Machine Learning

Jakub Adamczyk, Faculty of Computer Science, AGH

Time series forecasting

Part 2: statistical models

Agenda

- we will talk about two **most commonly used** model families:

    - ARIMA (autoregressive)

    - ETS (exponential smoothing)

- lastly, we will cover forecasting based on **regression (ML)**

- other methods are more niche, but very interesting, we'll mention them at the end

ARIMA

ARIMA - introduction

- **family of models:** AR, MA, ARMA, ARIMA, SARIMA, SARIMAX, …

- **idea:**
  - "history repeats itself"
  - recent values directly influence future values
  - model **autocorrelation** in the data
  - **short memory** models - based on recent values

- building blocks:
  - ARIMA = I(d) + ARMA(p,q) = I(d) + AR(p) + MA(q)
  - SARIMA = seasonality + ARIMA
  - SARIMAX = SARIMA + exogenous variables X
  - ARMA, ARMAX, SMAX etc.

Differencing - I(d)

- AR, MA, ARMA models **assume (require)** stationary time series

- first step: differencing of order *d*

- **automated algorithm:**

  - loop, KPSS test, take difference

  - typically $d_{max} = 2$

- you definitely should **manually verify** if possible:

  - trend still present - change detrending method and/or algorithm

  - outliers - various clipping / thresholding / anomaly removal methods

  - seasonality - use SARIMA

Autoregressive AR(p) models

- **idea:** next value depends linearly on the last p values

- **linear regression** using last p values, known as **lagged features:**

$$\hat{y}_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \epsilon_t$$

- **training method:** OLS, computed using SVD

- very fast, unique solution, numerically stable

- beginning values: either cut out, or assume $y_0 = 0, y_{-1} = 0, \dots$

- **why such simple model?**

  ○ closed likelihood formula, MLE estimation

  ○ can combine with MA(q) to create ARMA and ARIMA models

  ○ probabilistic forecasts, confidence intervals

  ○ allows very fast hyperparameter tuning via information criteria
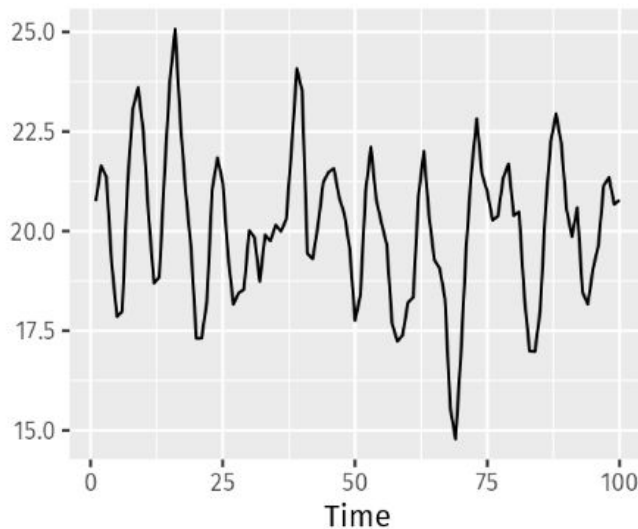
Autoregressive AR(p) processes

$$y_t = 18 - 0.8y_{t-1} + \epsilon_t$$
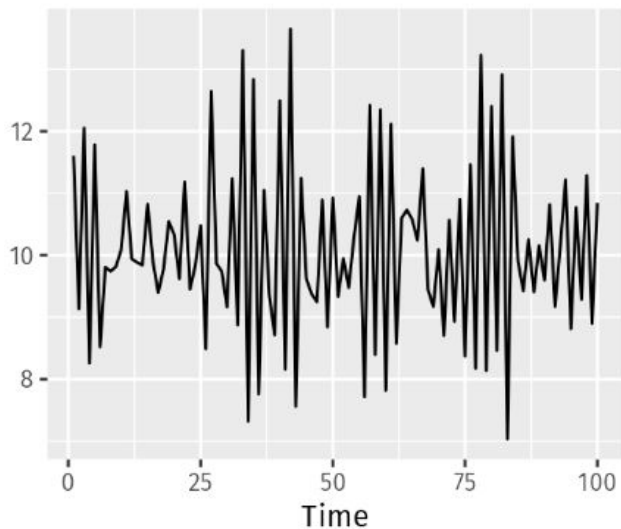
$$y_t = 8 + 1.3y_{t-1} - 0.7y_{t-2} + \epsilon_t$$

Moving average MA(q) models

- **be careful:** nothing to do with moving average, nor with exponentially weighted moving average (EWMA) model

- **idea:** next value depends on the last q errors

$$\hat{y}_t = c + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q} + \epsilon_t$$
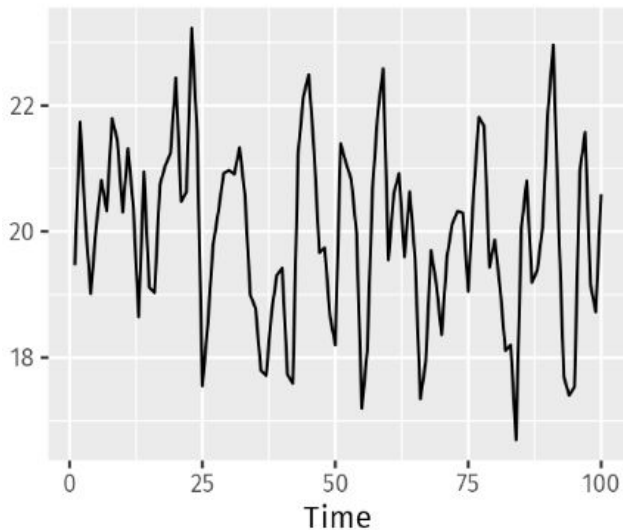
- allows for **correction** of the last errors, e.g. when trend direction changes

- assumes that last q errors are autocorrelated

- **problem:** those errors are not directly observable, so this is not linear regression

- however, if we **assume normal errors**, we can derive likelihood and do MLE estimation

$$\epsilon_t \sim \mathcal{N}(0, \sigma^2)$$
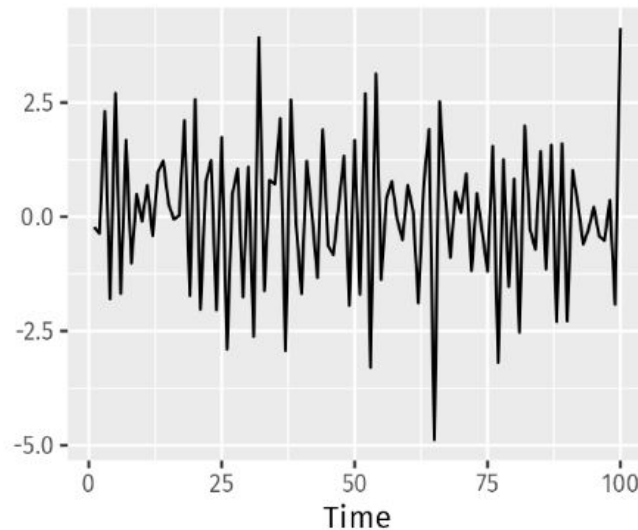
Moving average MA(q) processes

$$y_t = 20 + 0.8\epsilon_{t-1} + \epsilon_t$$

$$y_t = -\epsilon_{t-1} + 0.8\epsilon_{t-2} + \epsilon_t$$

MA(1)

MA(2)

ARMA(p,q) models

- **combination** of AR(p) and MA(q) into a single model:

$$\hat{y}_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t$$

- **idea:**
  - AR(p) models values autocorrelation
  - MA(q) corrects mistakes based on errors autocorrelation
- **problem:** since we have MA(q), we can't use OLS
- **training method:** there are a few, but MLE (or hybrid with MLE) is the most common one

- we won't be showing likelihood formulas, since they are quite complex; see additional resources if you're interested

Training ARMA(p,q) models

MLE:

- assume normal errors

- derive likelihood formula, and optimize NLL as cost function

- typically uses L-BFGS quasi-Newton optimizer

- popular: trivial code, strong statistical properties, probabilistic forecasting

Hannan-Rissanen method:

- start with AR(p), train with OLS

- compute errors - when they are known, we can train MA(q) with OLS

- then we have a choice:
  - repeat in a loop, until convergence ("pure" H-R)
  - switch to MLE, which typically rapidly converges (most common)

ARIMA(p,d,q) models

- ARIMA(p,d,q): first differencing I(d), then ARMA(p,q)

- **problem:**

    - how to choose p, d and q?

