Introduction to change point detection

Euan T. McGonigle

12th September

University of Southampton

Session Plan

First part:

- Learn about what change point detection is.
- Learn what the common change point methods are.
- Learn about some software we can use.

Second part:

- Write code and use software to try out change point algorithms.
- Compare methods and see the pros/cons.
- Perform change point analysis on some environmental data sets.

All materials for today are available on my Github page: `github.com/euanmcgonigle/IntroToCPD`.

Software we'll use

I'll be using R for the session, but if you are more comfortable then please use Python: both have everything we need for what we'll cover today.

If you want to use R, you'll need:

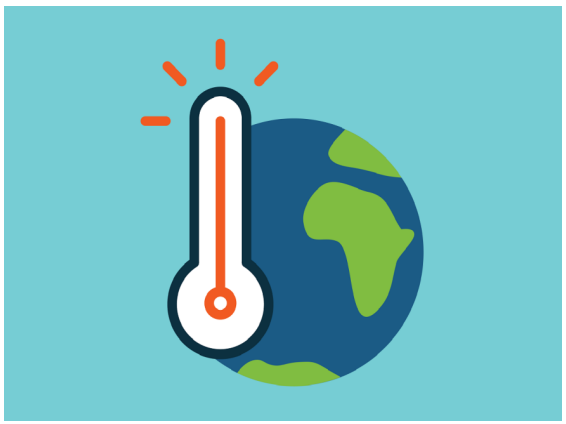
- `changepoint`
- `mosum`
- `EnvCpt`

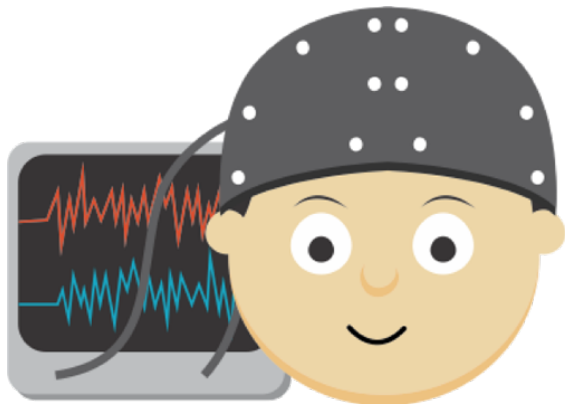
If you want to use Python, you'll need:

- `ruptures`
- `mosum`

Introduction

- Time series are now recorded in higher volumes over longer periods.
- Often, dynamics of time series are assumed stationary – unrealistic!
- Instead, we can develop nonstationary time series models.
- Could be with respect to mean level, spectral density, correlation, ...
- Examples: climatology, finance, neuroscience, ...
- 1 approach: piecewise stationarity via change point detection.





Introduction

- We might want to take a change point approach to model things like:
    1. air quality,
    2. frequencies of earthquakes,
    3. global sea temperatures,
    4. sentiment of text data like tweets.
- We'll focus on some common time series models to illustrate key ideas, but these techniques can be readily adapted to other more complicated models.

Change point detection (CPD)

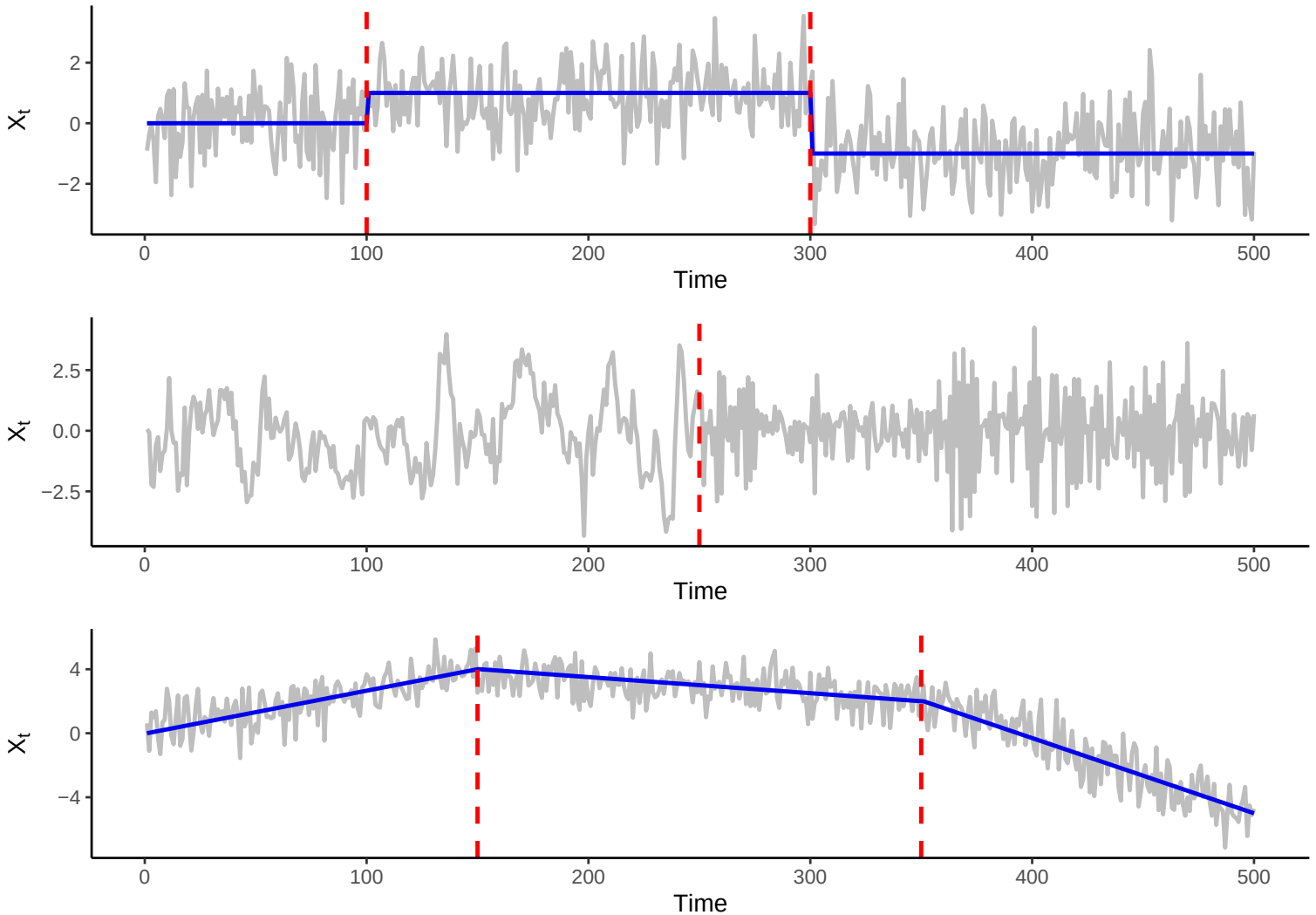
Change points: abrupt changes in the statistical properties of the time series. Also known as:

- data segmentation,
- break points,
- structural breaks,
- data shift/drift, ...

