Forecasting II

DS-6030 | Spring 2026

forecasting-2.pdf

Table of contents

1 Recap

Last time we covered the first half of the forecasting workflow:

> **Look at data → identify structure → build baselines** → choose model → evaluate with rolling CV → communicate uncertainty

Key ideas from last class:

- Forecasting is prediction under **extrapolation**: order matters, horizon h matters.
- Decomposition separates the predictable (trend, seasonality) from the remainder.
- Simple baselines (naive, seasonal naive, mean, trend) establish what "good" means.
- Feature engineering (lags, calendar indicators, Fourier terms) turns forecasting into a regression problem.

Today we'll cover the second half: honest evaluation, a survey of modeling approaches, and a deeper look at how uncertainty grows with horizon.

2 Time Series Cross-Validation

2.1 Why Not Regular CV?

In the past, we evaluated models using resampling (cross-validation, oob) with *random* splits of the data. **Why can't we do that here?**

Random splits destroy the temporal structure of the data. A random fold might put January 2017 in training and December 2016 in testing - which means we're using *future information* to predict the past. This is temporal leakage, and it produces optimistic performance estimates that don't reflect how the model will perform in deployment.

The fix: we need a CV strategy that **always predicts the future using only the past**. In other words, we need to setup exactly the situation we'll face when we deploy the model.

2.2 Rolling Origin Evaluation

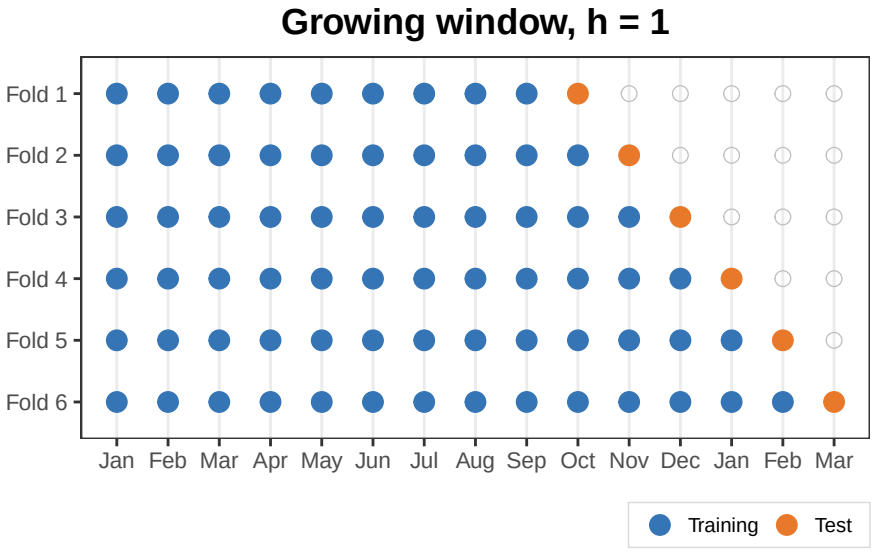
The idea is simple:

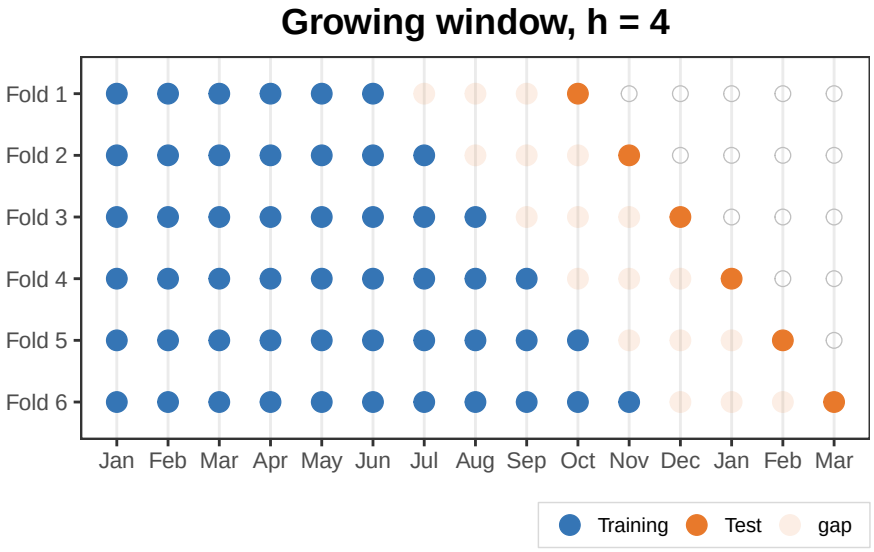
1. Train on data up to time t
2. Forecast h steps ahead
3. Record the error
4. Slide t forward
5. Repeat

This is sometimes called *time series cross-validation*, *rolling origin*, or *backtesting*.

2.2.1 Growing Window

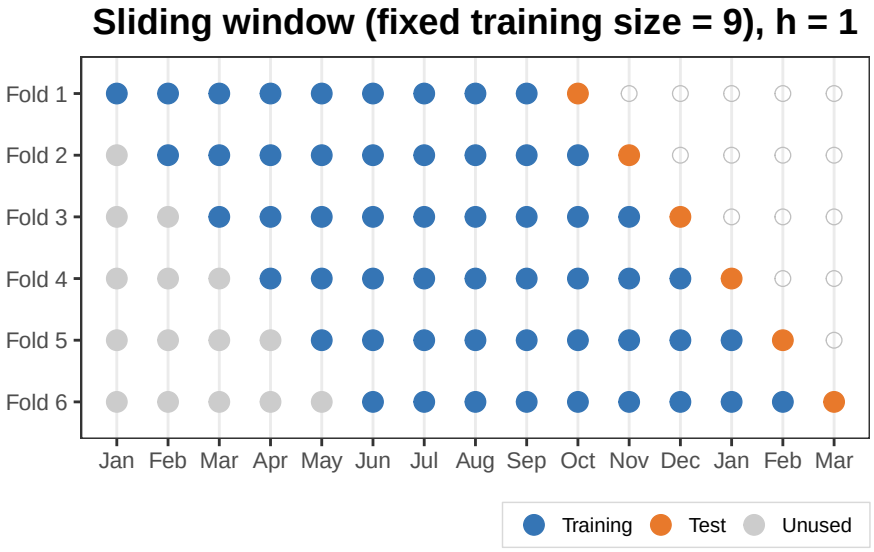
In the **growing** (or *stretching*) window approach, the training set gets larger as we slide forward. This is most useful when the series is short and there are no changepoints.

