Learning with Complex Data

Thomas Moreau

A word about Me

- PhD, tenured researcher at Inria
- https://tommoral.github.io/
- thomas.moreau@inria.fr
- **Research topics:** Optimization, deep learning, unsupervised learning, benchmarks, ML for Physiological and Physical Signals

You will learn…

- About the different types of data
- About method to cast complex data to simple ML tasks
- About the different tasks that can be considered with complex data
- About the caveat of model validation with complex data
- With some case studies with scientific data

Agenda

1. Complex tabular data

    - Encoding complex data.
    - Automated encoding with skrub.

2. Machine learning with signals (time series)

    - What differs with functional data?
    - How to get back to the classical ML framework?

3. Complex tasks with signals (forecasting, event detection)

    - Framing the problem.
    - How to get back to classical ML framework?
    - Model validation and stationarity

01 - Complex tabular data

Recall from yesterday…

- Machine learning aims at predicting a certain quantity y from data X
- For a regression task with $y \in \mathbb{R}$, with a typical linear model:

$$y = \theta^\top X \quad \text{with} \quad \theta, X \in \mathbb{R}^d$$

- But what if $X \notin \mathbb{R}^d$? Some features are categories, dates, strings…
- Or if we want more complex models? Think about polynomials

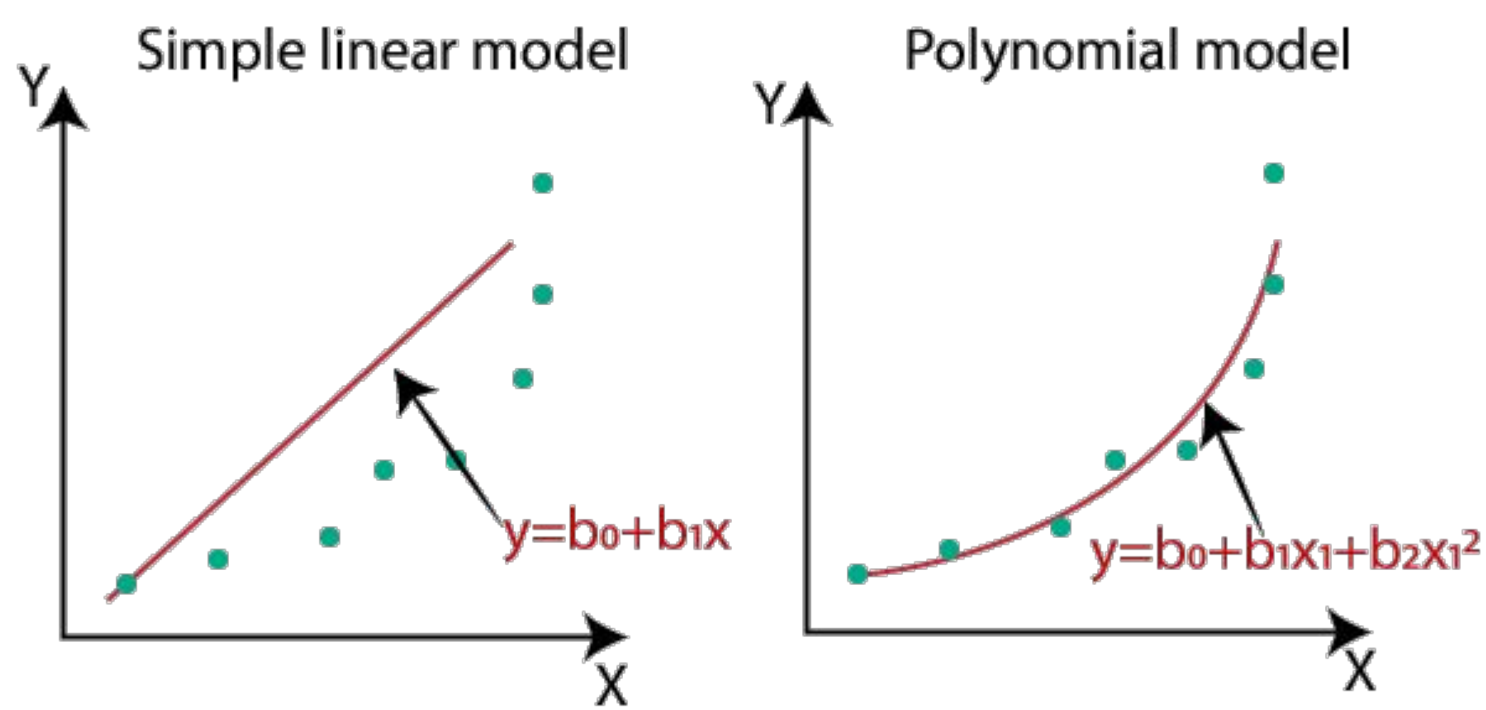
→ **Strategy: features extraction**

embed X as $\widetilde{X} = f(X) \in \mathbb{R}^d$ and learn $y = \theta^\top \widetilde{X}$

Data embedding

- Feature extraction, or data embedding is a process that goes from $X \in \mathcal{X}$ to an euclidean space $\widetilde{X} \in \mathbb{R}^d$.
- Typically, for categories, use Ordinal encoding or One Hot encoding.
- But this can cover more cases: textual data (n-grams), dates, …
- Can also extra *a priori* information: interaction, polynomial features, …

$$\widetilde{X} = \begin{bmatrix} X_i \\ X_i^2 \\ X_i X_j \\ \dots \end{bmatrix}$$



Encoding complex data with fuzzy logic

01-complex-tabular-data.ipynb

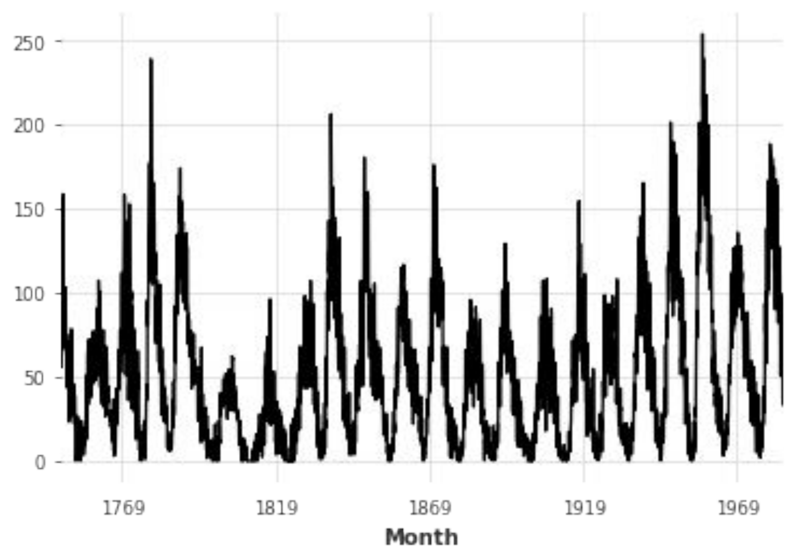
02 - Working with signals

What are the different types of signals?

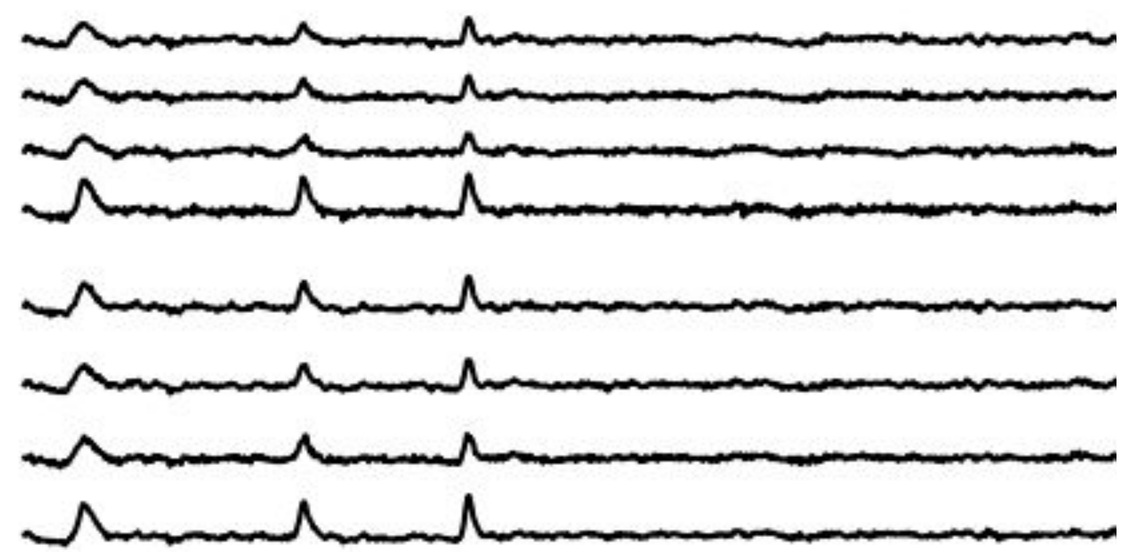
Can you give examples of cases where you would need to process signals with your applications?