We'll look at some of the most well-studied settings, and see how some simple ideas can be used to build powerful CPD algorithms.

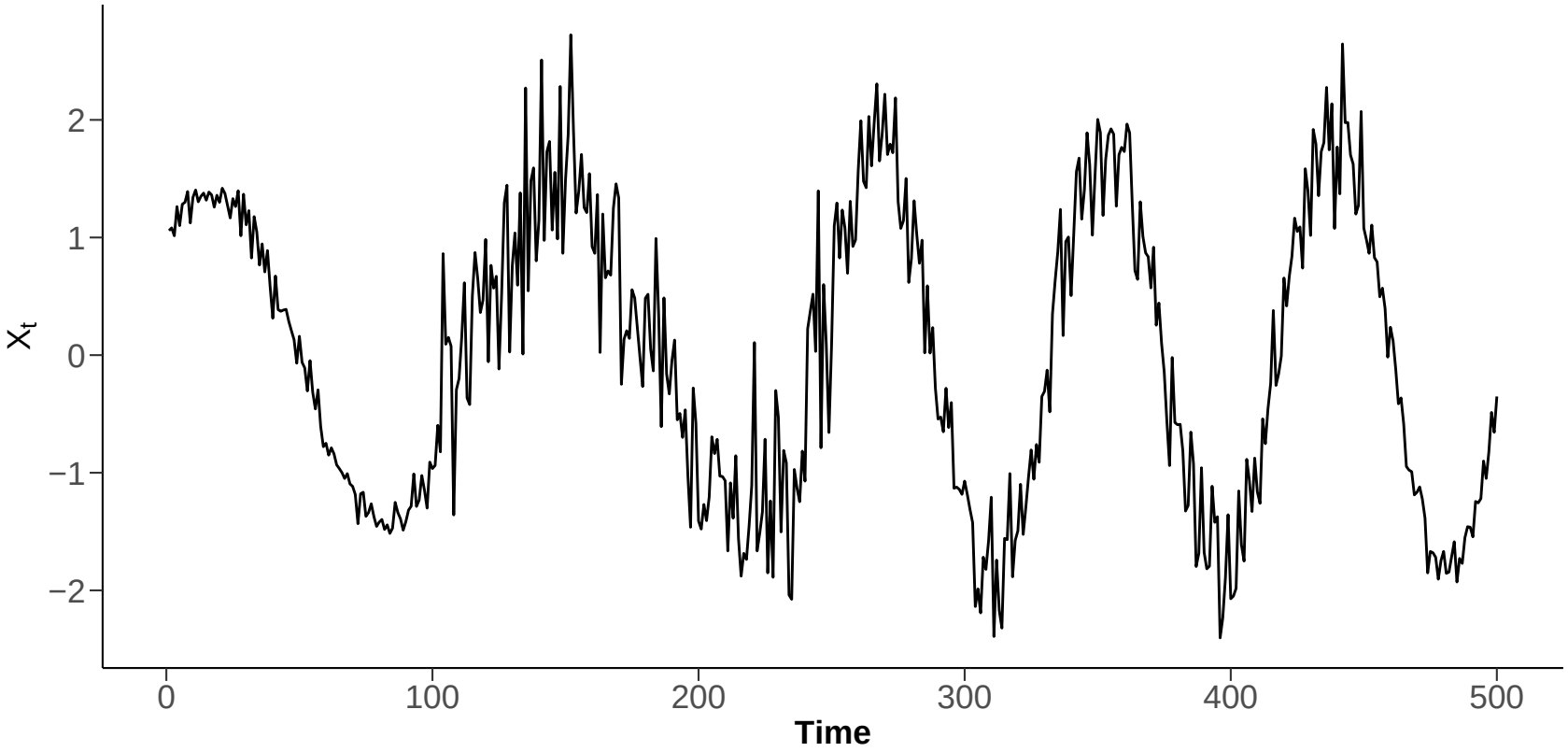
Change point detection (CPD)

Change points could occur in any properties of the data we want to model, such as the mean, dependence, and trend:



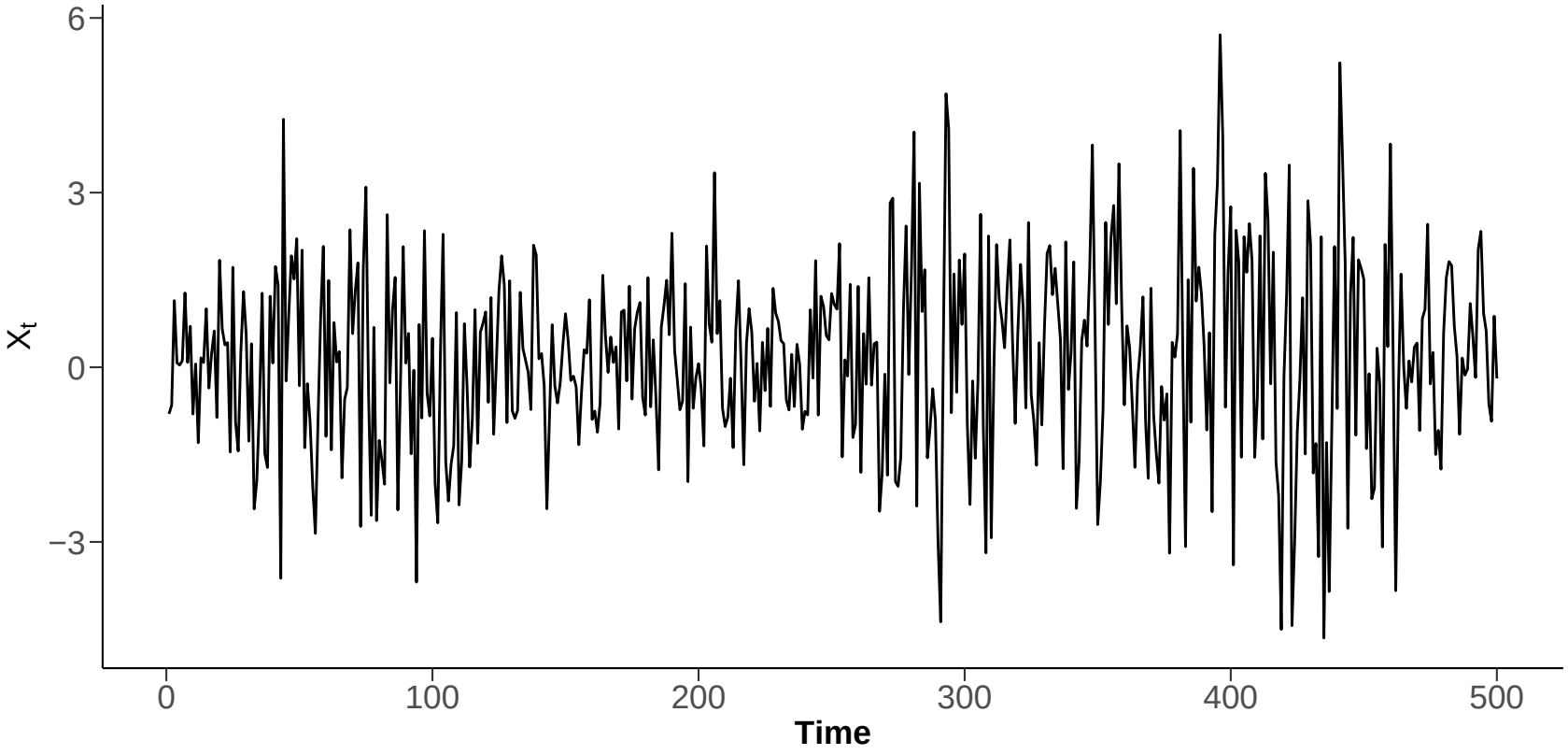
More complicated models

How many change points below? And where?



More complicated models

How many change points below? And where?



CPD: the goals

A time series is split into stationary segments defined by the change points. We want to:

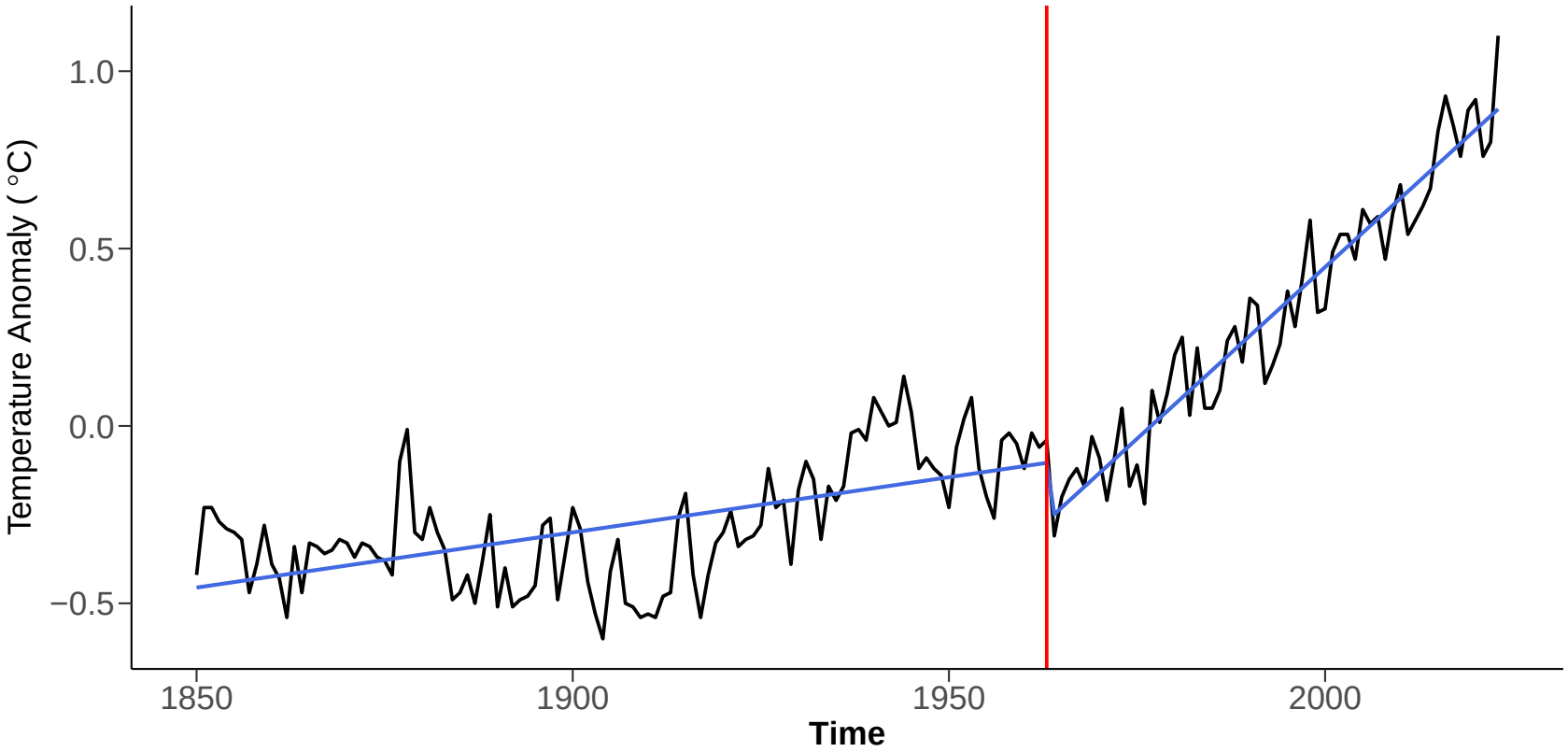
1. **Test:** Have changes occurred? If so, how many?

2. **Localise:** Where have the changes happened?

3. **Theoretically justify:** Statistical guarantee on number of changes and accuracy of estimated locations.