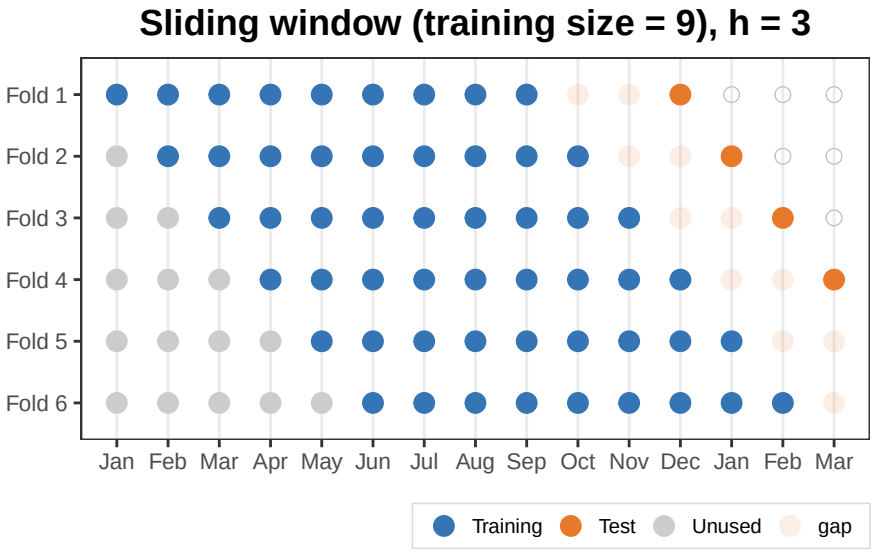




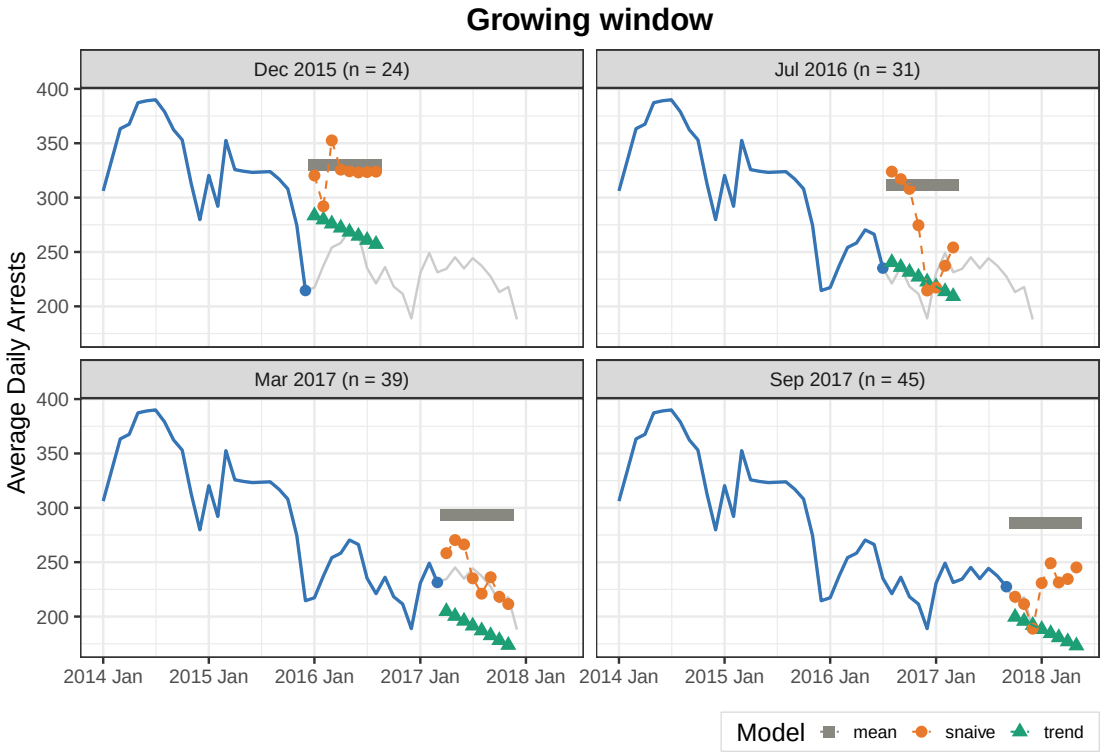
2.2.2 Sliding Window

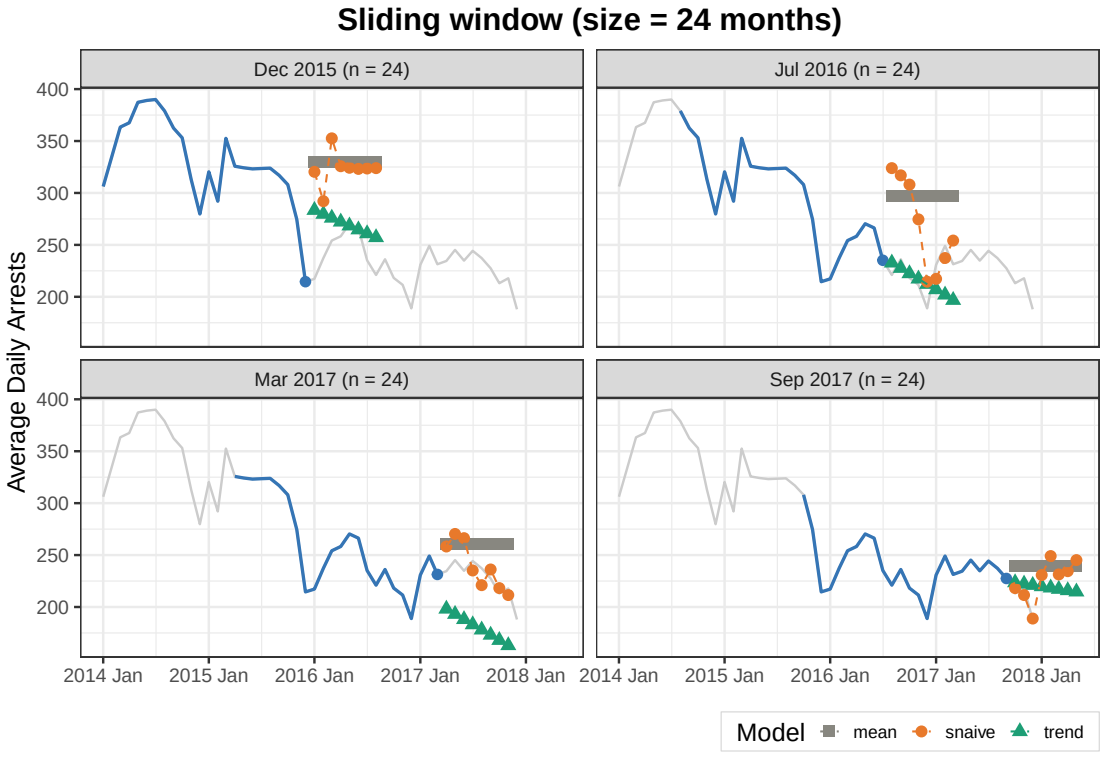
In the **growing** window approach, the training set keeps getting larger. But for long time series or series with **changepoints**, old data may hurt rather than help. The **sliding** window fixes the training size so that old observations drop off as new ones enter.


Sliding window (fixed training size = 9), h = 1
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar
Training
Test
Unused


Sliding window (training size = 9), h = 3
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar
Training
Test
Unused
gap

