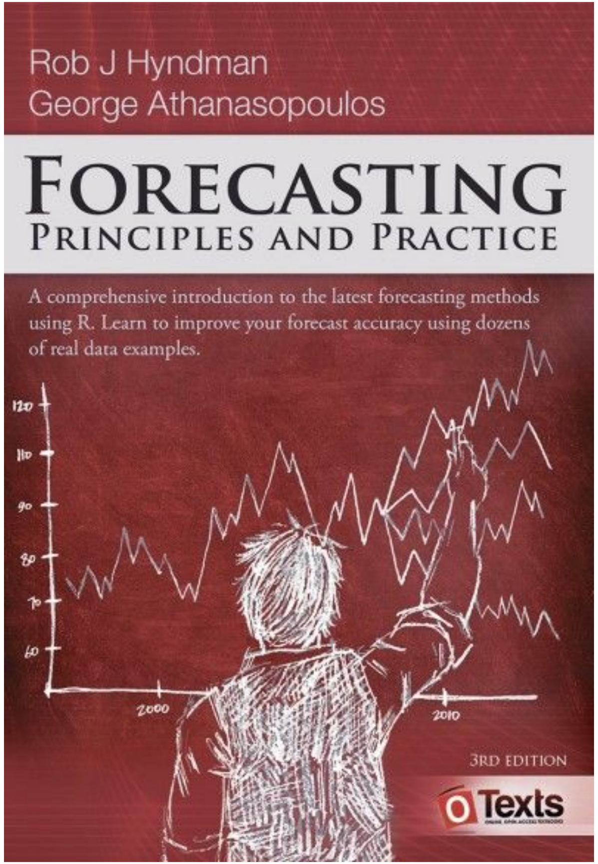
Machine Learning

Time series forecasting

Part 1:
introduction

Sources

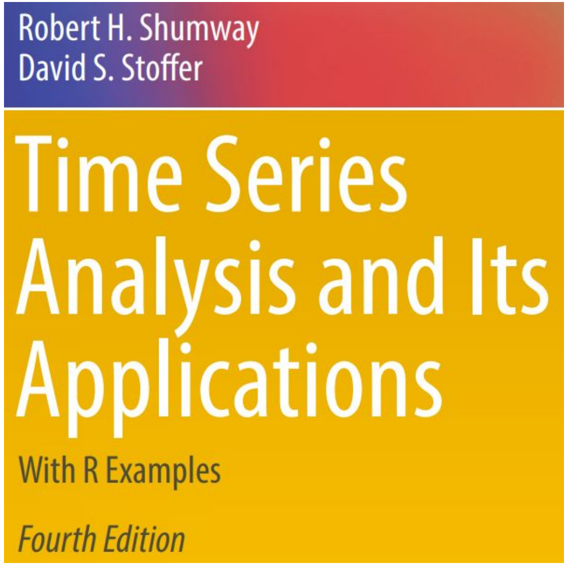


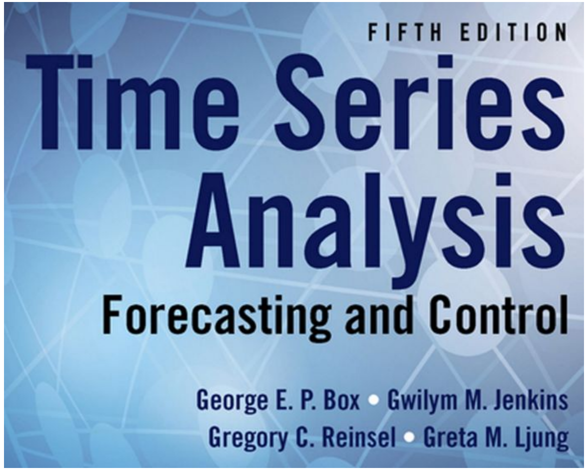
Time Series Analysis, Forecasting, and Machine Learning

Python for LSTMs, ARIMA, Deep Learning, AI, Support Vector Regression, +More Applied to Time Series Forecasting

Bestseller 4.7 ★★★★☆ (2,237 ratings) 8,376 students

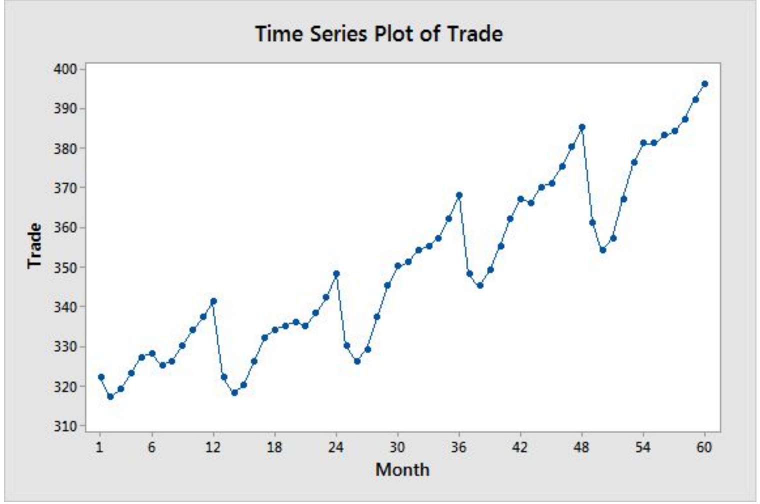
Created by Lazy Programmer Team, Lazy Programmer Inc.

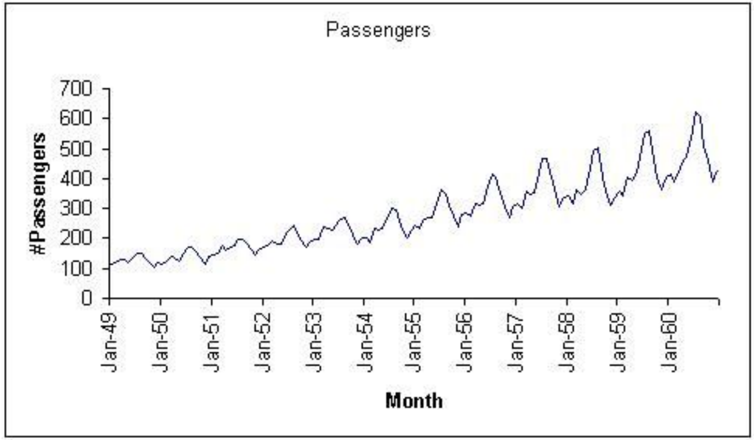




Time series

- sequence of **time-ordered values**, typically:
  - discrete time steps
  - continuous values
- time steps have given **frequency**, e.g. day, month
- we can have additional features, called **exogenous variables**
- we can have **multivariate** time series, i.e. many series in parallel, e.g. profits and costs, supply and demand
- related areas: signal processing, stochastic processes

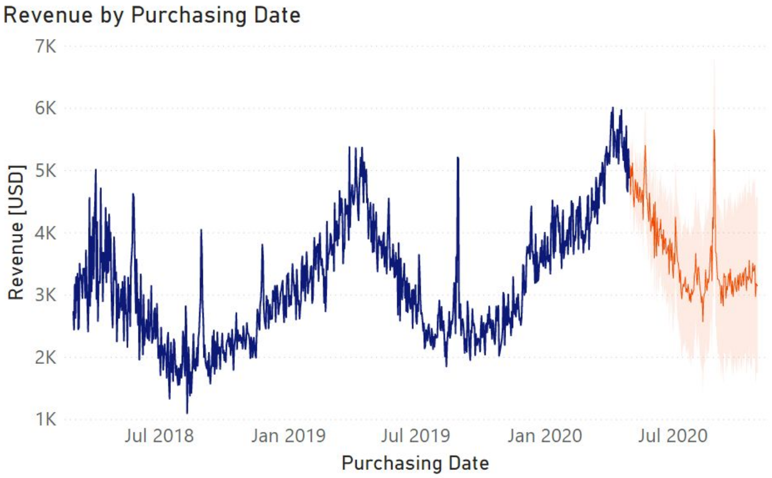


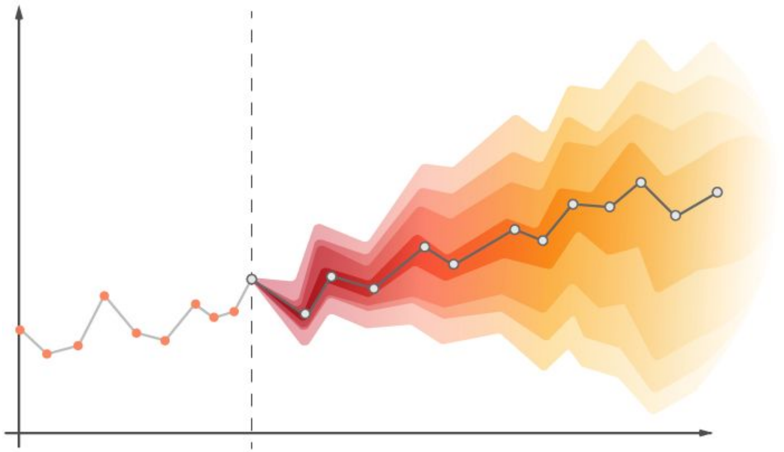


https://statisticsbyjim.com/time-series/time-series-analysis-introduction/
https://www.solver.com/time-series

Time series forecasting

- **forecasting** - predicting future values
- regression, but in time
- we can only use **past** values of time series
- optionally, also current exogenous variables
- **point forecasts** - predict a single value for a time step
- **probabilistic forecasts** - predict a range of possible values, for a given confidence interval (e.g. 50%, 95%)
- cost common task business-wise

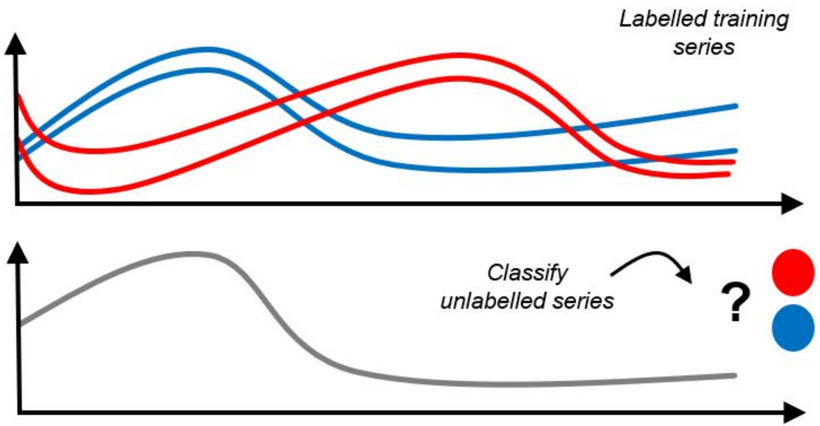




https://freedium.cfd/https://towardsdatascience.com/quick-and-easy-time-series-forecasting-in-powerbi-a-practical-guide-c31a858ee220
https://freedium.cfd/https://medium.com/analytics-vidhya/time-series-forecasting-c73dec0b7533

Time series classification

- classifying **the whole time series**
- typically for signals, e.g. in medicine (EEG, EKG)
- we know the whole signal, both previous and later data



https://www.sktime.net/en/v0.20.0/examples/02_classification.html

Time series classification - additional resources

- typically is based on a particular **feature extraction algorithm**
- most well-known algorithms:
    - Dynamic Time Warping (DTW)
    - shapelet transform
    - Time Series Forest
    - ROCKET, MiniROCKET
    - TSFRESH
    - various 1D CNNs
- J. Kezmann "All 8 Types of Time Series Classification Methods"
- "Time Series Classification: A Review of Algorithms and Implementations" J. Faouzi
- "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances" A. Bagnall et al.
- "Comparison of Manual and Automated Feature Engineering for Daily Activity Classification in Mental Disorder Diagnosis" J. Adamczyk, F. Malawski

Other time series tasks

- **anomaly detection / event detection**
  - intrusion detection
  - fault detection (predictive maintenance)
- **segmentation**
  - voice detection
  - speaker diarization ("who spoke when?")
- **clustering**
  - "Time-series clustering – A decade review" S. Aghabozorgi et al.