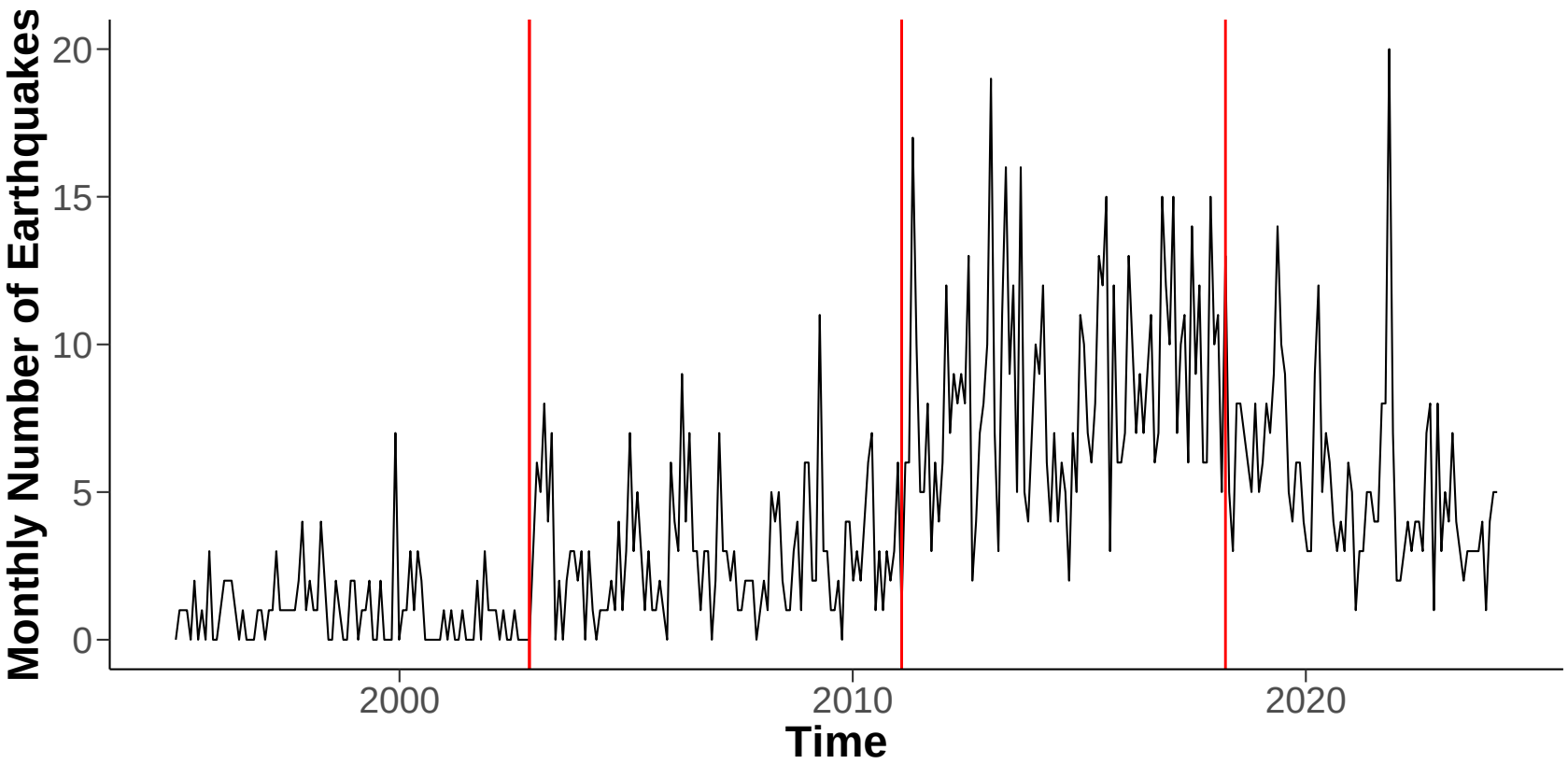
4. **Analyse post-segmentation:** Estimate parameters, quantify uncertainty, forecasting, causality?

Example application: global sea temperature anomalies



Example application: earthquake occurrences

Monthly number of earthquakes in the Groningen oil field:



Why CPD?

- The simplest possible nonstationary model.
- Very interpretable.
- Tends to lead to computationally tractable and exact solutions.
- Often we can tie change points to physical phenomena e.g. stock market crash, policy interventions, human-made emissions.
- Data-driven, "agnostic" intervention analysis.

Simple model – change in mean

Let's look at the canonical change in mean problem:

$$X_t = f_t + \varepsilon_t = \sum_{j=0}^{q} \mu_j \cdot \mathbb{I}\{\theta_j + 1 \leq t \leq \theta_{j+1}\} + \varepsilon_t, \quad t = 1, \ldots, n.$$

- f_t is the signal – piecewise constant mean function.
- f_t has jumps at the q change points $\theta_1, \ldots, \theta_q$.
- ε_t is a zero-mean noise term.

Note: for a general setting:

$$X_t = \sum_{j=0}^{q} X_t^{(j)} \cdot \mathbb{I}\{\theta_j + 1 \leq t \leq \theta_{j+1}\},$$

where distributions of $X^{(j)}$ and $X^{(j+1)}$ differ somehow.

Aside – online vs offline

Two main settings for CPD are **offline** and **online**.