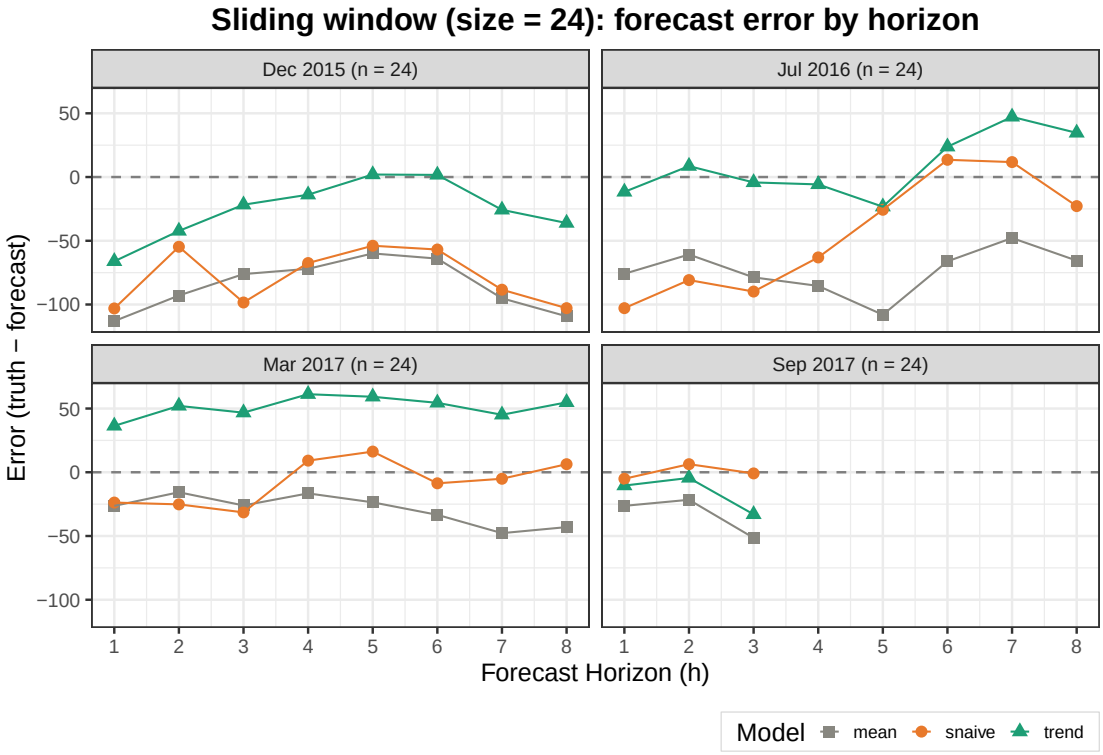
2.2.3 Chicago Arrests





2.3 Error as a Function of Horizon

This plot shows the prediction error as a function of forecast horizon h.



Your Turn #1 : Evaluating Over a Horizon

1. In Dec 2015 (top left) Why does the *trend* model have great prediction with $h = 5$?

2. Why does the seasonal naive (*snaive*) not work as well in 2015-2016?

3. Based on the 4 facets, what is the MAE at $h = 2$ for the *mean* model? What is it at $h = 8$?

4. Which model is better?

You may be tempted to aggregate errors across all h equally, but in practice you probably care most about a **specific** horizon.

- A retailer planning inventory might care about $h = 4$ weeks; a budget office might care about $h = 12$ months.

2.4 Evaluation Metrics

2.4.1 Setup: What Are We Averaging Over?

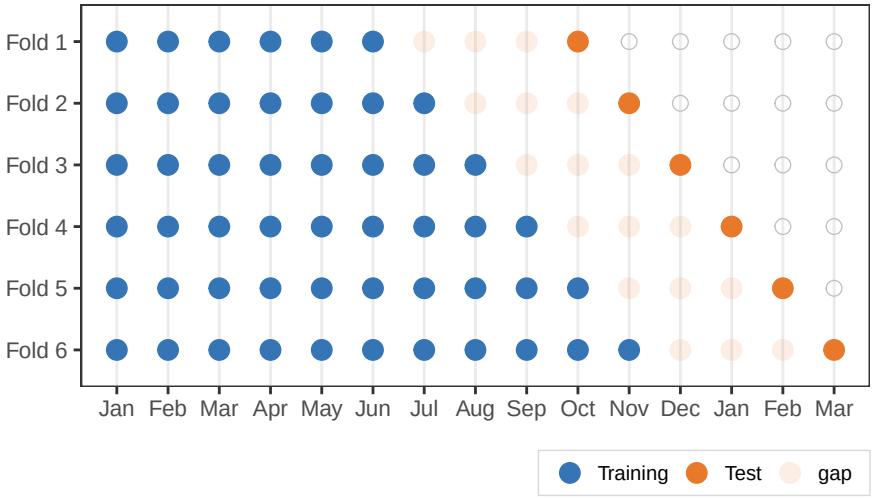
Suppose we use rolling origin CV with K folds. At each fold k, we train on data up to time t_k and forecast h steps ahead. This gives us K forecast errors:

$$e_k = y_{t_k+h} - \hat{y}_{t_k+h}$$

All semester we've been evaluating models using a **loss function** $L(y, \hat{y})$ that measures how bad a single prediction is. The metrics below are all just averages of a specific loss over the K CV folds:

$$\text{Metric} = \frac{1}{K}\sum_{k=1}^{K} L(y_{t_k+h},\ \hat{y}_{t_k+h})$$

The only thing that changes across metrics is the choice of L.



2.4.2 MAE (Mean Absolute Error)

$$L(y, \hat{y}) = |y - \hat{y}|$$

$$\text{MAE} = \frac{1}{K}\sum_{k=1}^{K} |e_k|$$

Familiar from regression. Easy to interpret: "on average, our forecast is off by X units." Same units as y.

2.4.3 RMSE (Root Mean Squared Error)

$$L(y, \hat{y}) = (y - \hat{y})^2$$

$$\text{RMSE} = \sqrt{\frac{1}{K}\sum_{k=1}^{K} e_k^2}$$

Also familiar. Same units as y (after taking the square root of the average loss), but penalizes large errors more heavily than MAE. Use RMSE when big misses are especially costly.

2.4.4 MAPE (Mean Absolute Percentage Error)

$$L(y, \hat{y}) = \left|\frac{y - \hat{y}}{y}\right|$$

$$\text{MAPE} = \frac{1}{K}\sum_{k=1}^{K}\left|\frac{e_k}{y_{t_k+h}}\right| \times 100$$

Scale-free ("on average, we're off by X%"), which sounds appealing. But two problems:

1. **Division by small values:** if y_{t_k+h} is near zero, the percentage blows up.
2. **Asymmetric penalties:** an over-forecast of 100 vs. actual of 50 gives $|-50/50| = 100\%$, but an under-forecast of 50 vs. actual of 100 gives $|50/100| = 50\%$. Same absolute error, different MAPE.

2.4.5 sMAPE (Symmetric Mean Absolute Percentage Error)

$$L(y, \hat{y}) = \frac{|y - \hat{y}|}{(|y| + |\hat{y}|)/2}$$

$$\text{sMAPE} = \frac{1}{K}\sum_{k=1}^{K}\frac{|e_k|}{(|y_{t_k+h}| + |\hat{y}_{t_k+h}|)/2} \times 100$$

Addresses the asymmetry in MAPE by averaging the actual and forecast in the denominator. Still has the near-zero problem, but the symmetry is an improvement.

2.4.6 MASE (Mean Absolute Scaled Error)

$$\text{MASE} = \frac{\frac{1}{K}\sum_{k=1}^{K}|e_k|}{\frac{1}{T-1}\sum_{t=2}^{T}|y_t - y_{t-1}|} = \frac{\text{MAE}}{\text{MAE}_{\text{naive}}}$$

MASE doesn't fit neatly into the $L(y, \hat{y})$ framework because it's a **ratio of two averages**: your model's MAE in the numerator, and the naive forecast's in-sample MAE in the denominator. The loss in the numerator is still $L(y, \hat{y}) = |y - \hat{y}|$, but the denominator provides the scaling.

- MASE < 1: your model beats naive
- MASE $= 1$: your model is no better than naive
- MASE > 1: naive is better — your model isn't earning its complexity

This is the metric recommended by Hyndman, and it's what the `accuracy()` function in `fpp3` reports. It directly encodes the baseline principle: **if your model can't beat naive, it's not useful.**

For seasonal data, the denominator can use the seasonal naive error instead:

$$\text{MASE}_{\text{seasonal}} = \frac{\frac{1}{K}\sum_{k=1}^{K}|e_k|}{\frac{1}{T-m}\sum_{t=m+1}^{T}|y_t - y_{t-m}|} = \frac{\text{MAE}}{\text{MAE}_{\text{snaive}}}$$

where m is the seasonal period (e.g., $m = 12$ for monthly data).

