








Time Series Analysis

Chapter 11: Foundation models for time series



Daniel Traian PELE

Bucharest University of Economic Studies
IDA Institute Digital Assets
Blockchain Research Center
AI4EFin Artificial Intelligence for Energy Finance
Romanian Academy, Institute for Economic Forecasting
MSCA Digital Finance

Acronyms used in this material (1/2)

- **AI**: Artificial Intelligence
- **AIC**: Akaike Information Criterion
- **API**: Application Programming Interface
- **AR**: AutoRegressive (model)
- **ARIMA**: AutoRegressive Integrated Moving Average (model)
- **BNR**: Banca Națională a României (the National Bank of Romania)
- **BY-NC**: Attribution–NonCommercial (Creative Commons licence terms)
- **CDF**: Cumulative Distribution Function
- **CPU**: Central Processing Unit
- **CRPS**: Continuous Ranked Probability Score
- **ENTSO-E**: European Network of Transmission System Operators for Electricity
- **ETS**: Error, Trend, Seasonality (exponential smoothing state space models)
- **FRED**: Federal Reserve Economic Data
- **GARCH**: Generalised AutoRegressive Conditional Heteroskedasticity

Acronyms used in this material (2/2)

- **GDP**: Gross Domestic Product
- **GIFT**: General Time Series Forecasting Model Evaluation (GIFT-Eval benchmark)
- **GPT**: Generative Pre-trained Transformer
- **GPU**: Graphics Processing Unit
- **HICP**: Harmonised Index of Consumer Prices
- **INS**: Institutul Național de Statistică — the National Institute of Statistics (Romania)
- **LLM**: Large Language Model
- **LOTSA**: Large-scale Open Time Series Archive (Moirai pretraining corpus)
- **MAE**: Mean Absolute Error
- **MASE**: Mean Absolute Scaled Error
- **RMSE**: Root Mean Squared Error
- **US**: United States
- **VAR**: Vector AutoRegression
- **WQL**: Weighted Quantile Loss

Today's question and route

- **Question**: can one large model, pretrained on many series, forecast a series it has never seen as well as a model built for that series?
  - until now (Chapters 0–10): one model per series, estimated on that series (ETS, ARIMA, GARCH, VAR)
  - Chapter 9: global machine-learning models, trained on the series of one data set
- **Route** of the chapter (self-study)
  - the idea of a foundation model; Transformers and attention in brief
  - from numbers to tokens: scaling, quantisation, patching; the main model families
  - probabilistic forecasts and their scores; an honest test on Romanian and US series
  - leakage and contamination; when these models help and when they do not

Learning outcomes

- Define a foundation model, zero-shot forecasting and fine-tuning
- Explain how a series becomes the input of a Transformer: mean scaling, quantisation, patching
- Describe Chronos, TimesFM, Moirai, Lag-Llama and TimeGPT and say how they differ
- Score a quantile forecast with the pinball loss, the CRPS, MASE and the coverage of an interval
- Run an open model on a CPU, compare it fairly with seasonal naive, ETS and ARIMA and recognise contamination

Prerequisites for Today

- Forecasting benchmarks and their evaluation (Chapters 0 and 4)
  - seasonal naive $\hat{y}_{T+h} = y_{T+h-m}$: the forecast repeats the value of the same season one cycle earlier
  - T: the last observed period; h: the horizon; m: the season length (12 for monthly data, 168 for hourly data with a weekly cycle); the hat marks a forecast
  - ETS; time-series cross-validation with a rolling origin; MAE, RMSE, MASE; the Diebold–Mariano test
- ARIMA and seasonal ARIMA (Chapters 3 and 4); machine learning basics (Chapter 9)
  - training, validation and test samples; a neural network as a flexible function with many parameters
  - global models: one model trained on many series
- Quantiles: q_τ is the value below which the variable falls with probability τ
  - $\tau \in (0, 1)$: the level; $q_{\mathbf{0.5}}$ is the median, $q_{\mathbf{0.1}}$ and $q_{\mathbf{0.9}}$ bound a central 80% interval

Three ways to forecast many series

- **Local** models
  - one model per series, estimated on that series only
  - ETS, ARIMA: few parameters, interpretable; 10 000 series mean 10 000 estimations
- **Global** models (Chapter 9)
  - one model trained on all the series of a data set
  - example: DeepAR (Salinas et al. (2020)), a recurrent network trained on all products of a retailer
- **Foundation** models
  - one model pretrained once on a very broad corpus of series from many domains
  - then used on new series and new data sets, without re-estimation
  - the term comes from Bommasani et al. (2021): models trained on broad data at scale, adaptable to many tasks

Zero-shot, fine-tuning, in-context

- **Context**
  - the recent history $y_{T-C+1}, \ldots, y_T$ passed to the model at forecast time (C = context length)
- **Zero-shot** forecasting
  - the pretrained weights stay fixed
  - only the context of the new series is used
  - no parameter is estimated on the target series: forecasting takes a fraction of a second
- **Fine-tuning**
  - further training of the pretrained weights on the target data
  - can help on unusual series; needs a validation sample and more computation
- The output is usually a set of **quantiles** $\hat{q}_{0.1}, \ldots, \hat{q}_{0.9}$ for each future step, not a single number

Pretraining at scale



A Google data centre, The Dalles, Oregon

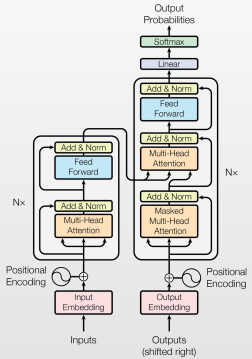
Photo: Visitor7 (2011); CC BY-SA 3.0; Wikimedia Commons

- Pretraining corpora mix public series from many domains
  - energy, transport, retail, weather, web traffic, finance and economics
  - plus synthetic series: random mixtures of real series and series drawn from Gaussian processes (Chronos)
- Scale: TimesFM was pretrained on about 10^{11} time points (Das et al. (2024))
- Pretraining is done once, by the developer, on many graphics processors (GPU)
  - the user only runs the model: the small versions run on an ordinary CPU

Recap: The foundation-model idea

- Local, global and foundation models differ in what the model is trained on
- Zero-shot: fixed weights, only the context of the new series
- The output is a set of quantiles: a probabilistic forecast

The Transformer (Vaswani et al., 2017)



The original diagram: encoder (left), decoder (right)

Figure: Vaswani et al. (2017), Google; CC BY-SA 4.0; Wikimedia Commons

- Vaswani et al. (2017): a network for sequences built on **attention**, without recurrence
  - designed for translation; behind most large language models (LLM)
- Input: a sequence of **tokens**, each turned into a vector (an embedding)
- **Encoder–decoder** architecture
  - the **encoder** reads the whole input
  - the **decoder** produces the output step by step
  - **decoder-only** models (GPT, TimesFM, Lag-Llama) forecast the next token from all previous ones
- Position information is added to each vector, since attention ignores order

Self-attention (1/2): the formula

- Each input vector x_i produces three vectors, by three linear maps learnt in training
  - a query $q_i = W_Q x_i$ (what position i looks for), a key $k_i = W_K x_i$ (what position i offers), a value $v_i = W_V x_i$ (what position i passes on)
  - W_Q, W_K, W_V: weight matrices learnt in training, the same for every position
- The output at position i is a weighted average of the values of all positions
  - $\text{out}_i = \sum_j w_{ij} v_j, \qquad w_{ij} = \dfrac{\exp(q_i^\top k_j / \sqrt{d})}{\sum_l \exp(q_i^\top k_l / \sqrt{d})}$
- Notation
  - j, l: indices that run over all input positions; w_{ij}: the weight that position i gives to position j
  - $q_i^\top k_j$: the scalar product of query and key, large when the two vectors point in the same direction (similarity score)
  - d: the length of the key vectors; dividing by $\sqrt{d}$ keeps the scores of moderate size
  - the exponential makes every weight positive and the denominator makes the weights of position i sum to 1