Time series forecasting - applications

- **business forecasting:**
  - demand, sales, production cost etc.
  - everywhere in business intelligence (BI)
- **finance & econometrics:**
  - short- and long-term macroeconomics, stock value, options, derivatives etc.
  - technical analysis and high frequency trading (HFT)
  - disclaimer: efficient market hypothesis, random walk hypothesis etc.
- **cloud & DevOps:**
  - network traffic, utilization forecasting etc.
  - VM predictive scaling

Forecasting - classical approaches

- **heuristics:**
  - trivial, but often surprisingly good results
  - crucial as baselines
  - e.g. average, last value

- **statistical models, simple ML:**
  - based on statistical analysis of time series
  - well researched
  - often give the best results, especially for one-dimensional data
  - e.g. ARIMA, ETS, Theta, TBATS, regression (e.g. boosting)

Forecasting - neural approaches

- **simple deep learning:**
  - often combination of statistics and simple MLPs
  - still strongly based on statistical time series analysis
  - better for complex data and multivariate time series
  - e.g. N-BEATS, N-HiTS, TSMixer, TiDE, recurrent neural networks (e.g. LSTM, GRU)

- **complex deep learning models:**
  - recently started to give reasonable results, but it's incredibly hard
  - e.g. TFT, PatchTST, Chronos, TimesFM, TimeGPT

Time series analysis and decomposition

Notation

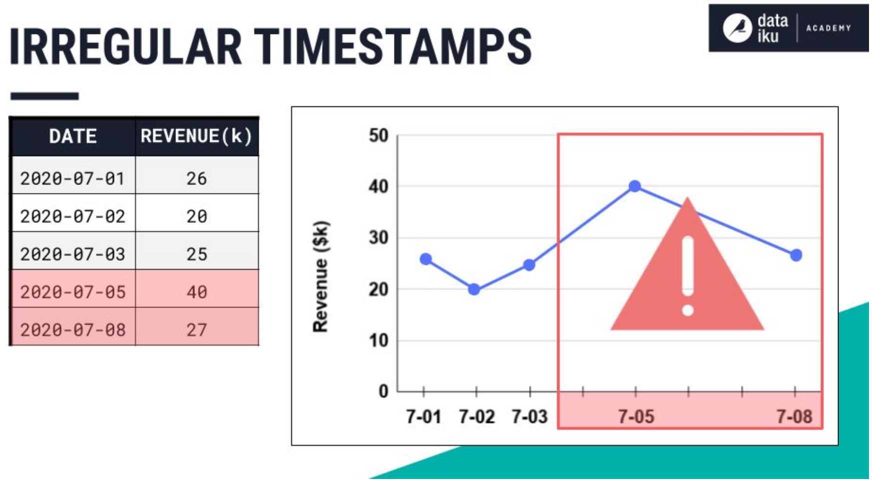
- t - time step index
- T - total number of time steps
- $t = 1, 2, \ldots, T$ - **indexed from 1**
- y_t - value at step t
- y_T - value at step T, last known value
- $\hat{y}_{T+h}$ - forecast h steps ahead
- $\hat{y}_{T+h|T}$ - forecast h steps ahead, explicitly using previous T values

Frequency

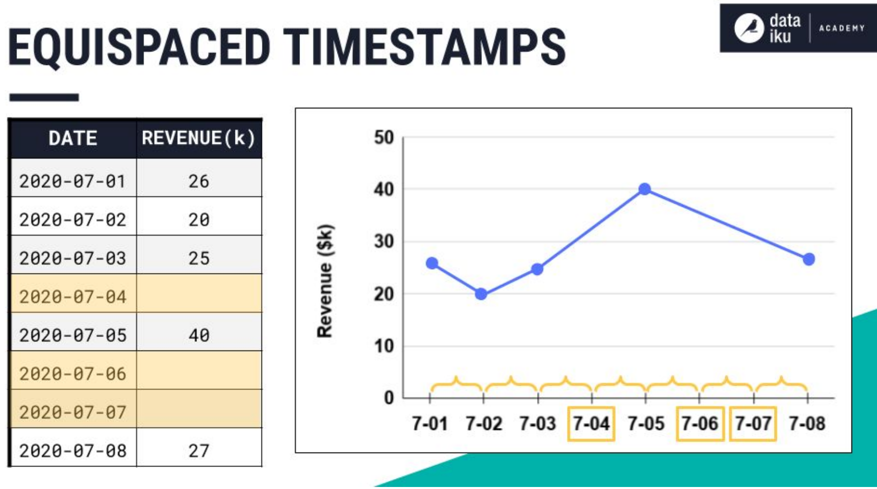
- **data frequency:** hourly, daily, monthly etc.
- typically related to the type and amount of data:
  - frequent: sensors, automated measurements etc.
  - infrequent: macroeconomics data, sales aggregates etc.
- basically all methods **require** single frequency
- **problems:**
  - missing data
  - too much data, unwanted variance (noise)

- if we have a lot of zeros, this is called **intermittent** time series, and requires dedicated models

Frequency



DATE	REVENUE(k)
2020-07-01	26
2020-07-02	20
2020-07-03	25
2020-07-05	40
2020-07-08	27



DATE	REVENUE(k)
2020-07-01	26
2020-07-02	20
2020-07-03	25
2020-07-04	
2020-07-05	40
2020-07-06	
2020-07-07	
2020-07-08	27

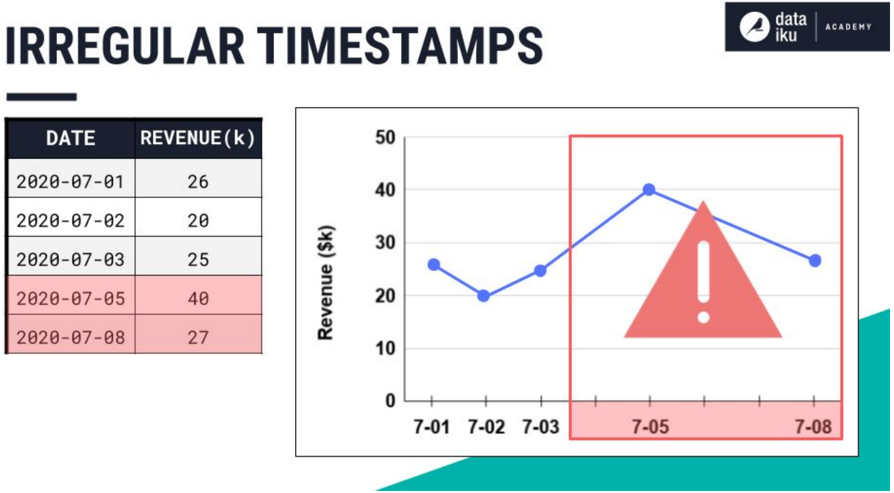
https://knowledge.dataiku.com/latest/ml-analytics/time-series/concept-resampling.html

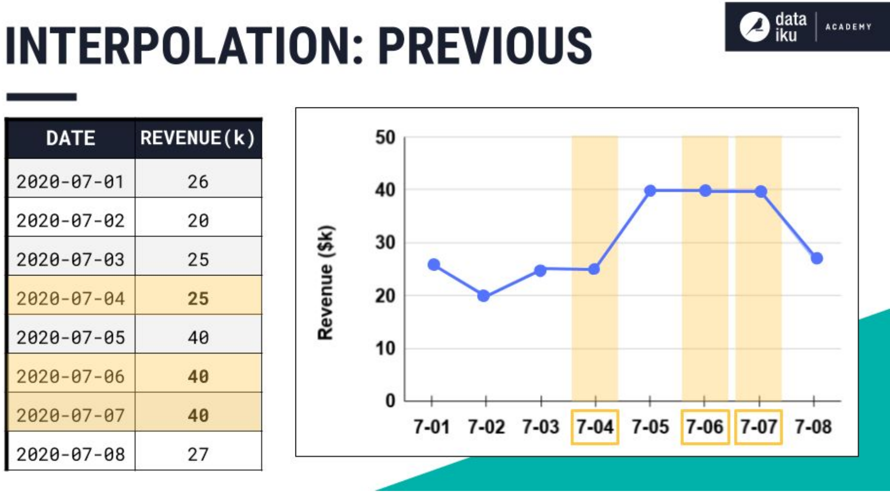
Resampling

- **aggregation** of existing data over time periods
- one of:
  - pick 1 data point every N values, e.g. last day of each month
  - "GROUP BY" over time, e.g. average, sum
- **reduces frequency**, e.g. daily sales -> total weekly sales

Forward fill

- **fills missing values** by copying last known value
- causal - we only use past knowledge
- most common in forecasting
- sometimes called "last value interpolation" or "previous value interpolation"





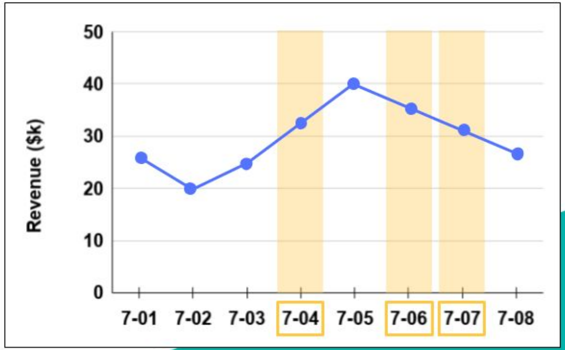
https://knowledge.dataiku.com/latest/ml-analytics/time-series/concept-resampling.html