2.4.7 Which Metric to Use?

You should know by now that the answer is that the correct metric is the one that most closely aligns with subsequent decision-making.

Metric	Loss $L(y, \hat{y})$	Units	Scale-free?	Handles zeros?	Baseline built in?
MAE	$\lvert y - \hat{y}\rvert$	Same as y	No	Yes	No
RMSE	$(y - \hat{y})^2$	Same as y	No	Yes	No
MAPE	$\left\lvert\frac{y-\hat{y}}{y}\right\rvert$	Percent	Yes	No	No
sMAPE	$\frac{\lvert y-\hat{y}\rvert}{(\lvert y\rvert+\lvert\hat{y}\rvert)/2}$	Percent	Yes	No	No
MASE	$\lvert y - \hat{y}\rvert$ / naive MAE	Unitless	Yes	Yes	Yes

Every metric is an average (or ratio of averages) of a loss function. This is the same $\widehat{\text{EPE}} = \frac{1}{K}\sum L(y_k, \hat{y}_k)$ pattern used for all model evaluation this semester.

3 Models

3.1 Exponential Smoothing (ETS)

> Forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older. In other words, the more recent the observation the higher the associated weight. This framework generates reliable forecasts quickly and for a wide range of time series, which is a great advantage and of major importance to applications in industry.
>
> - Hyndman and Athanasopoulos https://otexts.com/fpp3/expsmooth.html

3.1.1 Simple Exponential Smoothing (SES)

The simplest version, **Simple Exponential Smoothing (SES)**, uses a single tuning parameter $\alpha \in [0, 1]$ to update a "level" l_t:

$$l_t = \alpha\, y_t + (1 - \alpha)\, l_{t-1}$$

The forecast for all future horizons is flat: $\hat{y}(t + h \mid t) = l_t$.

The key intuition: α controls how much you trust new data versus your previous estimate.

- α close to 1: the level chases the data closely (high variance, low bias)
- α close to 0: the level is very smooth and slow to react (low variance, high bias)

This is yet another bias-variance tradeoff.

3.1.2 In-Class Activity: Exponential Smoothing by Hand

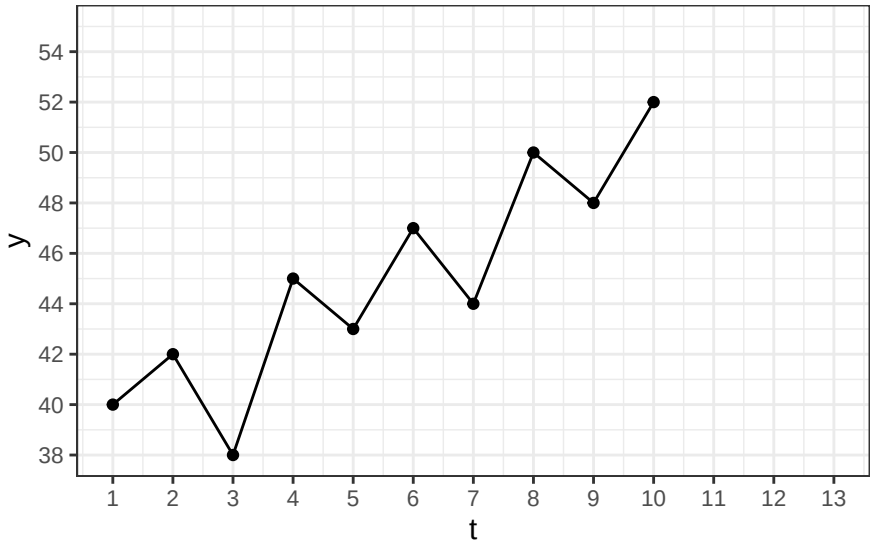
Smoothing by Hand

This activity connects exponential smoothing to Markov chains. Recall from last semester: in a Markov chain, the next state depends only on the current state. SES has the same property - the "state" l_t depends only on the current observation y_t and the previous state l_{t-1}:

$$l_t = \alpha\, y_t + (1 - \alpha)\, l_{t-1}$$

Instructions (teams of 2–3, ~15 minutes):

Here are 10 observed values from a time series. Compute the smoothed level l_t for two different values of α. Use $l_0 = y_1$ as the starting value.



t	y	level(a = 0.2)	level(a = 0.8)
1	40	40.0	40.0
2	42	40.4	41.6
3	38		
4	45		
5	43		
6	47		
7	44		
8	50		
9	48		
10	52		

Your Turn #2 : Smoothing Activity

1. Fill in the remaining rows for both values of α.
2. Plot both smoothed series (and the original data). Which α tracks the data more closely? Which is smoother?

3. Based on l_{10}, what is your forecast for $t = 11$? For $t = 13$? What do you notice?
4. How confident are you in that forecast at $h = 1$ versus $h = 5$?

3.1.3 Double Exponential (DES) / Holt

Holt's linear trend method (or double exponential smoothing) allows linear trends.

$$\text{Forecast Equation: } \hat{y}(t+h \mid t) = l_t + hb_t$$

for all $h > 0$. In this equation, l_t is the "level" and b_t is the trend.

The level is estimated by exponential smoothing using model parameter α.

$$\begin{aligned} l_t &= \alpha y_t + (1-\alpha)[l_{t-1} + b_{t-1}] \\ &= \alpha y_t + (1-\alpha)\hat{y}_{t|t-1} \end{aligned}$$

The slope is estimated by exponential smoothing using model parameter β.

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$$

3.1.4 Triple Exponential Smoothing / Holt-Winters

Holt-Winters method allows both a trend and seasonality.

$$\text{Forecast Equation: } \hat{y}(t+h \mid t) = l_t + hb_t + s_t(h)$$

for all $h > 0$. In this equation, l_t is the "level" and b_t is the trend, and $s_t(h)$ is the seasonality at forecast horizon h. If m is the period (e.g., $m = 12$ for monthly data, $m = 4$ for quarterly data), then

$$s_t(h) = \begin{cases} s_{t+h-m} & h \le m \\ s_{t+h-2m} & m < h \le 2m \\ s_{t+h-3m} & 2m < h \le 3m \\ \dots & \dots \end{cases}$$

which ensures that forecasts only use the data at the time the forecast is made.

The level equation averages the *deseasoned* level with the previous level and trend.

$$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)[l_{t-1} + b_{t-1}]$$

The slope is estimated by exponential smoothing using model parameter β (same as was done in DES).

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$$

The seasonality term uses parameter γ to mix the *detrended* residuals with the previous seasonality at lag m.

$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}$$

3.1.5 Multiplicative Seasonality

Instead of the seasonality component entering the model additively, it can be a multiplicative factor.