    - maybe don't use some, e.g. use ARI(p,d)?

    - higher p and q typically fits training data better, but also has higher risk of overfitting

- grid search CV - possible, but slow

- **AutoARIMA:**

    - d - statistical tests in a loop, up to d_{max}

    - p and q - build a grid and check it stepwise

    - use AIC (or AICc) to measure model quality

ARIMA - additional resources

- D. Childers - "ARIMA"

- J. Li - "ARMA Model"

- P. Cízek et al. - "Estimation of MA(q) and ARMA(p,q) Models"

- R. Nau - "Introduction to ARIMA models"

- M. Zhang - "Time Series: Autoregressive models AR, MA, ARMA, ARIMA"

- Real Statistics - "Calculate ARMA(p,q) coefficients using maximum likelihood"

- Statistics Ninja - "3 ARIMA Models - 3.5.2 Estimation - Maximum Likelihood Estimation"

- Lasse Engbo Christiansen - "02417 Lecture 12 part A: ARMA models on State space form"

Akaike's Information Criterion (AIC)

- **information criteria** measure model quality; AIC specifically is defined as:

$$AIC = -2\log(L) + 2k = 2 * NLL + 2k$$

  L - likelihood, how well model fits the data

  k - number of parameters, penalty for model complexity, "regularization"

- for ARIMA, $k = p + q$

- **alternative** to cross-validation score:
  - just calculate AIC using training data, and select the model with the lowest AIC
  - balances data fit and model complexity

- **pros:** speed, uses all data for training

- there are other information criteria, e.g. AICc, BIC, HQIC

- AICc corrects for small samples, useful e.g. in macroeconomics

Stepwise selection

- which p and q values should we check?

- **grid search:** very expensive, especially for expanding window

- **stepwise selection:**
  - check a few initial models

  - then go in "steps", checking neighboring values of p and q on the grid

  - typically gives very good results, despite greedy search

- traditionally checks at most 94 models
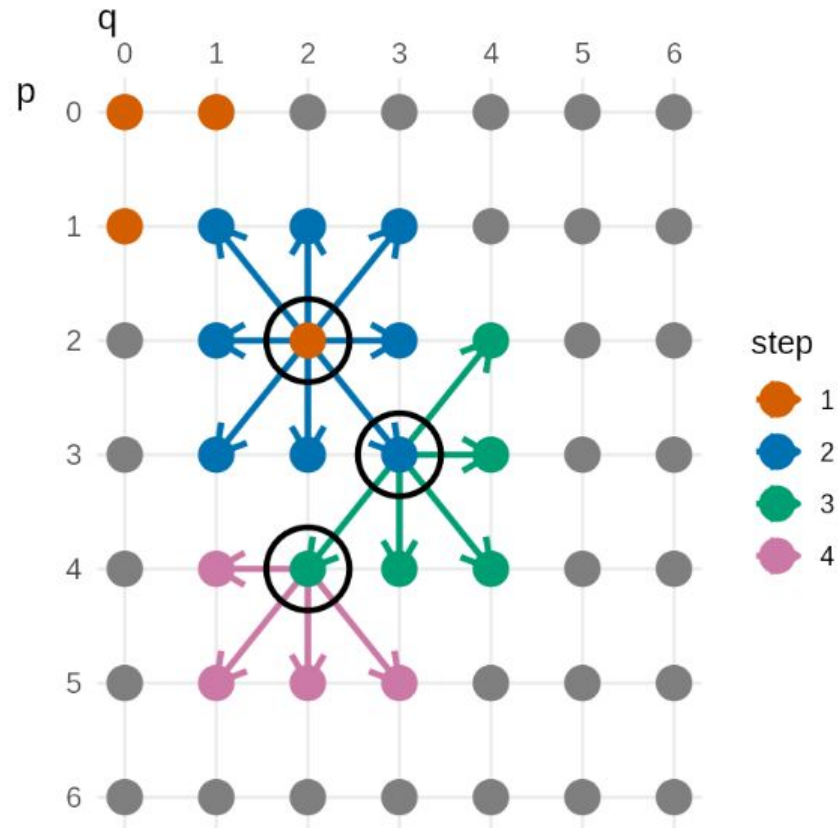
SARIMA

- **SARIMA = Seasonal ARIMA**

- simply add seasonal terms to ARIMA:

$$\hat{y}_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \sum_{i=1}^{P} \eta_i y_{t-i*m} + \sum_{i=1}^{Q} \psi_i \epsilon_{t-i*m} + \epsilon_t$$

  - SARIMA(p,d,q)(P,D,Q) with seasonality m

  - seasonal differencing

- **pros:** incorporates seasonality

- **cons:** computational cost, only single seasonality, not suitable for very long seasonal periods

- designed for relatively short seasonalities, e.g. 7, 14, 30 for daily data

Exogenous variables

- **exogenous variables** are additional features, incorporated in order to help the prediction

- e.g. for sales forecasting, variable "is weekend day" or "is a week before Black Friday"

- also known as:

  - auxiliary variables

  - covariates

- can be **static** or **dynamic:**

  - static - do not change with time, e.g. shop location, area in square meters

  - dynamic - change with time, e.g. "is there an ongoing promotion"

- matrix X with features, **known** at the moment of new forecast y_t

ARIMAX

- **ARIMA + exogenous variables**

- variants: ARMAX, ARIMAX, SARIMAX etc.

- they act as an additional linear regression; note that those variables are also differenced!

- ARMAX (for N exogenous variables, at time t):

$$\hat{y}_t = c + \sum_{i=1}^{N} w_i x_i^t + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t$$

- **pros:** often much better results, features are often quite obvious

- **cons:** computational cost, feature engineering

ARIMA extensions - additional resources

- FPP: [Estimation and order selection](), [ARIMA modelling in fable]()

- [Stack Overflow - Why SARIMA has seasonal limits?]()

- [Rob Hyndman - "Forecasting with long seasonal periods"]()

- [Rob Hyndman - "The ARIMAX model muddle"]()

ARIMA multi-step forecasts

- ARIMA by default gives formula for **one-step** forecast

- but how to do **multi-step** forecast?

- we just do forecasts one after another h times, and assume we are always correct

- last forecast is used as ground truth for the next forecast

- for MA(q) we assume new errors are zeros, e.g. $\epsilon_t = 0$

- model always makes a mistake, so we have **error compounding**

- this is a typical behavior for autoregressive models

- this is why confidence intervals get wider in time!

