



Time Series Analysis and Forecasting

# Chapter 8: Modern Extensions

Seminar



# Seminar Outline

- 1 Quiz: ARFIMA and Long Memory
- 2 Quiz: Machine Learning for Time Series
- 3 Quiz: LSTM Networks
- 4 True/False Questions
- 5 Practice Problems
- 6 Summary

## Quiz 1: Hurst Exponent

### Question

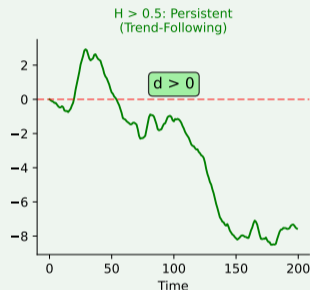
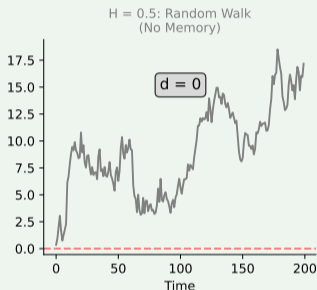
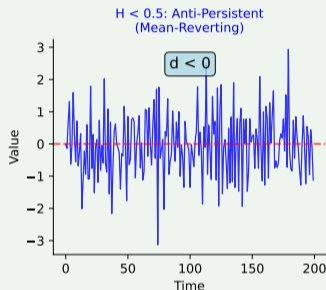
A time series has Hurst exponent  $H = 0.8$ . What does this indicate?

- ☐ 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** – trends continue
- $0 < H < 0.5$ : **Anti-persistence** – mean reversion

## Quiz 2: Fractional Differencing Parameter

### Question

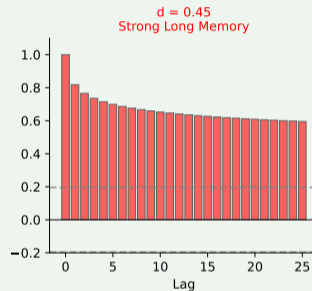
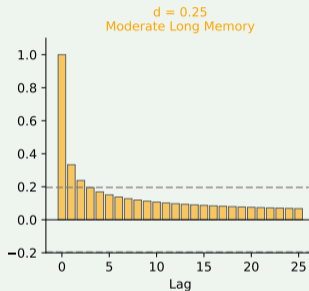
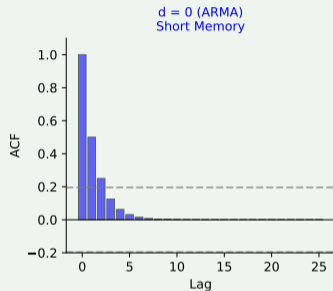
In the ARFIMA( $p, d, q$ ) model, the parameter  $d$  can take values:

- ☐ 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



Interpretation of  $d$  values:

- $d = 0$ : Stationary (ARMA)
- $0 < d < 0.5$ : Long memory, stationary
- $0.5 \leq d < 1$ : Long memory, non-stationary
- $d = 1$ : Unit root (classic ARIMA)

Relation to Hurst:  $d = H - 0.5$

## Quiz 3: Long Memory in Financial Series

### Question

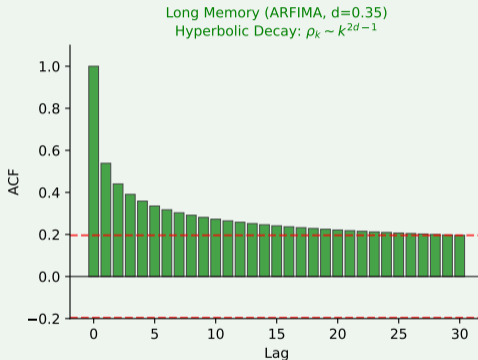
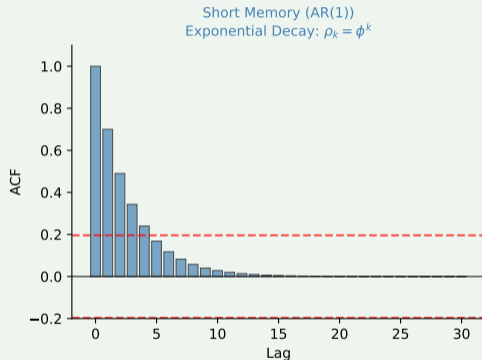
In which financial series is long memory most commonly documented?