The Holt-Winters multiplicative model using this form

$$\text{Forecast Equation: } \hat{y}(t+h \mid t) = (l_t + hb_t)s_t(h)$$

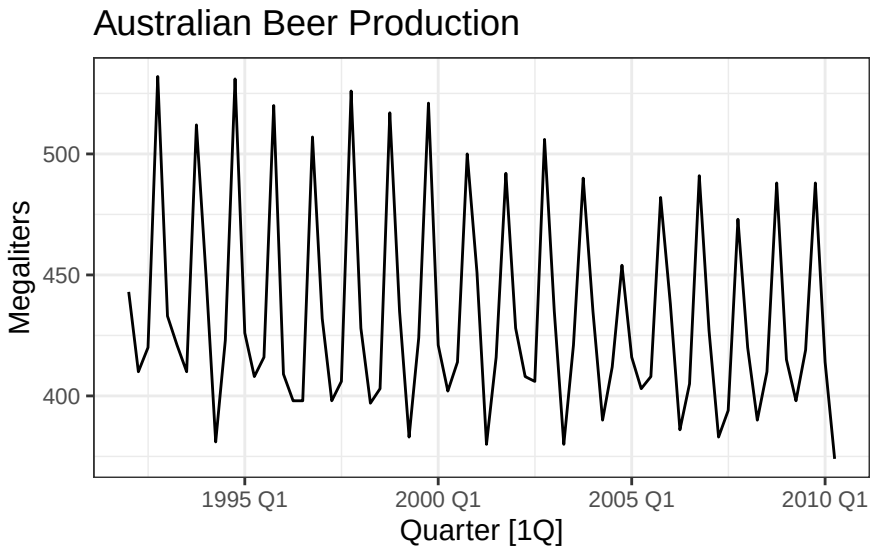
for all $h > 0$. In this equation, l_t is the "level" and b_t is the trend, and $s_t(h)$ is the seasonality at forecast horizon h.

The level, trend, and seasonality updates now account for the multiplicative term:

$$\begin{aligned} l_t &= \alpha \frac{y_t}{s_{t-m}} + (1-\alpha)(l_{t-1} + b_{t-1}) \\ b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \\ s_t &= \gamma \left(\frac{y_t}{l_{t-1} + b_{t-1}} \right) + (1-\gamma)s_{t-m} \end{aligned}$$

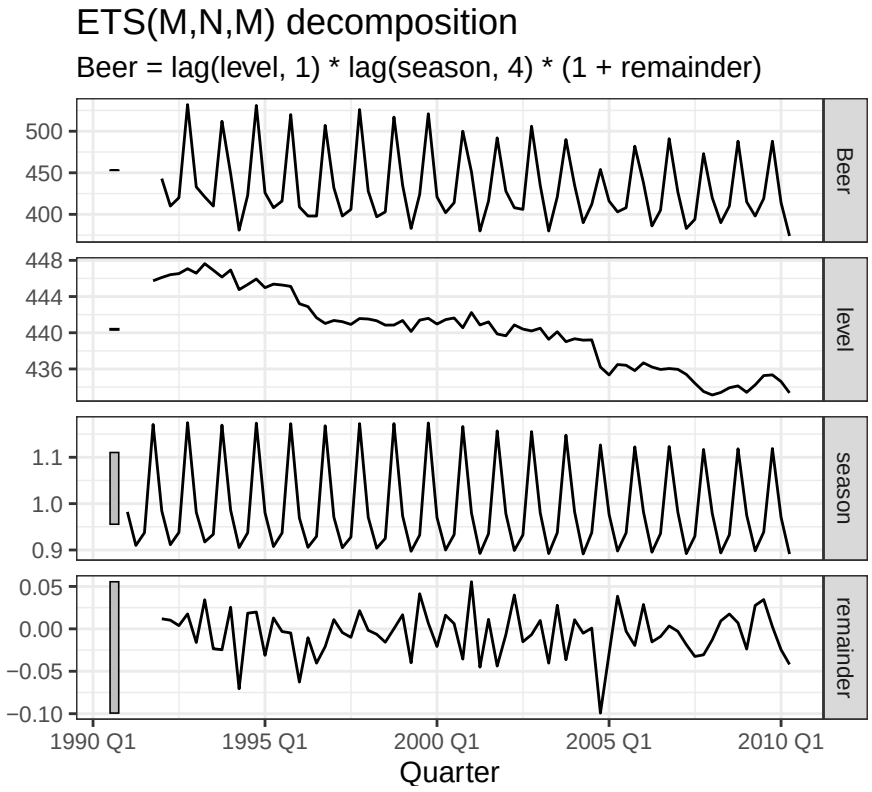
3.1.6 ETS in Practice

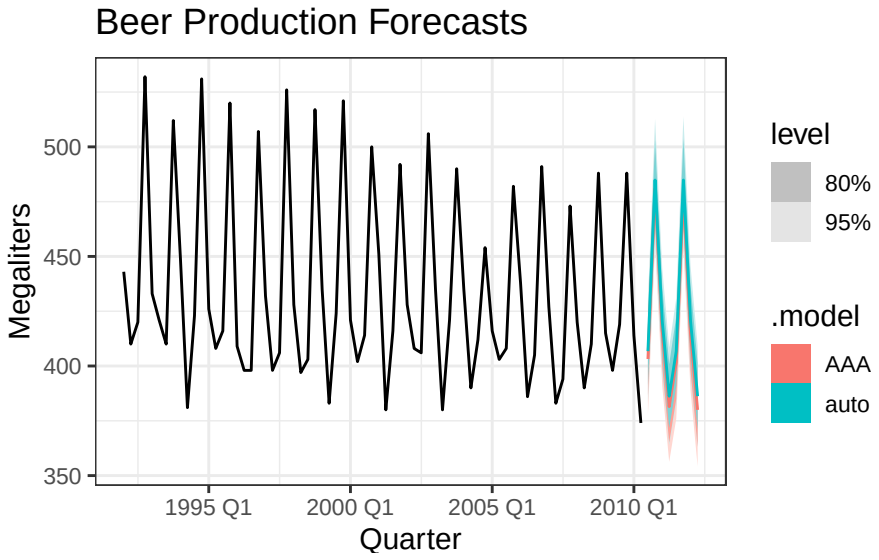
The ETS framework has three components - **E**rror, **T**rend, **S**easonality - each of which can take several forms. Instead of specifying these manually, we can let the software choose. This is just model tuning! Instead of the slower time series cross-validation, mathematical adjustments to the training loss (like AIC and BIC) are popular and very fast.




```
#> Series: Beer
#> Model: ETS(M,N,M)
#>   Smoothing parameters:
#>     alpha = 0.06872
#>     gamma = 0.1848
#>
#>   Initial states:
#>   l[0]  s[0]  s[-1]  s[-2]  s[-3]
#>  445.7 1.171 0.9373 0.9098 0.9822
```

```
#>
#>   sigma^2:  9e-04
#>
#>    AIC  AICc   BIC
#> 702.2 703.9 718.3
```





The details of how ETS estimates its components (the update equations, the taxonomy of 18 model variants, and the connection to state-space models) are covered in a bit more detail in the Appendix. For our purposes, the key ideas are:

1. ETS produces weighted averages of past data with exponentially decaying weights
2. Components (level, trend, seasonality) can each be additive, multiplicative, or absent
3. Automated selection (via AICc) works well in practice

3.2 ARIMA (Conceptual)

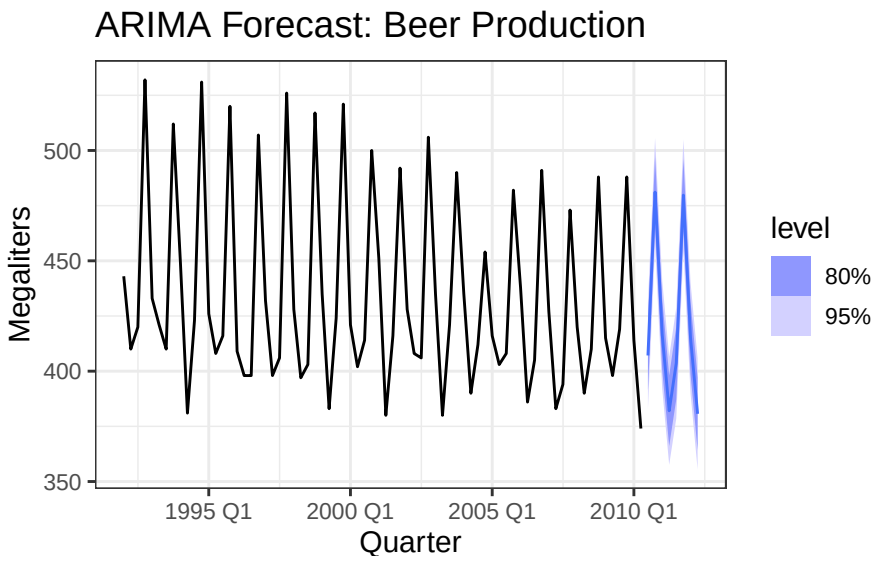
ARIMA models capture patterns through **lags** and **differencing** rather than through explicit trend/seasonal components.