ARIMA - pros and cons

Pros:

- elastic and powerful

- typically work great for data with strong autocorrelations

- automation, AutoARIMA

- seasonality (SARIMA)

- exogenous variables (ARIMAX)

Cons:

- stationarity assumption (hard to meet)

- do not work well for strong, complex trend and seasonality

- quite large computational cost

- only additive models

- often require complex preprocessing (Box-Cox, seasonal differencing, regular differencing etc.)

Exponential Smoothing (ETS)

ETS models

- ETS = ExponenTial Smoothing / Error-Trend-Seasonality (it's the same)

- **family of models**, which directly model average value, trend, seasonality, and residual error

- they use **exponential weighting** - newest observations are much more important than past ones

- they generally "smooth out" latest values, hence the name

- **do not** assume stationary data

- work well for **small data samples**, and are also extremely **fast** to train

Moving average smoothing

- a.k.a. **Simple Moving Average (SMA)** model

- **idea:** next value is an average of (last) previous values

- canonical version uses all historical data, and is just "predict average" baseline:

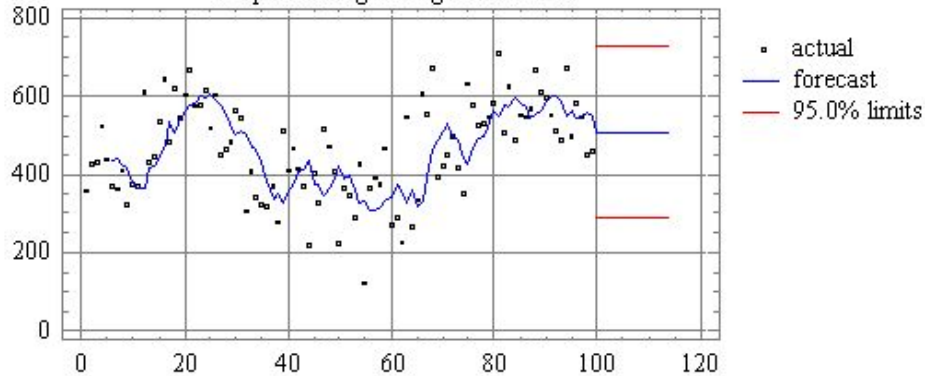$$\hat{y}_{T+h|T} = \mathrm{avg}(y) = \frac{y_1 + \ldots + y_T}{T}$$

- moving window version uses only last W values:

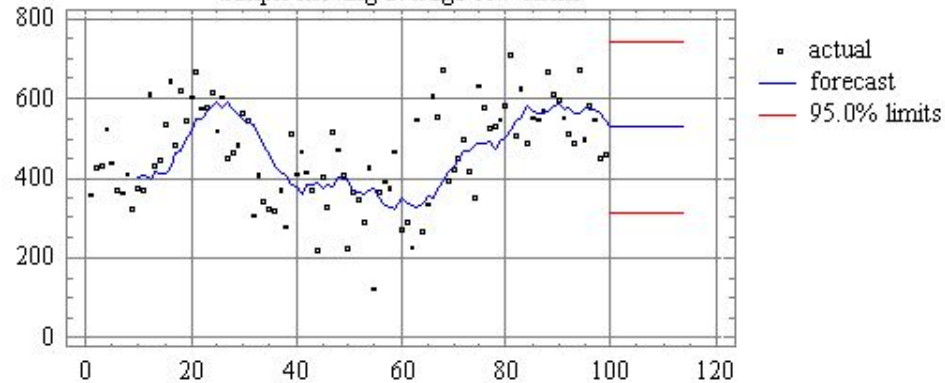$$\hat{y}_{T+h|T,T-1,\ldots,T-W} = \frac{y_T + y_{T-1} + \ldots + y_{T-W}}{W}$$

- works surprisingly well if data is slow to change

- prediction is a horizontal line (just like baseline)

Moving average smoothing

Simple Exponential Smoothing (SES)

- a.k.a. exponentially weighted moving average (EWMA)

- **idea:** last values are much more important, and weight falls exponentially fast

- weight of last observation is $\alpha \in (0, 1)$, and forecast is:

$$\hat{y}_{t+1} = \alpha y_t + \alpha(1 - \alpha)y_{t-1} + \alpha(1 - \alpha)^2 y_{t-2} + \dots$$

- still predicts a horizontal line, known as **level**

- smoothing, because it "smooths out" last values

- α parameter (how much to weight past values) is learned from data

Simple Exponential Smoothing (SES)
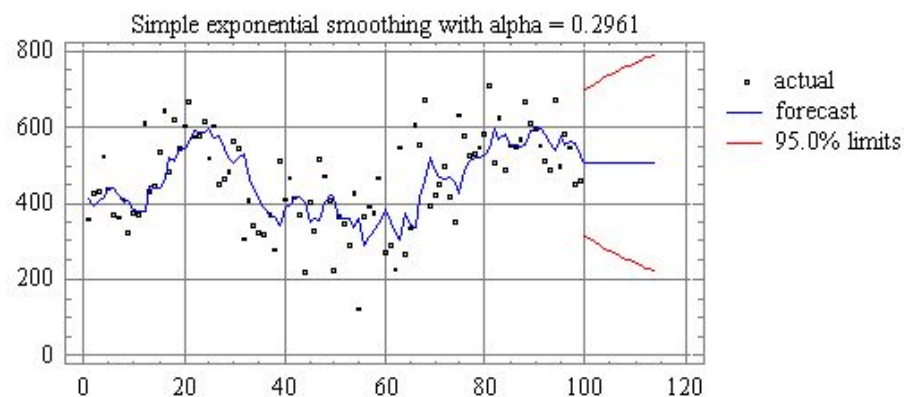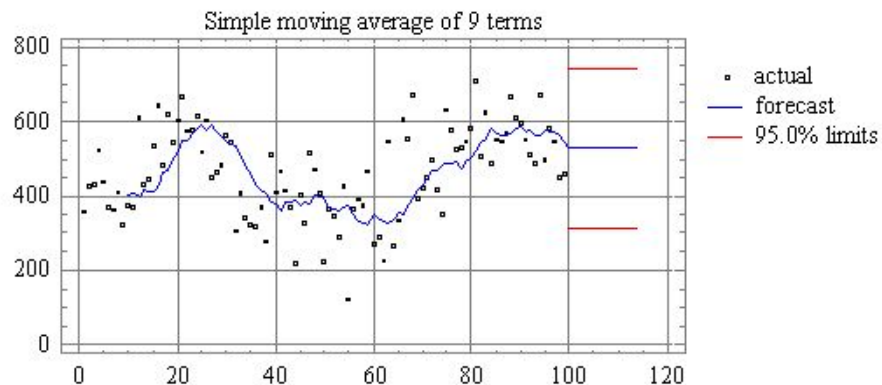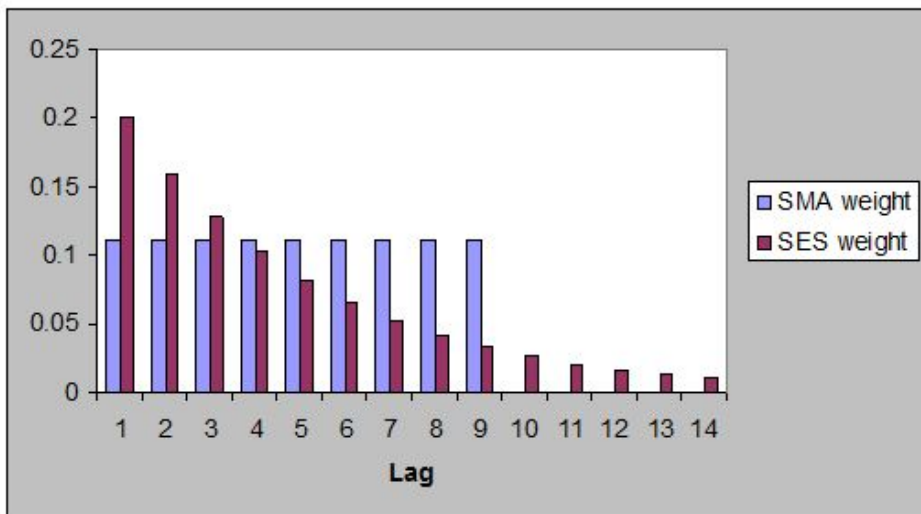
Component form