Self-attention (2/2): matrix form and intuition

- Matrix form of the same computation: $\text{softmax}(QK^\top/\sqrt{d})\,V$
  - Q, K, V: matrices whose rows are the vectors q_i, k_i, v_i; $QK^\top$ holds all the scores $q_i^\top k_j$
  - softmax: applied row by row, it turns the scores into the weights w_{ij} of the previous slide
- Intuition for series: today's step can look back at the same hour last week and give it a large weight
  - the weights are learnt from data, not fixed in advance as the lags of an AR(p)
- Cost: n inputs need n^2 comparisons
  - a context of 2048 values gives about 4 million comparisons in each attention layer

Worked example: attention weights

- One query and three keys; scores $q^\top k_j/\sqrt{d} = 2, 1, 0$; values $v_j = 10, 20, 30$
- Step 1: exponentiate: e^2, e^1, e^0
- Step 2: divide by the sum: weights 0.67, 0.24, 0.09 (they sum to 1)
- Step 3: output $= \sum_j w_j v_j = 14.2$
  - the highest score gets most of the weight, but every value contributes
- A foundation model stacks many such layers, each with several attention heads

Recap: Transformers

- Attention: each step is a learnt weighted average of all steps
- Encoder–decoder (Chronos) and decoder-only (TimesFM, Lag-Llama) designs
- The cost grows with the square of the number of inputs

Representing numerical series as model inputs

- Series differ in units and levels: load in GW, an index around 100, an exchange rate around 5
  - one model for all of them needs a common scale
- Three solutions used in practice
  - **scaling + quantisation**: each value becomes one token from a fixed vocabulary (Chronos)
  - **patching**: groups of consecutive values become one input vector (TimesFM, Moirai, Chronos-Bolt)
  - **digits as text**: a general LLM reads the numbers as strings (Gruver et al. (2023))
- In all cases the forecasts are transformed back to the original units

Mean scaling and quantisation (Chronos) (1/2)

- **Mean scaling**
  - each value of the context is divided by the mean absolute value of the context
  - $\tilde{x}_t = x_t / s, \qquad s = \frac{1}{C} \sum_{t=1}^{C} |x_t|$
  - x_t: the t-th value of the context, $t = 1, \ldots, C$; C: the context length; s: the scale; $\tilde{x}_t$: the scaled value
  - the scaled context has values of order one, whatever the units
- **Quantisation**
  - each scaled value is replaced by the number of a bin
  - $[-15, 15]$ is divided into 4093 equal bins of width 0.0073; the token of $\tilde{x}_t$ is the number of the nearest bin centre
  - plus special tokens (padding, end of sequence)

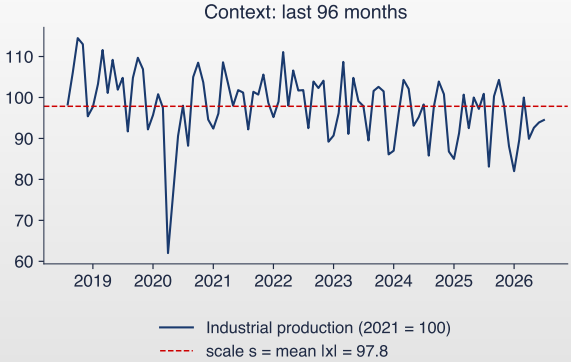
Mean scaling and quantisation (Chronos) (2/2)

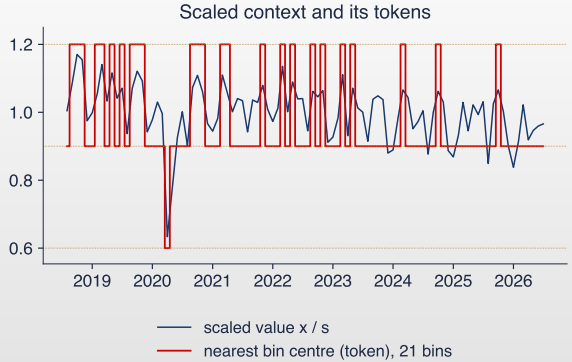
- Training: a language model (T5) learns to predict the next token
  - loss: the cross-entropy $-\log \hat{p}(z_{t+1})$, where z_{t+1} is the true next token and $\hat{p}(z_{t+1})$ the probability the model gave it
  - it is 0 when that probability is 1 and grows as the probability falls
- Forecast in three steps
  - sample many token paths from the model
  - map each token back to its bin centre and multiply by s
  - read the quantiles of the sampled values at each future step
- Limit: scaled values outside $[-15, 15]$ cannot be produced

Worked example: scaling and tokens

- Context: the Romanian unemployment rate (%, seasonally adjusted, Eurostat), March 2026 to August 2026
  - $x = 6.5, 6.8, 6.7, 6.8, 6.4, 6.4$
- Step 1: scale s = mean of $|x_t| = 6.60$
- Step 2: scaled values $x_t/s = 0.985, 1.030, 1.015, 1.030, 0.970, 0.970$
- Step 3: nearest bin centres (0 to 4092): tokens 2180, 2187, 2184, 2187, 2178, 2178
  - equal values give equal tokens; a change of 0.1 points moves the token by several bins
- The model never sees the level 6.6%: only the shape relative to the scale

Scaling and quantisation on real data (1/2)





Scaling and quantisation on real data (2/2)

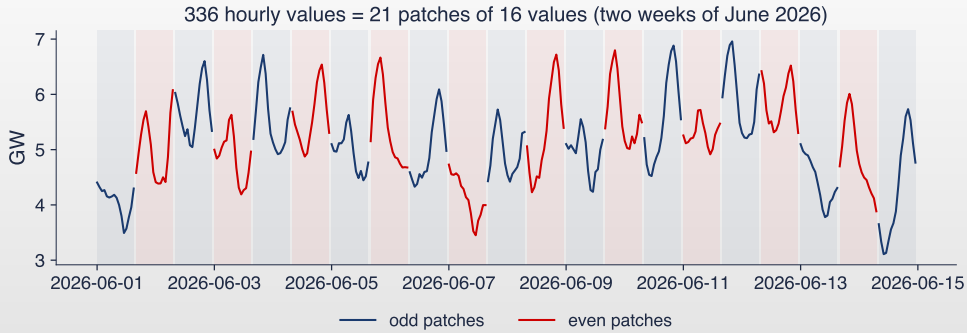
- Romanian industrial production (Eurostat, 2021 = 100), the last 96 months; scale $s = 97.8$
- Right: the scaled context and its tokens on a coarse grid of 21 bins (Chronos uses 4093)
- After scaling, all values lie between 0.63 and 1.17: a few per cent of the range $[-15, 15]$
- On the coarse grid the 96 months use only 3 tokens: the April 2020 fall (the lockdown) is one separate token
- With 4093 bins the grid is fine enough to keep the seasonal shape
- One extreme value inflates s and shrinks all the other scaled values: outliers in the context matter

TSA_ch11_tokens_scores

Patching

- **Patch**
  - a block of P consecutive values, mapped to one input vector by a small network
  - PatchTST (Nie et al. (2023)): "a time series is worth 64 words"
- Two gains
  - fewer inputs: 2048 values with $P = 16$ give 128 patches, so attention needs 128^2 instead of 2048^2 comparisons
  - each input carries a local shape (a slope, a peak), not a single noisy number
- Used by TimesFM (input patches of 32), Moirai (patch size by frequency), Chronos-Bolt and Chronos-2 (16)
  - the output can also be a whole patch of future values at once, which is faster than token by token

Patching the Romanian electricity load



- 336 hourly values (two weeks of June 2026, ENTSO-E) cut into 21 patches of 16 values

TSA_ch11_tokens_scores

Interpreting patching

- A day has 24 hours, so a patch of 16 covers two thirds of a day: patches do not align with the daily cycle
- Attention links patches: the morning rise of Monday can be matched with that of the previous Monday
- A weekly cycle (168 hours) is visible only if the context covers at least one week: 11 patches