The core ideas:

- **AR (Autoregressive):** use past values $y_{t-1}, y_{t-2}, \ldots$ as predictors. This is exactly the lag feature approach from Lecture 1, but with constraints that ensure stability.
- **I (Integrated / Differencing):** instead of modeling y_t directly, model the *changes* $y_t - y_{t-1}$. This removes trends without having to specify the trend's shape. We saw this idea briefly in Lecture 1 with the first-differenced arrest data.
- **MA (Moving Average):** model the forecast errors themselves as having temporal structure.

In practice, automated tools select the right ARIMA specification:

```
#> Series: Beer
#> Model: ARIMA(0,0,1)(0,1,1)[4] w/ drift
#>
#> Coefficients:
#>           ma1     sma1  constant
#>       -0.2071  -0.7612   -1.3801
#> s.e.   0.1174   0.0878    0.3404
#>
#> sigma^2 estimated as 149.3:  log likelihood=-274.8
#> AIC=557.5   AICc=558.1   BIC=566.5
```



We won't go deeper into ARIMA mechanics; a key point is that it's a complementary approach to ETS. Where ETS decomposes the series into components, ARIMA works with differences and lagged values. Automated tools make both easy to apply, and in practice they often produce similar forecasts.

3.3 Regression / ML-Based Approaches

This is what we covered in forecasting-1: construct time features (lags, calendar indicators, Fourier terms) and feed them to an ML or Statistical Model.

This is the **most flexible** approach and the practical default for most applied data science work:

- Handles multiple exogenous predictors naturally
- Scales to many series
- You can use any model: elastic net, boosted trees, random forests
- Prophet fits here too - it's a regression model with trend changepoints and Fourier seasonality, plus ridge penalties

The tradeoff: it requires **more data** and **more feature engineering** than ETS or ARIMA.

3.3.1 Facebook Prophet

Facebook's Prophet is a regression-based forecasting model that fits naturally into the feature engineering framework. It specifies an additive decomposition:

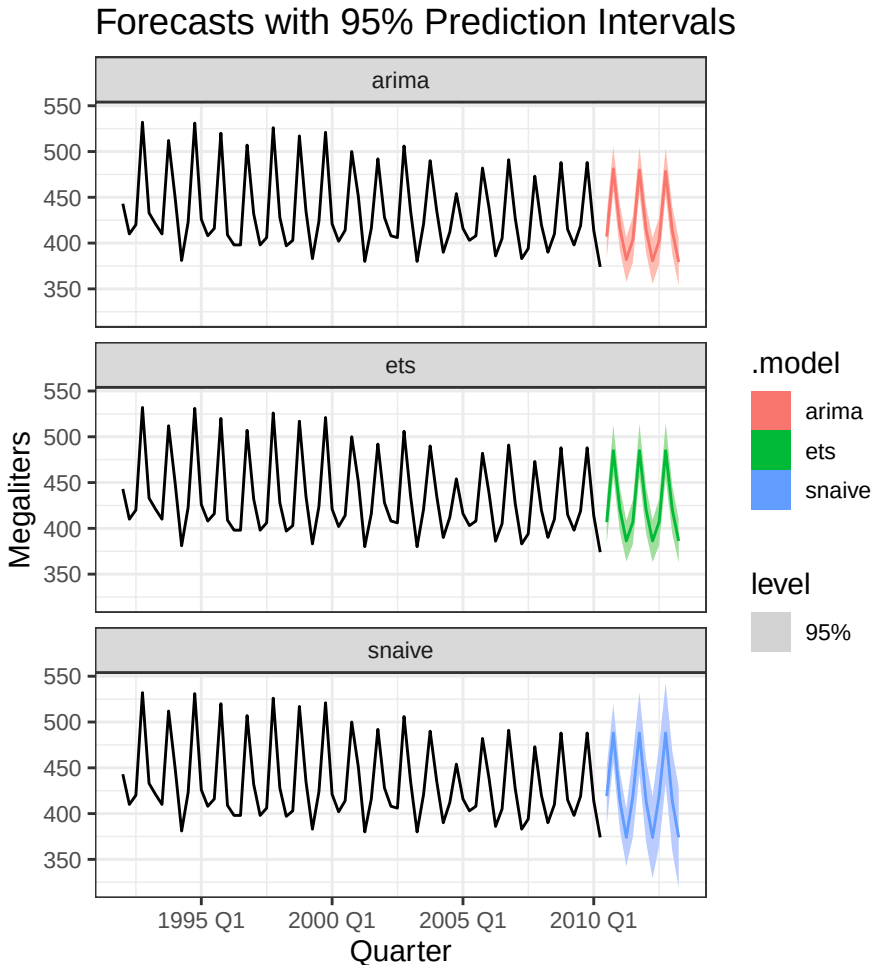
$$\hat{y}_t = \hat{T}_t + \hat{S}_t + \hat{H}_t$$

where T is trend, S is seasonality, and H is for holidays or special times. Under the hood, this is using a Generative Additive Model (GAM) framework to fit the model to the data.

A few more details in the Appendix.

4 Prediction Intervals: Uncertainty Grows with Horizon

We saw earlier that forecasts have prediction intervals that widen with h. Now let's see this with a real model.



Notice how the intervals **fan out** as the forecast horizon increases. This is a fundamental feature of forecasting: the further ahead you predict, the less certain you are.

For *multi-step forecasts*, models often feed predictions back into themselves - so errors compound as h increases. This is one reason intervals widen even beyond what you'd expect from static uncertainty.

4.0.1 Intervals as a Decision-Making Tool

A forecast of "130 megaliters next quarter" means something very different with an interval of $[125, 135]$ versus $[100, 160]$.

If your prediction intervals are too wide to be actionable at the horizon your stakeholder cares about, **that's important information.** Saying "we can't forecast this reliably beyond 2 quarters" is more valuable than producing a meaningless point forecast at $h = 12$.

5 Common Pitfalls

A few things that frequently go wrong in forecasting:

1. Using random CV instead of rolling origin. This is the most common beginner mistake. Random splits create temporal leakage and produce overoptimistic error estimates.

2. Extrapolating trends that aren't stable. A linear trend fit to recent data will project indefinitely. If the trend is driven by a temporary cause, the forecast will be wildly wrong.

3. Overfitting seasonality on short series. If you only have 2 years of monthly data, your seasonal estimates are based on just 2 observations per month. Adding Fourier terms or month indicators with this little data is asking for trouble.