Univariate vs Multivariate time-series

Sun spots data



EEG data



$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{t-1} \\ x_t \end{bmatrix}$$

Time

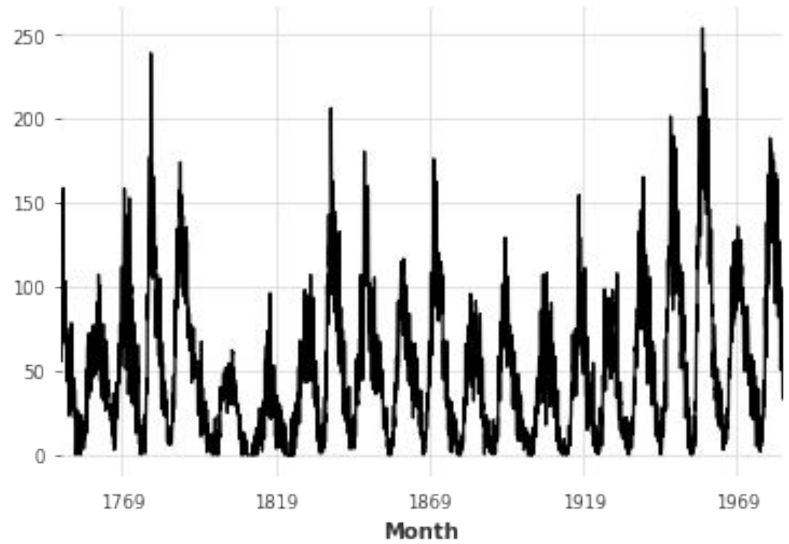
Features

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^p \\ x_2^1 & x_2^2 & \dots & x_2^p \\ \vdots & \dots & \dots & \vdots \\ x_{t-1}^1 & x_{t-1}^2 & \dots & x_{t-1}^p \\ x_t^1 & x_t^2 & \dots & x_t^p \end{bmatrix}$$