Recap: From numbers to tokens

- Mean scaling removes units; quantisation turns values into tokens (Chronos)
- Patching groups values: shorter sequences, local shapes
- The forecast is transformed back to the original units

Chronos, Chronos-Bolt and Chronos-2 (Amazon)



The Amazon Spheres, Seattle

Photo: Andy Li (2018); CC0; Wikimedia Commons

- **Chronos** (Ansari et al. (2024))
  - T5 encoder–decoder on scaled and quantised values
  - samples token paths
- **Chronos-Bolt** (November 2024)
  - patches of 16 values and a direct output of the quantiles 0.1, . . . , 0.9
  - sizes used here: tiny 9 and small 48 million parameters; context up to 2048 values
- **Chronos-2** (Ansari et al. (2025), October 2025)
  - 119 million parameters, 21 quantile levels, covariates and groups of related series
- Open weights (Apache-2.0 licence); Python package `chronos-forecasting`; all run on a CPU

TimesFM (Google) and Lag-Llama

- **TimesFM** (Das et al. (2024))
  - decoder-only Transformer, input patches of 32, output patches of 128 values
  - pretrained on real series (Google Trends, Wikipedia page views) and synthetic ones; about 200 million parameters
  - open weights (Apache-2.0); later versions add quantile outputs
- **Lag-Llama** (Rasul et al. (2023))
  - decoder-only, built on the Llama architecture of language models
  - inputs: lagged values chosen by frequency (for example 7 days, 12 months, 24 hours) plus calendar features
  - output: the parameters of a Student-t distribution for the next step; sampling gives paths
  - small and open (Apache-2.0); meant to be fine-tuned

Moirai (Salesforce) and TimeGPT



Salesforce Tower, San Francisco

Photo: Saggittarius A (2020); CC BY-SA 4.0; Wikimedia Commons

- **Moirai** (Woo et al. (2024))
  - "universal" forecaster
  - any number of variables in one sequence; patch size chosen by frequency
  - output: a mixture of distributions; corpus LOTSA, about 27 billion observations
  - open weights under a non-commercial licence (CC BY-NC 4.0)
- **TimeGPT** (Garza et al. (2023))
  - closed model, available only through a paid API
  - the weights and the training data cannot be inspected; the version can change; the data leave your computer
  - not used in this course: the results could not be reproduced

The families side by side

Model	Architecture	Input	Output	Access
Chronos (2024)	encoder–decoder (T5)	tokens (bins)	sampled paths	Apache-2.0
Chronos-Bolt (2024)	encoder–decoder	patches of 16	9 quantiles	Apache-2.0
Chronos-2 (2025)	encoder	patches + covariates	21 quantiles	Apache-2.0
TimesFM (2024)	decoder	patches of 32	patches of 128	Apache-2.0
Moirai (2024)	encoder	patches, any variables	mixture	CC BY-NC 4.0
Lag-Llama (2023)	decoder (Llama)	lagged values	Student-t	Apache-2.0
TimeGPT (2023)	encoder–decoder (few details)	–	points, intervals	paid API

- A further line of work feeds the digits to general LLMs (Gruver et al. (2023)); Tan et al. (2024) show that the language-model part adds cost but no accuracy

Recap: The model families

- Chronos: tokens; Chronos-Bolt and Chronos-2: patches and direct quantiles
- TimesFM and Lag-Llama: decoder-only; Moirai: any number of variables
- Open weights make results reproducible; a paid API does not

Quantile forecasts

- A **probabilistic forecast** gives the distribution of y_{T+h} given the data up to T
  - foundation models report it through quantiles $\hat{q}_\tau$ at levels $\tau = 0.1, \ldots, 0.9$
- Point forecast: the median $\hat{q}_{0.5}$
- Central 80% interval: $[\hat{q}_{0.1}, \hat{q}_{0.9}]$; a **fan chart** draws several such bands
- Two qualities to check
  - **calibration**: the 80% interval contains about 80% of the outcomes
  - **sharpness**: among calibrated forecasts, narrower is better (Gneiting and Raftery (2007))
- For ETS and ARIMA we use Normal quantiles $\hat{q}_\tau = \hat{y}_{T+h} + z_\tau \hat{\sigma}_h$
  - z_τ: the τ quantile of the standard Normal distribution ($z_{0.9} = 1.28$); $\hat{\sigma}_h$: the estimated standard deviation of the h-step forecast error

The pinball loss

- The **pinball** (quantile) loss scores one forecast quantile $\hat{q}_\tau$ against the outcome y
  - $L_\tau(y, \hat{q}_\tau) = \tau\,(y - \hat{q}_\tau)$ if $y \geq \hat{q}_\tau$, and $(1-\tau)(\hat{q}_\tau - y)$ otherwise
- Notation
  - τ: the quantile level; $\hat{q}_\tau$: the forecast quantile; y: the value observed later
- Reading
  - the loss is 0 when $\hat{q}_\tau = y$ and grows linearly with the distance between them
  - an outcome above the quantile is weighted by τ, one below by $1 - \tau$
  - its expected value is smallest at the true quantile: reporting the true quantile is the best strategy

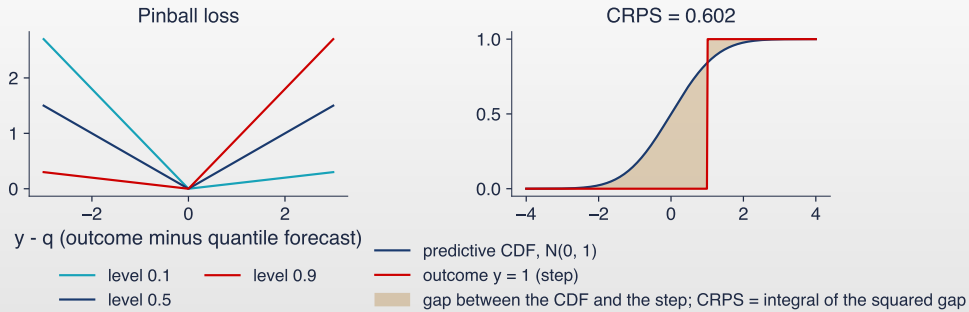
The CRPS

- The **CRPS** (continuous ranked probability score, Matheson and Winkler (1976)) scores a whole forecast distribution against the outcome y
  - $\mathrm{CRPS}(F, y) = \int (F(z) - \mathbf{1}\{z \geq y\})^{\mathbf{2}}\, dz = 2 \int_{\mathbf{0}}^{\mathbf{1}} L_\tau(y, F^{-\mathbf{1}}(\tau))\, d\tau$
- Notation
  - F: the forecast cumulative distribution function (CDF); z: the integration variable, running over all possible values
  - $\mathbf{1}\{z \geq y\}$: the indicator, 1 if $z \geq y$ and 0 otherwise (the CDF of a forecast that puts all mass on y)
  - $F^{-\mathbf{1}}(\tau)$: the τ quantile of F; the second form is the pinball loss averaged over all levels
- Reading
  - CRPS ≥ 0, in the units of y; 0 only for a forecast that puts all its mass on the outcome; smaller is better
  - for a point forecast it equals $|y - \hat{y}|$: the CRPS generalises the MAE

The weighted quantile loss (WQL)

- **WQL** (weighted quantile loss)
  - the CRPS approximated with the 9 levels 0.1, ..., 0.9 and divided by the size of the series
  - WQL $= \frac{1}{9} \sum_\tau \frac{2 \sum_t L_\tau(y_t, \hat{q}_{\tau,t})}{\sum_t |y_t|}$
- Notation
  - τ: runs over the 9 levels; t: runs over all forecast steps of all origins
  - $\hat{q}_{\tau,t}$: the forecast τ quantile for step t; y_t: the outcome at step t
- Reading
  - the denominator $\sum_t |y_t|$ removes the units: series with different levels can be compared
  - 0 for a perfect forecast; smaller is better; used by Chronos and GIFT-Eval