Interpolation

- **fills missing values** by using some interpolation scheme
- e.g. linear, polynomial, (historical) average/median
- **only** for time series classification - uses future data and can't extrapolate

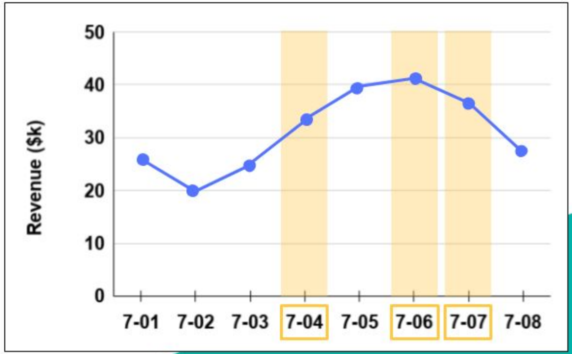
INTERPOLATION: LINEAR

DATE	REVENUE(k)
2020-07-01	26
2020-07-02	20
2020-07-03	25
2020-07-04	32.5
2020-07-05	40
2020-07-06	35.67
2020-07-07	31.33
2020-07-08	27



INTERPOLATION: QUADRATIC

DATE	REVENUE(k)
2020-07-01	26
2020-07-02	20
2020-07-03	25
2020-07-04	33.35
2020-07-05	40
2020-07-06	41.16
2020-07-07	36.82
2020-07-08	27



https://knowledge.dataiku.com/latest/ml-analytics/time-series/concept-resampling.html

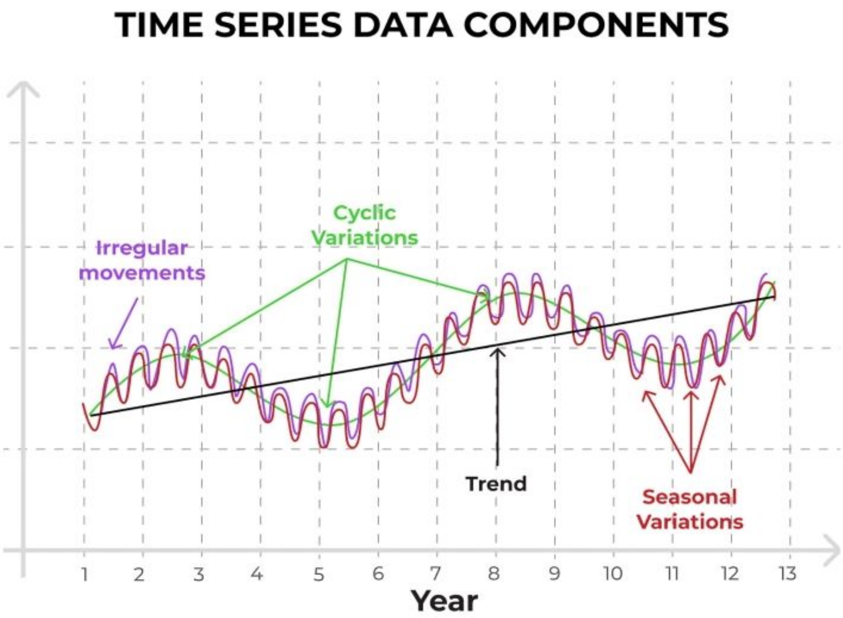
Time series components

- **time series decomposition** - break the data into simpler components
- most common one:

data = trend + seasonality + cycle + remainder

- **trend** - general tendency to go up/down
- **seasonality** - regular, periodic changes
- **cycle** - irregular cycles, no constant periodicity
- **remainder** - all other irregular changes

- seasonality is assumed to be known or suspected based on domain knowledge



https://codeit.us/blog/machine-learning-time-series-forecasting

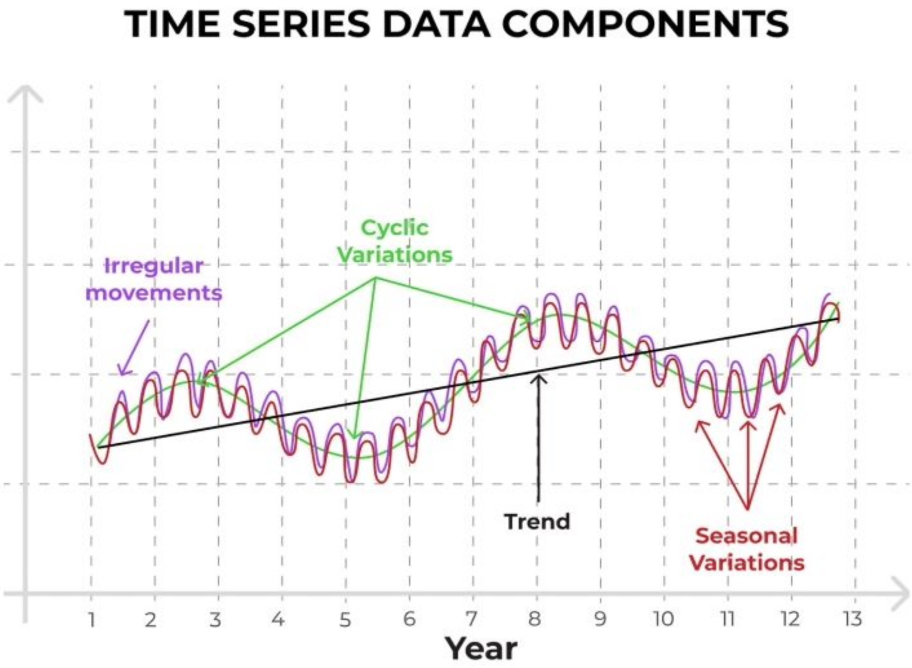
Time series components

Trend-cycle:

- combined trend and cycle, often called just "trend"
- allows irregular trend, going up or down
- data = trend + seasonality + remainder

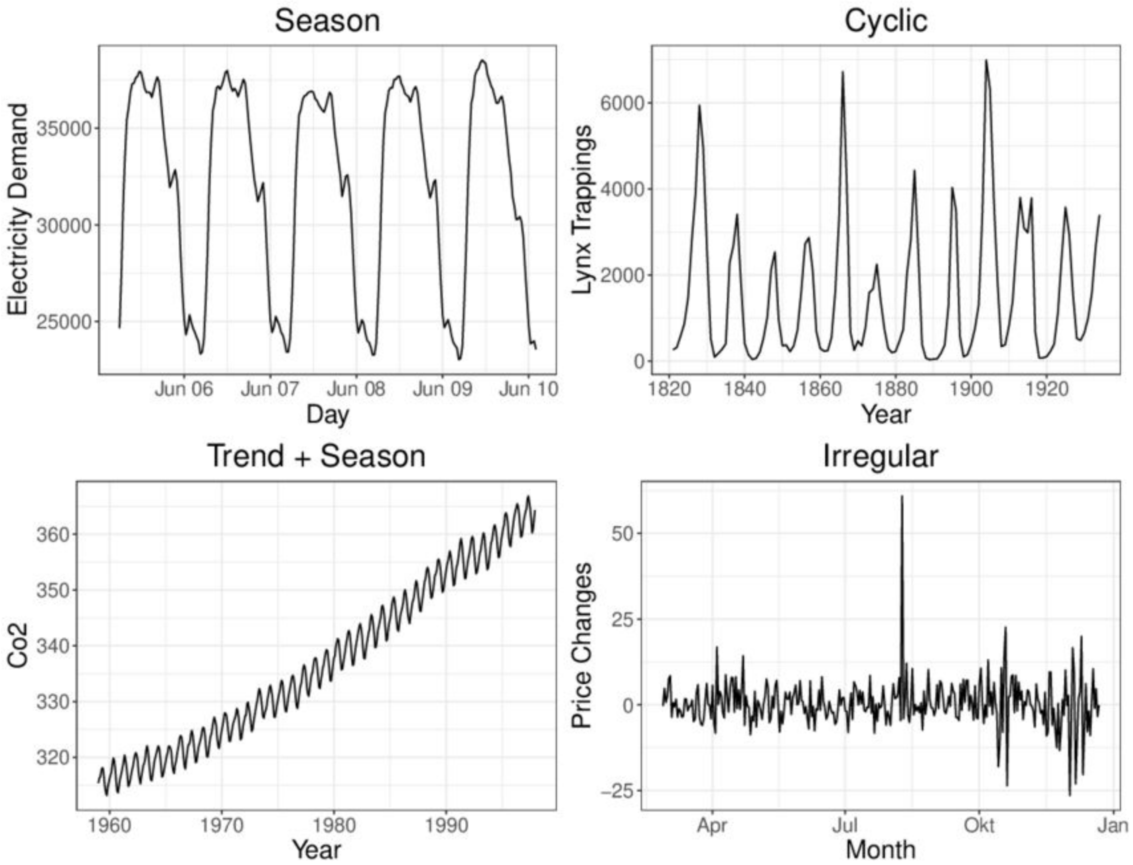
Multiple seasonalities:

- e.g. weekly and yearly
- often hard to incorporate this, but is often important



https://codeit.us/blog/machine-learning-time-series-forecasting

Time series components



https://www.researchgate.net/publication/348592737_Automated_Hybrid_Time_Series_Forecasting_Design_Benchmarking_and_Use_Cases

Additive vs multiplicative decompositions

- **additive decomposition** assumes independent (additive) components:

$$y_t = T_t + S_t + R_t$$

- not necessarily true, e.g. Black Friday sales spike (seasonality) will be more visible as the store gets bigger (trend)
- **multiplicative decomposition** reflects this idea:

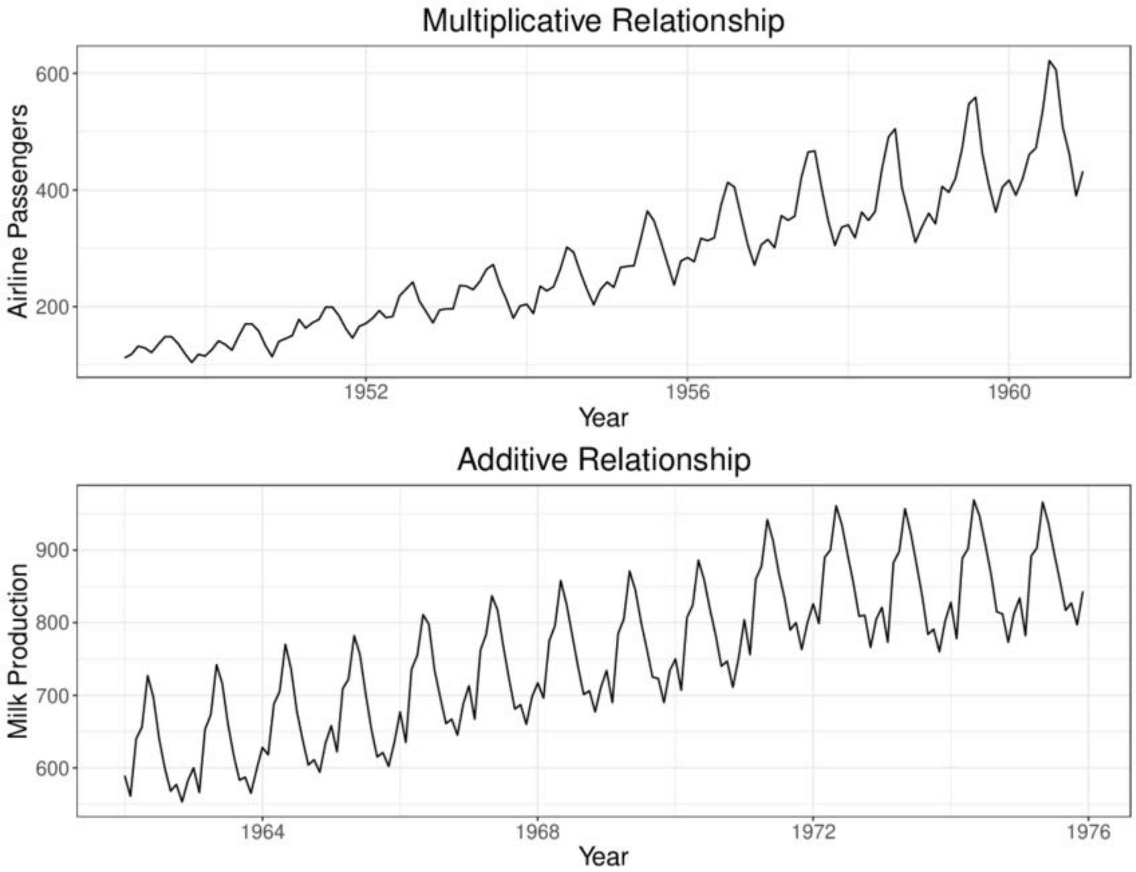
$$y_t = T_t * S_t * R_t$$

- we can often go from multiplicative to additive relation by using **log-transform:**

$$\log(y_t) = \log(T_t) + \log(S_t) + \log(R_t)$$

- additive models are often more stable, i.e. numerically + reasonable predictions
- we will generally assume additivity

Additive vs multiplicative decompositions

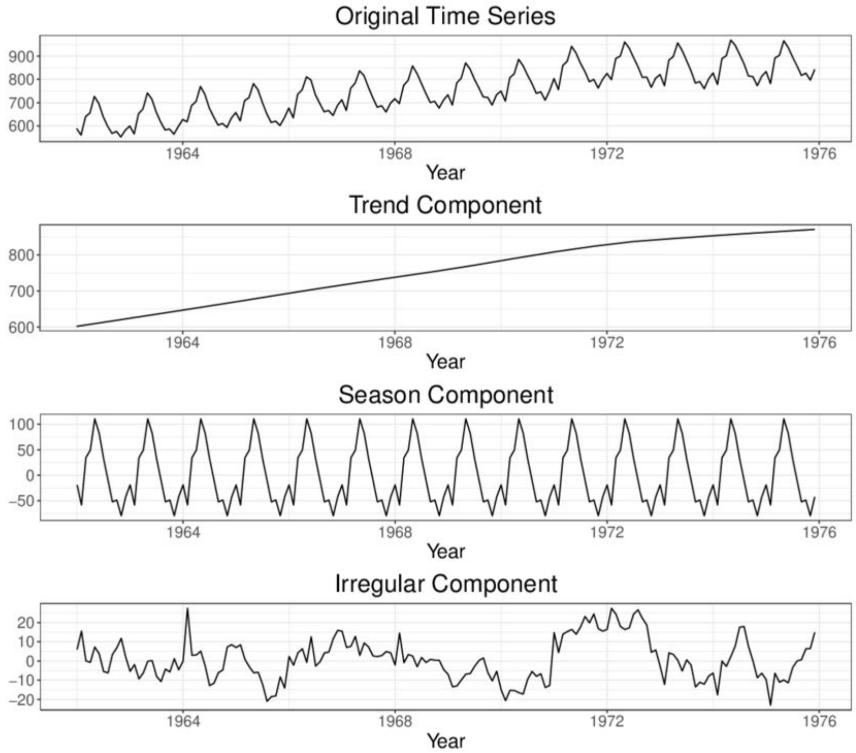


https://www.researchgate.net/publication/348592737_Automated_Hybrid_Time_Series_Forecasting_Design_Benchmarking_and_Use_Cases

STL decomposition

- **Seasonal and Trend decomposition using LOESS (STL)**
- decomposition algorithm, based on weighted regression
- assumes additive model:

  trend + seasonality + remainder
- many variants and extensions
- **pros:** simple, intuitive, robust variant (robust to outliers)
- **cons:** single seasonality, only additive model



https://www.researchgate.net/publication/348592737_Automated_Hybrid_Time_Series_Forecasting_Design_Benchmarking_and_Use_Cases

Decomposition - additional resources

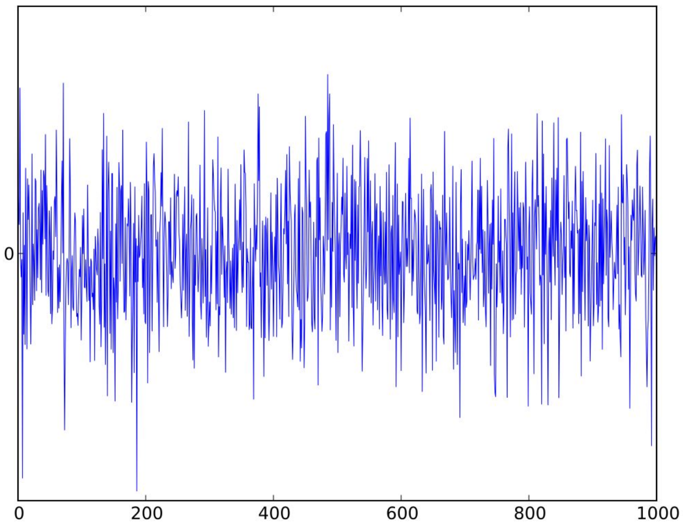
- FPP: Time series patterns, Time series components, STL decomposition
- Statistics Canada - Trend-cycle estimates
- STL Algorithm Explained
- "STL: A Seasonal-Trend Decomposition Procedure Based on Loess" R. Cleveland et al.
- "MSTL: A Seasonal-Trend Decomposition Algorithm for Time Series with Multiple Seasonal Patterns" K. Bandara et al.
- "STR: Seasonal-Trend Decomposition Using Regression" A. Dokumentov, R. Hyndman

Stationarity, variance and transformations

Strong stationarity

- **strong stationarity** - time series values have the same distribution for a given time window, no matter the time
- joint probability distribution of $y_{t_1}, \ldots, y_{t_1+\tau}$ and $y_{t_2}, \ldots, y_{t_2+\tau}$ is the same for all t_1 and t_2
- **unrealistic**, data always changes, but useful for theory
- strongly stationary process - **Gaussian white noise**