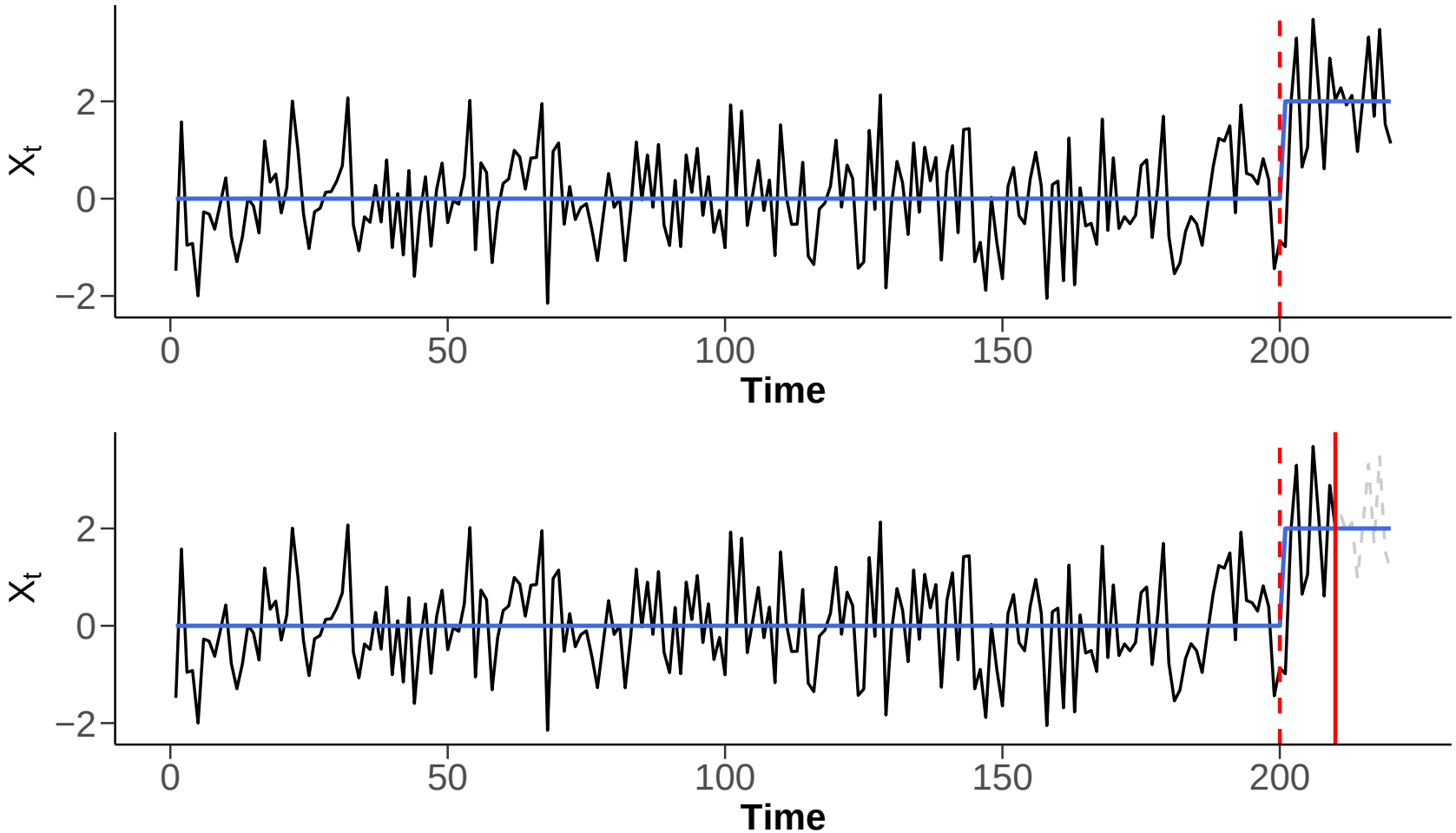
Online:

- Processes the data as it arrives, or in batches.
- Main aim is to quickly detect new changes.
- Applications include process control monitoring, fault detection.

Offline:

- Processes all the data in one go, after it has all been collected.
- Main aim is accurate detection of changes.
- Applications include genome analysis.

Aside – online vs offline



Simple model – change in mean

What do we want to do?

- Estimate the number of change points q.
- Estimate the change point locations $\theta_1, \ldots, \theta_q$.

How do we do that?

Many approaches, 2 of which are:

- Localised testing using a test statistic.
- Cost function based on data fit.

Localised testing

- Scan data for candidate change point estimators.
- Typically involve computing a test statistic of the form $\widehat{\sigma}_{s,e}^{-1}|\mathcal{T}_{s,k,e}|$, where

$$\mathcal{T}_{s,k,e} = \sqrt{\frac{(k-s)(e-k)}{e-s}}\left(\frac{1}{k-s}\sum_{t=s+1}^{k} X_t - \frac{1}{e-k}\sum_{t=k+1}^{e} X_t\right),$$

  and $\widehat{\sigma}_{s,e}$ is a estimator of $\sigma_{s,e}$, a measure of variability of $\{X_t\}_{t=s}^{e}$.
- Compute $\mathcal{T}_{s,k,e}$ for all k over a range of intervals to find the most likely change point locations.
- Often compare $\widehat{\sigma}_{s,e}^{-1}|\mathcal{T}_{s,k,e}|$ to a threshold D to test for changes.

CUSUM approaches: idea

- Simplest example is the cumulative sum (CUSUM):

$$\mathcal{T}_{0,k,n} = \sqrt{\frac{k(n-k)}{n}} \left(\frac{1}{k} \sum_{t=1}^{k} X_t - \frac{1}{n-k} \sum_{t=k+1}^{n} X_t \right).$$

- To find a single change θ_1 we compute $\mathcal{T}_{0,k,n}$ for all $1 \leq k \leq n-1$, and set

