



Diogo Ribeiro

ESMAD - Escola Superior de Média Arte e Design

Lead Data Scientist, Mysense.ai

✉ dfr@esmad.ipp.pt  0009-0001-2022-7072

November 11, 2025

Outline

- 1 Introduction to Time Series
- 2 Classical Time Series Models
- 3 Vector Autoregression (VAR)
- 4 State Space Models
- 5 Forecasting
- 6 Modern Deep Learning Approaches
- 7 Best Practices and Pitfalls
- 8 Conclusion

What is a Time Series?

[

title=Time Series] A **time series** is a sequence of observations $\{y_t\}_{t=1}^T$ indexed by time, where y_t represents the value at time t .

Key Characteristics:

- **Temporal ordering:** Observations have a natural sequence
- **Dependence:** Values are typically correlated over time
- **Components:** Trend, seasonality, cycles, irregular

Applications:

- Finance: Stock prices, returns, volatility
- Economics: GDP, inflation, unemployment
- Climate: Temperature, precipitation
- Business: Sales, demand, web traffic

Time Series Components

Additive Decomposition:

$$y_t = T_t + S_t + C_t + \epsilon_t \quad (1)$$

where:

- T_t = **Trend** (long-term movement)
- S_t = **Seasonality** (regular periodic fluctuations)
- C_t = **Cycle** (longer-term oscillations)
- ϵ_t = **Irregular/Noise** (random variation)

Multiplicative Decomposition:

$$y_t = T_t \times S_t \times C_t \times \epsilon_t \quad (2)$$

Used when variation increases with level (common in economics).

[

title=Weak (Covariance) Stationarity] A time series is weakly stationary if:

1. $\mathbb{E}[y_t] = \mu$ for all t (constant mean)
2. $\text{Var}(y_t) = \sigma^2$ for all t (constant variance)
3. $\text{Cov}(y_t, y_{t-h}) = \gamma_h$ depends only on lag h

[

title=Why Stationarity Matters] Most time series models assume stationarity. Non-stationary series can lead to:

- Spurious regressions
- Invalid statistical inference

Common transformations:

1. Differencing:

- First difference: $\Delta y_t = y_t - y_{t-1}$
- Seasonal difference: $\Delta_s y_t = y_t - y_{t-s}$

2. Detrending:

- Remove linear trend: $\tilde{y}_t = y_t - (\hat{\alpha} + \hat{\beta}t)$
- Remove polynomial trend

3. Log transformation:

- Stabilize variance: $\log(y_t)$
- Box-Cox transformation: $y_t^{(\lambda)} = \frac{y_t^\lambda - 1}{\lambda}$

Autoregressive (AR) Models

[

title=AR(p) Model]

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t \quad (3)$$

where $\epsilon_t \sim \text{WN}(0, \sigma^2)$ (white noise).

Interpretation:

- Current value depends on p past values
- ϕ_i coefficients measure persistence
- Order p selected via AIC/BIC

[

Moving Average (MA) Models

[

title=MA(q) Model]

$$y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q} \quad (4)$$

where $\epsilon_t \sim \text{WN}(0, \sigma^2)$.

Interpretation:

- Current value is linear combination of white noise terms
- Always stationary
- Shocks have limited effect (only q periods)

ACF vs PACF:

- AR(p): ACF decays, PACF cuts off after lag p
- MA(q): ACF cuts off after lag q , PACF decays

ARMA and ARIMA Models

[
title=ARMA(p,q) Model] Combines AR and MA components:

$$y_t = c + \sum_{i=1}^p \phi_i y_{t-i} + \epsilon_t + \sum_{j=1}^q \theta_j \epsilon_{t-j} \quad (5)$$

[
title=ARIMA(p,d,q) Model] ARMA applied to d -differenced data:

$$\phi(B)(1 - B)^d y_t = \theta(B)\epsilon_t \quad (6)$$

where B is the backshift operator: $By_t = y_{t-1}$.

[

title=SARIMA(p, d, q)(P, D, Q) $_s$ Model]

$$\phi(B)\Phi(B^s)(1-B)^d(1-B^s)^D y_t = \theta(B)\Theta(B^s)\epsilon_t \quad (7)$$

where s is the seasonal period.

Components:

- (p, d, q) : Non-seasonal AR, differencing, MA orders
- $(P, D, Q)_s$: Seasonal AR, differencing, MA orders
- s : Seasonal period (e.g., 12 for monthly data)

[

[

title=VAR(p) Model] For k -dimensional time series $\mathbf{y}_t = (y_{1t}, \dots, y_{kt})'$:

$$\mathbf{y}_t = \mathbf{c} + \mathbf{A}_1 \mathbf{y}_{t-1} + \dots + \mathbf{A}_p \mathbf{y}_{t-p} + \boldsymbol{\epsilon}_t \quad (8)$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ and \mathbf{A}_i are $k \times k$ coefficient matrices.