$$y_t \sim \mathcal{N}(0, \sigma^2)$$

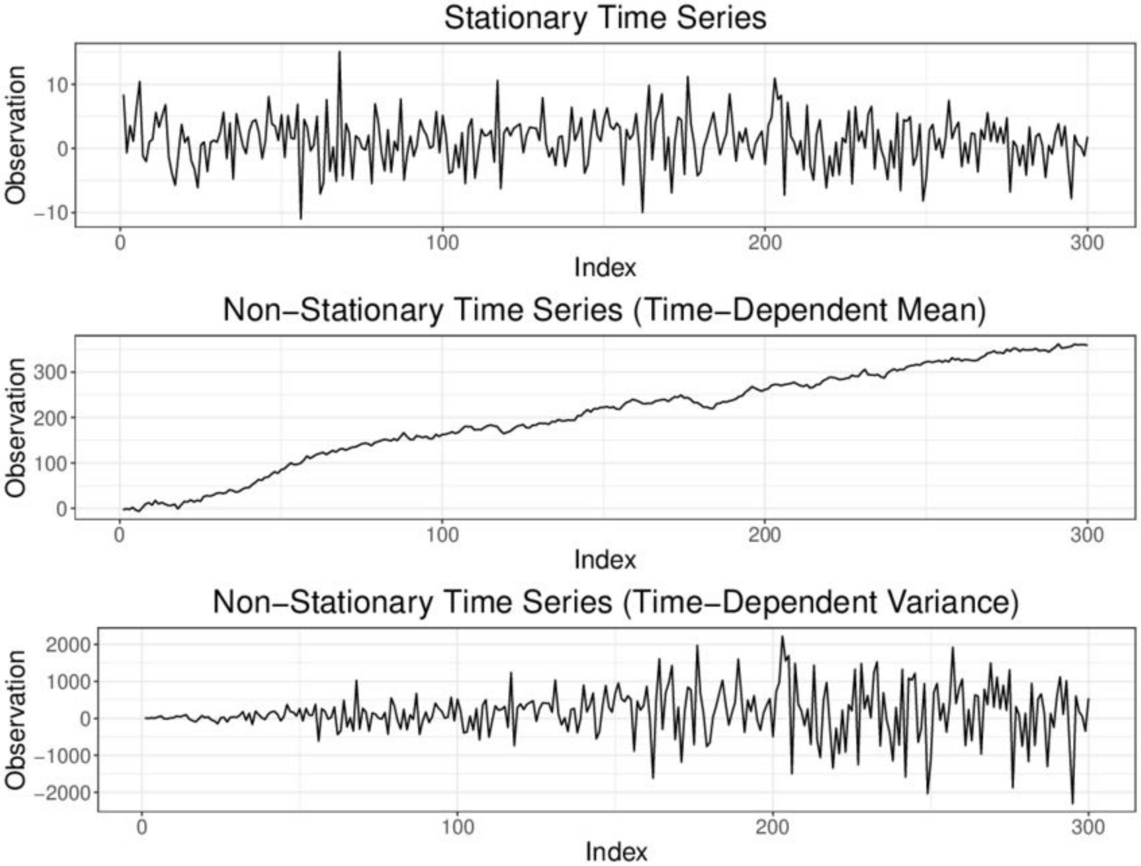


https://en.wikipedia.org/wiki/White_noise

Stationarity

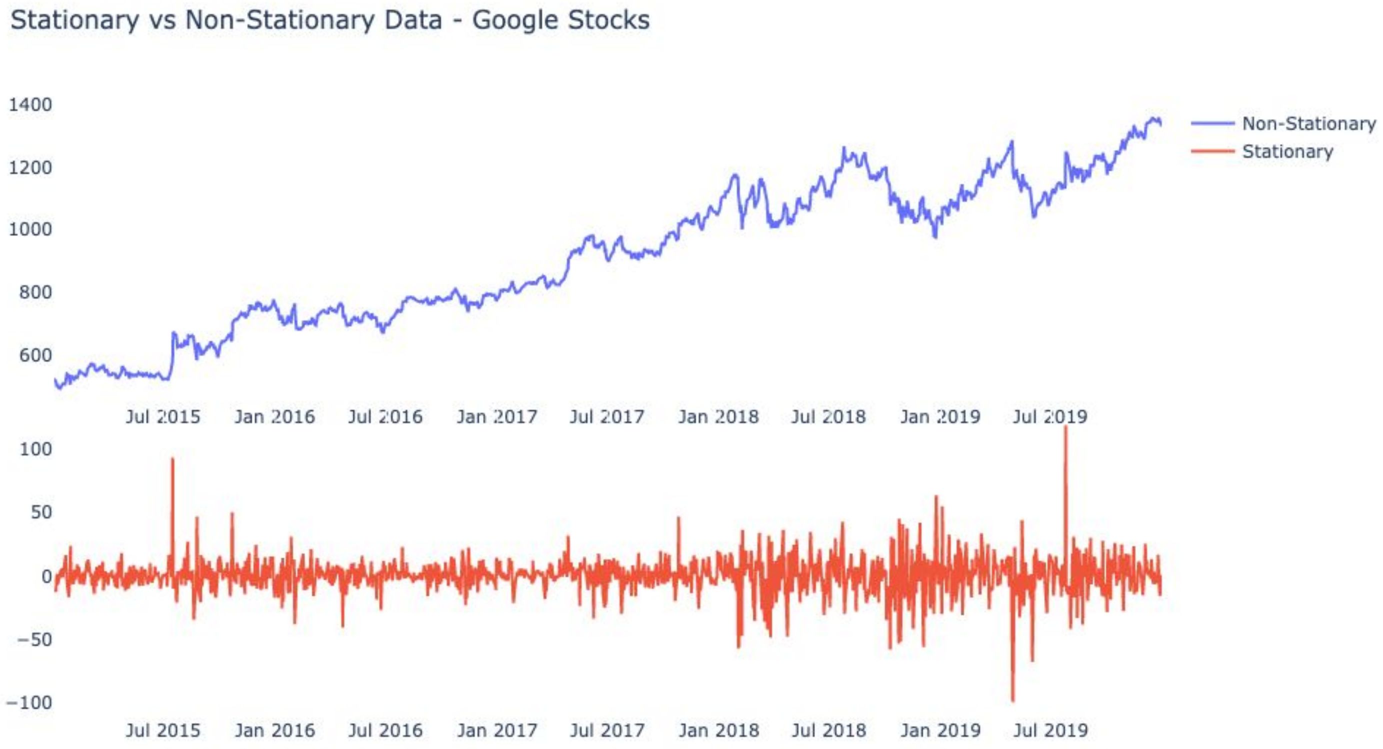
- **weak (wide-sense) stationarity** - mean and autocovariance are constant, and variance is finite
- often called just "stationarity", **good enough** in practice
- **what this means:**
  - when plotted, time series look similar in all places
  - no trend, no seasonality, finite and stable values
  - reasonably similar statistical properties everywhere
- **easier to forecast** stationary data, since training and testing data are similar
- for variance, we want **homoscedasticity**, i.e. finite and time-independent variance

Stationarity



https://www.researchgate.net/publication/348592737_Automated_Hybrid_Time_Series_Forecasting_Design_Benchmarking_and_Use_Cases

Stationarity



https://blogs.cisco.com/analytics-automation/arima2

Autocovariance

- **autocovariance** - covariance of time series with itself, at a given **time lag** *k*

$$\gamma(t, t-k) = \mathrm{Cov}[y_t, y_{t-k}] = \mathbb{E}[(y_t - \mu)(y_{t-k} - \mu)]$$

- how much past and future values of time series are related
- sample autocovariance:

$$\hat{\gamma}(t, t-k) = \frac{1}{T} \sum_{t=k+1}^{T} (y_t - \mu)(y_{t-k} - \mu)$$

- **autocorrelation** = normalized autocovariance (divided by standard deviation)
- high autocovariance = future values directly depend on past values *k* lags before

Stationarity testing

- **manually:**
  - graphical plots, e.g. ACF, PACF
  - shockingly, many people stop here, and cite "subjectivity" as a problem in time series forecasting…
- **statistical tests:**
  - standard ML solution
  - Kwiatkowski–Phillips–Schmidt–Shin (KPSS) - typically best in practice
  - Augmented Dickey-Fuller (ADF) - traditional, but overly pessimistic when automated
  - additional resources: link 1, link 2

Stationarization

- making time series **more stationary**
- best-effort tools, often won't work really well, but good enough

Problem	Standard solution
Trend	Differencing
Seasonality	Seasonal differencing
Unstable variance and/or autocovariance	Log, sqrt, Box-Cox transformations

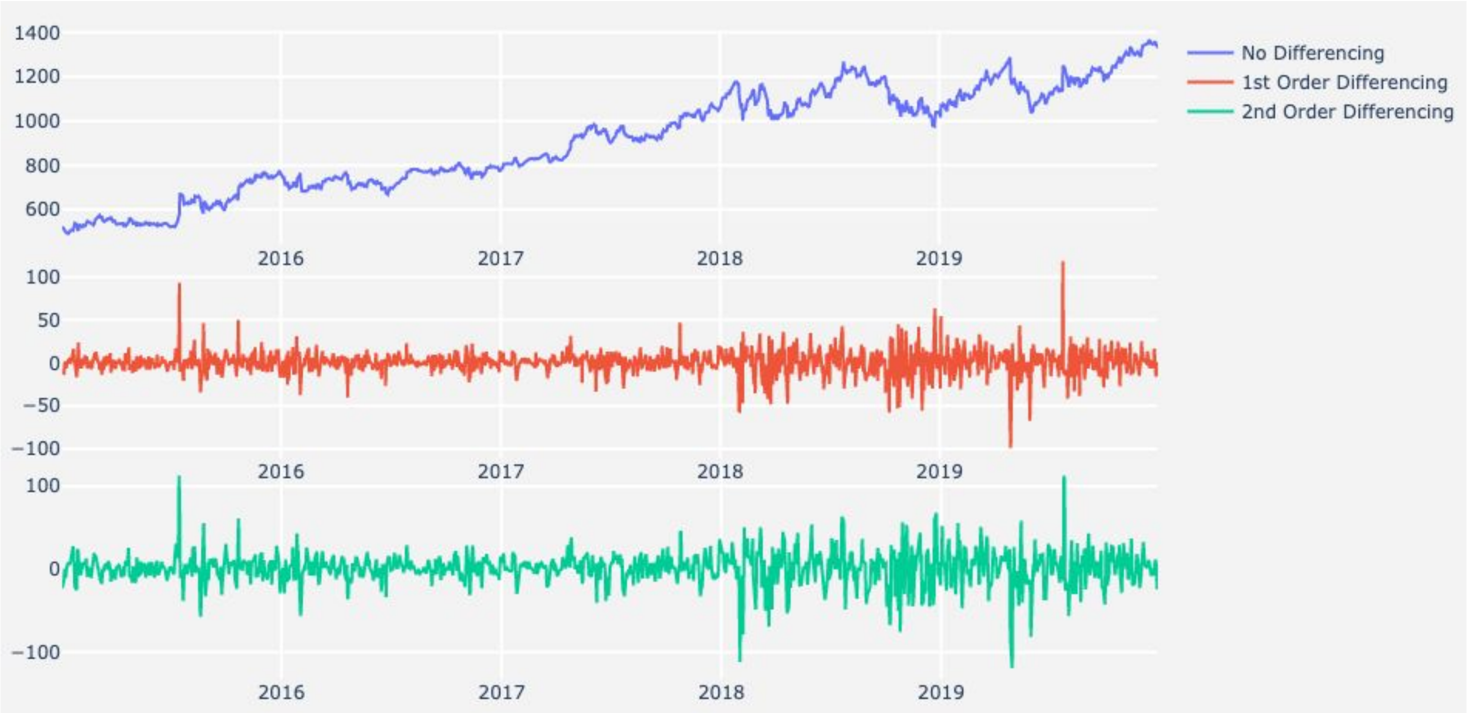
Differencing

- **differencing** - changing values into change values (deltas) between time steps:

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

- detrends, i.e. removes trend, makes time series **mean-stationary**
- almost always 1-2 times is enough
- note the similarity to **derivatives** - this is a crude (1st order), discrete approximation of the function derivative, i.e. rate of change
- **does not always work** - some data just cannot be made easily mean-stationary, but this typically results in a good enough result
- trained models forecast differences - remember to **invert** during prediction!
- need to remember the first value from the original series for inversion

Differencing



https://blogs.cisco.com/analytics-automation/arima2

Automatic differencing algorithm

- repeat in a loop:
    - test stationarity with KPSS test
    - if stationary, break
    - else, take a difference $y'_t = y_t - y_{t-1}$

- **differencing order** - number of differences
- marked as *d*
- typically we limit d_{max} in this loop, often 2

Seasonal differencing

- **seasonal differencing** - removes seasonality of order *m*:

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

- typically we first do seasonal differencing, then regular differencing
- statistical tests
  - Osborn-Chui-Smith-Birchenhall (OCSB)
  - Canova-Hansen (CH)
- automated just like regular differencing