- to add further elements (trend, seasonality), it's convenient to write SES in so-called **component form**

- SES component formula explicitly models forecast (forecast equation) and level (level equation):

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

last forecast
(here, just the level)

l - level

- recurrent formula is convenient for multi-step forecasts

- for SES this is trivial, but further ETS expansions add:

  - more elements to the first formula (forecast equation)

  - further equations below

Holt model (double smoothing)

- a.k.a. Holt linear trend model (author - Charles Holt)

- **idea:** add linear trend, by another smoothing

- estimates trend line with slope b_t (trend equation):

$$\hat{y}_{t+h|t} = l_t + b_t * h$$
$$l_t = \alpha y_t + (1 - \alpha)(l_{t-1} + b_{t-1})$$
$$b_t = \beta(l_t - l_{t-1}) + (1 - \beta)b_{t-1}$$

line: $y = b + ax$

last forecast is now
level + trend

trend = slope = difference in level

- level and trend both use exponential smoothing, hence the name

- equations are still "weight * curr_val + (1 - weight) * prev_val", just now we have 2

- training now learns α and β

Damped trend

- **problem:** Holt's method trend goes to infinity, which gives unrealistic forecasts for longer horizons

- **damped trend:** multiply the trend in each step by $\phi \in [0, 1]$, which decreases it

$$
\begin{aligned}
\hat{y}_{t+h|t} &= l_t + (\phi + \phi^2 + \ldots + \phi^h) * b_t \\
l_t &= \alpha y_t + (1 - \alpha)(l_{t-1} + \phi b_{t-1}) \\
b_t &= \beta(l_t - l_{t-1}) + (1 - \beta)\phi b_{t-1}
\end{aligned}
$$

- typically $\phi \in [0.8, 0.98]$

- we'll omit this for readability in the next slides, but can be used for all ETS models

- typically **much better** forecasts, especially for longer horizons

- estimating one extra parameter ϕ is typically cheap

Holt's model (double smoothing)
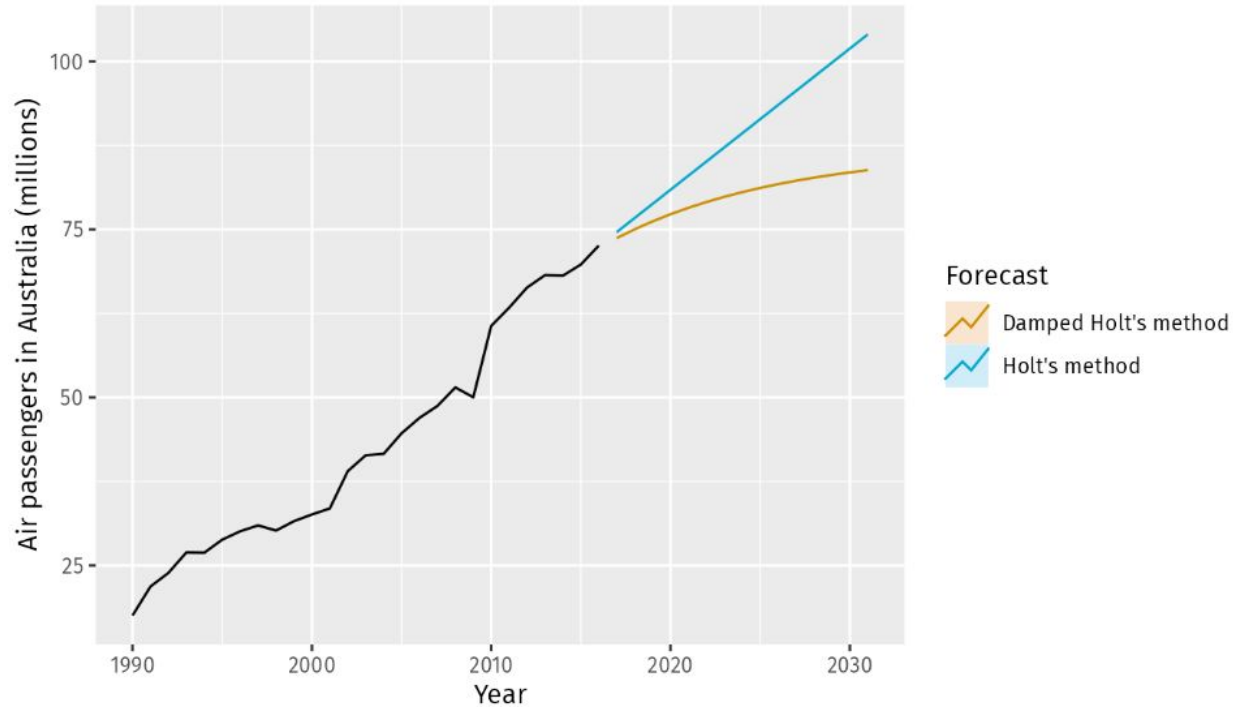
Holt-Winters model (triple smoothing)

- **idea:** add seasonality with third smoothing

- for given seasonality m, add third parameter γ :

same moment from
the last season

$$
\begin{aligned}
\hat{y}_{t+h|t} &= l_t + b_t * h + s_{t+h-m(k+1)} \\
l_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \\
b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \\
s_t &= \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}
\end{aligned}
$$

$$k = \lfloor \frac{h-1}{m} \rfloor$$

y = level + trend + seasonal --> seasonal = y - level - trend

- level equation changes, since we now estimate the level during the given season, so we have to subtract the seasonal part

- assumes additive seasonality (like e.g. STL); there is also a multiplicative variant