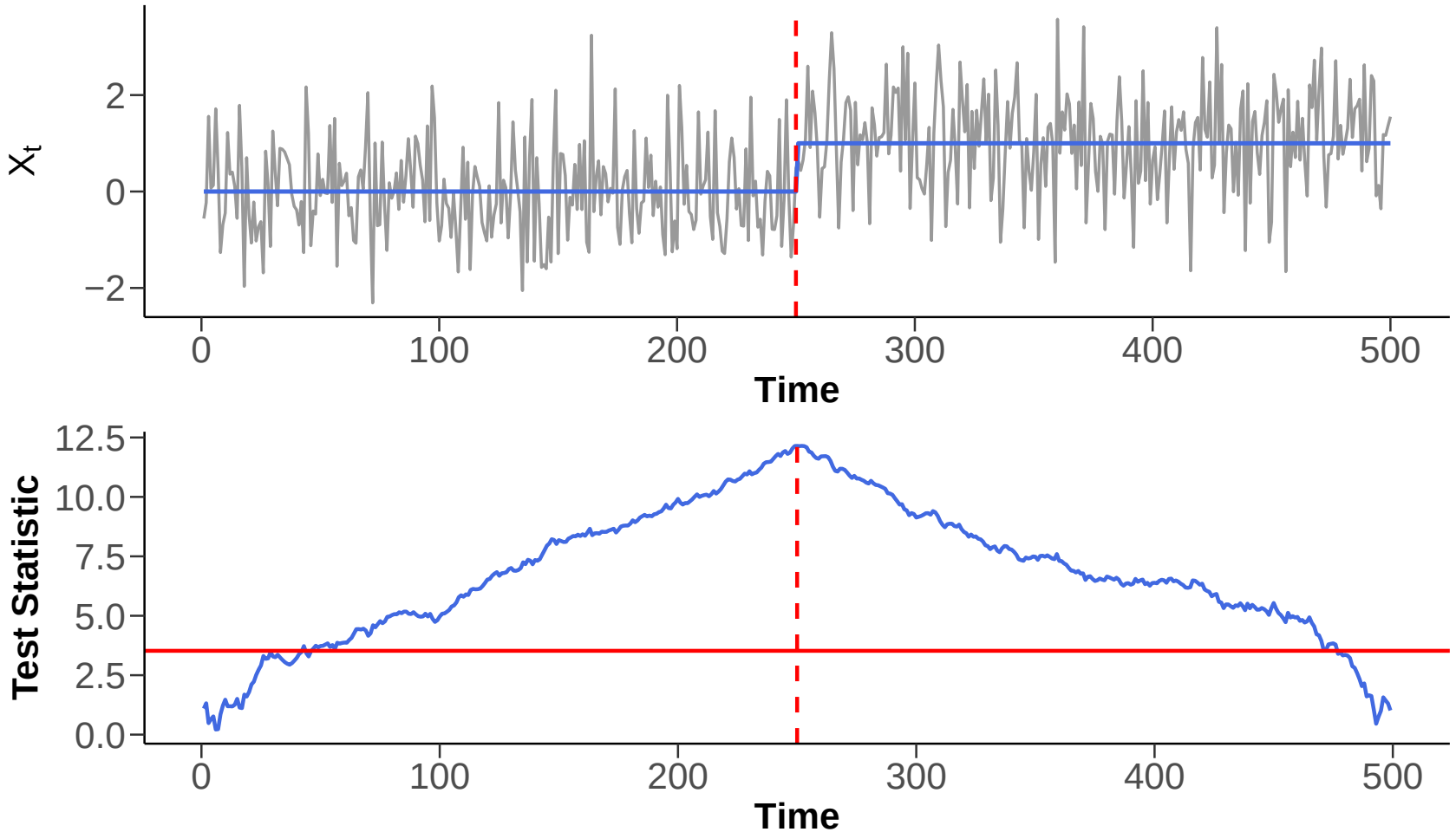
$$\widehat{\theta}_1 = \underset{1 \leq k \leq n-1}{\arg\max} \, |\mathcal{T}_{0,k,n}|$$

  to be the change point estimator.

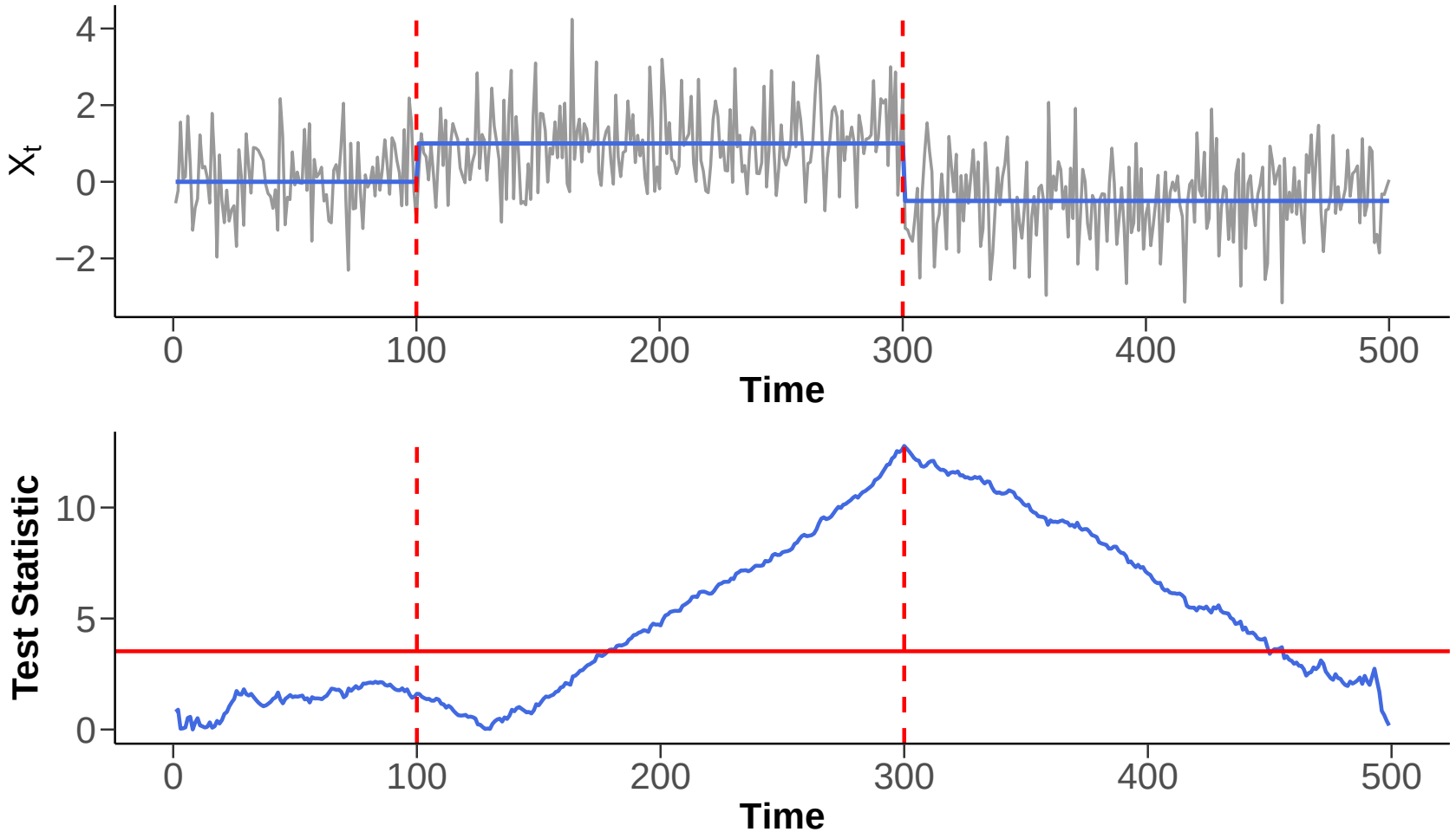
- We declare $\widehat{\theta}_1$ as a change if $\hat{\sigma}^{-1} |\mathcal{T}_{0,\widehat{\theta}_1,n}| > D$.

- Many CPD methods start with the CUSUM as the basic idea, but use fancier ideas.