Stationarity - additional resources

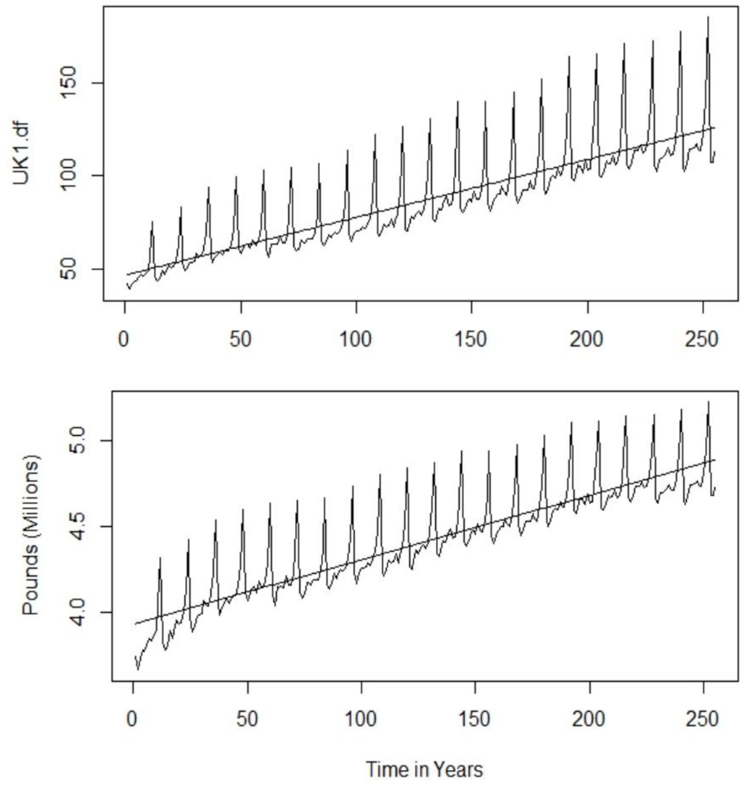
- we explicitly omit unit roots for simplicity, but they are interesting additional material
- "Nonstationary time series transformation methods: An experimental review" E. Ogasawara
- ritvikmath "Unit Roots : Time Series Talk"
- Lazarski Open Courses "Time Series Econometrics: Unit root testing"
- "An Introduction to Stationarity and Unit Roots in Time Series Analysis"

Variance-stabilizing transformations

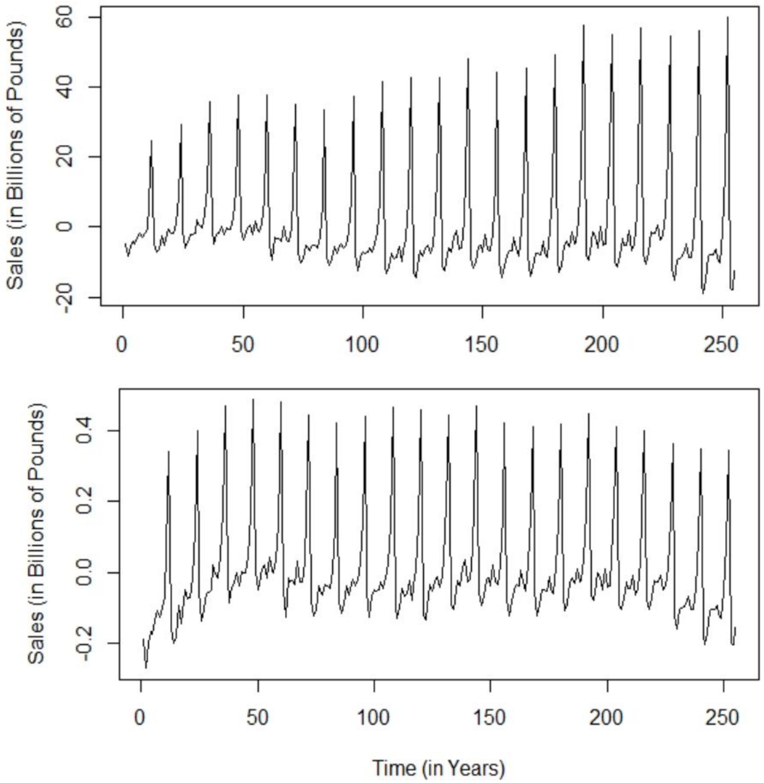
- **variance-stabilizing transforms** make variance more regular, smoothing values and removing outliers
- typical: log, sqrt, inverse, exp
- sometimes called "variance stationarization"
- **advantages:**
  - reduce heteroscedasticity (time-dependence of variance)
  - values distribution closer to Gaussian (normal)
  - change multiplicative relations to additive (log-transform)
  - greater numerical stability, smaller absolute values
- **problem:** how to choose the transformation?

Variance-stabilizing transformations

Original data vs log-transform



Detrended data vs log-transform



https://stats.stackexchange.com/questions/508475/stabilizing-the-variation-in-a-time-series

Box-Cox transformation

- **Box-Cox transformation (power transform)** is a generalized formula:

$$y_{Box-Cox} = \begin{cases} \frac{y^{\lambda}-1}{\lambda}, & \text{if } \lambda \neq 0 \\ \ln(y), & \text{if } \lambda = 0 \end{cases}$$

- covers many cases, e.g. log (λ=0), sqrt (λ=0.5), no transform (λ=1)
- λ is **estimated** from data, typically via maximum likelihood estimation (MLE)

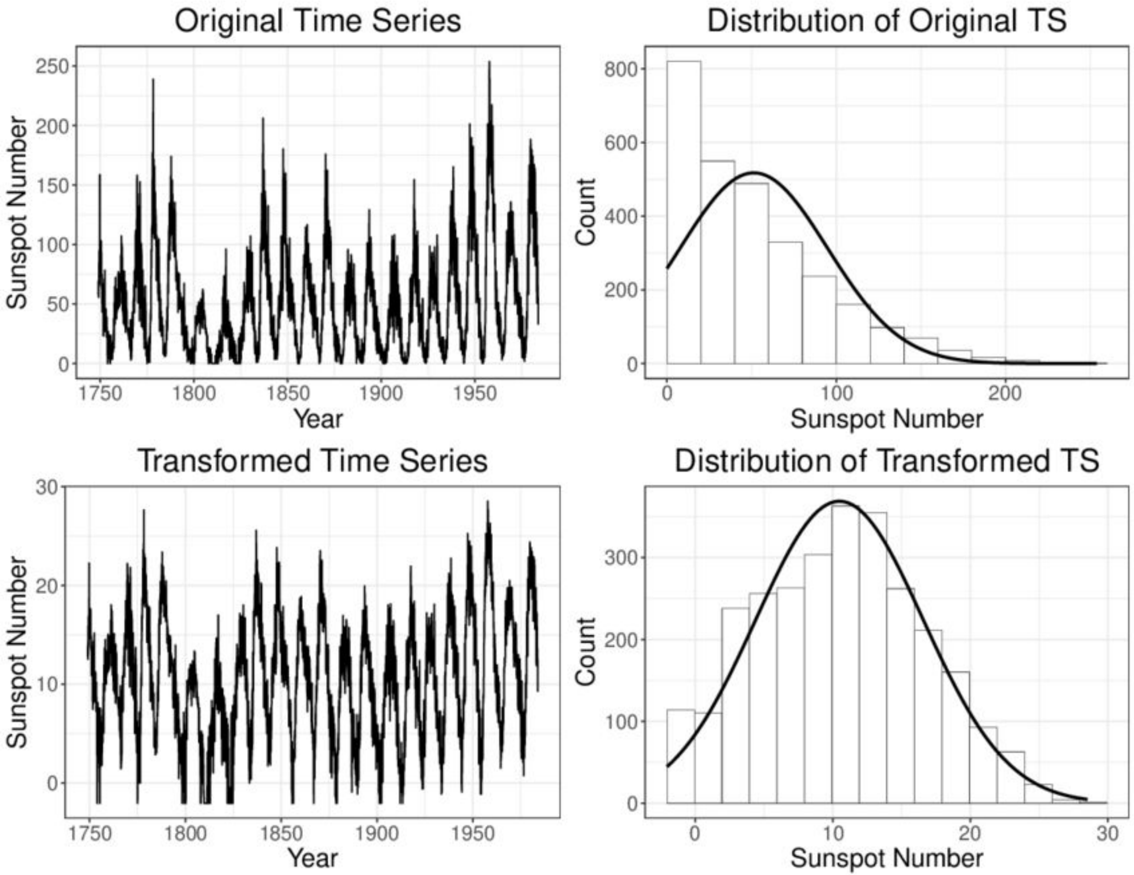
Pros:

- automated
- can learn new transforms, e.g. λ=0.25
- often the best results (but not always)

Cons:

- requires positive values (but adding small value like 1e-5 works)
- does not incorporate seasonality

Box-Cox transformation



https://www.researchgate.net/publication/348592737_Automated_Hybrid_Time_Series_Forecasting_Design_Benchmarking_and_Use_Cases

V.s. transformations - additional resources

- CrossValidated - When (and why) should you take the log of a distribution (of numbers)?
- CrossValidated - BOX-COX TRANSFORMATION always stabilize variance
- P. Li "Box-Cox Transformations: An Overview"

- Yeo-Johnson transformation - allows negative values
- Guerrero's method - incorporates seasonality
- Transform Data with Hyperbolic Sine - various methods for time series with negative values
- "Variance stabilizing transformations for electricity spot price forecasting" B. Uniejewski, R. Weron, F. Ziel