- ☐ 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 in finance:

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

**Why?** Volatility clustering – turbulent periods follow turbulent periods. This is the basis for FIGARCH and HAR-RV models.

## Quiz 4: Feature Engineering

### Question

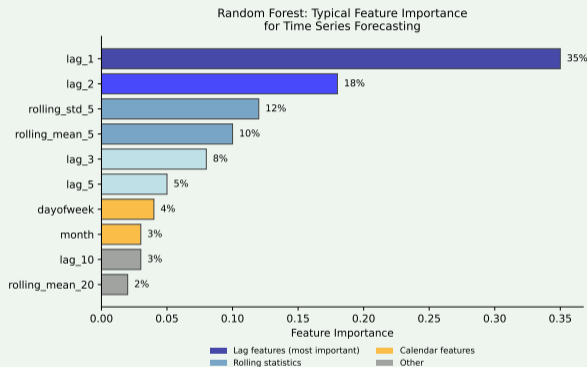
To apply Random Forest to time series, we must create:

- ☐ 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:** moving average, moving std, min/max
- **Calendar features:** day of week, month, etc.

## Quiz 5: Time Series Cross-Validation

### Question

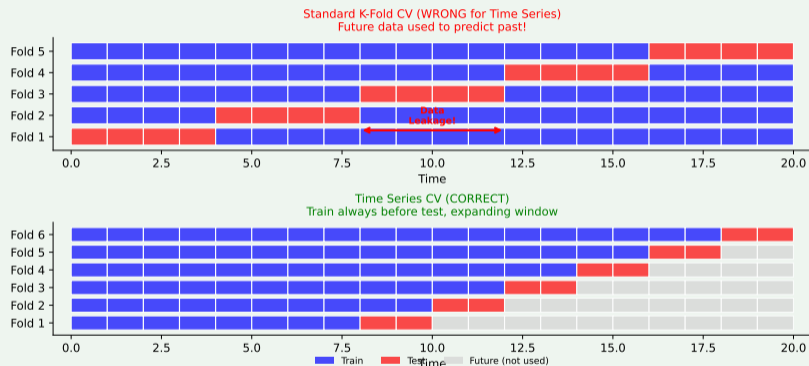
Why can't we use standard k-fold cross-validation for time series?

- ☐ 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 randomly
- Trains on future data, tests on past
- **Data leakage**  $\Rightarrow$  overestimated performance

## Quiz 6: Feature Importance in Random Forest

### Question

Feature importance in Random Forest for time series helps us:

- ☐ 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

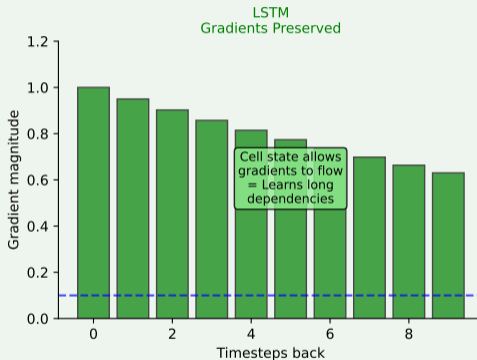
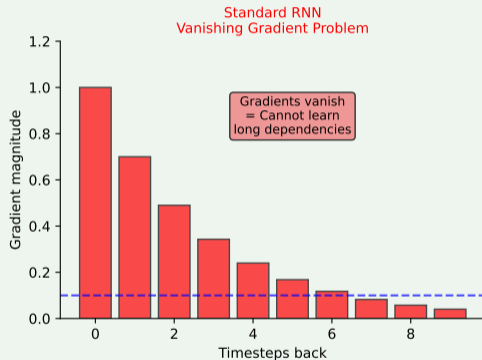
What is the main advantage of LSTM over simple RNNs?

- ☐ 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.

