30: Time Series – Classical Methods

When the order of the rows is the whole point

Every model so far assumed the rows were interchangeable

Regression, trees, logistic regression – all of them treat the dataset as a *bag* of i.i.d. rows. Shuffle the rows, nothing changes.

Time series breaks that assumption. Row order *is* the signal. Yesterday's value tells you a lot about today's. You can only train on the past, and you must predict the future – never the other way around.

Examples everywhere: daily sales, electricity demand, website traffic, the dram-to-dollar exchange rate, sensor readings, COVID case counts, your heart rate.

Two lectures. **Today (30):** the classical statistical toolkit – decomposition, stationarity, ARIMA, exponential smoothing. **Next (31):** treating forecasting as a supervised ML problem.

Outline

What makes time series different

One column, indexed by time – and the index carries information.

The vocabulary

A time series is a sequence $y_1, y_2, \ldots, y_T$ sampled at regular steps (hourly, daily, monthly, ...).

Two jobs:

- **Forecasting** – predict future values $y_{T+1}, \ldots, y_{T+h}$. (h = forecast *horizon*.)
- **Understanding** – decompose, explain, find structure.

Two shapes of data:

- **Univariate** – one series.
- **Multivariate** – many series, possibly with exogenous drivers.

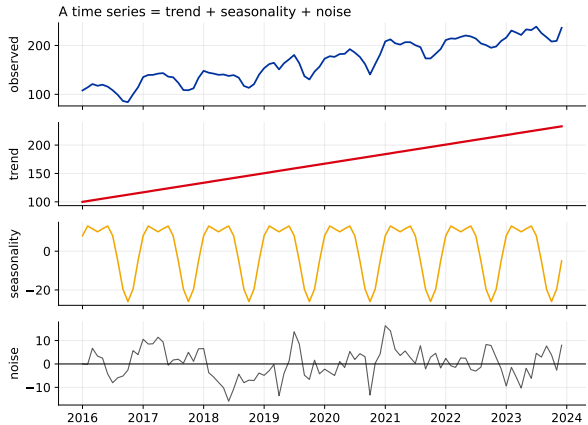
The one rule that governs everything: information flows forward in time only. Any value, feature, or statistic you feed the model must have been *knowable at prediction time*. Violate this and you get "leakage" – great backtests, useless in production.

The four components of a series

A series is usefully thought of as a sum (or product) of:

- **Trend** T_t – long-run drift up or down.
- **Seasonality** S_t – a pattern that repeats on a *fixed* period (24h, 7d, 12mo).
- **Cycle** – swings with no fixed period (business cycles).
- **Noise / residual** R_t – what's left.

Additive: $y_t = T_t + S_t + R_t$.
Multiplicative: $y_t = T_t \cdot S_t \cdot R_t$ (season grows with level – take log to make it additive).



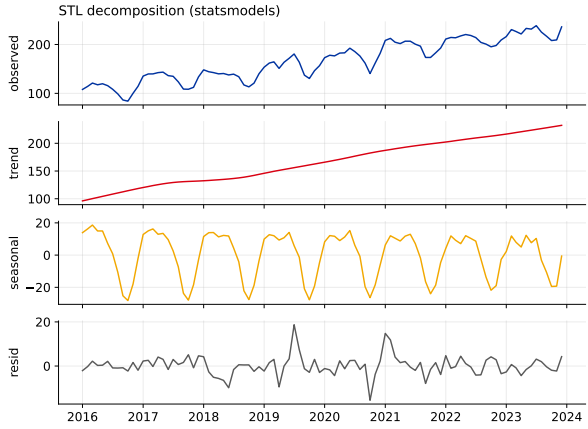
Decomposition: STL

STL = "Seasonal-Trend decomposition using Loess". It splits the series into trend + seasonal + residual, robustly.

Why bother?

- *See* the structure before modeling.
- Deseasonalize (analyze the trend cleanly).
- The residual is where anomalies show up.

```
from statsmodels.tsa.seasonal
    import STL
res = STL(y, period=12).fit()
res.trend, res.seasonal, res.
    resid
```



Stationarity and differencing

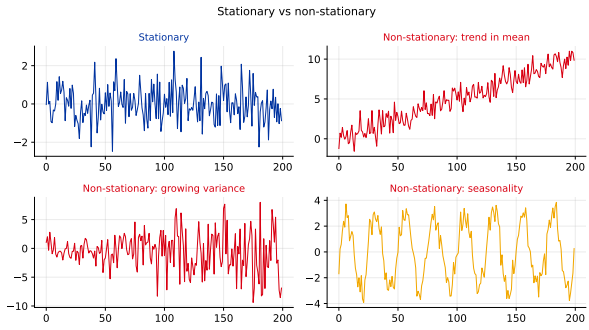
The classical models need a series whose behaviour does not drift.