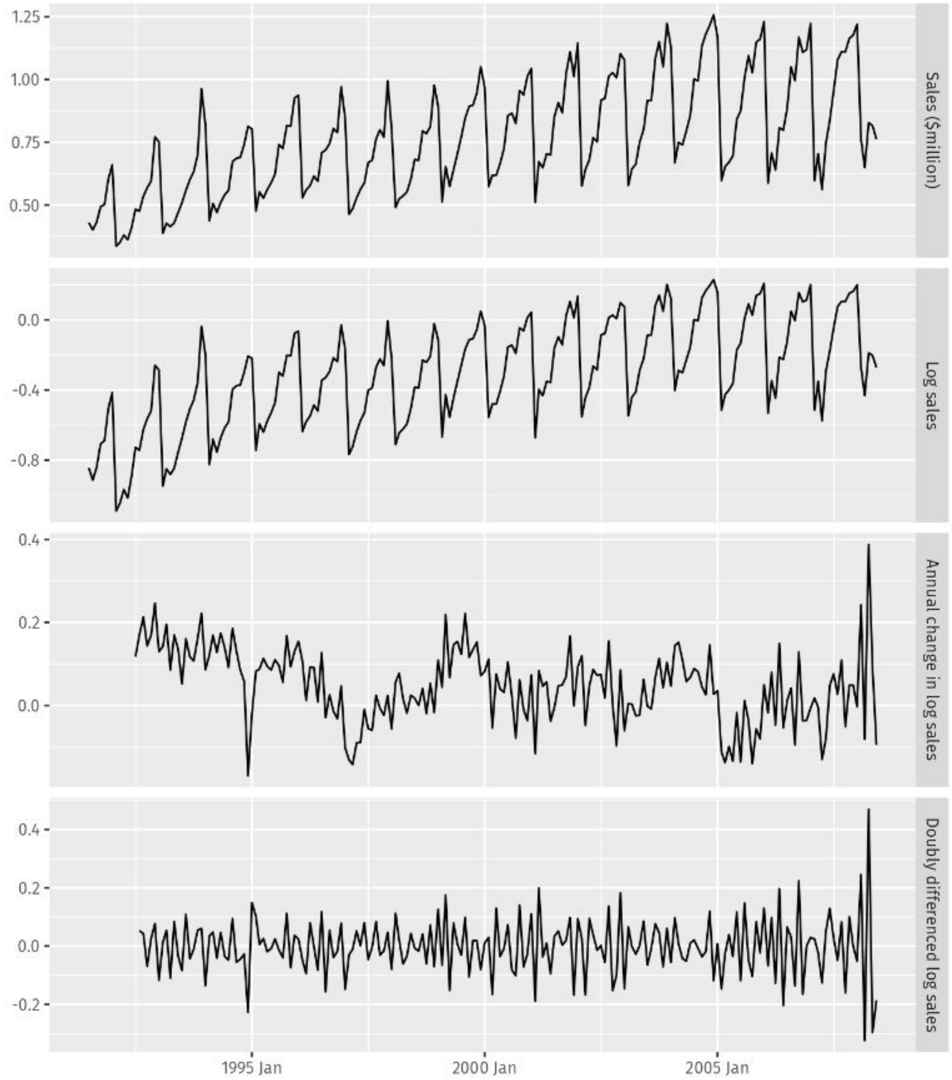
Transformations example

Data

-> log

-> seasonal differencing

-> differencing



https://otexts.com/fpp3/stationarity.html

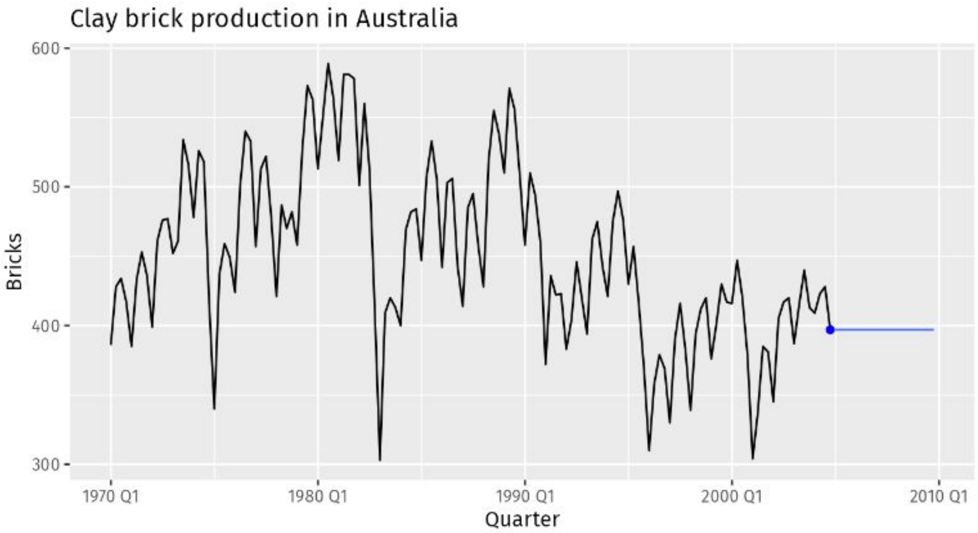
Baselines

Predict last

- forecast:

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

- often works very well in economy and finance
- the best possible forecast if the data is a **random walk**
- often called just "naive" or "naive forecast"



https://otexts.com/fpp3/simple-methods.html

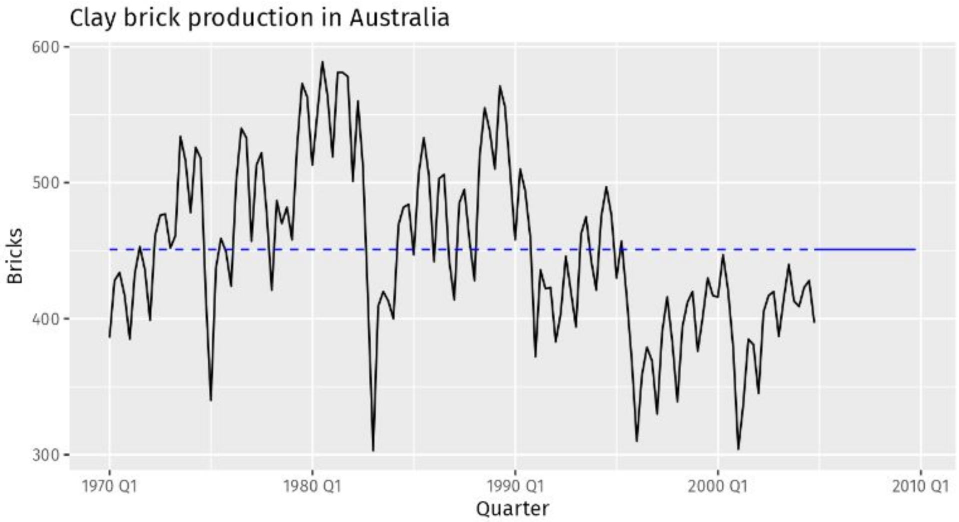
Average, median

- forecast:

$$\hat{y}_{T+h|T} = \text{avg}(y) = \frac{y_1 + y_2 + ... + y_T}{T}$$

$$\hat{y}_{T+h|T} = \text{median}(y)$$

- works well if the data is mainly trend and seasonality
- average is the optimal forecast for **white noise**
- median is rare in literature, but robust and gives good results



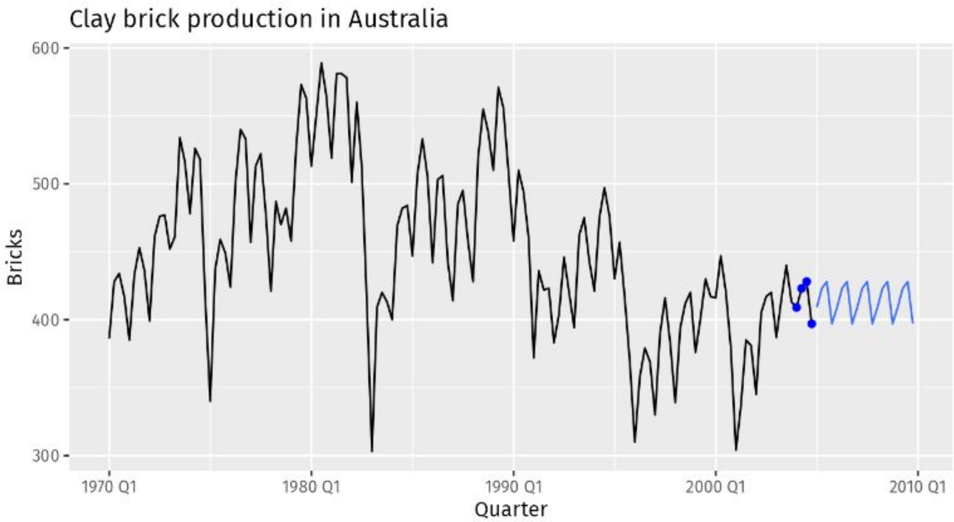
https://otexts.com/fpp3/simple-methods.html

Naive seasonal

- forecast:

$$\hat{y}_{T+h|T} = y_{T+h-m\lfloor \frac{h-1}{m} \rfloor}$$

- copy the value of the same moment from the last season, e.g.:
  - monthly data, yearly seasonality
  - assume the same sales, as the same month the last year
- works well for strongly seasonal data, e.g. sales, demand



https://otexts.com/fpp3/simple-methods.html

Why baselines are important

["Forecast evaluation for data scientists: common pitfalls and best practices" H. Hewamalage et al.]

- fair evaluation - baseline not present in the paper
- complex Autoformer model outperformed by a simple naive forecast…
- even for long time horizons!

Table 3 Results from the naïve forecast and the Autoformer model on the exchange rate dataset

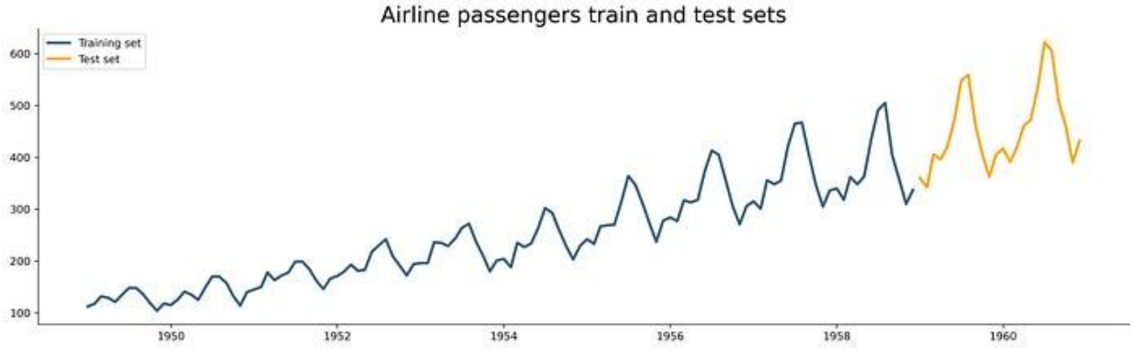
Horizon	Naïve		Autoformer (Rerun)		Autoformer (Original Paper)	
	MAE	MSE	MAE	MSE	MAE	MSE
96	**0.192**	**0.078**	0.279	0.149	0.323	0.197
192	**0.282**	**0.158**	0.399	0.299	0.369	0.300
336	**0.388**	**0.287**	0.504	0.460	0.524	0.509
720	**0.694**	**0.817**	0.963	1.552	0.941	1.447

Best models shown in boldface font

Forecasts evaluation

Data splitting

- **never** split randomly - that would be data leakage and using future data
- **time split / chronological split:**
    - older data - training, newest data - testing
    - final model is retrained on entire data
    - always useful when we have time information (not only for time series)



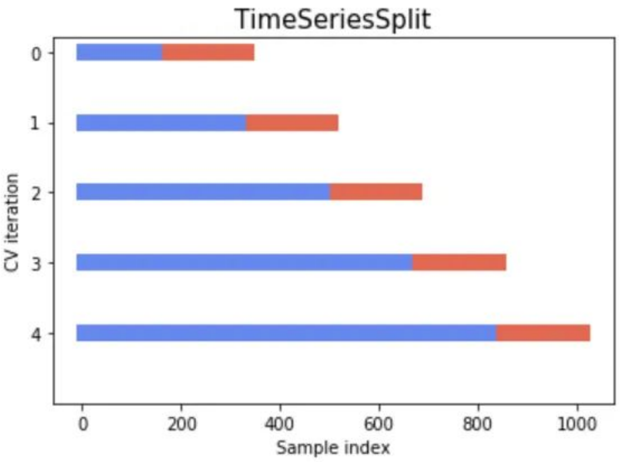
https://freedium.cfd/https://towardsdatascience.com/time-series-from-scratch-train-test-splits-and-evaluation-metrics-4fd654de1b37