Interpretation:

- Each variable depends on its own lags and lags of all other variables
- Captures dynamic interactions between variables
- $(\mathbf{A}_i)_{jk}$ measures effect of variable k at lag i on variable j

1. Granger Causality:

- Variable x Granger-causes y if past values of x help predict y
- Test: H_0 : All coefficients of x lags in y equation are zero

2. Impulse Response Functions (IRF):

- Trace effect of one-unit shock to variable j on all variables over time
- $\text{IRF}(h) = \frac{\partial \mathbf{y}_{t+h}}{\partial \epsilon_{jt}}$
- Visualize dynamic relationships

3. Forecast Error Variance Decomposition:

- Decompose forecast error variance into contributions from each shock
- Measures relative importance of each variable

State Space Representation

[

title=State Space Model] **State equation:**

$$\mathbf{x}_t = \mathbf{F}_t \mathbf{x}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \mathbf{w}_t \quad (9)$$

Observation equation:

$$\mathbf{y}_t = \mathbf{H}_t \mathbf{x}_t + \mathbf{v}_t \quad (10)$$

where $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$ and $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$.

Components:

- \mathbf{x}_t : Hidden state (unobserved)
- \mathbf{y}_t : Observations
- \mathbf{F}_t : State transition matrix
- \mathbf{H}_t : Observation matrix

Optimal recursive algorithm for linear Gaussian state space models.

Prediction step:

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t \hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}_t \mathbf{u}_t \quad (11)$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t' + \mathbf{Q}_t \quad (12)$$

Update step:

$$\mathbf{K}_t = \mathbf{P}_{t|t-1} \mathbf{H}_t' (\mathbf{H}_t \mathbf{P}_{t|t-1} \mathbf{H}_t' + \mathbf{R}_t)^{-1} \quad (13)$$

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t (\mathbf{y}_t - \mathbf{H}_t \hat{\mathbf{x}}_{t|t-1}) \quad (14)$$

$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \mathbf{P}_{t|t-1} \quad (15)$$

where \mathbf{K}_t is the Kalman gain.

Applications of State Space Models

1. Structural Time Series:

- Decompose series into trend, seasonal, cycle
- Handle missing data and irregular spacing
- Example: Local level model, local linear trend

2. Dynamic Linear Models:

- Time-varying regression coefficients
- Bayesian inference via MCMC or particle filters

3. ARMA as State Space:

- Any ARMA model can be written in state space form
- Enables ML estimation via Kalman filter

Point Forecasts

h-step ahead forecast:

$$\hat{y}_{T+h|T} = \mathbb{E}[y_{T+h}|y_1, \dots, y_T] \quad (16)$$

For ARIMA models:

- 1-step: Direct substitution
- Multi-step: Iterative forecasting
- Forecast variance increases with horizon h

[

title=AR(1) Forecast] Model: $y_t = 0.8y_{t-1} + \epsilon_t$

Forecasts:

$$\hat{y}_{T+1|T} = 0.8y_T$$

$$\hat{y}_{T+2|T} = 0.8\hat{y}_{T+1|T} = 0.8^2 y_T$$

$$\hat{y}_{T+h|T} = 0.8^h y_T \rightarrow 0 \text{ as } h \rightarrow \infty$$

Prediction Intervals

95% prediction interval:

$$\hat{y}_{T+h|T} \pm 1.96 \sqrt{\text{Var}(y_{T+h} - \hat{y}_{T+h|T})} \quad (17)$$

Forecast error variance:

- Increases with forecast horizon
- Depends on model parameters and residual variance
- For ARMA: Analytical formula available

[

title=Bootstrap Prediction Intervals] For complex models without analytical variance:

1. Fit model and obtain residuals
2. Resample residuals with replacement
3. Generate forecast paths

Forecast Evaluation Metrics

Common metrics for evaluating forecasts:

- **Mean Absolute Error (MAE):**

$$\text{MAE} = \frac{1}{n} \sum_{t=1}^n |y_t - \hat{y}_t| \quad (18)$$

- **Root Mean Squared Error (RMSE):**

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2} \quad (19)$$

- **Mean Absolute Percentage Error (MAPE):**

$$\text{MAPE} = \frac{100}{n} \sum_{t=1}^n \left| \frac{y_t - \hat{y}_t}{y_t} \right| \quad (20)$$

- **Symmetric MAPE (sMAPE):**

Recurrent Neural Networks (RNN)

[

title=RNN Architecture] Hidden state update:

$$\mathbf{h}_t = \tanh(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_h) \quad (22)$$

Output:

$$\mathbf{y}_t = \mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y \quad (23)$$

Challenges:

- **Vanishing gradients:** Hard to learn long-term dependencies
- **Exploding gradients:** Unstable training

Solution: Long Short-Term Memory (LSTM) and GRU

Long Short-Term Memory (LSTM)

LSTM cell with gating mechanisms:

Forget gate: $\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)$

Input gate: $\mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i)$

Candidate: $\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c)$

Cell state: $\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$

Output gate: $\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o)$

Hidden state: $\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$

Key features:

- Cell state \mathbf{c}_t provides long-term memory
- Gates control information flow
- Mitigates vanishing gradient problem

Sequence-to-Sequence (Seq2Seq):

- Encoder: Maps input sequence to context vector
- Decoder: Generates output sequence from context
- Used for variable-length forecasting

Attention Mechanism:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right) \mathbf{V} \quad (24)$$

- Allows decoder to focus on relevant parts of input
- Addresses bottleneck of fixed-size context vector
- Improves long-range dependencies

Transformers for Time Series

Self-Attention for Temporal Data:

- Parallel processing (no sequential constraint)
- Direct modeling of pairwise dependencies
- Positional encoding for temporal information

Recent architectures:

- **Temporal Fusion Transformer (TFT):**

- Multi-horizon forecasting
- Variable selection
- Interpretable attention weights

- **Autoformer:**

- Decomposition-based architecture
- Auto-correlation mechanism
- Strong seasonality modeling

- **Informer:**

- Efficient for long sequences
- ProbSparse attention
- Reduced computational complexity

Combining classical and deep learning:

1. ES-RNN (Smyl, 2020):

- Exponential smoothing + LSTM
- Winner of M4 forecasting competition
- Combines statistical principles with DL flexibility

2. N-BEATS (Oreshkin et al., 2019):

- Pure deep learning, no domain knowledge
- Interpretable decomposition (trend + seasonality)
- Strong performance on multiple benchmarks

3. Prophet (Facebook):

- Additive model with automatic changepoint detection
- Handles missing data and outliers
- User-friendly for business forecasting

Model Selection and Validation

Time Series Cross-Validation:

- **Rolling window:** Train on $[1, t]$, test on $[t + 1, t + h]$
- **Expanding window:** Increasingly larger training sets
- Never use future data for training (data leakage!)

Model Selection Criteria:

- **AIC:** $-2 \log L + 2k$ (penalizes complexity)
- **BIC:** $-2 \log L + k \log n$ (stronger penalty)
- **Out-of-sample error:** Most reliable for forecasting

Overfitting Warning

Complex models (many parameters, deep networks) can overfit!

- Use validation set
- Regularization (L1/L2, dropout)
- Ensemble methods

Common Pitfalls

1. Ignoring stationarity:

- Test for unit roots (ADF, KPSS)
- Transform data appropriately

2. Data snooping:

- Don't repeatedly test on same validation set
- Use separate test set for final evaluation

3. Spurious patterns:

- Be skeptical of perfect fits
- Check residual diagnostics

4. Structural breaks:

- Economy/system changes over time
- Consider adaptive methods or regime-switching models

5. Scale issues:

- Neural networks sensitive to input scale
- Normalize/standardize features

Summary

Classical Methods:

- ARIMA: Foundation of time series analysis
- VAR: Multivariate relationships
- State Space: Flexible framework, optimal filtering

Deep Learning:

- LSTM/GRU: Handle long-term dependencies
- Transformers: State-of-the-art for many tasks
- Hybrid: Combine strengths of both approaches

Key Takeaways:

- No universal best method
- Understand your data and problem
- Validate carefully with proper time series CV
- Consider ensemble of multiple approaches

Classical Methods:

- Hamilton (1994). *Time Series Analysis*
- Box, Jenkins, Reinsel (2015). *Time Series Analysis: Forecasting and Control*
- Lütkepohl (2005). *New Introduction to Multiple Time Series Analysis*

Deep Learning:

- Goodfellow, Bengio, Courville (2016). *Deep Learning*
- Lim, Zohren (2021). "Time-series forecasting with deep learning: a survey"

Applied:

- Hyndman, Athanasopoulos (2021). *Forecasting: Principles and Practice*
- Tsay (2014). *Multivariate Time Series Analysis*

Acknowledgments

- ESMAD for institutional support
- Mysense.ai for industry applications
- Time series research community

Generated with \LaTeX Beamer

Theme: ESMAD Professional Academic Style

Diogo Ribeiro

ESMAD - Escola Superior de Média Arte e Design
Lead Data Scientist, Mysense.ai



dfr@esmad.ipp.pt



0009-0001-2022-7072



github.com/diogoribeiro7



linkedin.com/in/diogoribeiro7

This presentation is part of the academic materials repository.
Available at: <https://github.com/diogoribeiro7/academic-presentations>