What is stationarity?

A series is (weakly) **stationary** if its statistical behaviour does not depend on *when* you look:

- constant mean (no trend),
- constant variance (no growing swings),
- autocovariance depends only on the *lag*, not on absolute time.

Why we care: ARMA-type models assume stationarity. Trends and seasonality must be removed *first*, then added back to the forecast.



Differencing makes a series stationary

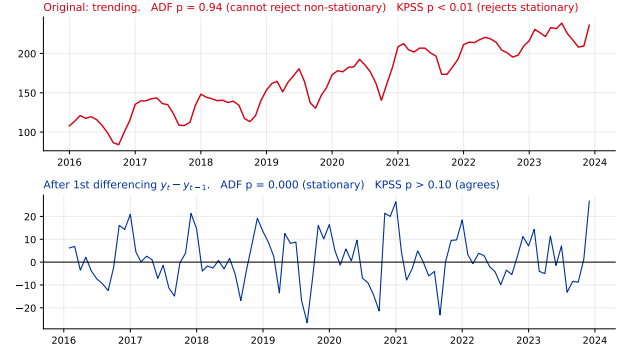
Differencing: model the change instead of the level.

$$y'_t = y_t - y_{t-1}$$

- Removes a trend (do it d times – usually $d = 1$).
- **Seasonal differencing** $y_t - y_{t-m}$ removes a season of period m.

Testing it – two tests, opposite nulls (small p rejects, so they point opposite ways):

- **Augmented Dickey-Fuller** (**ADF**) – H_0: non-stationary.
- **Kwiatkowski-Phillips-Schmidt-Shin** (**KPSS**) – H_0: stationary.



Predict-first: if one difference is good, are two better?

One difference took us from ADF $p = 0.94$ to $p = 0.000$. The series looks stationary.

Tempting thought: difference once more, just to be safe.

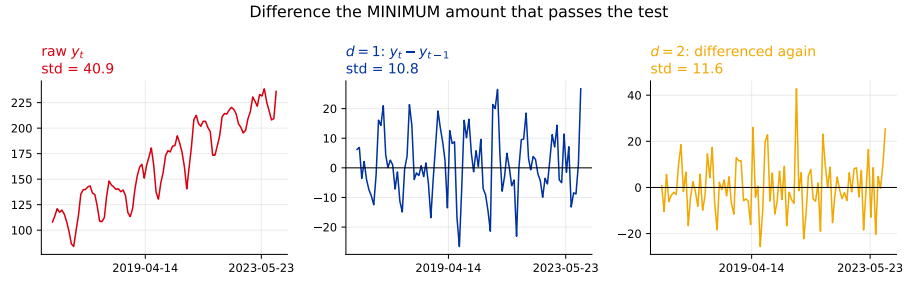
Predict: what happens to the spread of the values?

Predict-first: if one difference is good, are two better?

One difference took us from ADF $p = 0.94$ to $p = 0.000$. The series looks stationary.

Tempting thought: difference once more, just to be safe.

Predict: what happens to the spread of the values?



The spread goes *up*: 40.9 → 10.8 → 11.7. No trend was left to remove, so the second difference only differenced the **noise** – and the difference of noise is noisier still.

Autocorrelation

How a series correlates with delayed copies of itself.

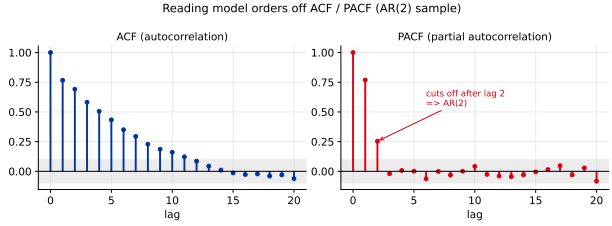
ACF and PACF

ACF (autocorrelation): correlation between y_t and y_{t-k} for each lag k. Includes indirect effects.