The pinball loss and the CRPS



- Left: L_τ against $y - \hat{q}$ for $\tau = 0.1, 0.5, 0.9$. Right: the forecast N(0, 1), the outcome $y = 1$
- N(μ, σ^2): the Normal distribution with mean μ and variance σ^2

TSA_ch11_tokens_scores

Interpreting the scores

- At $\tau = 0.9$ an outcome above the quantile costs 9 times more than one below: the 0.9 quantile should be exceeded only 10% of the time
- At $\tau = 0.5$ the loss is half the absolute error: the median is scored like a point forecast
- CRPS of N(0, 1): 0.234 if $y = 0$, 0.602 if $y = 1$; N(0, 4) at $y = 1$ gives 0.663: a wider band is penalised even when it covers the outcome
- The CRPS rewards calibration and sharpness at the same time

Worked example: pinball losses

- Forecast of next month's unemployment rate: $\hat{q}_{0.1} = 6.5$, $\hat{q}_{0.5} = 7.0$, $\hat{q}_{0.9} = 7.6$; outcome $y = 7.2$
- $\tau = 0.1$: y is above, loss $0.1 \cdot (7.2 - 6.5) = 0.07$
- $\tau = 0.5$: y is above, loss $0.5 \cdot (7.2 - 7.0) = 0.10$
- $\tau = 0.9$: y is below, loss $(1 - 0.9) \cdot (7.6 - 7.2) = 0.04$
- Approximate CRPS with these three levels: $\frac{2}{3}$ (sum of losses) $= 0.140$ percentage points
  - the outcome lies inside the 80% interval [6.5; 7.6]: it counts as covered in the coverage score (two slides on)

Point accuracy: the MASE

- **MASE** (mean absolute scaled error, Hyndman and Koehler (2006))
  - the MAE of the forecast divided by the in-sample MAE of the seasonal naive
  - $\text{MASE} = \dfrac{\frac{1}{h}\sum_{j=1}^{h} |y_{T+j} - \hat{y}_{T+j}|}{\frac{1}{T-m}\sum_{t=m+1}^{T} |y_t - y_{t-m}|}$
- Notation
  - numerator: the mean absolute error of the forecasts $\hat{y}_{T+j}$ over the h steps after the origin T
  - denominator: the mean absolute error of the seasonal naive forecast y_{t-m} inside the sample $t = m+1, \ldots, T$; m: the season length
- Reading
  - below 1: better than the in-sample seasonal naive; above 1: worse; free of units
  - example: MAE 0.30, in-sample naive MAE 0.25, so MASE = 1.2

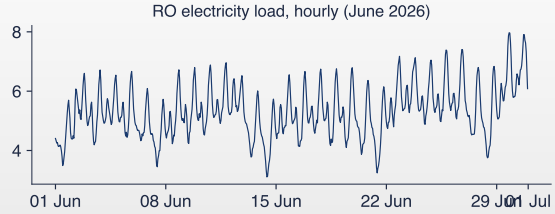
Coverage and relative scores

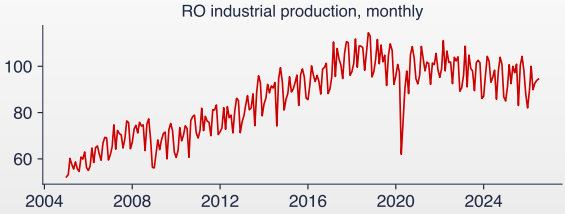
- **Coverage**
  - the share of outcomes inside the 80% interval
  - $\text{Cov} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{\hat{q}_{0.1}^{(i)} \le y_i \le \hat{q}_{0.9}^{(i)}\}$; n: the number of forecasts; $\hat{q}_{\tau}^{(i)}$: the quantile of forecast i; $\mathbf{1}\{\cdot\}$: 1 if the outcome is inside, 0 otherwise
  - target 80%; 7 of 10 inside means 70%: intervals somewhat too narrow
- **Relative scores**
  - a model's MASE or WQL divided by that of the seasonal naive on the same origins
  - below 1: the model beats the seasonal naive
  - averaged over the S series with the geometric mean $(r_1 r_2 \cdots r_S)^{1/S}$, r_k: the ratio for series k; so 2 and 0.5 cancel out

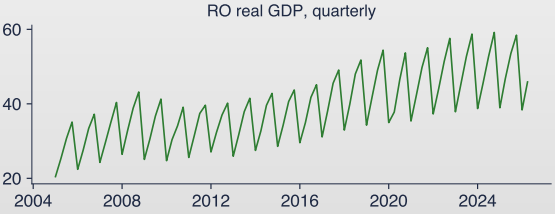
Recap: Scores

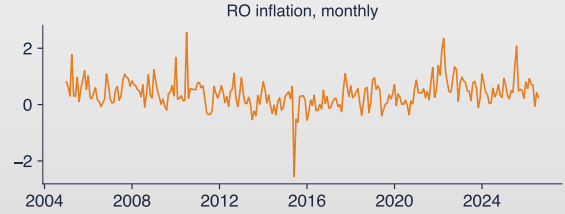
- Pinball loss for each quantile; CRPS for the whole distribution; WQL as its scaled approximation
- MASE for the median; coverage for calibration
- Report scores relative to the seasonal naive forecast

Seven series for the test (1/3)

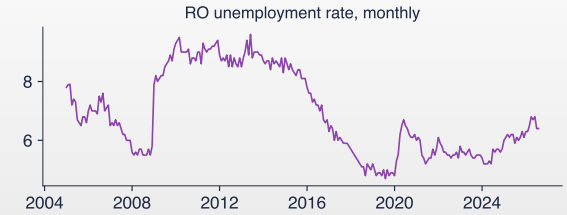


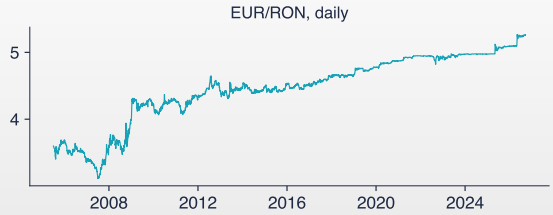


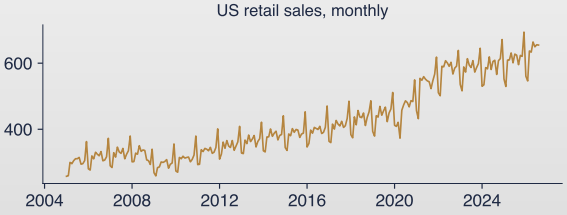




Seven series for the test (2/3)





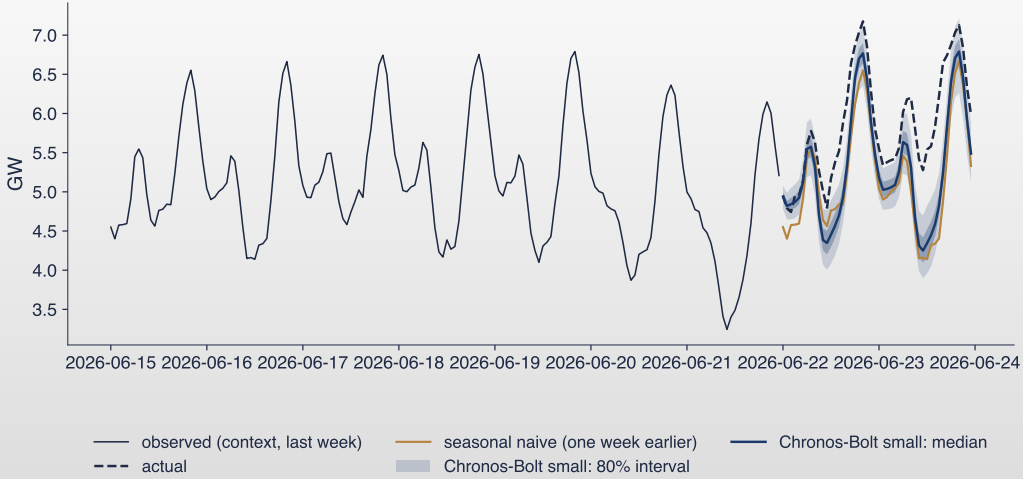


Seven series for the test (3/3)