Time split - problems

- **problem 1:**
  - assume 80-20% split, with test set being 2 years
  - we retrain models once a week
  - clients typically make forecasts with 1-3 months horizon
  - longer horizonts = less precise forecasts
  - evaluating directly on the whole 2 years would be unrealistic!
- **problem 2:**
  - we still want to use cross-validation, e.g. for hyperparameter tuning
  - but we need to take time into consideration

Time series CV

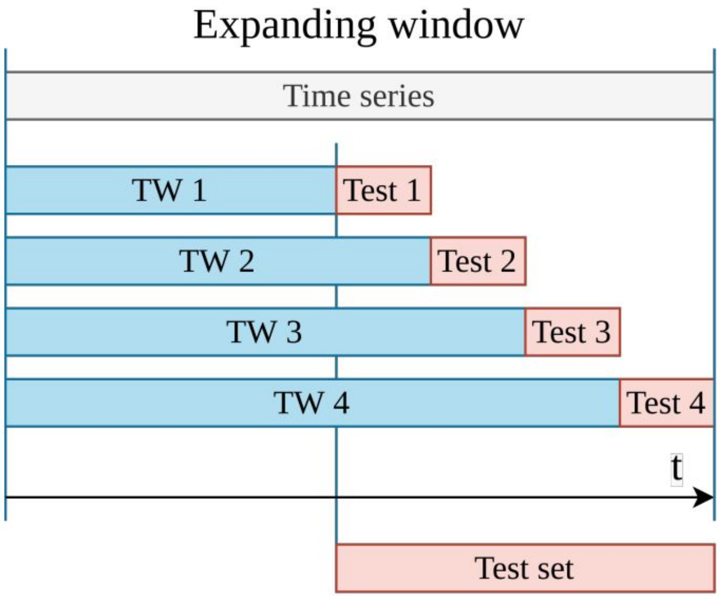
- time series CV has 2 main variants: k-fold split and expanding window
- **k-fold split:**
  - divide data into k equally sized parts (folds)
  - train on first k folds, evaluate on k+1
  - commonly used for selecting hyperparameters (validation)
- **problems:**
  - data-hungry neural models underperform initially
  - single step - forces equal retraining frequency and forecasting horizon



https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4

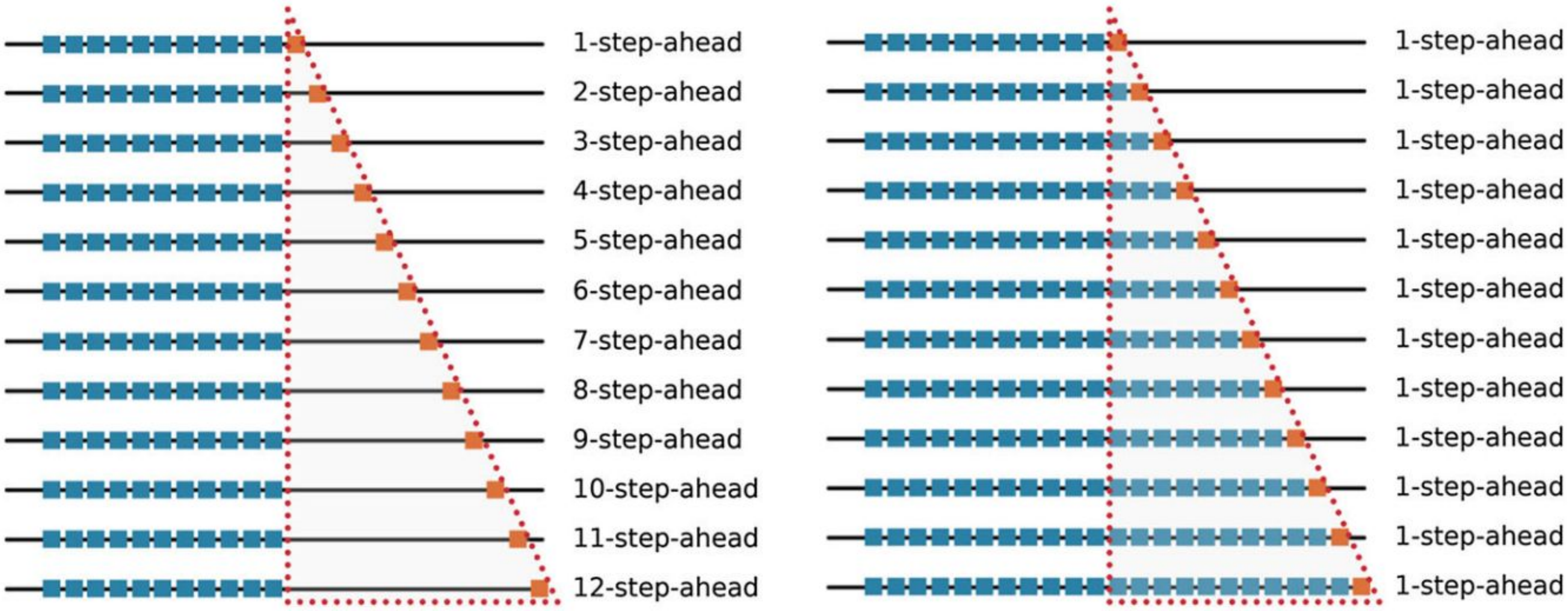
Time series CV

- **expanding window** uses configurable steps instead of k equally-sized parts
- 3 parameters: initial data size, retraining step, forecasting horizon
- **expanding window:**
  - start with given percentage of data
  - forecast and evaluate with given horizon
  - go forward a given number of steps and retrain
- **pros:** elastic and realistic, great for testing
- **cons:** computationally expensive



https://www.researchgate.net/figure/Sliding-vs-expanding-window-strategies_fig2_351885630

Train-test vs expanding window



"Forecast evaluation for data scientists: common pitfalls and best practices" H. Hewamalage et al.

Quality metrics

- **classic regression measures:**
  - RMSE, MAE etc.
  - well-known, easily understandable
  - relatively sensitive to outliers
  - scale-dependent (can be good or bad)
  - can use median (MdAE) or geometric mean (GMAE) instead of regular mean
- **measures relative to true y:**
  - MAPE (Mean Absolute Percentage Error), SMAPE (Symmetric MAPE)
  - measure error in percentage, relative to true y
  - many problems, e.g. numerically unstable, biased towards overforecast
  - you **should never** use those

Problems with MAPE - additional resources

- Open Forecasting "Avoid using MAPE!"
- M. Ganguly "Pitfalls of MAPE as a forecast accuracy metric"
- CrossValidated - What are the shortcomings of the Mean Absolute Percentage Error (MAPE)?
- CrossValidated - Is MAPE a good error measurement statistic? And what alternatives are there?

Mean Absolute Scaled Error (MASE)

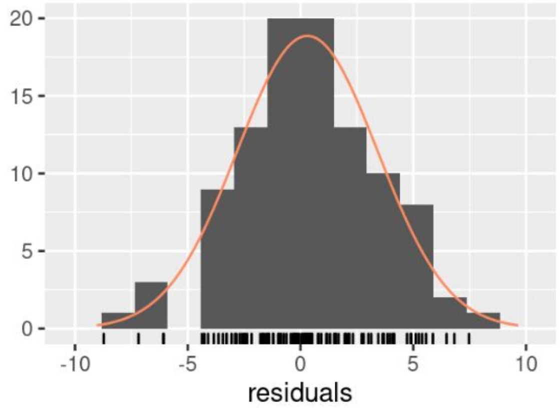
- **forecasting quality measure**, defined as MAE on test set, divided by MAE of naive 1-step forecast on the training set (in-sample error)

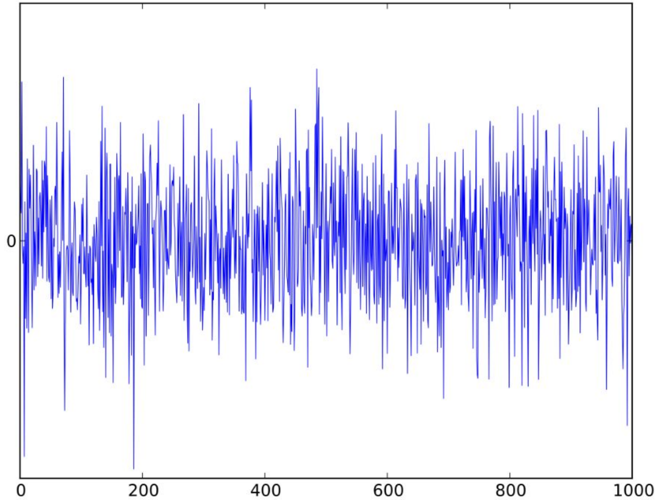
$$MASE(y, \hat{y}) = \frac{MAE_{test}(y, \hat{y})}{\frac{1}{T-1} \sum_{i=1}^{T} |y_i - y_{i-1}|}$$

- has a lot of nice features:
  - **scale-invariant** - good forecasts have values in range [0, 1)
  - **interpretable** - value below 1 means that forecast is better than naive baseline
  - **symmetric** - equally penalizes under- and overforecast, and works well both for small and large values
  - **numerically stable** - no problems with zero / small values
- the only metric you need in addition to scale-dependent ones

Residuals analysis

- **residual error** of a model is: $\epsilon_t = y_t - \hat{y}_t$
- **assumption 1:**
  - zero-centered, normally-distributed errors:
    $\epsilon_t \sim \mathcal{N}(0,\, \sigma^2)$
  - Shapiro-Wilk or Anderson-Darling test
- **assumption 2:**
  - homoscedastic errors, without autocorrelation
  - Ljung-Box test
- in other words, errors should be a Gaussian white noise
- if false, we have an "incorrect" model, but it can still be very useful





https://robjhyndman.com/hyndsight/forecast8/index.html, https://en.wikipedia.org/wiki/White_noise

Evaluation - additional resources

- FPP: Evaluating point forecast accuracy
- "Forecast evaluation for data scientists: common pitfalls and best practices" H. Hewamalage et al.
- "On the use of cross-validation for time series predictor evaluation" C. Bergmeir, J. M. Benítez

- Wikipedia - Mean absolute scaled error
- CrossValidated - Interpretation of mean absolute scaled error (MASE)
- CrossValidated - Interpretation of scaled error measures

Questions?