PACF (partial): correlation at lag k *after* removing the effect of the shorter lags. The direct link only.

They are the classic way to read off model orders:

- PACF cuts off after lag $p \Rightarrow$ **AR(p)**.
- ACF cuts off after lag $q \Rightarrow$ **MA(q)**.
- Both decay $\Rightarrow$ mix (ARMA).



The shaded band is the "not significantly different from zero" region ($\pm 1.96/\sqrt{T}$).

The ARIMA family

The workhorse of classical forecasting.

Building up: AR, MA, ARMA

AR(p) – autoregressive. Regress on your own past:

$$y_t = c + \sum_{i=1}^{p} \phi_i \, y_{t-i} + \varepsilon_t$$

"Tomorrow looks like a weighted blend of recent days."

MA(q) – moving average. Regress on past *shocks*:

$$y_t = c + \varepsilon_t + \sum_{j=1}^{q} \theta_j \, \varepsilon_{t-j}$$

"A surprise last week still echoes this week."

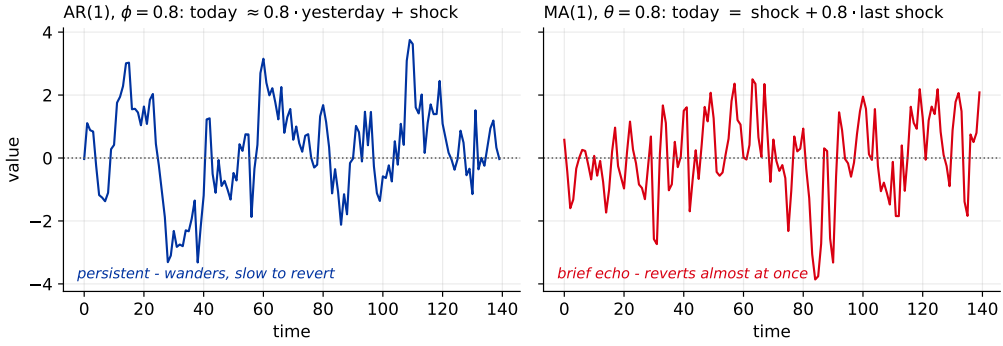
ARMA(p, q) combines both – for a *stationary* series.

ARIMA(p, d, q) adds the "I" = Integrated = difference d times first, so it handles trending series directly.

- p – AR order (from PACF)
- d – differencing (from ADF)
- q – MA order (from ACF)

ARIMA(0,1,0) is just the random walk – the "tomorrow = today" baseline.

What AR and MA actually look like



Same machinery, different memory: an **AR** term makes the series *persist* (today leans on yesterday's *value*); an **MA** term is a short *echo* of the last random *shock*. ARMA blends the two.

Careful: "moving average" means two different things

The MA in ARIMA is not a rolling mean.

What you already know – a *rolling* or *moving* average: a sliding mean of the **values**, used to smooth a curve.

$$\tfrac{1}{3}\left(y_{t-1} + y_{t-2} + y_{t-3}\right)$$

You compute it *from data you can see*.

MA(q) in ARIMA – a weighted sum of the past **shocks** (the model's own errors):

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \dots$$

The ε's are *unobserved*. You never measure them; the fitting procedure infers them. That is why MA orders cannot be read off the data directly the way a rolling mean can – and why they need the ACF.

Two unrelated ideas, one name, purely for historical reasons. Next lecture we build rolling-mean *features* (`y.shift(1).rolling(7).mean()`) and feed them to a tree – that is the *other* moving average. Keep them apart.

SARIMA: adding seasonality

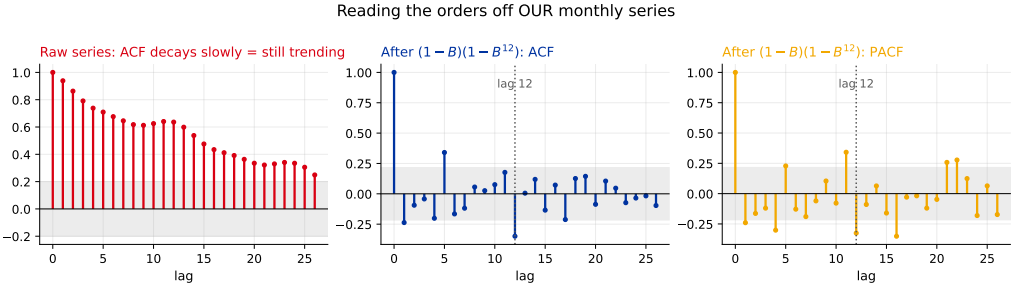
Real series are seasonal, so add a second, seasonal triple:

$$\text{SARIMA}(p, d, q)(P, D, Q)_m$$

- lower-case (p, d, q) – the short-term, non-seasonal part;
- upper-case (P, D, Q) – the same idea at the seasonal lag;
- m – periods per season (12 monthly, 7 daily-with-weekly, 24 hourly-with-daily).

One m only. Hourly electricity demand has a daily *and* a weekly *and* a yearly season – SARIMA can model one of them. That limitation is exactly why Fourier seasonal terms, and models like Prophet, exist.

Choosing the orders – on our own series



Left: the raw ACF decays slowly (lag $1 = 0.94$) – classic "still trending", so we need $d \geq 1$.

Middle: after $(1 - B)(1 - B^{12})$ – one ordinary and one seasonal difference – one clear *negative* spike at **lag 12** (-0.35). A single negative seasonal spike in the ACF is the signature of a **seasonal MA** term, so $Q = 1$.

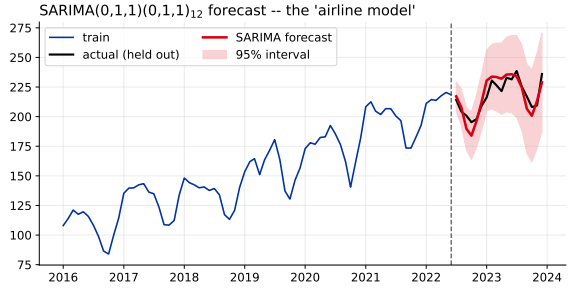
That gives SARIMA$(0, 1, 1)(0, 1, 1)_{12}$ – known as the **airline model**, because Box and Jenkins fitted exactly this to monthly airline passengers.

Did reading the plot actually help?

We could have guessed a seasonal *AR* term instead. Compare the two by **Akaike Information Criterion** (**AIC**) – fit quality penalised by the number of parameters, lower is better:

orders	AIC	test MAE
$(1,1,1)(1,1,0)_{12}$	356.4	7.24
$(0,1,1)(0,1,1)_{12}$	**343.8**	**6.07**

The plot was right: the seasonal MA wins on both, and we never touched the test set to find that out.



The forecast comes with a **prediction interval** – uncertainty grows with the horizon. A point forecast without an interval is only half an answer.

In practice: don't hand-tune p, d, q at all. Use `pmdarima.auto_arima` or `statsforecast.AutoARIMA`, which search orders by AIC for you. Read the plot to *understand*, let the search *choose*.

Exponential smoothing

Weighted averages where the recent past counts most.

From a simple average to Holt-Winters

The family is written **ETS – Error, Trend, Seasonality**, the three components you switch on or off.

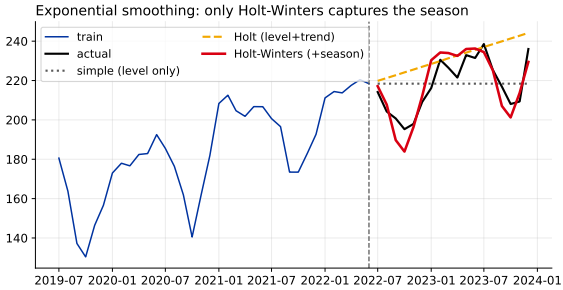
Simple Exponential Smoothing (**SES**) – level only. Weights decay geometrically into the past:

$$\hat{y}_{t+1} = \alpha y_t + (1-\alpha)\hat{y}_t$$

Flat forecast – no good if there is a trend or season.

Holt – adds a trend term (level + slope). Forecast is a line.

Holt-Winters – adds a seasonal term. Level + trend + season, each with its own smoothing rate (α, β, γ).



Baselines and evaluation

A forecast is only "good" relative to the dumbest thing that works.

Always start with a naive baseline

Before any model, write down the trivial forecast:

- **Naive:** $\hat{y}_t = y_{t-1}$ (tomorrow = today).
- **Seasonal naive:** $\hat{y}_t = y_{t-m}$ (this December = last December).
- **Drift:** last value + average change.