- Romania (ENTSO-E)
  - hourly electricity load
- Romania (Eurostat)
  - industrial production
  - real GDP, not seasonally adjusted
  - HICP inflation; unemployment rate
- EUR/RON (BNR reference rate)
- US retail sales (FRED RSXFSN)
- Strong seasonality: load (daily and weekly cycles), industrial production and US retail sales (yearly), GDP (quarterly)
- Close to a random walk: EUR/RON and the unemployment rate; noisy and short-lived: monthly inflation
- Breaks: the 2020 pandemic (industrial production, GDP, retail); the 2022 inflation surge
- Lengths from 126 quarters (GDP) to 39 408 hours (load): very different amounts of context

TSA_ch11_tokens_scores

Romanian electricity load, 48 hours ahead (1/2)



Romanian electricity load, 48 hours ahead (2/2)

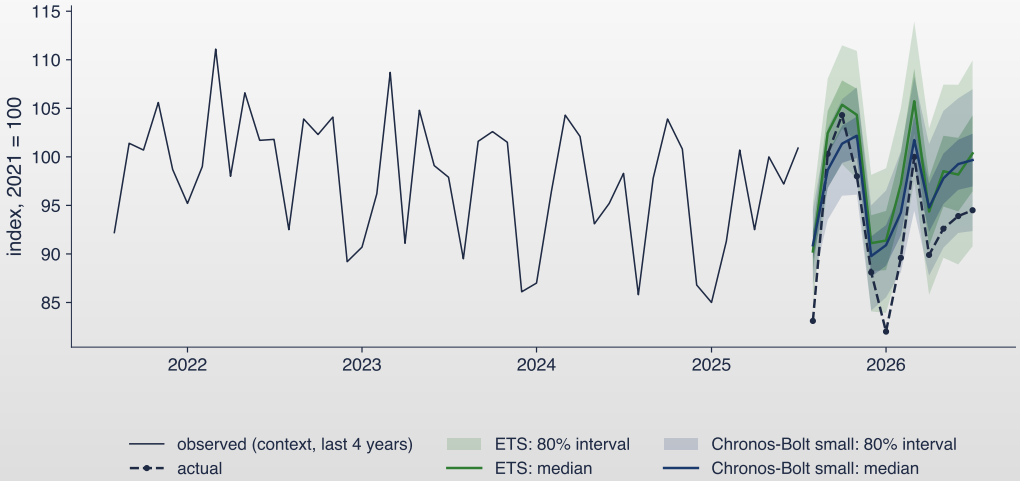
- Chronos-Bolt small, zero-shot, context of 2048 hours (85 days); origin 22 June 2026, 00:00; bands: 50% and 80% central intervals
- The model reproduces the daily shape (night trough, evening peak) without any estimation on this series
- MASE: Chronos-Bolt 1.31, seasonal naive 1.61: better than repeating last week
- But the actual load stays above the forecast all day: only 29% of the 48 hours fall in the 80% band
- The model has no temperature or calendar input: a change in the weather that the past 85 days do not announce cannot be anticipated
- One origin proves little: the systematic test follows

TSA_ch11_zero_shot

Worked example: scoring one hour

- Hour 24 of the load forecast (22 June 2026, 23:00): $\hat{q}_{0.1} = 5.1837$, $\hat{q}_{0.5} = 5.4763$, $\hat{q}_{0.9} = 5.7758$ GW; outcome 5.8033 GW
- All three quantiles are below the outcome, so each loss is $\tau\,(y - \hat{q}_\tau)$
  - $\tau = 0.1$: 0.062; $\tau = 0.5$: 0.163; $\tau = 0.9$: 0.025 GW
- The outcome is above $\hat{q}_{0.9}$: coverage indicator 0 for this hour
- Averaged over many hours and origins, these losses give the WQL of the next section

Romanian industrial production, 12 months ahead (1/2)

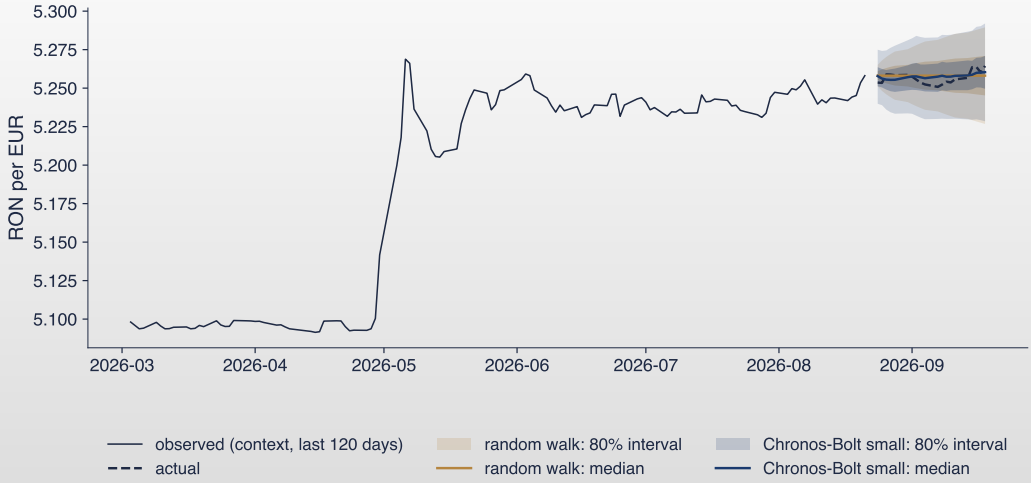


Romanian industrial production, 12 months ahead (2/2)

- Origin August 2025, forecasts to July 2026; Chronos-Bolt small (zero-shot) and ETS (estimated on the series), both with 80% bands
- Both models copy the seasonal pattern (the August and December dips) and keep the recent level
- MASE: Chronos-Bolt 1.04, ETS 1.20, seasonal naive 0.67: on this origin the simplest forecast wins
- The weak production of 2026 is not announced by the context; the 80% bands cover 83% (Chronos-Bolt) and 83% (ETS) of the months
- The bands of Chronos-Bolt are narrower than those of ETS at long horizons

TSA_ch11_zero_shot

EUR/RON, 20 days ahead (1/2)



EUR/RON, 20 days ahead (2/2)

- BNR reference rate; origin 24 August 2026, last value before it 5.2581; Chronos-Bolt small against the random walk with Normal bands
- The median after 20 days is 5.2604, practically the last value 5.2581: the model has learnt to behave like a random walk
- Width of the 80% band at 20 days: 0.064 (Chronos-Bolt) and 0.063 (random walk): almost the same uncertainty
- MAE over the 20 days: 0.0032 against 0.0035: a tie in practice
- A model cannot extract what the past does not contain: exchange rates stay close to unpredictable (Chapter 1)

TSA_ch11_zero_shot

Recap: Zero-shot forecasts

- Without estimation, the model copies cycles and levels from the context
- It cannot anticipate what the context does not show (heat, a downturn)
- On a random walk it returns a random walk

