-generated code may run without errors and look professional. That does not mean it is correct.

# Thank You!

## Questions?

Seminar materials are available at: <https://danpele.github.io/Time-Series-Analysis/>



Quantlet



Quantinar



## Bibliography I

### Fundamental Textbooks

- ▣ Hyndman, R.J., & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice*, 3rd ed., OTexts.
- ▣ Shumway, R.H., & Stoffer, D.S. (2017). *Time Series Analysis and Its Applications*, 4th ed., Springer.
- ▣ Brockwell, P.J., & Davis, R.A. (2016). *Introduction to Time Series and Forecasting*, 3rd ed., Springer.

### Financial Time Series

- ▣ Tsay, R.S. (2010). *Analysis of Financial Time Series*, 3rd ed., Wiley.
- ▣ Franke, J., Härdle, W.K., & Hafner, C.M. (2019). *Statistics of Financial Markets*, 4th ed., Springer.

## Bibliography II

### Modern Approaches and Machine Learning

- ▣ Nielsen, A. (2019). *Practical Time Series Analysis*, O'Reilly Media.
- ▣ Petropoulos, F., et al. (2022). *Forecasting: Theory and Practice*, International Journal of Forecasting.
- ▣ Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 Competition, International Journal of Forecasting.

### Online Resources and Code

- ▣ **Quantlet**: <https://quantlet.com> — Code repository for statistics
- ▣ **Quantinar**: <https://quantinar.com> — Quantitative methods learning platform
- ▣ **GitHub TSA**: [https://github.com/QuantLet/TSA/tree/main/TSA\\_ch8](https://github.com/QuantLet/TSA/tree/main/TSA_ch8) — Python code for this seminar