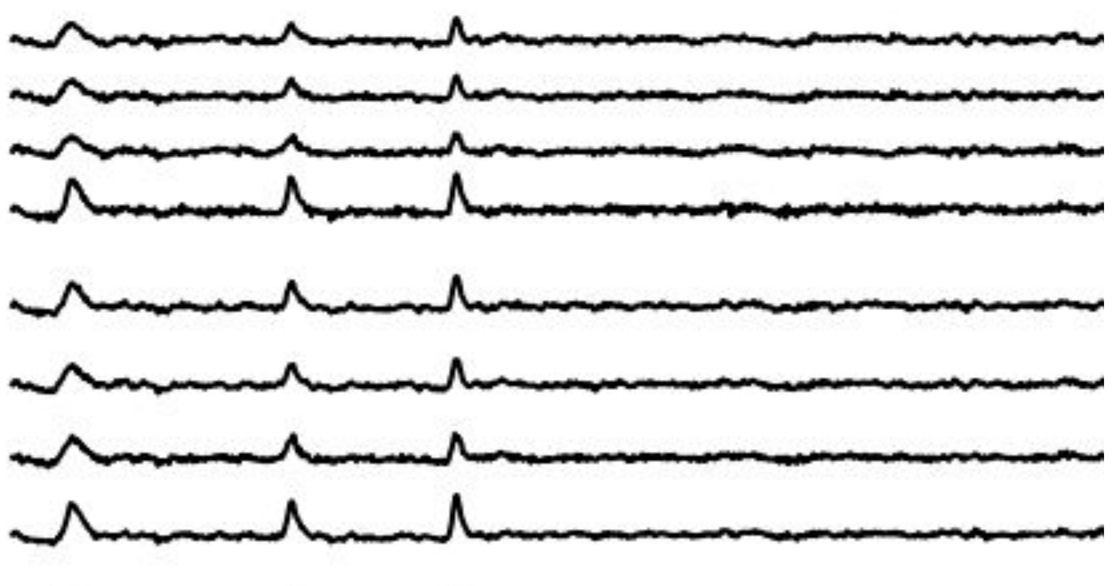
Time

Different flavors of signals

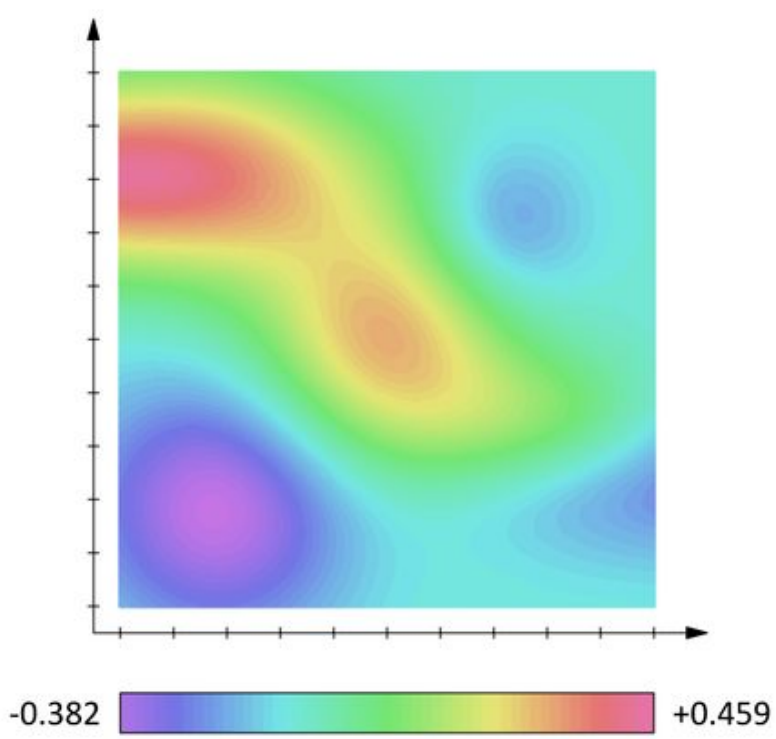
Sunspots data



EEG data

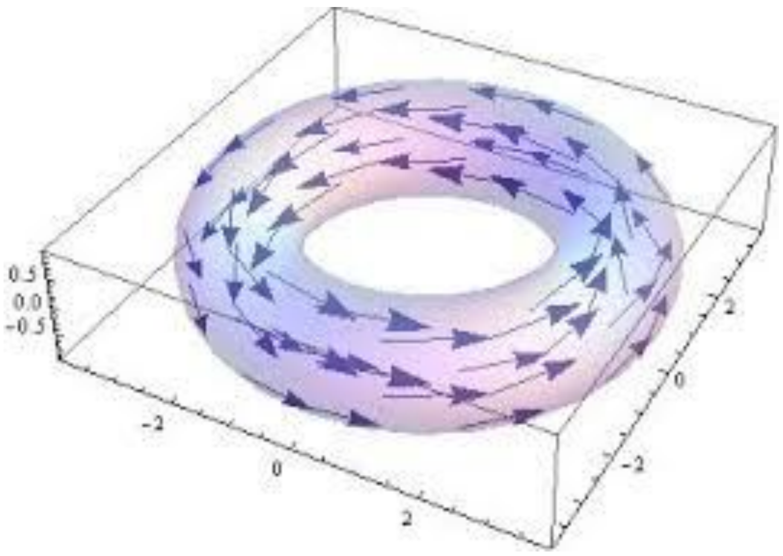


Temperature field



Images

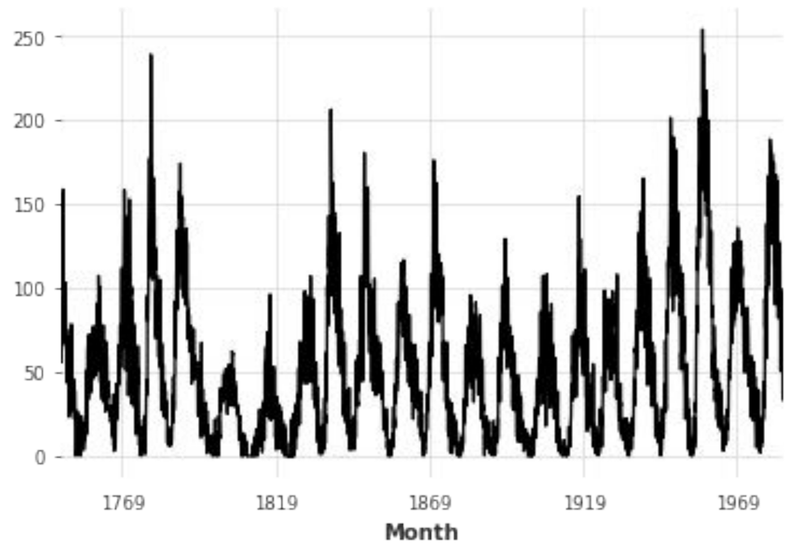
Tokamak vector field



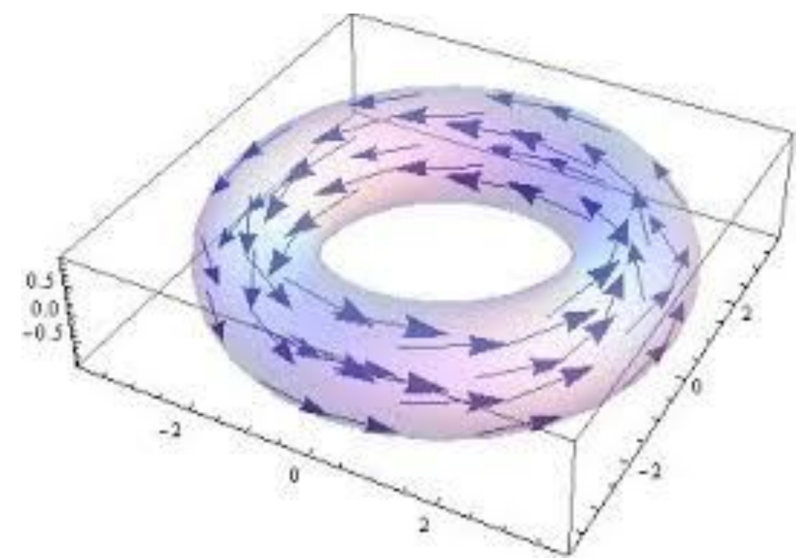
How do you define a signal?

$$X_t = f(t) \in \mathbb{R}^d \quad \text{for} \quad t \in \Omega$$

Sunspots data



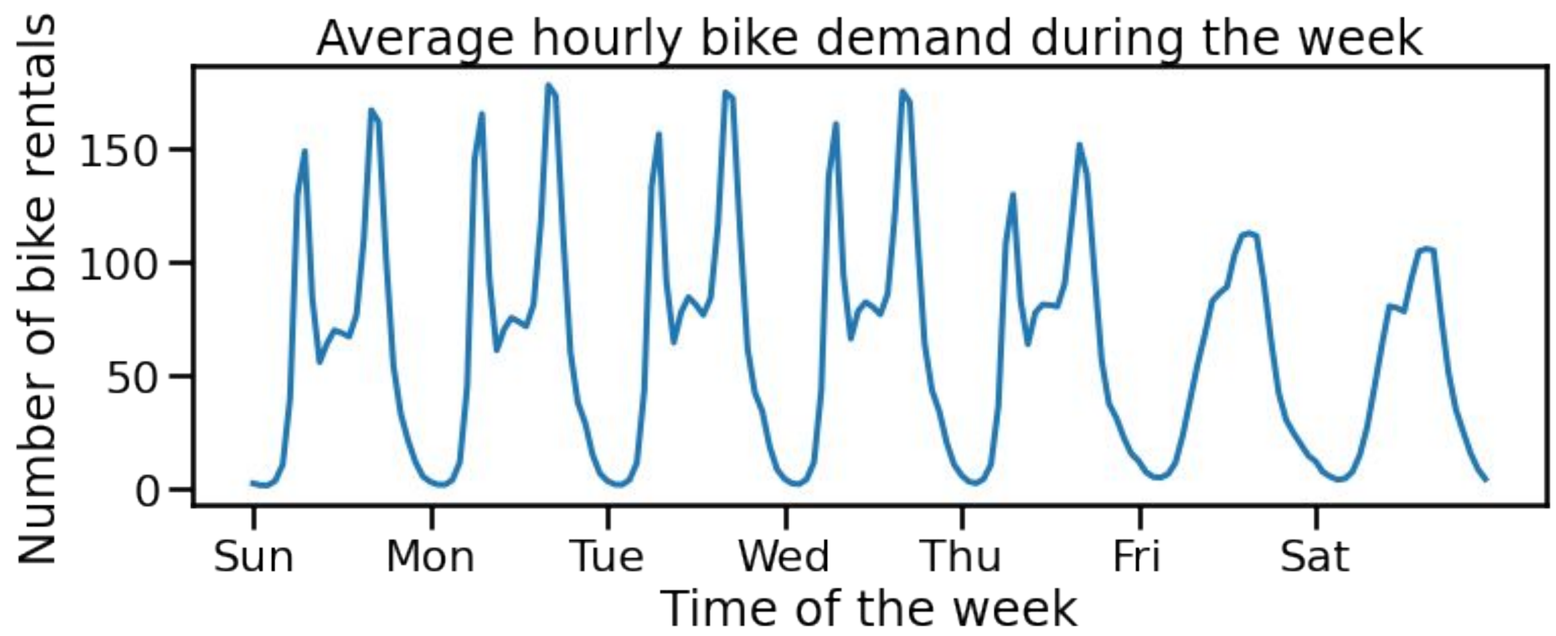
Tokamak
vector field



For univariate time-series, Ω is a time segment and $d = 1$

For vector-fields, Ω is a the domain of definition 3D + the time.