If your fancy model can't beat seasonal naive, it is not helping. This happens far more often than people admit.

Error metrics

- **Mean Absolute Error** (**MAE**) – in the target's units.
- **Root Mean Squared Error** (**RMSE**) – penalizes big misses more.
- **Mean Absolute Percentage Error** (**MAPE**) – breaks when $y \approx 0$, and punishes over-forecasting more than under-forecasting.
- **Mean Absolute Scaled Error** (**MASE**) – scale-free; the one to report.

MASE, by hand

$$\text{MASE} = \frac{\text{MAE of your forecast on the } \textbf{test} \text{ set}}{\text{MAE of the seasonal naive forecast } \textbf{in-sample}\text{, on the training set}}$$

Denominator – walk the *training* series and predict each month with the same month last year. Say those errors are $|-12|, 20, |-18|, 22$:

$$\frac{12+20+18+22}{4} = 18$$

Numerator – your model's test errors, say $5, |-9|, 7, 3$:

$$\frac{5+9+7+3}{4} = 6$$

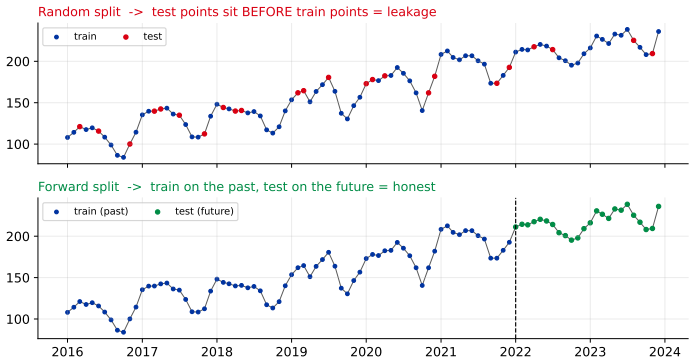
$$\text{MASE} = \frac{6}{18} = \mathbf{0.33}$$

Your errors are a third of what naive repetition would cost. **Below 1 is a win, above 1 means you lost to copying last year.**

Note the asymmetry: the denominator is measured *in-sample*, on train. So the seasonal naive evaluated on the *test* set does **not** have to score exactly 1.00 – it can land either side. Next lecture ours comes out at 1.22.

Evaluate on the future, never by random split

You cannot shuffle-and-split a time series: that trains on the future to predict the past.



This "time-aware validation" is the hinge between the two lectures. We build the full machinery – leakage, lag features, `TimeSeriesSplit` – in lecture 31.

One more trap: even the baseline has to respect the arrow

Seasonal naive means "this December = last December". Writing that as `y.shift(12)` across a test block looks harmless – and is wrong the moment your horizon exceeds one season.

Forecasting 18 months from one origin, steps 13–18 would reach back only 12 and land *inside the test window* – the "baseline" reads the future it is supposed to predict.

On our series that single fix moves the seasonal naive from MASE **0.90** to **1.22**.

Which is the whole point: a leaky baseline is the most dangerous kind. It sets the bar *too high*, so a good model looks mediocre – and you go tune something that was never the problem.

The honest version always reaches back to an *observed* value:

$$\hat{y}_{T+k} \;=\; y_{T+k-m\lceil k/m \rceil}$$

Rule of thumb for picking an approach: short, single, strongly-seasonal series → ARIMA / ETS are hard to beat. Many related series, long history, exogenous drivers → the ML and deep methods of the next lecture start to win.

Recap – classical time series

Ideas that transfer:

- Time order is information; only train on the past.
- Decompose into trend + season + residual (STL).
- Difference the *minimum* amount; check with ADF **and** KPSS.
- ACF/PACF read off AR/MA orders; confirm with AIC.
- MA(q) is about *shocks*, not a rolling mean.
- ARIMA: autocorrelation-based. ETS: component-based.
- Score with MASE – and make sure the baseline doesn't peek.

Next (31): ML for time series. Reframe forecasting as supervised learning –

- time-aware splits and leakage,
- lag / rolling / calendar features,
- `TimeSeriesSplit` cross-validation,
- gradient boosting (and the trend-extrapolation trap),
- deep learning and the new foundation models.

Questions?

Next: forecasting as a supervised ML problem.