- only short seasonality supported, shorter than for ARIMA

Holt-Winters model (triple smoothing)
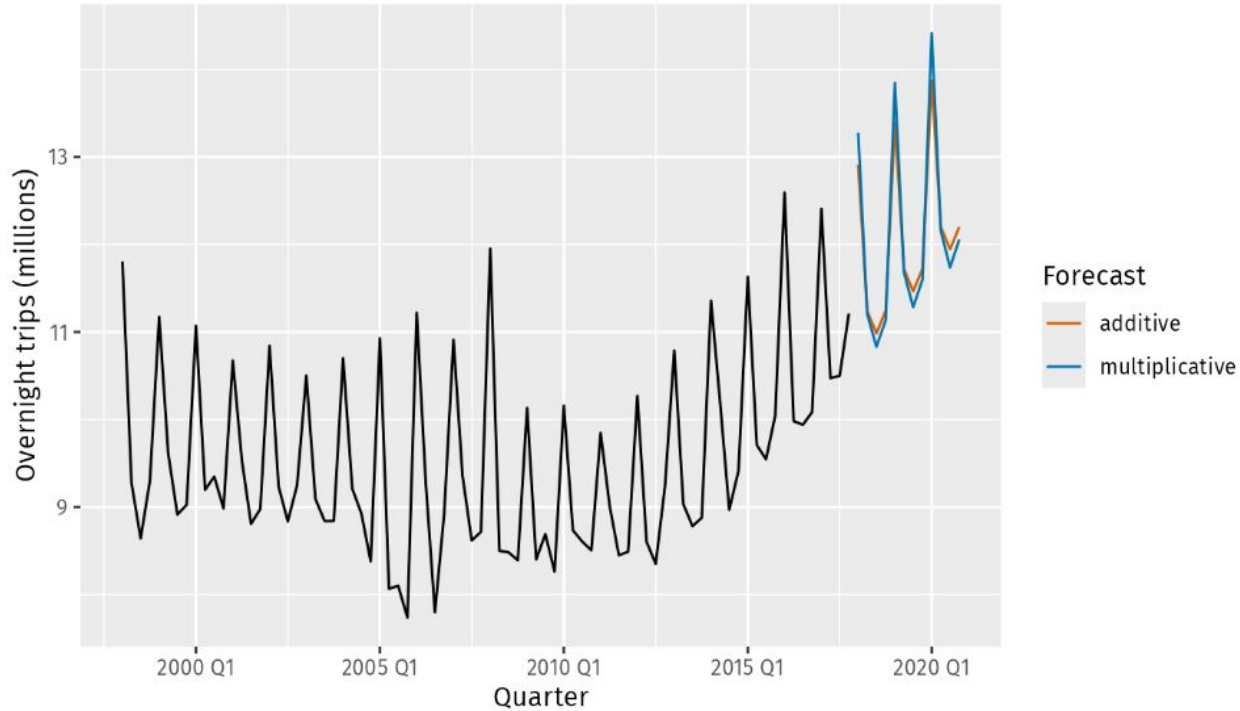
ETS models comparison

Holt-Winters Exponential Smoothing
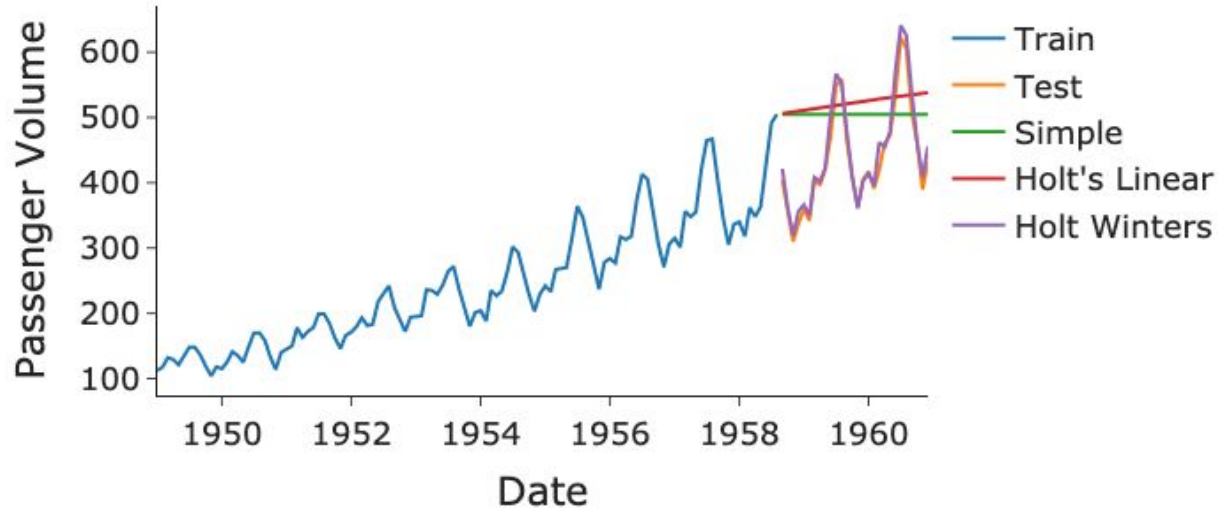
Training ETS models

- **training:** MLE

- **algorithm:** typically L-BFGS

- likelihood equations are complex, and their derivation is **very** complex, so we omit them here

- in practice, training is very fast

- ETS models are very strongly related to state-space models (SSMs) and Kalman filters

ETS models and SSMs - additional resources

- FPP slides - State space models

- "A state space framework for automatic forecasting using exponential smoothing methods" R. Hyndman et al.

- "Forecasting with Exponential Smoothing: The State Space Approach" R. Hyndman et al.

- D. Childers - "State space models"

ETS models and ARIMA

- all ETS models shown were **purely additive:**

  y = level + trend + seasonality + error

- in particular, we assume that prediction errors are additive (sum up with time)

- it can be shown that all purely additive ETS models have ARIMA counterparts

- but this **does not** mean that ETS is just a subset of ARIMA!

  - they don't assume stationarity

  - damped trend variants

  - there are many multiplicative variants, with no ARIMA counterpart

  - much faster training

ETS models family

- full grid of possible ETS models:

    - **E (error):** additive (A), multiplicative (M)

    - **T (trend):** no trend (N), additive (A), additive damped trend (A_d)

    - **S (seasonality):** no seasonality (N), additive (A), multiplicative (M)

- multiplicative trend is typically omitted, due to unstable models and bad forecasts

- additive and multiplicative error give the same point forecasts, but differ in confidence intervals

- if you're interested - <u>full table of ETS models</u>, with formulas

ETS models and ARIMA

Model	ETS formula	ARIMA formula
Simple exponential smoothing	(A,N,N)	(0,1,1)
Holt model	(A,A,N)	(0,2,2)
Holt model with damped trend	(A,A_d,N)	(1,1,2)
Holt-Winters model	(A,A,A)	$(0,1,m+1)(0,1,0)_m$
Holt-Winters with damped trend	(A,A_d,A)	$(1,0,m+1)(0,1,0)_m$
Holt-Winters with multiplicative seasonality	(A,A,M)	-

AutoETS

- **problem:** how to choose ETS model?

- we have MLE = we can compute AIC

- AutoETS works just like AutoARIMA (but typically much faster)

- **be careful:**

  - AIC-based approach is the same, but likelihood has different values, since those are completely different formulas and models!

  - we can't compare AIC between ARIMA and ETS, it can be used only to select models in one family

ETS - additional resources

- FPP: [Exponential smoothing](#)

- [Crunching the Data - "When to use exponential smoothing models"](#)

- [R. Tibshirani - "Exponential Smoothing With Trend and Seasonality"](#)

- [I. Svetunkov - "Why you should care about Exponential Smoothing"](#)

- [Cross Validated - Holt Winters with exogenous regressors in R](#)

ETS - pros and cons

Pros:

- great results, especially for strong trend and seasonality

- very fast

- good for small data

- many variants (including multiplicative), quite flexible

- can often model longer dependencies than ARIMA

Cons:

- not good when data often has dynamic or drastic changes

- cannot model very complex relations

- typically not good for long forecasting horizons

- only single seasonality

- only relatively short seasonal periods supported
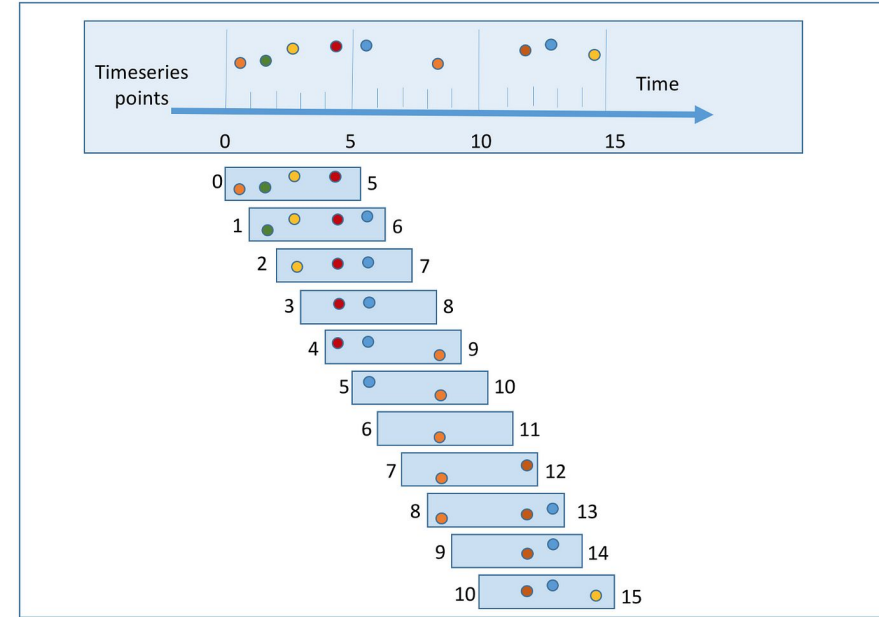
- no exogenous variables support

Forecasting as ML regression

Regression-based forecasting

- forecasting = regression + time

- **idea:** treat it exactly like regression problem, using time-based features

- typically called just "ML" in time series literature, and deep learning is always named explicitly

- features can only use data available **at the time of forecast**

- feature engineering relies on past data aggregates and exogenous variables

- **popular regressors:**

  - LightGBM boosting (rarely other ones)

  - linear models, in particular robust models (quantile / LAD, Huber)

- typically **state-of-the-art (SOTA)** in forecasting competitions, and often in business applications

Moving window features

- a. k. a. sliding window, rolling window

- **statistics** of the last N values, typically multiple ones

- e.g. mean, sum, min, max, stddev, quantiles

- **moving average** measures latest trend, e.g. average sales in the last 7, 14, 30 days

- **standard deviation** measures volatility, i.e. strength of shocks and changes

- **expanding window** can also be used, with values from the start of the time series; this captures more global properties

Date / event features

- exogenous variables describing **important dates** throughout the time period

- also called "event features", since they often describe whether a special event takes place

- application- and domain-dependent

- examples:

  - **quarter** - sales often drop in 4th quarter

  - **is a Black Friday week** - a lot of promos and higher sales

  - **is December / January** - much higher sales before holidays, drastic fall after

  - **days in month** - 28 vs 31 days impact sales

  - **is weekend / is Sunday / is a holiday** - high impact on business days, also shops can be closed

Encoding date and time

- including **date and time features** is not straightforward:
  - we have a "wraparound", e.g. hours 1 and 23 are far as numbers, but close in reality
  - seasonality can have similar impact, e.g. similar weather each July
  - same for other cases, e.g. days in month (1st and 31st), months
  - formally called **cyclical** nature of time
- **naive** options:
  - ordinal encoding (just integers), one-hot encoding (independent binary features)
  - very simple, sometimes work, but don't encode the cyclicity
  - **pros:** works well when particular value is important, e.g. given month
  - **cons:** high dimensionality, sparse data, doesn't work well if data ranges are important (e.g. afternoon hours)
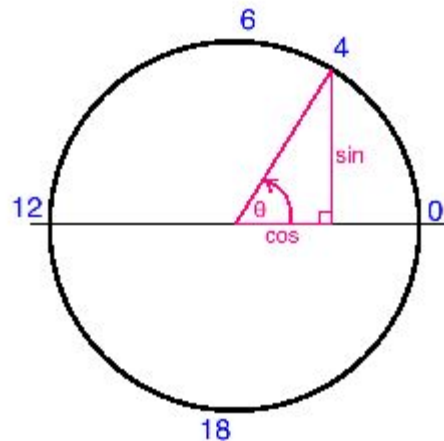
Cyclical encoding

- **idea:** encode timestamp using a **trigonometric basis**, as a point on a unit circle

- uses sine and cosine parts, so 2 features, with **radians** as unit:

$$x(a) = \sin\left(\frac{a * 2\pi}{\max(a)}\right) \qquad y(a) = \cos\left(\frac{a * 2\pi}{\max(a)}\right)$$

a - timestamp as integer, e.g. hour, month
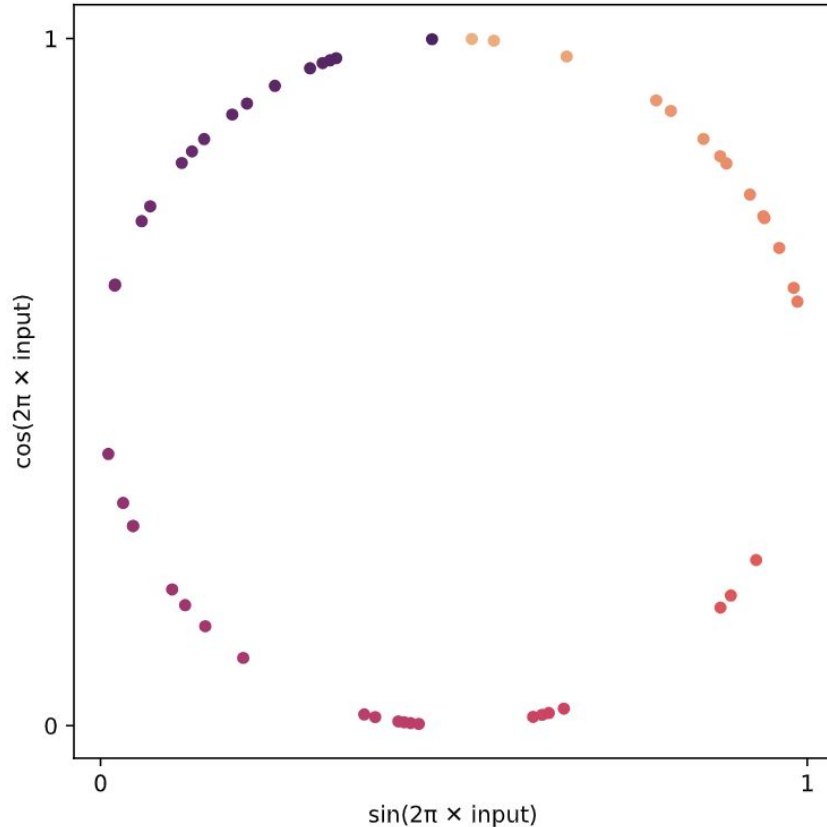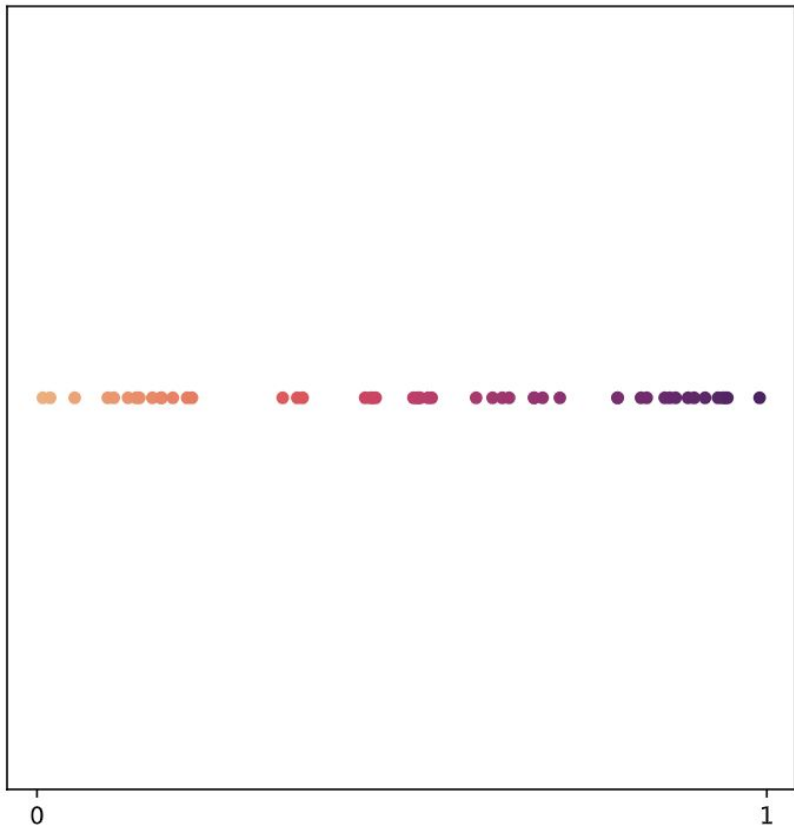
max(a) - maximal period value, e.g. 24 for hours, 12 for months