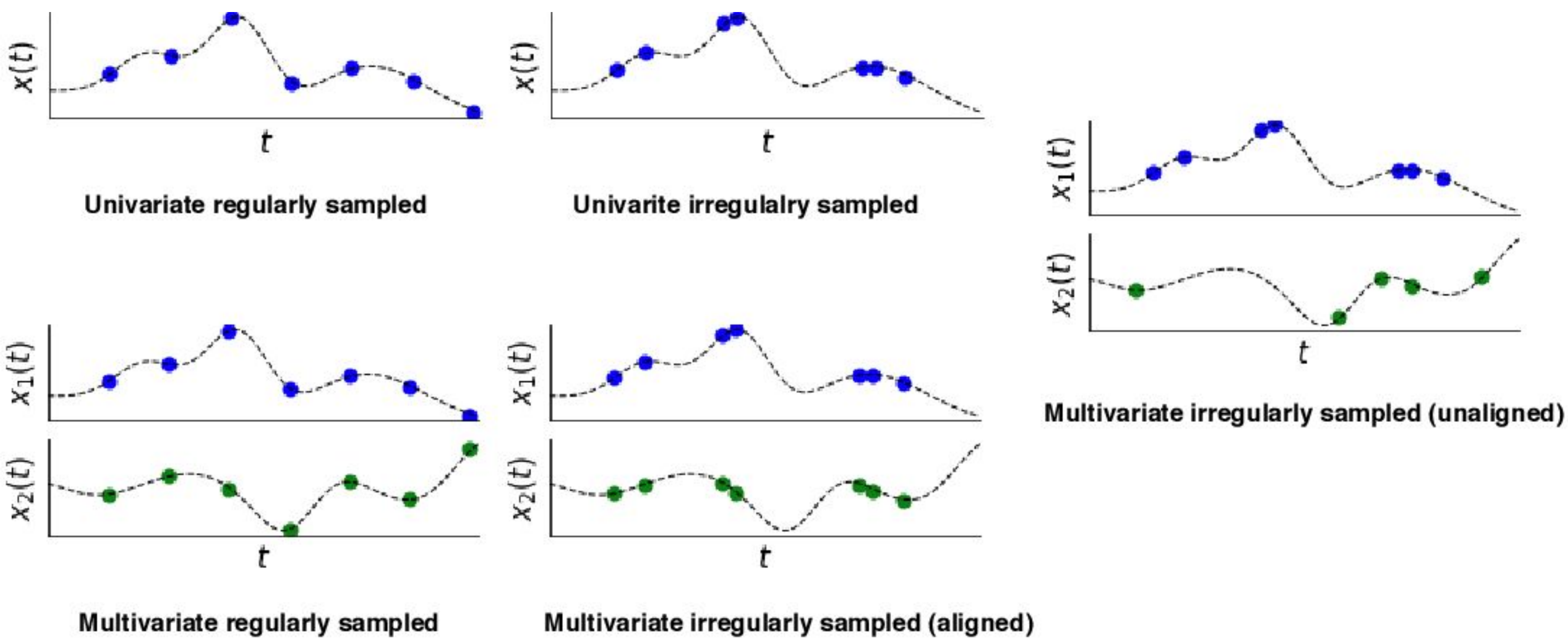
Signals with covariates



$$X = \text{concat}\left(\underbrace{\begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^p \\ x_2^1 & x_2^2 & \dots & x_2^p \\ \vdots & \dots & \dots & \vdots \\ x_t^1 & x_t^2 & \dots & x_t^p \end{bmatrix}}_{\text{time-series}}, \underbrace{\begin{bmatrix} z_1^1 & \dots & z_1^q \\ z_2^1 & \dots & z_2^q \\ \vdots & \dots & \vdots \\ z_t^1 & \dots & z_t^q \end{bmatrix}}_{\text{covariates}} \right)$$

(e.g. holidays, weather etc.)

Regularly vs Irregularly sampled signals



- Regular grids can also have missing values.
- Sometimes, irregular signals can be resampled (NUFFT)

Learning to look at Signals

00-times_series_visualization.ipynb

What are time series problems

- Classification (e.g. brain computer interfaces, diagnosis from physiological signals like ECG)
- Regression (e.g. brain age prediction from EEG)
- Forecasting (e.g. sales & energy consumption, online user traffic, weather forecast, stock market)
- Anomaly/Event detection (e.g. factory monitoring)
- Segmentation / change point detection (e.g. detecting changes on user demand)
- Survival analysis (churn, attrition, …)

Focus on signal classification

- We have N signals $X^{(i)}$ and we want to predict $y^{(i)}$.

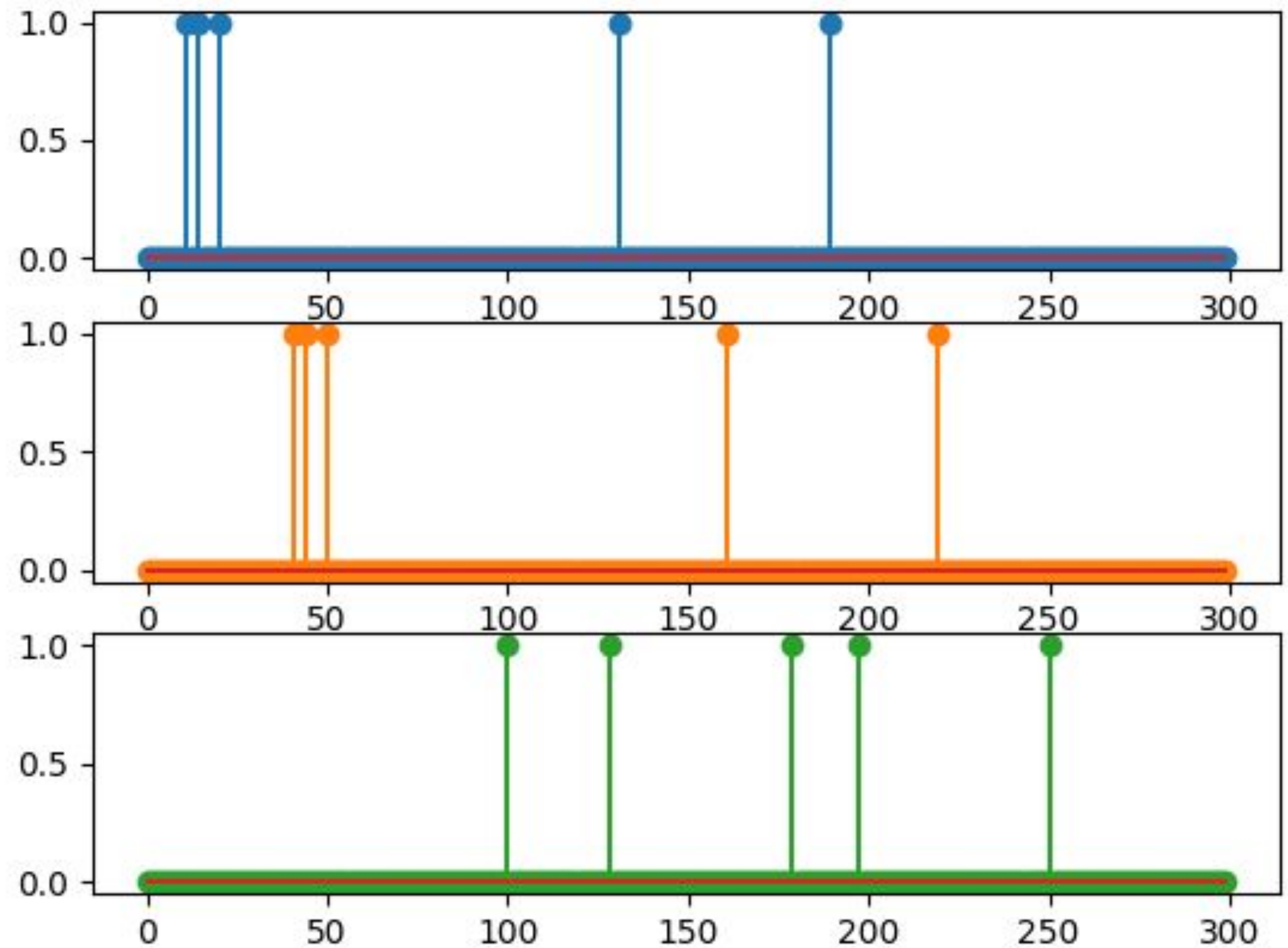
- Consider the full signal to make the prediction.

Problems

- For features, there is a notion of distance between features: we need to account for this.
- Data are "functional data": dedicated metrics, notion of scale, spectral content, patterns, different lengths/scales
- Generic APIs (like scikit-learn) are much harder to design
- The ecosystem is a lot more scattered / messy…

On the notion of alignment

Which signal is closer to the blue one?



They are at the same Euclidean distance!

Moreover, each $X^{(i)}$ can have its own domain Ω_i, with potentially different sizes.