The test protocol

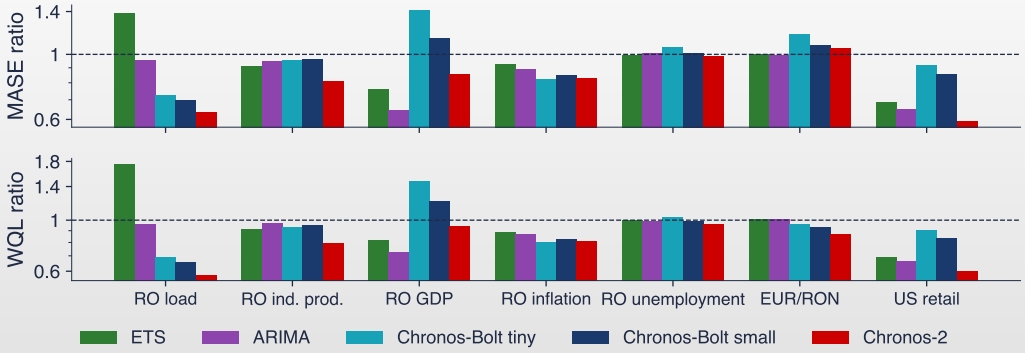
- **Rolling origin** (Chapter 4)
  - at each origin every model forecasts h steps from the data up to the origin only
  - load: $h = 48$ hours, weekly origins July 2025 to June 2026; monthly series: $h = 12$, monthly origins from 2016; GDP: $h = 4$, from 2012; EUR/RON: $h = 20$ days, every 20 days from 2016
  - 708 origins in total; context: all the past, at most 2048 observations
- **Benchmarks**, re-estimated at every origin
  - seasonal naive (the naive forecast for unemployment and EUR/RON)
  - ETS with damped trend and additive seasonality; ARIMA with orders chosen once by AIC before the first origin (load: ARIMA(2,0,2) on weekly differences)
- **Foundation models**, zero-shot
  - Chronos-Bolt tiny and small, Chronos-2
- Same origins, same context, same horizon, same scores: MASE, WQL, coverage

Results: WQL relative to the seasonal naive

Series	Origins	ETS	ARIMA	Bolt tiny	Bolt small	Chronos-2
RO load (hourly)	52	1.75	0.96	0.69	0.66	0.58
RO industrial production	116	0.92	0.97	0.94	0.95	0.79
RO real GDP	55	0.82	0.72	1.48	1.22	0.94
RO inflation	117	0.88	0.87	0.81	0.82	0.81
RO unemployment	117	1.00	0.99	1.03	0.99	0.96
EUR/RON (daily)	134	1.01	1.01	0.96	0.93	0.87
US retail sales	117	0.69	0.66	0.90	0.84	0.60
Geometric mean		0.97	0.87	0.95	0.90	0.78

- Below 1: better than the seasonal naive; each value is the WQL of the model divided by that of the seasonal naive on the same origins

MASE and WQL by series and model



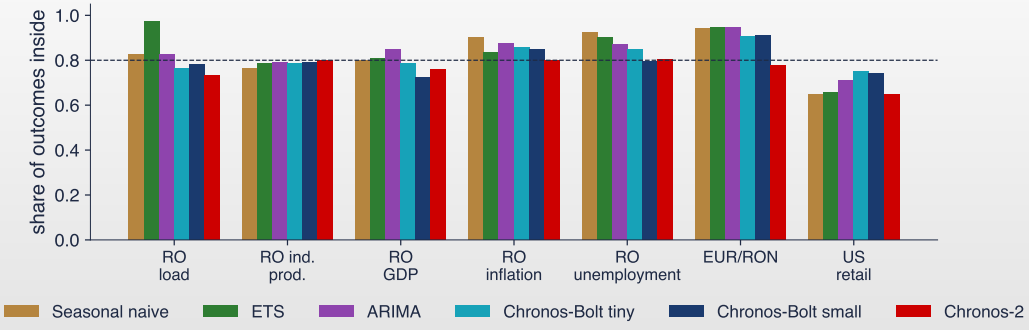
- Ratios to the seasonal naive forecast (log scale; the dashed line is 1)

TSA_ch11_benchmark

Interpreting the benchmark

- Hourly load: the foundation models win clearly (WQL ratio 0.58 for Chronos-2, 0.66 for Bolt small, 0.96 for ARIMA); ETS with a daily season only fails (1.75)
- Romanian GDP (short quarterly series): ARIMA is best (0.72); Chronos-Bolt is worse than the seasonal naive (1.22)
- Unemployment: all models are close to 1; EUR/RON: close to 1 except Chronos-2 (WQL 0.87), whose median is still no better than the naive (MASE 1.05)
- Geometric mean of WQL: Chronos-2 0.78, ARIMA 0.87, Bolt small 0.90, Bolt tiny 0.95, ETS 0.97; Chronos-2 is best on 5 of 7 series
- A well-specified ARIMA is as good as the small foundation models; a single winner on all series does not exist

Coverage of the 80% intervals



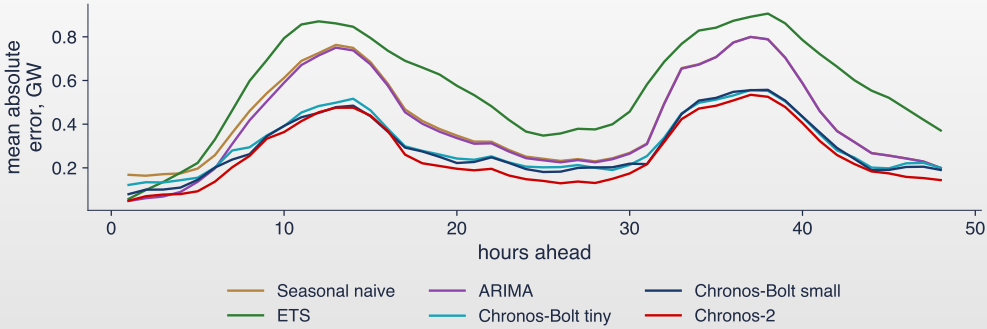
Share of outcomes inside $[\hat{q}_{0.1}, \hat{q}_{0.9}]$, all origins and steps; the dashed line is the target 0.8

TSA_ch11_benchmark

Interpreting the coverage

- Average coverage: seasonal naive 83%, ETS 84%, ARIMA 84%, Bolt small 80%, Chronos-2 76%
- Chronos-2 has the best WQL but intervals that are too narrow: sharp, not fully calibrated
- US retail sales: every model covers well below 80% (from 64.7% for the seasonal naive and 65.0% for Chronos-2 to 75.2% for Bolt tiny): the 2020 shock and the 2021 jump surprise all of them
- ETS and ARIMA bands assume Normal errors and constant variance: too wide in calm periods, too narrow in crises

Romanian load: error by horizon



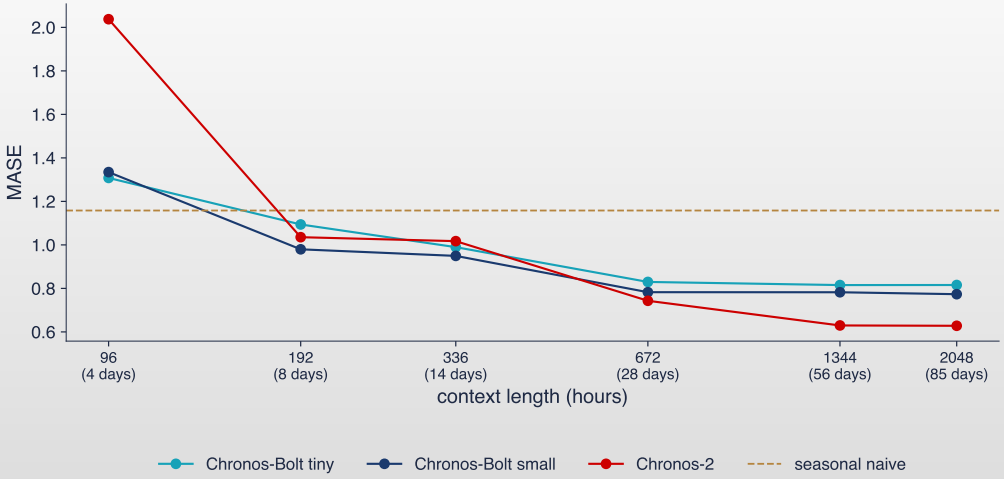
⊡ Mean absolute error (GW) of the median forecast for each of the 48 hours ahead, over 52 weekly origins