CUSUM approaches: single change



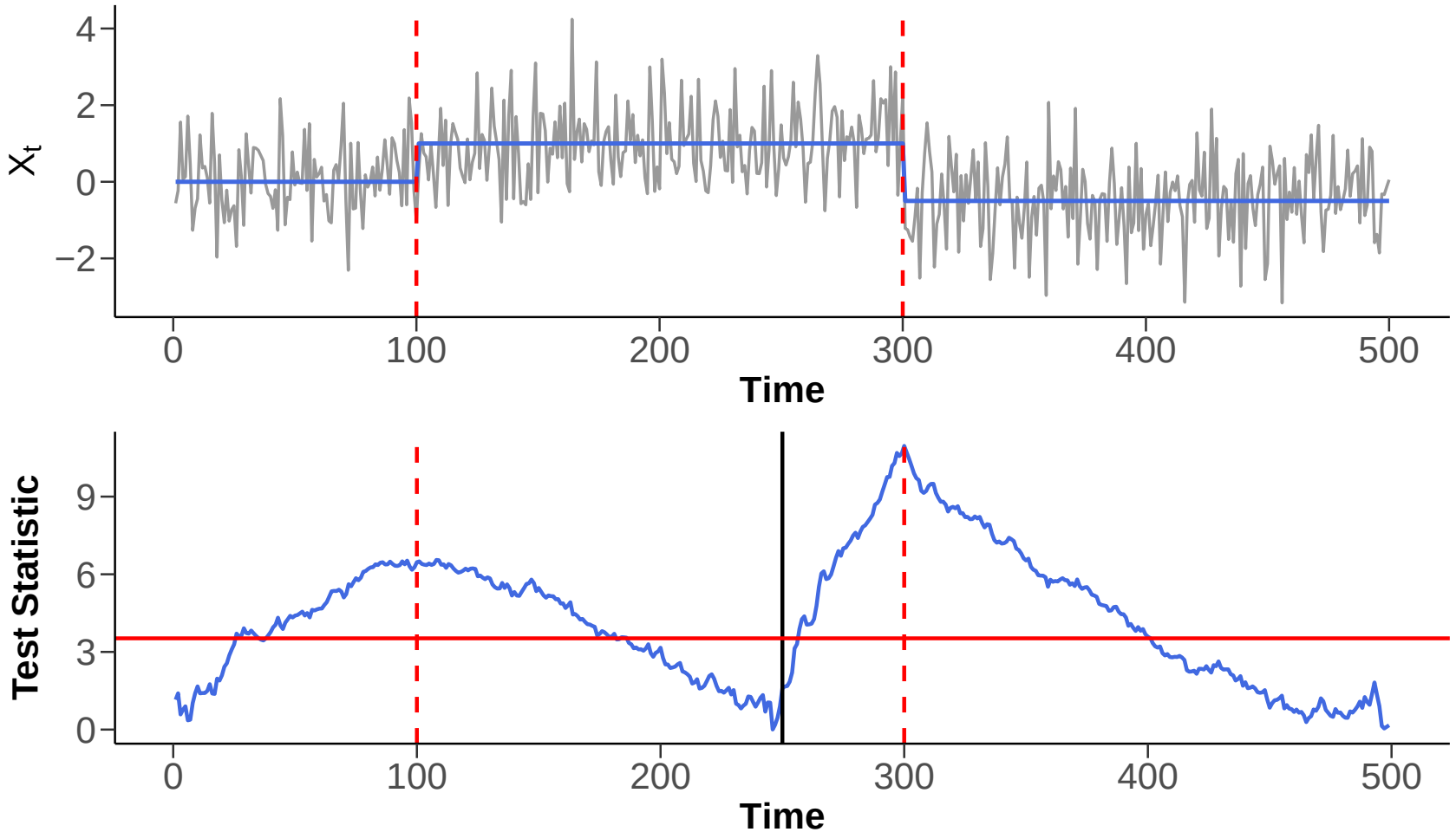
CUSUM approaches: multiple changes



CUSUM approaches: extensions

- It's easy to extend the global CUSUM to look for multiple change points.
- In Binary Segmentation (BS), after calculating $\widehat{\theta}_1$, we then perform 2 CUSUMs on the intervals $(1, \widehat{\theta}_1)$ and $(\widehat{\theta}_1 + 1, n)$, and rinse and repeat.
- Wild binary segmentation extends this by using BS on a set of randomly generated intervals.
- Many other state-of-the-art methods devise sophisticated "interval generation" techniques, upon which CUSUMs are performed.

CUSUM approaches: smart interval generation



MOSUM approaches

There are many other CPD methods that use localised testing.