On the notion of alignment

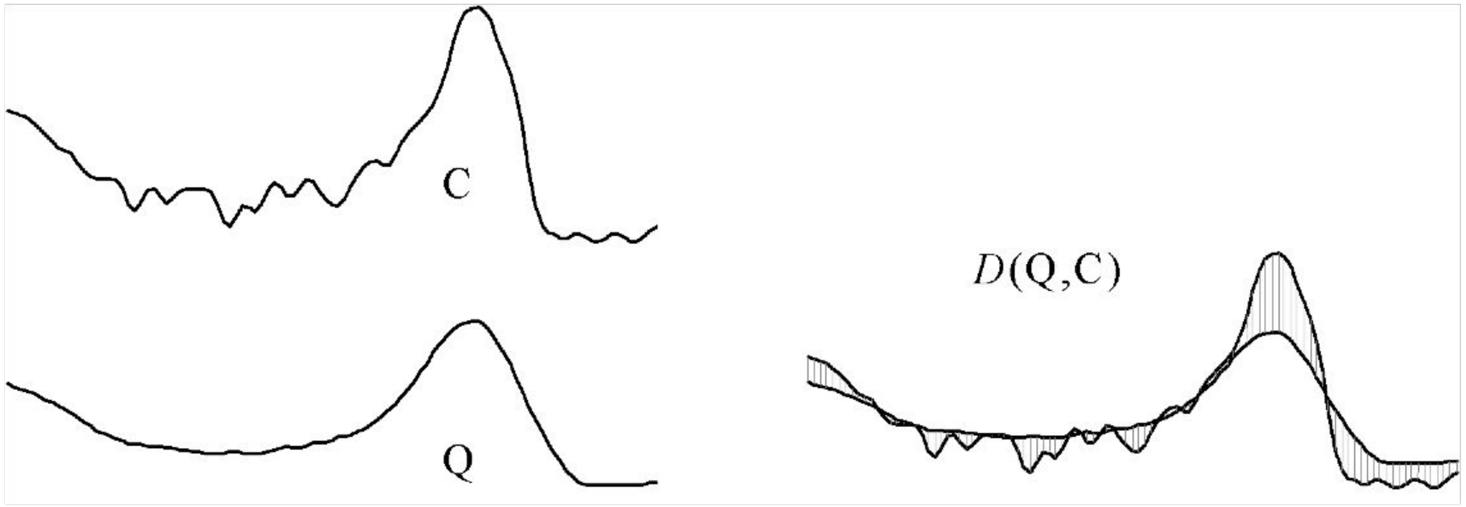
To cope with this, there are several solutions:

- Extract features invariant to alignment
    - Global features: mean, variance, frequencies, …
    - Features based on convolutions
      e.g. w/ deep learning.
- Use metrics that adapt to unalign signals

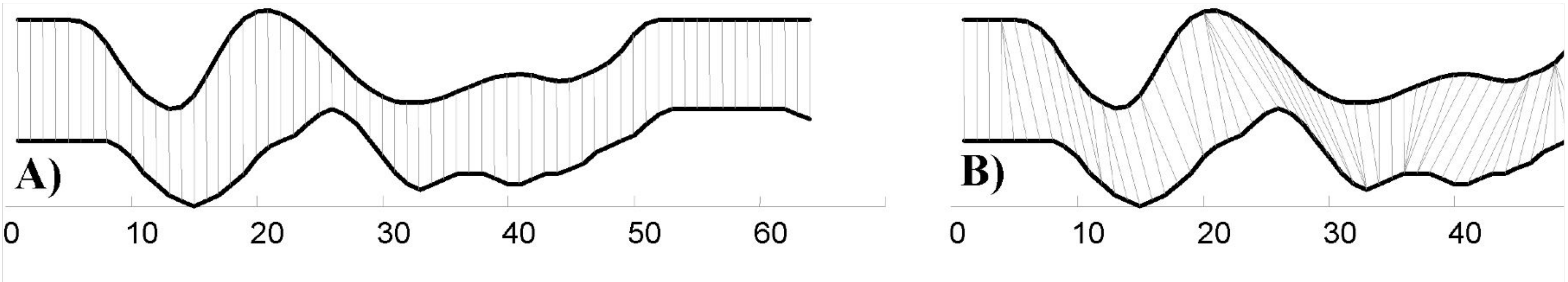
Metrics on time series

- Euclidean distance

$$D(x^1, x^2) = \sqrt{\sum_t (x_t^1 - x_t^2)^2}$$



- Dynamic Time Warping (DTW)

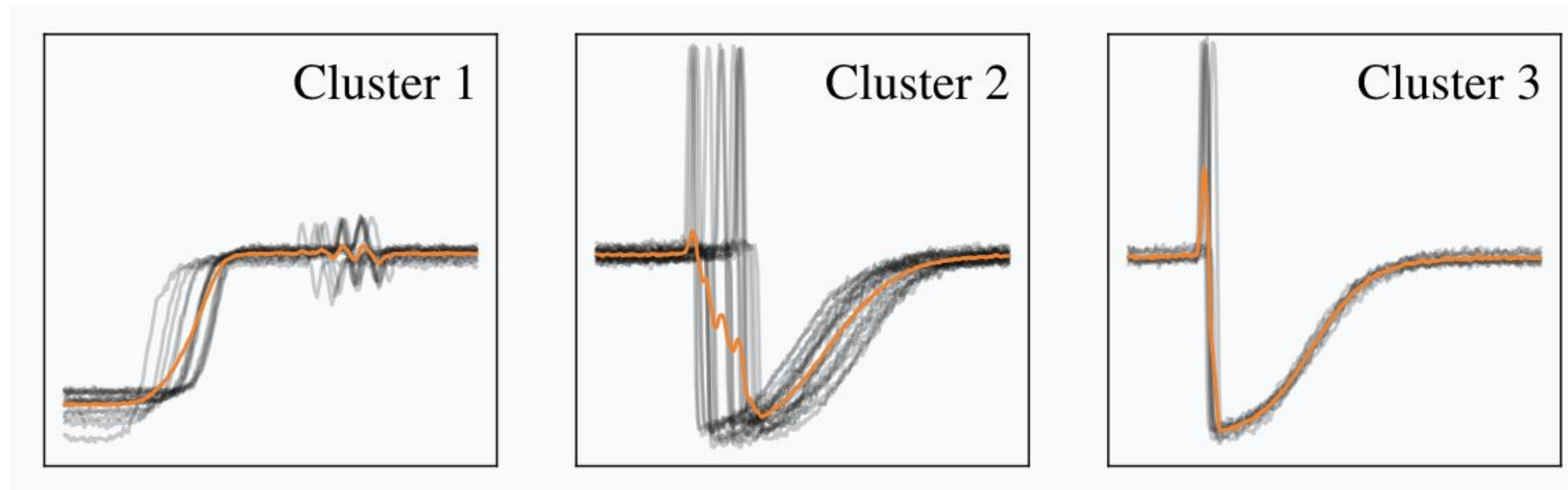


Metrics/distances/similarities are everywhere in ML:
K-means, K-nearest neighbors, kernels / SVMs, etc.