4. Ignoring changepoints. If the data-generating process changed (new policy, economic shock, pandemic), data from before the change may hurt your model. This is why sliding windows can outperform growing windows.

5. Reporting point forecasts without intervals. A forecast without uncertainty is incomplete. Stakeholders need to know how much to trust the prediction.

These are all variations of themes from earlier in the course: overfitting, distribution shift, and the importance of honest evaluation.

6 Summary

We've now covered the full forecasting workflow:

> **Look at data → identify structure → build baselines → choose model → evaluate with rolling CV → communicate uncertainty**

Key takeaways from today:

- **Time series CV** (rolling origin) is the honest way to evaluate forecasts. Random CV creates temporal leakage.
- **Error grows with horizon** h: decide whhich horizon matters for your problem.
- **Three model families:** ETS (weighted averages), ARIMA (lags + differencing), regression/ML (feature engineering).
- **Prediction intervals widen** as h increases. Intervals are as important as point forecasts.
- **Always compare against baselines.** If your model can't regularly beat seasonal naive, it's not earning its complexity.

7 Appendix A: ETS Taxonomy and Update Equations

7.1 Simple Exponential Smoothing (SES)

The SES model produces a **flat** forecast using one parameter α:

$$\text{Forecast: } \hat{y}(t+h \mid t) = l_t \tag{1}$$

$$\text{Level: } l_t = \alpha y_t + (1-\alpha) l_{t-1} \tag{2}$$

There are several equivalent ways to write the level update:

As an innovation correction:

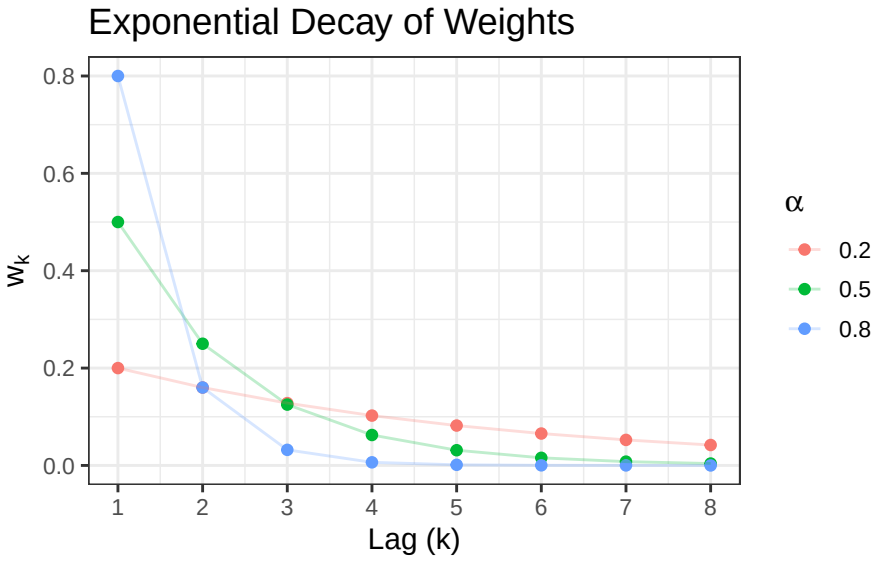
$$l_t = l_{t-1} + \alpha(y_t - \hat{y}_{t|t-1})$$

The level adjusts by a fraction α of the forecast error. This connects to the "learning from mistakes" intuition.

As a weighted average of all past data:

$$l_t = \alpha \sum_{k=0}^{t-1} (1-\alpha)^k y_{t-k} + (1-\alpha)^t l_0$$

where $w_k = \alpha(1-\alpha)^k$ are the weights. These decay geometrically:



7.2 Holt's Linear Trend (Double Exponential Smoothing)

Adds a trend component b_t with its own smoothing parameter β:

$$\text{Forecast: } \hat{y}(t+h \mid t) = l_t + h\, b_t \tag{3}$$

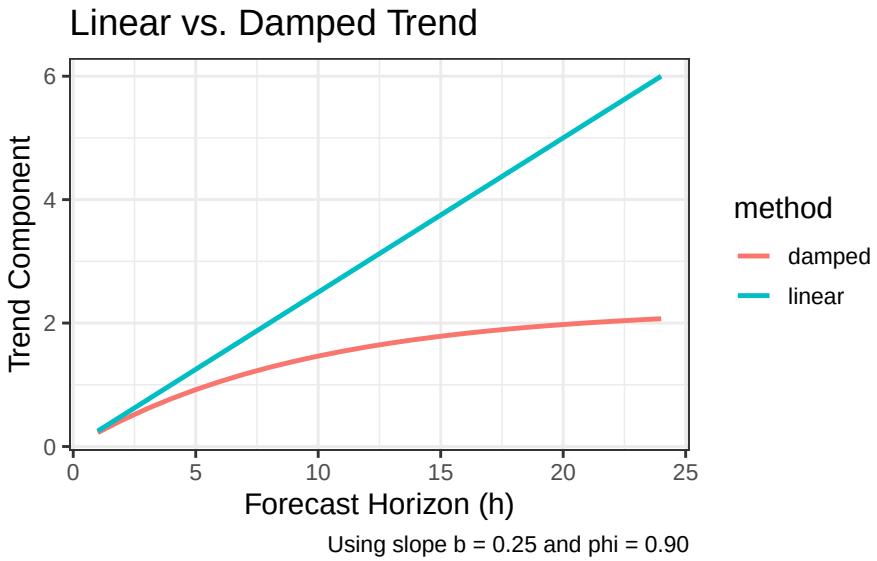
$$\text{Level: } l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{4}$$

$$\text{Trend: } b_t = \beta(l_t - l_{t-1}) + (1-\beta) b_{t-1} \tag{5}$$

The trend b_t is a smoothed estimate of the slope. The forecast is now linear in h rather than flat.

7.2.1 Damped Trend

For longer horizons, linear trends can produce unreasonable forecasts. The **damped** method replaces hb_t with $(\phi + \phi^2 + \ldots + \phi^h)b_t$ where $\phi \leq 1$:



7.3 Holt-Winters (Triple Exponential Smoothing)

Adds a seasonal component s_t with smoothing parameter γ and seasonal period m:

Additive seasonality:

$$\text{Forecast: } \hat{y}(t+h \mid t) = l_t + h\, b_t + s_{t+h-m} \tag{6}$$

$$\text{Level: } l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{7}$$

$$\text{Trend: } b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{8}$$

$$\text{Season: } s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \tag{9}$$

Multiplicative seasonality:

$$\text{Forecast: } \hat{y}(t+h \mid t) = (l_t + h\, b_t) \cdot s_{t+h-m} \tag{10}$$

$$\text{Level: } l_t = \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{11}$$

$$\text{Trend: } b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{12}$$

$$\text{Season: } s_t = \gamma\frac{y_t}{l_{t-1} + b_{t-1}} + (1-\gamma)s_{t-m} \tag{13}$$

Use multiplicative seasonality when the seasonal swings grow proportionally with the level.

7.4 The Full ETS Taxonomy

The ETS framework is named for its three components, each of which can take different forms:

- **E**rror: Additive (A) or Multiplicative (M)
- **T**rend: None (N), Additive (A), or Additive Damped (Ad)
- **S**eason: None (N), Additive (A), or Multiplicative (M)

This gives $2 \times 3 \times 3 = 18$ possible model specifications. The model is identified by a three-letter code, e.g., ETS(A,Ad,M) has additive errors, damped additive trend, and multiplicative seasonality.