TSA_ch11_benchmark

Interpreting the error by horizon

- First hour: ARIMA 0.05 and Chronos-2 0.05 GW, far better than the seasonal naive 0.17: the last observation matters most
- After one day ARIMA is back to the naive level (0.24 against 0.25 GW): its correction fades out
- Chronos-2 keeps an advantage over the whole horizon (0.14 GW at 48 hours against 0.20)
- ETS is good for a few hours and then the worst: a daily season cannot represent the weekend

How much context is needed? (1/2)

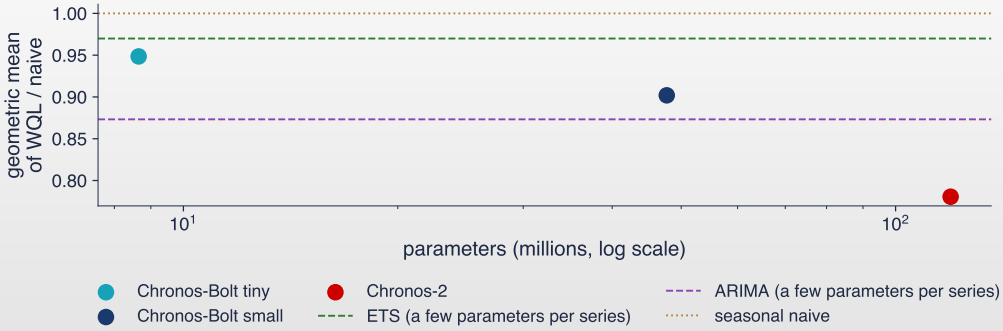


How much context is needed? (2/2)

- MASE on Romanian load for contexts of 96 to 2048 hours; 26 weekly origins (the last half year); the dashed line is the seasonal naive
- With 96 hours (4 days) every model is worse than the seasonal naive: Chronos-Bolt small 1.33, Chronos-2 2.04, naive 1.16
- From 192 hours (more than one week) on, the weekly cycle is in the context and the error drops
- Chronos-2 keeps improving up to 1344 hours (0.63); Chronos-Bolt flattens at 672 hours (0.78)
- Rule: give the model several full seasonal cycles; a zero-shot model only repeats what it sees

TSA_ch11_context_contamination

Size and accuracy



⊡ Geometric mean over the seven series of the WQL ratio to the seasonal naive, against the number of parameters

TSA_ch11_context_contamination

Interpreting size and cost

- Parameters: Bolt tiny 9 million, Bolt small 48 million, Chronos-2 119 million; ETS and ARIMA: a few per series
- The larger models score better here, but Chronos-2 also differs in architecture and training data: size alone is not the cause; ARIMA (0.87) sits between Bolt small and Chronos-2
- Time per load forecast on a laptop CPU: Bolt small 19 ms, Chronos-2 43 ms, ARIMA 114 ms, ETS 411 ms
- Zero-shot inference is cheap; the cost was paid once, in pretraining

Landmark benchmarks

- **M3** (Makridakis and Hibon (2000))
  - 3003 series
  - sophisticated methods did not beat simple ones on average
- **M4** (Makridakis et al. (2020))
  - 100 000 series
  - combinations and a hybrid of exponential smoothing and a neural network won
- **Monash archive** (Godahewa et al. (2021))
  - public data sets with fixed horizons and benchmark results
- **GIFT-Eval** (Aksu et al. (2024))
  - many domains and frequencies, MASE and CRPS, a public leaderboard of foundation models
  - separates the test data from a pretraining corpus, to limit contamination
- Critical studies: simple linear models beat early Transformer forecasters (Zeng et al. (2023)); LLM parts add no accuracy (Tan et al. (2024))

Advantages and limitations of foundation models

- **Help**
  - many series, little time per series; strong and regular seasonality; long contexts (load, sales, traffic)
  - a quick, strong benchmark before building a dedicated model; probabilistic output at no extra cost
- **Do not help**
  - series close to a random walk (exchange rates, returns): nothing to learn from the past
  - short series (a few dozen quarters) where a careful ARIMA is better
  - when drivers matter (temperature, prices, policy) and are not in the context
  - when the model must be explained parameter by parameter, or a causal question is asked
- Always compare with seasonal naive, ETS and ARIMA on the same origins; test differences with Diebold–Mariano (Diebold and Mariano (1995))

Recap: The evaluation

- Chronos-2 is best on average, especially for hourly load; ARIMA is close and wins on GDP
- On random-walk-like series no model beats the naive forecast
- Intervals of foundation models tend to be too narrow; context must cover the seasons

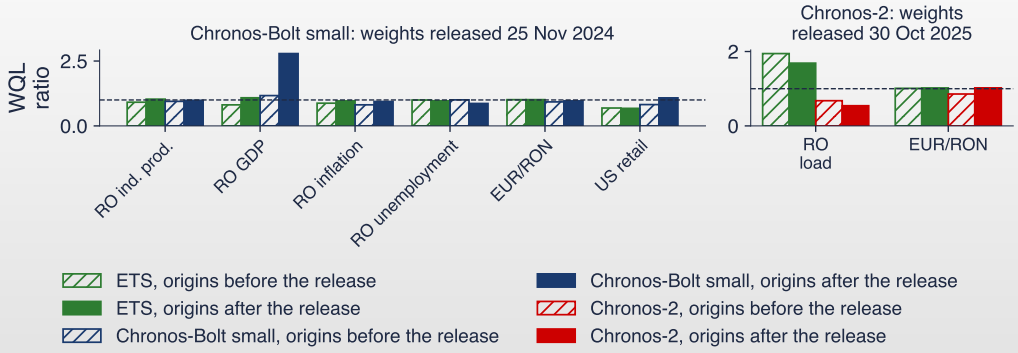
Leakage in your own pipeline

- **Leakage**
  - information from the test period reaches the forecasts (Hewamalage et al. (2023))
- Typical mistakes
  - scaling or standardising with the mean of the whole sample
  - choosing ARIMA orders, the context length or the model version after looking at test errors
  - using revised data (GDP) as if it had been known at the origin
  - random train/test splits of a time series (Chapter 9)
- In this chapter: each context ends at the origin; ARIMA orders were fixed before the first origin; Chronos scales by the context only
  - remaining issue: Eurostat GDP and production are the latest vintage, not real-time data

Contamination of benchmarks

- **Contamination**
  - the test series (or series highly correlated with it) were in the pretraining corpus
  - the score then measures memory, not forecasting: an optimistic bias
- Chronos reports two benchmarks: in-domain (data sets used in training) and zero-shot (data sets not used) (Ansari et al. (2024))
  - public series such as M4 or US macro data are likely in many corpora; FRED is public, so is Eurostat
- Remedies
  - test only on data published after the release date of the weights
  - live benchmarks: forecasts submitted before the outcome exists
  - report the training corpus and the release date of every model you use

A contamination check: before and after the release



- WQL ratio to the seasonal naive for origins before (hatched) and after (solid) the release of the weights; ETS on the same origins as control

TSA_ch11_context_contamination

Interpreting the contamination check

- Chronos-2 on EUR/RON: 0.86 before its release (124 origins), 1.02 after (10 origins): the advantage disappears
- Chronos-2 on load: 0.68 before, 0.54 after: the gain is real, also on data it could not have seen
- Chronos-Bolt small, US retail: 0.82 before, 1.08 after, while ETS is stable (0.69, 0.68)
- Caution: 3 to 35 origins after the release are few; the after period also differs (2025–2026); a drop is a warning, not a proof
- Chronos-2 was released in October 2025: its monthly results above use only origins before that date

Recap: Leakage and contamination

- Leakage is under your control: use only data available at each origin
- Contamination is not: the pretraining corpus may contain your series
- Check results on data released after the model