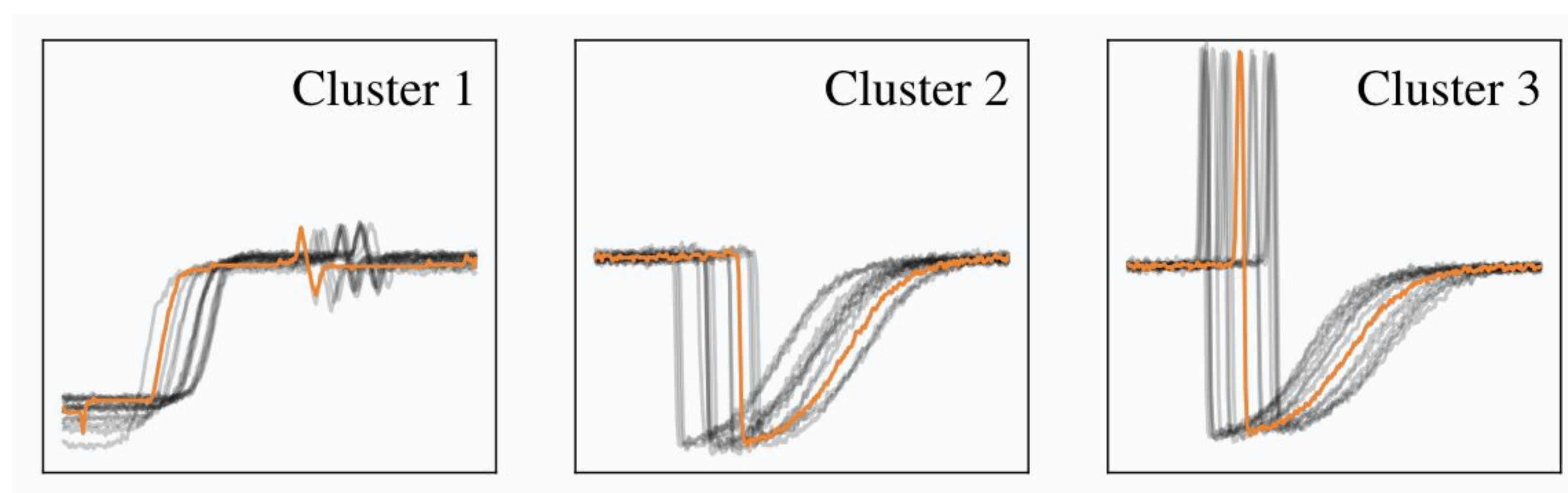
Clustering signals

Dataset

K-means result with L2 distance



K-means result with DTW distance



Source: https://rtavenar.github.io/blog/dtw.html

Classifying signals

02-classifying-signals.ipynb

03 - Complex tasks with complex data

A side note on samples in TS

We have considered simple tasks where we have one label per signal: $$p(y^i|x^i_{[1,T]})$$

However, classic tasks with signals involve $p(y_t|x_t)$

- Horizon vs the context of the prediction

$$p(y_{[t,t+H]}|x_t) \qquad p(y_t|x_{[t,t-C]})$$

- One can also extract features

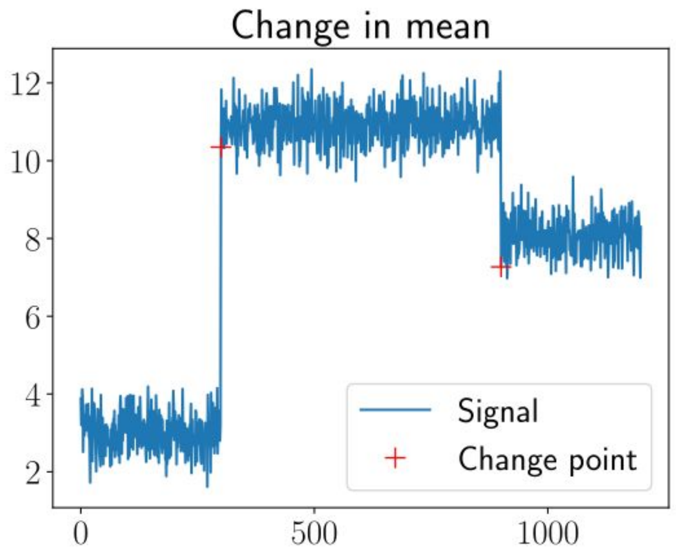
$$p(y_t|z_t) \quad \text{where} \quad z_t = f(x_{[1,t]})$$

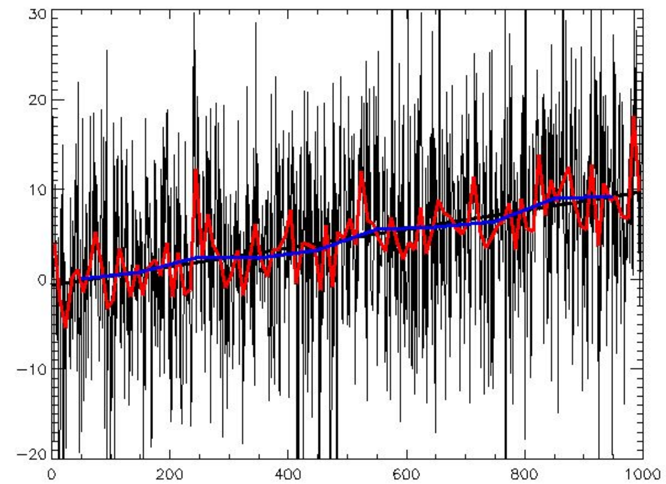
What is different with these tasks?

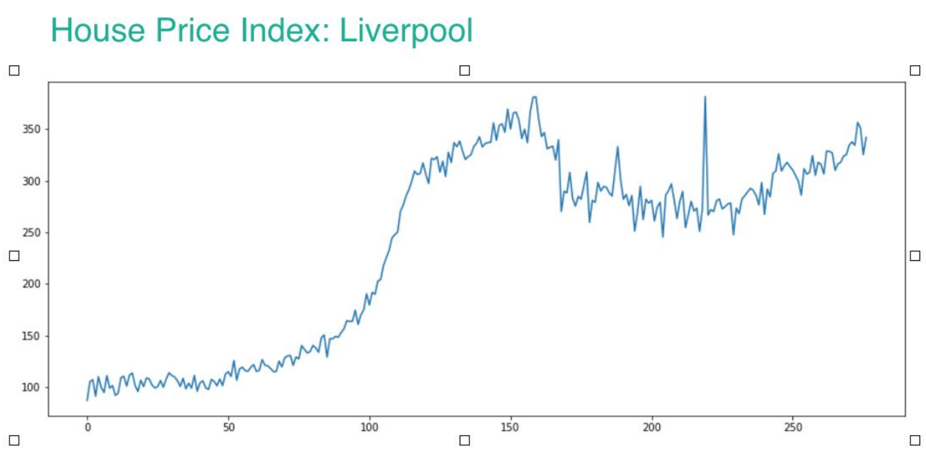
- Samples from a same signal are not *i.i.d.* anymore: samples have regularity, they are autocorrelated
- Generic APIs (like scikit-learn) are once again much harder to design.
- The ecosystem is scattered / messy…
- One can learn with as single “sample”.
- The statistical power comes from stationarity.

Looking at time series (and some jargon)

Smoothness, noise and stationarity







- Smoothness measure how close two samples can be.

- Noise can hide the overall pattern

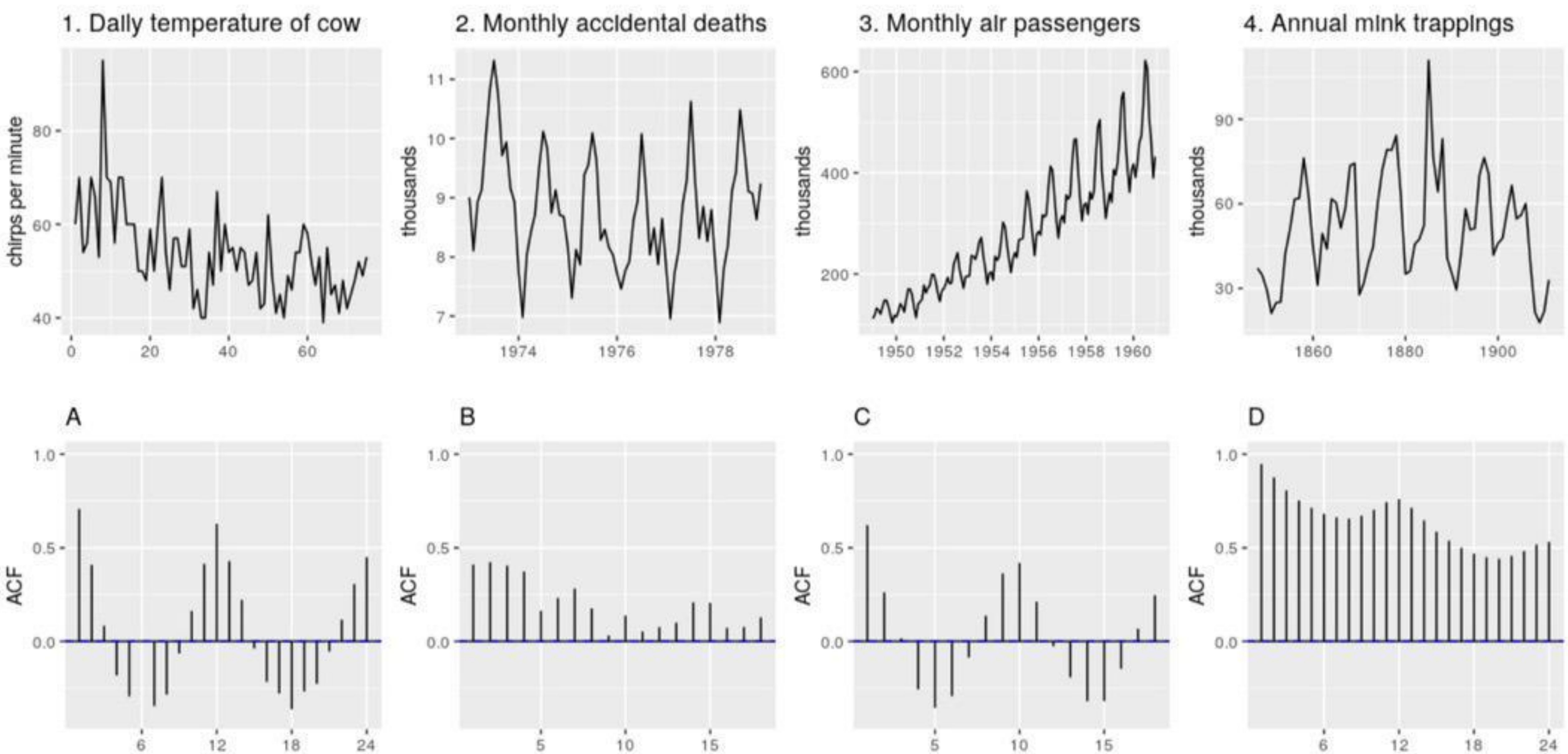
- Non-stationarity are points where the statistics or the dynamic of the signal change