- this way we have a natural wraparound, like on a clock - large values are close (in radians) to small ones

- **pros:** encodes cyclicity, works well for time periods

- **cons:** 2 separate features, which are not directly connected (tree-based models split on single feature)

Cyclical encoding - visualization

Cyclical encoding - additional resources

- "Cyclical Encoding: An Alternative to One-Hot Encoding for Time Series Features" P. Hayden

- "Encoding Cyclical Features for Deep Learning" A. Van Wyk

- "The best way to encode dates, times, and other cyclical features" H. Pim

- "Why We Need Encoding Cyclical Features" A. Kud

- "Understand the capabilities of cyclic encoding" S. Matsumoto

- "Feature Engineering - Handling Cyclical Features" D. Kaleko

- "Handling cyclical features, such as hours in a day, for machine learning pipelines with Python example" A. Sefidian

Fourier features

- **idea:** model seasonality with sines and cosines with varying frequency - **Fourier terms**

- also called **harmonic regression**

- seasonality is **periodic** by definition, so Fourier approximation works really well

- for time series with period m, it creates pairs of features:

$$x_k(t) = \sin\left(\frac{2k\pi t}{m}\right) \quad x_{k+1}(t) = \cos\left(\frac{2k\pi t}{m}\right)$$

$$x_1(t) = \sin\left(\frac{2\pi t}{m}\right), x_2(t) = \cos\left(\frac{2\pi t}{m}\right), x_3(t) = \sin\left(\frac{4\pi t}{m}\right), x_4(t) = \cos\left(\frac{4\pi t}{m}\right), \ldots$$

- more terms = better approximation, but often **just a few** suffice

- K is a hyperparameter, also feature selection can be used

- K has max value $m/2$, when it's equal to one-hot encoding

Fourier features

- pure harmonic regression:

$$\hat{y}(t) = \sum_{k=1}^{K} \left(a_k \sin \left(\frac{2k\pi t}{m} \right) + b_k \sin \left(\frac{2k\pi t}{m} \right) \right)$$

a_k, b_k - weights

- **pros:**
  - models **long periods** well - for low K the patterns are smooth and change slowly
  - models **non-integer periods**
  - useful for high frequency and long seasonality

Harmonic regression example

- K=1, 2, …, 6

- larger K = learns more complex patterns

Total monthly eating-out expenditure

https://otexts.com/fpp3/dhr.html
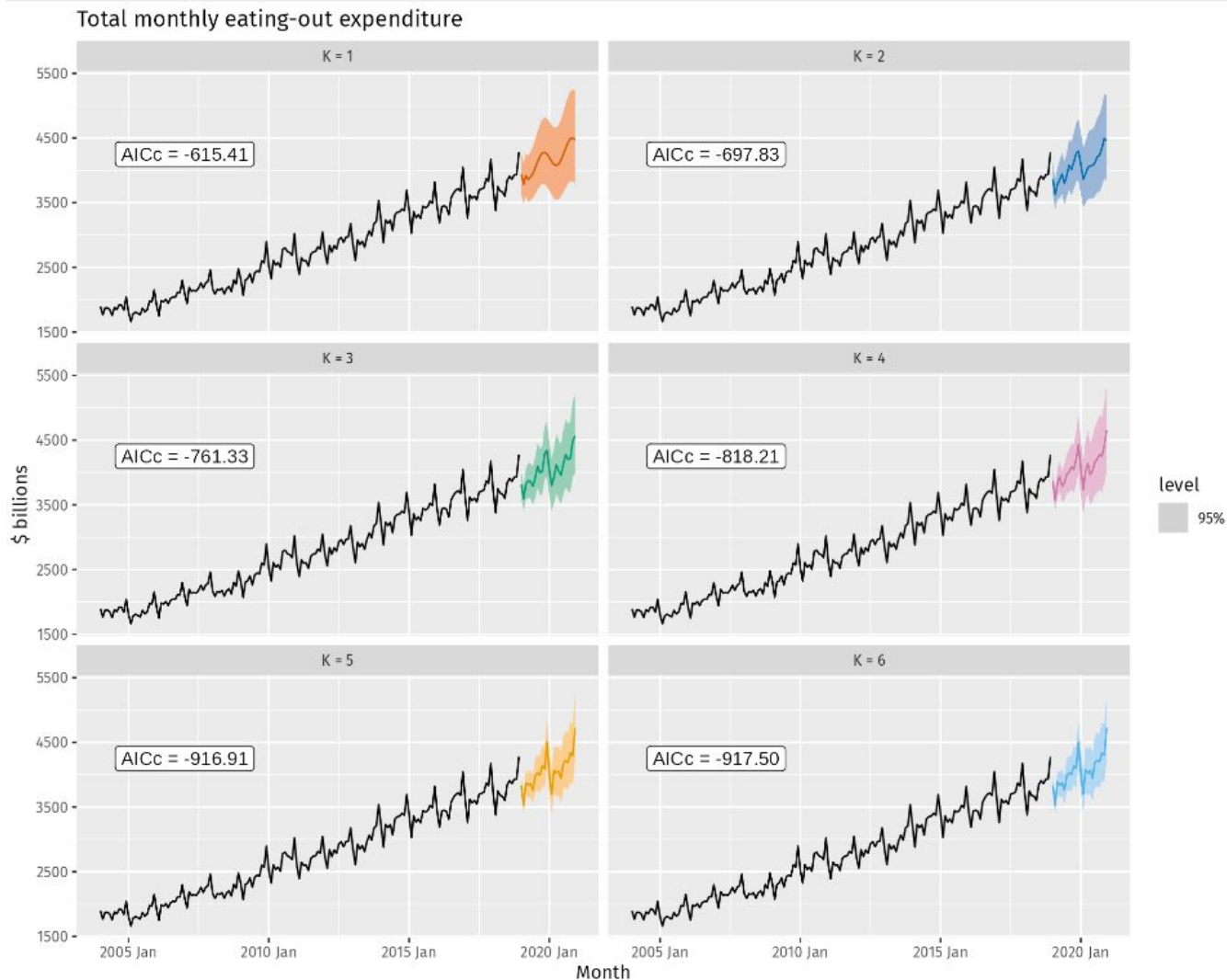
Fourier features - additional resources