Trend	**Seasonal**		
	N	**A**	**M**
N	$\hat{y}_{t+h\|t} = \ell_t$ $\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}$	$\hat{y}_{t+h\|t} = \ell_t + s_{t+h-m(k+1)}$ $\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)\ell_{t-1}$ $s_t = \gamma(y_t - \ell_{t-1}) + (1-\gamma)s_{t-m}$	$\hat{y}_{t+h\|t} = \ell_t s_{t+h-m(k+1)}$ $\ell_t = \alpha(y_t/s_{t-m}) + (1-\alpha)\ell_{t-1}$ $s_t = \gamma(y_t/\ell_{t-1}) + (1-\gamma)s_{t-m}$
A	$\hat{y}_{t+h\|t} = \ell_t + hb_t$ $\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$	$\hat{y}_{t+h\|t} = \ell_t + hb_t + s_{t+h-m(k+1)}$ $\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$ $s_t = \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}$	$\hat{y}_{t+h\|t} = (\ell_t + hb_t)s_{t+h-m(k+1)}$ $\ell_t = \alpha(y_t/s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$ $s_t = \gamma(y_t/(\ell_{t-1} + b_{t-1})) + (1-\gamma)s_{t-m}$
$\mathbf{A_d}$	$\hat{y}_{t+h\|t} = \ell_t + \phi_h b_t$ $\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + \phi b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)\phi b_{t-1}$	$\hat{y}_{t+h\|t} = \ell_t + \phi_h b_t + s_{t+h-m(k+1)}$ $\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + \phi b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)\phi b_{t-1}$ $s_t = \gamma(y_t - \ell_{t-1} - \phi b_{t-1}) + (1-\gamma)s_{t-m}$	$\hat{y}_{t+h\|t} = (\ell_t + \phi_h b_t)s_{t+h-m(k+1)}$ $\ell_t = \alpha(y_t/s_{t-m}) + (1-\alpha)(\ell_{t-1} + \phi b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)\phi b_{t-1}$ $s_t = \gamma(y_t/(\ell_{t-1} + \phi b_{t-1})) + (1-\gamma)s_{t-m}$

```
#> # A tibble: 9 x 4
#>   .model   AIC  AICc   BIC
#>   <chr>  <dbl> <dbl> <dbl>
#> 1 AAM     704.  707.  725.
#> 2 ANM     705.  707.  721.
#> 3 ANA     706.  707.  722.
#> 4 AAdM    704.  708.  727.
#> 5 AAA     706.  709.  727.
#> 6 AAdA    707.  711.  730.
#> # i 3 more rows
```

In practice, `ETS(Beer)` with no specification lets the software search over all 18 variants and select the best by AICc.

8 Appendix B: Prophet

Facebook's Prophet is a regression-based forecasting model that fits naturally into the feature engineering framework. It specifies an additive decomposition:

$$\hat{y}_t = \hat{T}_t + \hat{S}_t + \hat{H}_t$$

where T is trend, S is seasonality, and H is for holidays or special times.

Trend: Piecewise linear (or logistic) with automatically detected changepoints. This is equivalent to the ReLU basis functions and lasso approach we discussed: place many potential changepoints and use an L1 penalty to select only the important ones.

Seasonality: Fourier basis functions (as covered in the forecasting-1.pdf Appendix) with a ridge penalty on the coefficients. Multiple seasonal periods are supported (e.g., weekly + yearly for daily data).

Holidays: Indicator variables for special days, also with ridge penalties.

Prophet is essentially a penalized regression model with clever feature engineering built in. It works well for business time series with strong seasonal effects, multiple seasonalities, and known special events.

9 Appendix C: Forecasting Contest Winners

Here are some descriptions of winning submissions to forecasting contests.

Note: This material was generated with ChatGPT; there may be errors and hallucinations.

If you only read a few things, start with the M4 and M5 papers plus Hyndman's short history of forecasting competitions.

9.1 General overview

1. A brief history of forecasting competitions Rob J. Hyndman Why read it: a short and very readable overview of what forecasting competitions are, why they matter, and what the field has learned from them. Link: A brief history of forecasting competitions

9.2 Classical forecasting competitions

2. M4 Competition: large-scale benchmark across many series and methods Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos Why read it: the main results paper for one of the most influential forecasting competitions. Good if you want the big picture. Link: The M4 Competition: 100,000 time series and 61 forecasting methods
3. M4 winning solution: ES-RNN Slawek Smyl Why read it: the winning hybrid method that combined exponential smoothing ideas with a recurrent neural network. Good lesson: strong forecasting solutions often combine classical structure with machine learning rather than replacing one with the other. Link: M4 Forecasting Competition: Introducing a New Hybrid ES-RNN Model
4. M5 Accuracy: Walmart retail forecasting Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos Why read it: the best competition to read if you want examples of hierarchical forecasting, feature engineering, and large-scale ML methods for many related time series. Link: M5 accuracy competition: Results, findings, and conclusions
5. M5 Accuracy winning write-up Yeonjun In and STU Why read it: practical details on how the winning solution used multiple direct and recursive LightGBM models and blended them. Link: 1st place solution
6. M5 Uncertainty: quantiles and prediction intervals Spyros Makridakis et al. Why read it: especially relevant if you care about probabilistic forecasting and interval forecasts rather than only point predictions. Link: The M5 uncertainty competition: Results, findings, and conclusions
7. M5 Uncertainty winning solution A. David Lainder and Russell D. Wolfinger Why read it: a clear paper on gradient boosted trees for probabilistic forecasting, with a useful emphasis on augmentation, tuning, and cross-validation. Link: Forecasting with gradient boosted trees: augmentation, tuning, and cross-validation strategies
8. Tourism forecasting competition George Athanasopoulos et al. Why read it: a strong classical forecasting paper using real tourism demand series across multiple frequencies. Very good if you want a less "Kaggle" and more traditional applied forecasting example. Link: The tourism forecasting competition

9. M6 Competition Spyros Makridakis et al. Why read it: optional, but useful if you want to see how forecasting was evaluated in a financial setting where investment decisions matter too. Link: The M6 forecasting competition: Bridging the gap between forecasting and investment decisions

9.3 Finance-flavored Kaggle competitions

10. Jane Street Market Prediction Why read it: not a classical forecasting contest, but a very interesting example of high-dimensional financial prediction. Competition link: Jane Street Market Prediction
11. Jane Street Market Prediction, 1st place Yirun Zhang Why read it: a strong example of a supervised autoencoder plus MLP approach. Link: Yirun's 1st place solution
12. Jane Street Market Prediction, 3rd place Martin B.B. Why read it: a striking example of deep MLP ensembles succeeding with relatively limited hand-crafted feature work. Link: 3rd place solution: Ensembles of deep (49 layer) MLPs
13. Jane Street Real-Time Market Data Forecasting Why read it: another finance-flavored competition, but this one adds real-time inference constraints and deployment concerns. Competition link: Jane Street Real-Time Market Data Forecasting
14. Jane Street Real-Time Market Data Forecasting winner walkthrough Team Patrick Yam Why read it: a good high-level walkthrough of how the winning team handled the real-time setting and latency constraints. Link: Kaggle Winners Walkthroughs: Jane Street Real-Time Market Data Forecasting

9.4 Suggested reading path for this course

1. Hyndman history paper
2. M4 results paper
3. M5 Accuracy results paper
4. M5 Accuracy winning write-up
5. M5 Uncertainty results paper
6. M5 Uncertainty winning solution paper

That sequence will give you a good mix of 1. broad perspective, 2. classical forecasting benchmarks, 3. modern feature-based / ML forecasting, and 4. probabilistic forecasting.