Autocorrelation

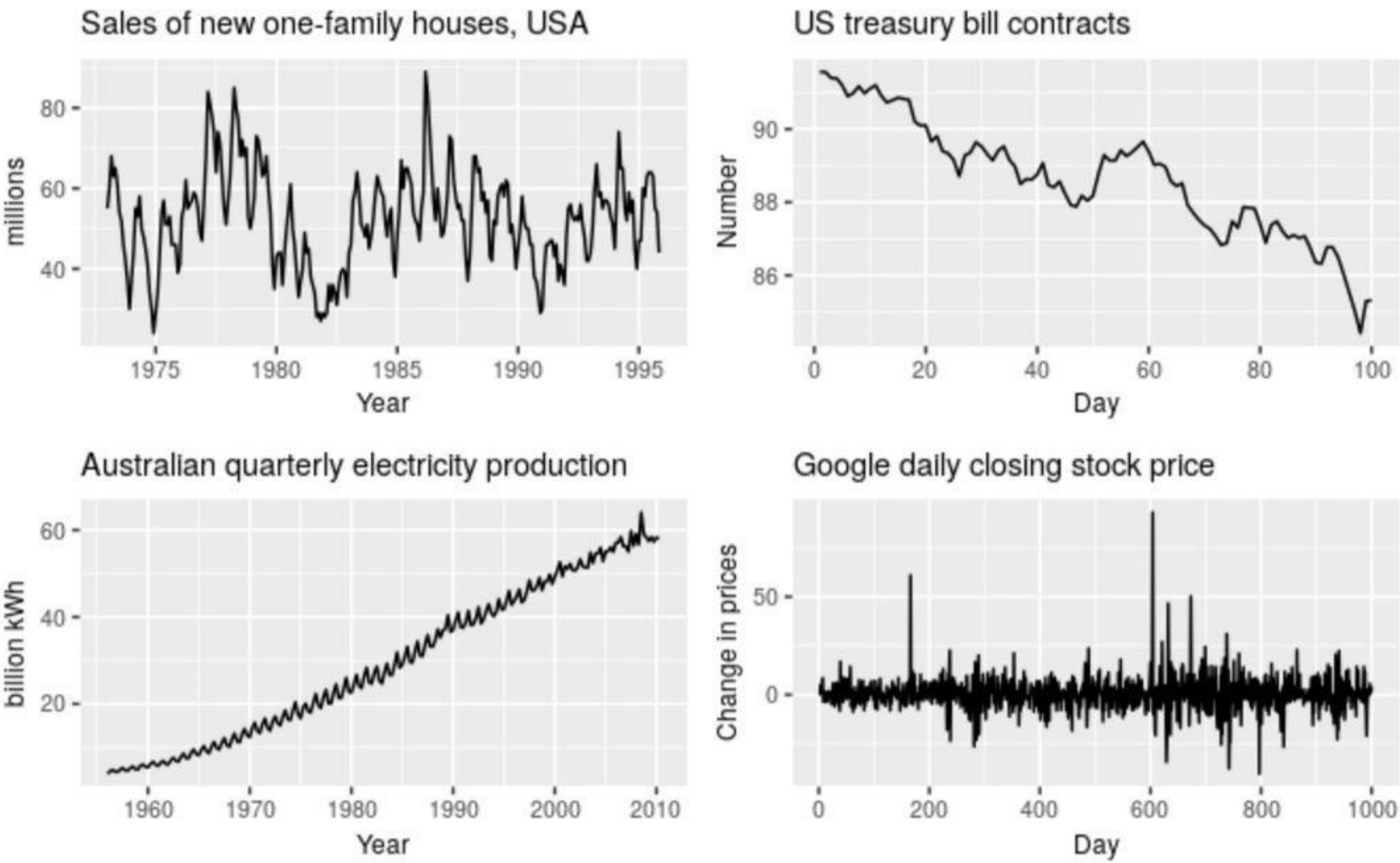
- Measures the correlation between x_t and x_{t-k}

$$r_k = \frac{\sum_{t=k+1}^{T} (x_t - \bar{x})(x_{t-k} - \bar{x})}{\sum_{t=1}^{T} (x_t - \bar{x})^2}$$

- Can you match the autocorrelation function (ACF) to the data?



Trend? Seasonality? Cyclic behavior?



Evaluating complex tasks with signals

03-evaluating-ts-models.ipynb

We'll now focus on forecasting

$$p(y_t|x_{[t,t-C]}) \quad \text{with} \quad y_t = \begin{cases} x_{t+1} \\ x_{[t+1,t+H]} \end{cases}$$

From Linear Models to auto-regressive (AR) models

$$y = \sum_{j=1}^{p} x_j \theta_j + \theta_0$$

Example: $$\text{sales(t)} = \underbrace{\theta_0 + \theta_1 t}_{\text{trend}} + \underbrace{\theta_2 \cos\left(\frac{2\pi t}{12}\right)}_{\text{seasonal}}$$

vs

$$x_t = \sum_{k=1}^{p} x_{t-k} \theta_k + \theta_0$$

Example: $$\text{sales(t)} = \theta_0 + \underline{\theta_1 \text{sales(t} - 1)} + \underline{\theta_2 \text{sales(t} - 2)}$$

Previous values

Recurrent vs. non-recurrent model

$$x_t = f_\theta(t) + g_{\theta'}(z_t)$$

fct. of t

covariates

- Non-recurrent model
- Prediction can be done together for all time points

VS.