- FPP: [Fourier features](#)

- ["Forecasting with long seasonal periods" R. Hyndman](#)

- ["Forecasting with daily data" R. Hyndman](#)

- ["Forecasting with weekly data" R. Hyndman](#)

- [Sktime docs: Fourier features](#)

- [Cross Validated - Importance of Fourier terms in time series forecasting](#)

- [Cross Validated - Fourier terms to model seasonality in ARIMA models](#)

- ["Modeling variable seasonal features with the Fourier transform" F. Andrei](#)

Time series features - additional resources

- FPP: [Some useful predictors](#)

- ["Must-Know Base Tips for Feature Engineering With Time Series Data" V. Shkulov](#)

- [scikit-learn docs: Time-related feature engineering](#)

- ["Practical Guide for Feature Engineering of Time Series Data" J. Gordon](#)

- ["Feature-based time-series analysis" B. Fulcher](#)

Combining everything

- those features can be used **in conjunction** with statistical models, as exogenous variables

- using ML regression is not required

- you can even combine the two, in so-called **dynamic regression:**

  - feature engineering + ML-based regression

  - subtract forecast from time series

  - model and forecast the residuals with ARIMA

  - if using Fourier terms, known as **dynamic harmonic regression (DHR)**

- alternatively, ARIMA / ETS forecasts can be input features for ML regression in a form of **stacking** (a kind of ensemble learning)

- those combinations are particularly useful for long seasonal periods and long forecasts

Dynamic regression - additional resources

- FPP: Dynamic regression models (whole book chapter)

- "The ARIMAX model muddle" R. Hyndman

Regression-based forecasting - pros and cons

Pros:

- can encode rich additional data

- works great, typically SOTA with good feature engineering

- can learn complex patterns, long time dependencies etc.

- many algorithms to choose from

- explainability

- LightGBM pros, e.g. fast, widely used, well researched

Cons:

- have to do feature engineering

- MLOps complexity, have to build and maintain the whole pipeline

- underperforms for univariate time series

- LightGBM cons, e.g. quite easily overfits, requires a lot of hyperparameter tuning

Why not
deep learning?

Deep learning is hard

- deep learning often **underperforms**, especially for:
  - univariate time series
  - small data
  - short forecasting horizons
  - high frequency data
  - important exogenous variables (they typically can't be used)
- many other **problems:**
  - slow and costly to train
  - hard to train: unstable, overfit easily, many hyperparameters to tune
  - high hardware requirements for inference

Statistical methods still work well

- statistical models have **a lot of advantages:**

  - great for univariate data

  - automated variants, work out-of-the-box

  - fast training and inference

- they often **give good results:**

  - frequently not exactly SOTA, but very close

  - good enough for most practical use cases

  - M4 competition (2018): all pure ML models failed to outperform statistical methods

  - M5 competition (2020): 92.5% of submissions, including ML, deep learning and hybrids, did not outperform AutoETS

Boosting-based ML is still SOTA

- LightGBM **typically wins** forecasting competitions:

  - great for multivariate time series

  - gains a lot from feature engineering

  - works surprisingly well for learning cross-series information

  - very fast training (a lot of hyperparameter tuning, though)

- **disclaimer:** competitions are often unrealistic, e.g. lots of data, overoptimization for test set, typically huge unstable ensembles win

- but LightGBM is **always** in top 3, typically wins, so it says a lot

- in M5 competition, winner and 4 out of 5 top solutions used LightGBM ensembles

- a lot of other examples and analysis: "Kaggle forecasting competitions: An overlooked learning opportunity" C. Bojer, J. Meldgaard

Further reading

- J. Dancker "Why You (Currently) Do Not Need Deep Learning for Time Series Forecasting"

- State of Competitive Machine Learning 2023 - Time Series Forecasting

- "Statistical, machine learning and deep learning forecasting methods: Comparisons and ways forward" S. Makridakis et al.

- "M5 accuracy competition: Results, findings, and conclusions" S. Makridakis

- "Statistical vs Deep Learning forecasting methods" - Nixtla

- "Deep Learning Is Not What You Need" V. Manokhin

Questions?

Other forecasting algorithms

ARFIMA

- **FI = fractional differencing**, with order d in range (0,1)

- keeps the shape between original data and stationary one, still partially incorporating the trend

- ARFIMA models have long memory, keeping long dependencies in the data

- particularly useful in macroeconomics and finance, which have

- Wikipedia - ARFIMA

- "ARIMA and ARFIMA models" C. Baum

- "AutoRegressive Fractionally Integrated Moving Average (ARFIMA) model" J. Tanaka

BATS and TBATS

- BATS and TBATS models are arguably the most modern statistical models for time series

- evolution combining most previous important building blocks:

  - **T**rigonometric seasonality - Fourier features

  - **B**ox-Cox transformation

  - **A**RMA errors

  - **T**rend and **S**easonal components

- trend and seasonality are modelled like in ETS, Box-Cox and ARMA errors like in ARIMA

- "Forecasting time series with complex seasonal patterns using exponential smoothing" A. De Livera et al.

- "Forecasting Time Series with Multiple Seasonalities using TBATS in Python" G. Skorupa

ARCH and GARCH

- ARCH = autoregressive conditional heteroskedasticity

- ARCH and GARCH explicitly **model the autocorrelation of variance**

- ARCH uses AR(p), and GARCH uses ARMA(p,q)

- useful in finance in econometric, where shocks, sudden changes etc. are particularly relevant and can have long influence

- so-called volatility and volatility clustering

- R. Engle and C. Granger won Nobel's Prize in economics for inventing the ARCH model

- Wikipedia - ARCH and GARCH

- "Time Series Model(s)—ARCH and GARCH" R. Kumar

Other interesting models

- Theta
  - Theta: remove seasonality, model with OLS regression and SES, combine
  - expanded by Four Theta and Optimized Theta methods
- Croston method
  - for intermittent time series (lots of zeros) and count data
  - 2 SES models: one to forecast non-zero period, second for amount for next period
- vector autoregression (VAR)
  - extension of ARIMA family for multivariate time series
  - FPP: VAR
  - "Vector AutoRegressive Moving Average Models: A Review" M.C. Düker et al.

Prophet

- ["Forecasting at scale" S. Taylor, B. Letham](#)

- this is **not** an interesting model, but rather a dangerous one!

- Generalized Additive Model (GAM):

  y = trend + seasonality + holidays + noise

- Bayesian MCMC estimation, with probabilistic forecasts

- at its heart, this is just **curve fitting** to historical data, and it assumes constant variance

- both too simple and too complicated to be really useful

Prophet

- ["Zillow, Prophet, Time Series, & Prices"](#)

- ["Facebook Prophet, Covid and why I don't trust the Prophet" S. Seitz](#)

- ["Is Facebook's "Prophet" the Time-Series Messiah, or Just a Very Naughty Boy?"](#)

- ["Shortcomings of Facebook Prophet for Time Series Analysis"](#)

- ["Comparing Prophet and Deep Learning to ARIMA in Forecasting Wholesale Food Prices" L. Menculini et al.](#)

- ["A Worrying Analysis of Probabilistic Time-series Models for Sales Forecasting" S. Jung et al.](#)

- ["Cash flow prediction: MLP and LSTM compared to ARIMA and Prophet" H. Weytjens](#)