**LSTM Solution:**

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

## Quiz 8: Data Preparation for LSTM

### Question

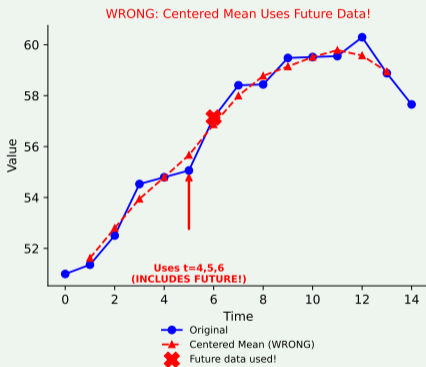
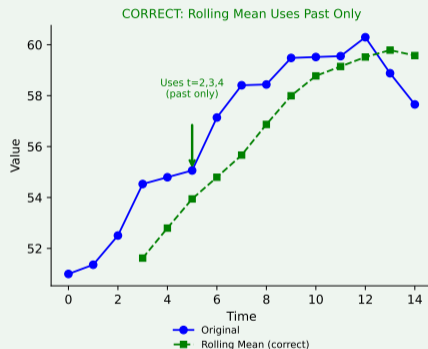
Before training an LSTM, data should be:

- ☐ 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

## Quiz 9: LSTM Hyperparameters

### Question

Which is NOT a typical LSTM hyperparameter?

- ☐ 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

## Quiz 10: Model Selection

### Question

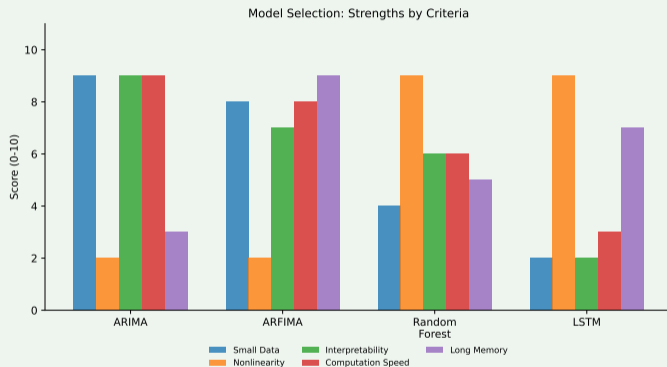
When comparing ARFIMA, Random Forest, and LSTM for forecasting:

- ☐ A) LSTM always wins because it's deep learning
- ☐ B) ARFIMA is always best for financial data
- ☐ C) The best model depends on data characteristics and problem requirements
- ☐ D) Random Forest can't be used for time series

*Answer on next slide...*

## Quiz 10: Answer

Answer: C – Depends on data and requirements



### Model Selection Guidelines:

- **ARFIMA:** Long memory, interpretability needed
- **Random Forest:** Nonlinear relations, feature importance
- **LSTM:** Long sequences, complex patterns, sufficient data

## True/False Questions

Determine if each statement is True or False:

- ❶ The Hurst exponent  $H = 0.5$  indicates long memory.
- ❷ ARFIMA reduces to ARIMA when  $d$  is an integer.
- ❸ Standard k-fold CV is appropriate for time series data.
- ❹ LSTM can capture long-range dependencies better than simple RNNs.
- ❺ Feature importance in Random Forest proves causality.
- ❻ Normalizing data is optional when training neural networks.

*Answers on next slide...*

# True/False: Solutions

- ❶ The Hurst exponent  $H = 0.5$  indicates long memory.

$H = 0.5$  means random walk (no memory). Long memory:  $H > 0.5$ .

FALSE

- ❷ ARFIMA reduces to ARIMA when  $d$  is an integer.

With  $d = 0$  or  $d = 1$ , ARFIMA becomes standard ARMA or ARIMA.

TRUE

- ❸ Standard k-fold CV is appropriate for time series data.

Use time series split to maintain temporal order and avoid data leakage.

FALSE

- ❹ LSTM can capture long-range dependencies better than simple RNNs.

Cell state and gates allow gradients to flow without vanishing.

TRUE

- ❺ Feature importance in Random Forest proves causality.

Shows predictive power, not causal relationship.

FALSE

- ❻ Normalizing data is optional when training neural networks.

Critical for convergence and stability with activation functions.

FALSE

## 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:

- ❶ **Feature engineering:**
  - Lags:  $y_{t-1}, y_{t-2}, \dots, y_{t-7}$
  - Rolling mean/std: 7, 14, 30 days
- ❷ **Train/Test split:** TimeSeriesSplit( $n\_splits=5$ )
- ❸ **Model:** RandomForestRegressor( $n\_estimators=100$ )
- ❹ **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

## Summary: When to Use Each Method

### 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

## 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:  $\frac{1}{n} \sum \mathbf{1}[\text{sign}(\Delta y) = \text{sign}(\Delta \hat{y})]$

## LSTM

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

# Thank You!

Questions?

`danpele@ase.ro`