- Recurrent model
- Prediction is done time-by-time
- Prediction is reinjected into the model

$$x_t = f_\theta(x_{t-1}, \ldots, x_{t-p}) + g_{\theta'}(z_t)$$

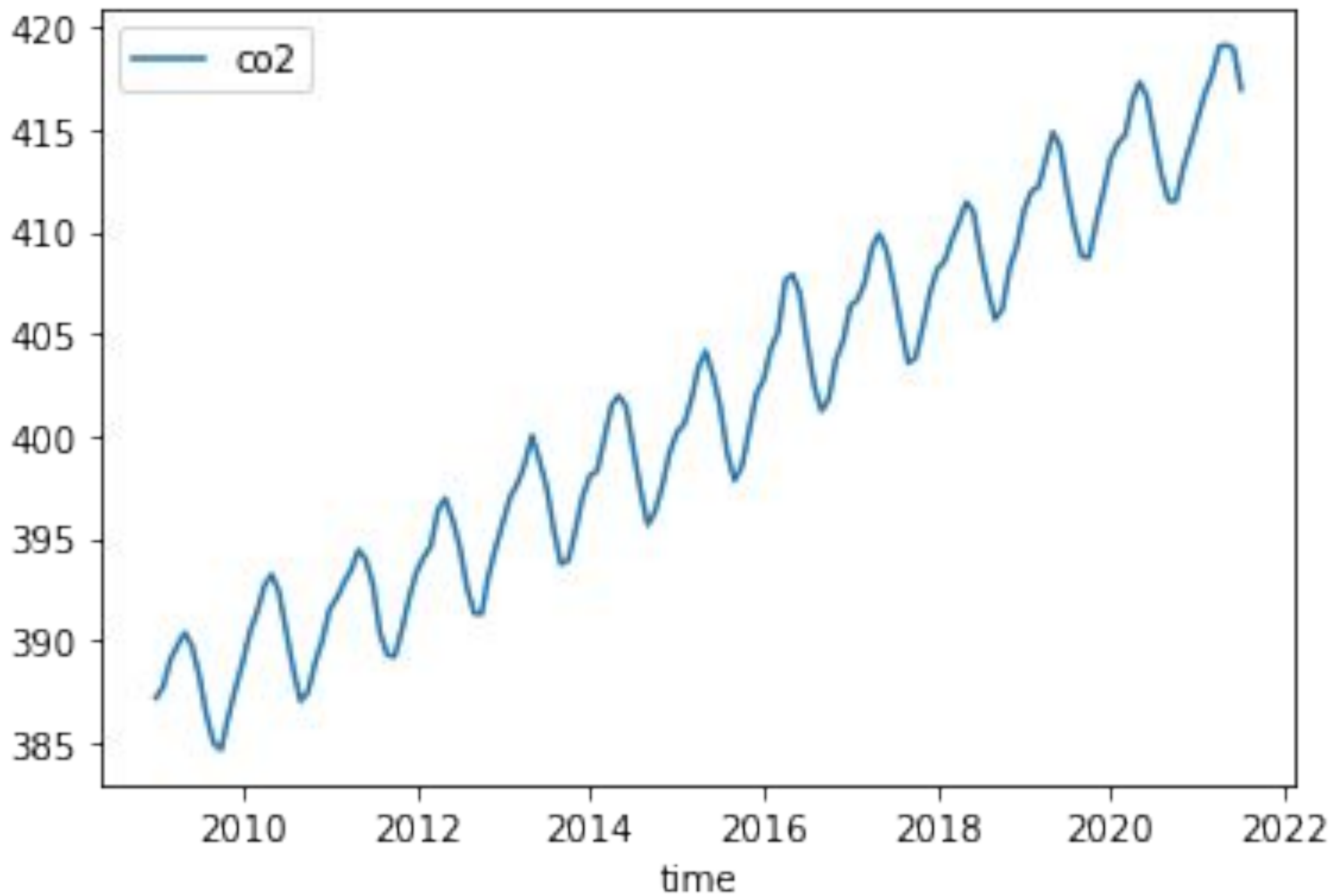
fct. of delayed features

covariates

Forecasting with a linear model

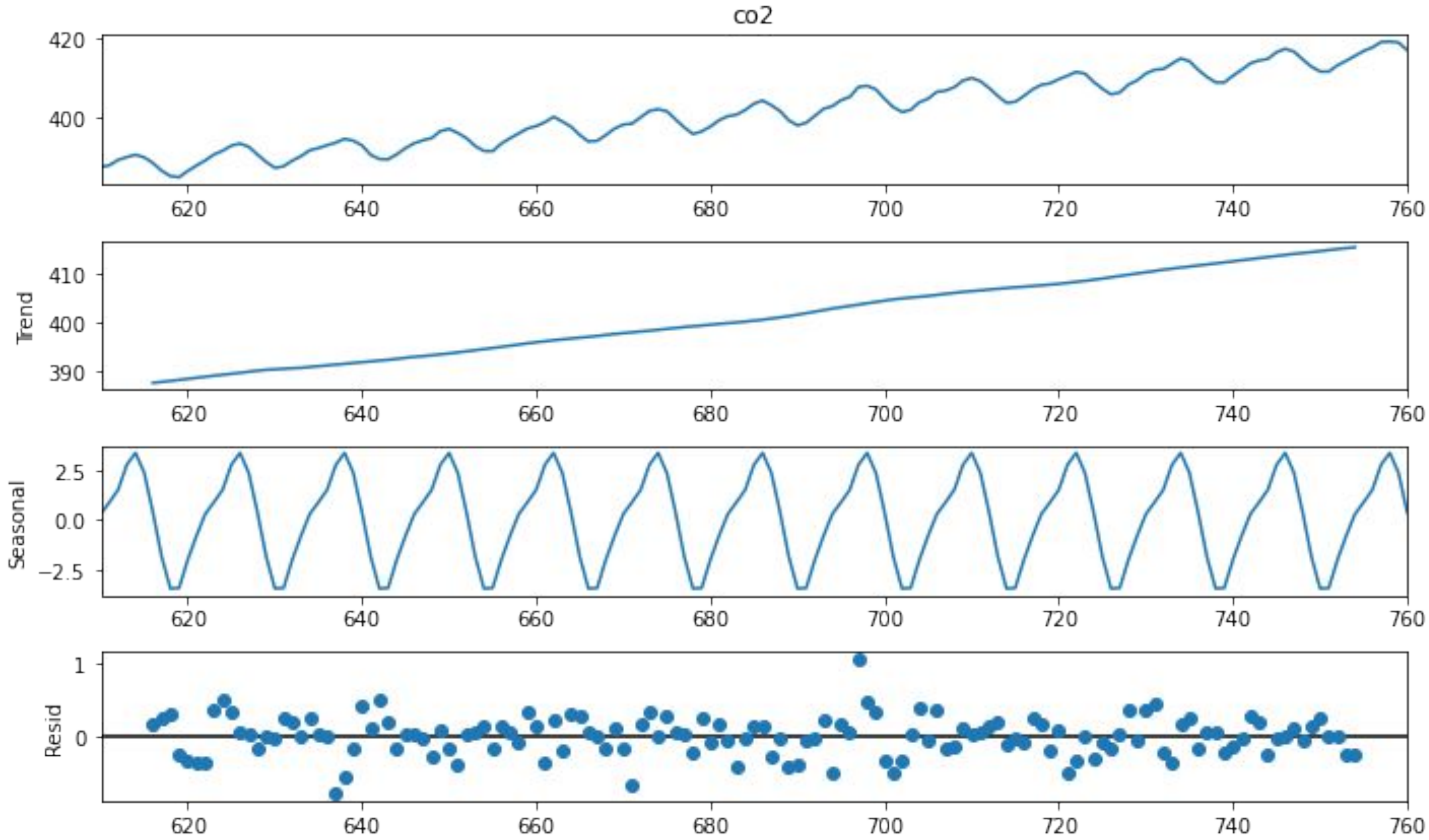
A non-recurrent model: learn $t \mapsto x(t)$

$$x_t = \theta_0 + \theta_1 t + \theta_2 \cos\left(\frac{2\pi t}{12}\right) + \theta_3 \sin\left(\frac{2\pi t}{12}\right) + \epsilon$$



FORECASTING WITH A LINEAR MODEL

data = trend + seasonal + noise



See: `from statsmodels.tsa.seasonal import seasonal_decompose`

FORECASTING WITH A LINEAR MODEL

Recurrent model: $x_t = \sum_{k=1}^{p} x_{t-k}\theta_k + \theta_0$ AR(p)

Hankel Matrix a.k.a. "Temporal embedding"

This corresponds to learning with windows of the signal.

This is a very classical pattern for signals!

- Forecasting,
- Event or object detection,
- Anomaly detection.

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} \Rightarrow \begin{bmatrix} a & b & c & d & e \\ b & c & d & e & f \\ c & d & e & f & g \\ d & e & f & g & h \\ e & f & g & h & i \end{bmatrix}$$

Also called patch based analysis.

A case study

—

Forecasting Bike usage

04-forecasting-bikes.ipynb

Questions?