Possible contribution of AI

- **Code**
  - a first draft of a rolling-origin backtest, of the pinball loss and the WQL, of a fan chart
- **Reading**
  - a summary of a model paper (architecture, corpus, licence, release date)
- **Exploration**
  - run several open models on many Romanian series from Eurostat or INS
- Example prompt
  - `Write Python code that downloads Romanian monthly industrial production (Eurostat sts_inpr_m, not seasonally adjusted), forecasts 12 months ahead from monthly origins since 2016 with amazon/chronos-bolt-small (package chronos-forecasting, CPU), ETS and the seasonal naive, and reports MASE, the weighted quantile loss and the coverage of the 80% interval.`

Checks you must run

- The package and the model names: `BaseChronosPipeline`, `amazon/chronos-bolt-small`; an invented class or model name fails at once
- The context ends at the origin: no scaling, ordering or tuning with test data
- The quantile levels: Chronos-Bolt gives only 0.1 to 0.9; a "1% quantile" from it is an extrapolation
- The benchmarks: seasonal naive, ETS and ARIMA on the same origins; scores relative to the naive forecast
- The release date of the weights against the test period; claims of "state of the art" need a source
- Every cited paper: it must exist; check the arXiv number or the DOI

Key takeaways

- A foundation model is pretrained once on many series and forecasts new series zero-shot, from the context only
- Series become Transformer inputs by scaling, quantisation (Chronos) or patching (TimesFM, Moirai, Chronos-Bolt)
- Outputs are quantiles: score them with the pinball loss, the CRPS or WQL, and check coverage
- On our seven series: large gains on hourly load, none on random-walk-like series, losses on short quarterly GDP; ARIMA remains a strong benchmark
- Contamination can inflate published scores: test on data released after the model

Key formulas

Quantity	Formula
Attention	$\mathrm{softmax}(QK^\top/\sqrt{d})\,V$
Mean scaling	$\tilde{x}_t = x_t/s, \quad s = \frac{1}{C}\sum_{t=1}^{C}\lvert x_t\rvert$
Pinball loss	$L_\tau(y,\hat{q}_\tau) = \max\{\tau(y-\hat{q}_\tau),\ (\tau-1)(y-\hat{q}_\tau)\}$
CRPS	$\int (F(z) - \mathbf{1}\{z \geq y\})^2\,dz = 2\int_0^1 L_\tau(y, F^{-1}(\tau))\,d\tau$
WQL	$\frac{1}{9}\sum_\tau 2\sum_t L_\tau(y_t,\hat{q}_{\tau,t}) \big/ \sum_t \lvert y_t\rvert$
MASE	$\frac{1}{h}\sum_j \lvert y_{T+j} - \hat{y}_{T+j}\rvert \Big/ \frac{1}{T-m}\sum_{t>m}\lvert y_t - y_{t-m}\rvert$
Coverage	$\frac{1}{n}\sum_i \mathbf{1}\{\hat{q}^{(i)}_{0.1} \leq y_i \leq \hat{q}^{(i)}_{0.9}\}$, target 0.8

Self-assessment (1/2)

- **Question**: what is the difference between zero-shot forecasting and fine-tuning?
  - **Answer**: zero-shot keeps the pretrained weights fixed and uses only the context; fine-tuning trains the weights further on the target data
- **Question**: a context is 10, 20, 30. What are s and the scaled values in Chronos?
  - **Answer**: $s = 20$; scaled values 0.5, 1, 1.5
- **Question**: $\hat{q}_{0.9} = 5$ and $y = 6$. What is the pinball loss?
  - **Answer**: $0.9 \cdot (6 - 5) = 0.9$
- **Question**: how many patches of 16 does a context of 512 values give?
  - **Answer**: 32

Self-assessment (2/2)

- **Question**: an 80% interval covers 95% of the outcomes. Is this good?
  - **Answer**: no: the intervals are too wide (not sharp); the CRPS penalises this
- **Question**: why can a foundation model not beat the random walk on EUR/RON?
  - **Answer**: the changes of the rate are close to unpredictable from the past; no model can extract information that the context does not contain
- **Question**: a model released in 2025 scores very well on test data from 2018–2023. What do you check?
  - **Answer**: whether those series were in its pretraining corpus; repeat the test on data published after the release
- **Idea for a project**: compare Chronos-Bolt, Chronos-2 and ARIMA on all Romanian monthly series of one Eurostat data set, before and after the release dates
- Next: Chapter 12, spectral analysis

References (1/3)

- Aksu, T., Woo, G., Liu, J., Liu, X., et al. (2024). GIFT-Eval: A benchmark for general time series forecasting model evaluation. arXiv:2410.10393
- Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., et al. (2024). Chronos: Learning the language of time series. *Transactions on Machine Learning Research*. arXiv:2403.07815
- Ansari, A. F., Shchur, O., Küken, J., Auer, A., et al. (2025). Chronos-2: From univariate to universal forecasting. arXiv:2510.15821
- Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., et al. (2021). On the opportunities and risks of foundation models. arXiv:2108.07258
- Das, A., Kong, W., Sen, R., & Zhou, Y. (2024). A decoder-only foundation model for time-series forecasting. *Proceedings of the 41st International Conference on Machine Learning (ICML)*. arXiv:2310.10688
- Diebold, F. X., & Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3), 253–263. doi:10.1080/07350015.1995.10524599
- Garza, A., Challu, C., & Mergenthaler-Canseco, M. (2023). TimeGPT-1. arXiv:2310.03589
- Gneiting, T., & Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477), 359–378. doi:10.1198/016214506000001437

References (2/3)

- Godahewa, R., Bergmeir, C., Webb, G. I., Hyndman, R. J., & Montero-Manso, P. (2021). Monash time series forecasting archive. *NeurIPS Track on Datasets and Benchmarks*. arXiv:2105.06643
- Gruver, N., Finzi, M., Qiu, S., & Wilson, A. G. (2023). Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36. arXiv:2310.07820
- Hewamalage, H., Ackermann, K., & Bergmeir, C. (2023). Forecast evaluation for data scientists: Common pitfalls and best practices. *Data Mining and Knowledge Discovery*, 37(2), 788–832. doi:10.1007/s10618-022-00894-5
- Huang, C., & Petukhina, A. (2022). *Applied Time Series Analysis and Forecasting with Python*. Springer. doi:10.1007/978-3-031-13584-2
- Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice* (3rd ed.). OTexts. otexts.com/fpp3
- Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4), 679–688. doi:10.1016/j.ijforecast.2006.03.001
- Makridakis, S., & Hibon, M. (2000). The M3-Competition: Results, conclusions and implications. *International Journal of Forecasting*, 16(4), 451–476. doi:10.1016/S0169-2070(00)00057-1
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1), 54–74. doi:10.1016/j.ijforecast.2019.04.014

References (3/3)

- Matheson, J. E., & Winkler, R. L. (1976). Scoring rules for continuous probability distributions. *Management Science*, 22(10), 1087–1096. doi:10.1287/mnsc.22.10.1087
- Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with Transformers. *International Conference on Learning Representations (ICLR)*. arXiv:2211.14730
- Rasul, K., Ashok, A., Williams, A. R., Ghonia, H., et al. (2023). Lag-Llama: Towards foundation models for probabilistic time series forecasting. arXiv:2310.08278
- Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3), 1181–1191. doi:10.1016/j.ijforecast.2019.07.001
- Tan, M., Merrill, M. A., Gupta, V., Althoff, T., & Hartvigsen, T. (2024). Are language models actually useful for time series forecasting? *Advances in Neural Information Processing Systems*, 37. arXiv:2406.16964
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30. arXiv:1706.03762
- Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., & Sahoo, D. (2024). Unified training of universal time series forecasting Transformers. *Proceedings of the 41st International Conference on Machine Learning (ICML)*. arXiv:2402.02592
- Zeng, A., Chen, M., Zhang, L., & Xu, Q. (2023). Are Transformers effective for time series forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9), 11121–11128. doi:10.1609/aaai.v37i9.26317