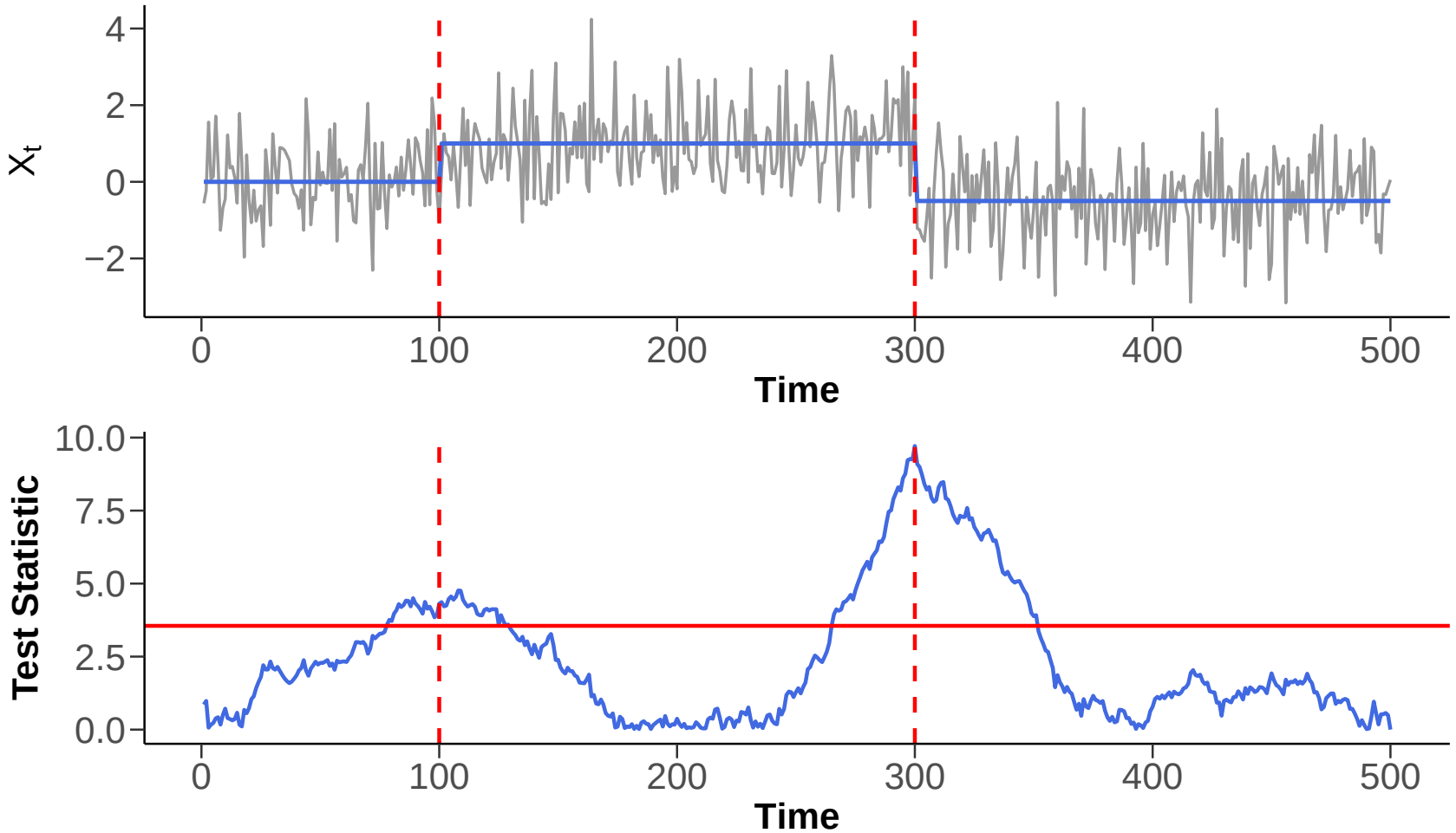
- The **MOSUM** approach uses moving windows of size G to the left and right of candidate change k.
- This allows it to simultaneously find multiple changes.
- For window size G, the MOSUM detector $T_G(k)$ is given by

$$\mathcal{T}_G(k) = \mathcal{T}_{k-G,k,k+G} = \frac{1}{\sqrt{2G}} \left(\sum_{t=k+1}^{k+G} X_t - \sum_{t=k-G+1}^{k} X_t \right)$$

  for $k = G, \ldots, n - G$.

- Changes declared as all local maximisers with $\widehat{\sigma}^{-1}|\mathcal{T}_G(k)| > D$.

MOSUM approaches: multiple change points



MOSUM approaches: extensions

- Again, this is a simple idea that can be extended into more sophisticated algorithms.
- Uncertainty quantification on change locations via bootstrap.
- Instead of using G data points on each side of the window, can use asymmetric sizes.
- A multiscale version of MOSUM using multiple bandwidths helps to detect changes of different sizes and different segment lengths.
- Can easily be generalised to multivariate data and other types of change.

MOSUM approaches: extensions

- Again, this is a simple idea that can be extended into more sophisticated algorithms.
- Uncertainty quantification on change locations via bootstrap.
- Instead of using G data points on each side of the window, can use asymmetric sizes.
- A multiscale version of MOSUM using multiple bandwidths helps to detect changes of different sizes and different segment lengths.
- Can easily be generalised to multivariate data and other types of change.

Cost function approaches

Another common approach is using cost functions.

- A cost function $\mathcal{C}(X_{s:e})$ measures how well (badly) the model fits the data on the segment $\{X_t\}_{t=s}^{e}$.
- For 1 change point, we can measure the cost of fitting a change point at time k as $\mathcal{C}(X_{1:k}) + \mathcal{C}(X_{(k+1):n})$.
- For finding q change points, we can minimise the equation

$$\sum_{j=1}^{q+1} \mathcal{C}(X_{(\theta_{j-1}+1):\theta_j}) + f(q),$$

  where $f(q)$ is a penalty to avoid overfitting.
- Twice the negative log likelihood is a commonly used cost function.

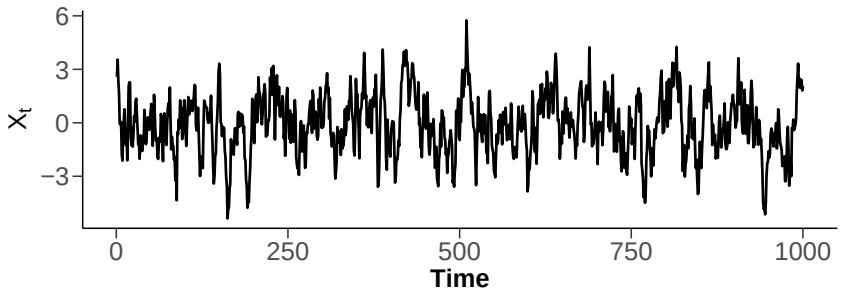
Cost function approaches: PELT

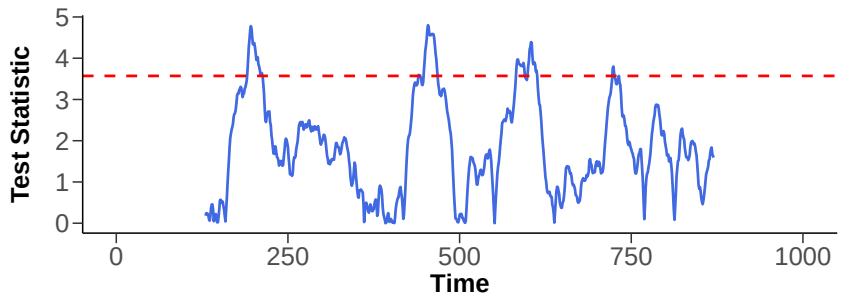
- Trying to calculate this minimum exhaustively is not feasible.
- Thankfully it can be calculated recursively: the minimal cost for $X_{1:s}$ can be expressed in terms of that of $X_{1:t}$ for $t < s$.
- This is still complexity $\mathcal{O}(n^2)$: the Pruned Exact Linear Time (PELT) method improves this to expected complexity $\mathcal{O}(n)$.
- Achieves this by pruning the set of possible change point locations in a clever way, using properties of the cost function.
- For change in mean, equivalent to least-squares estimation.

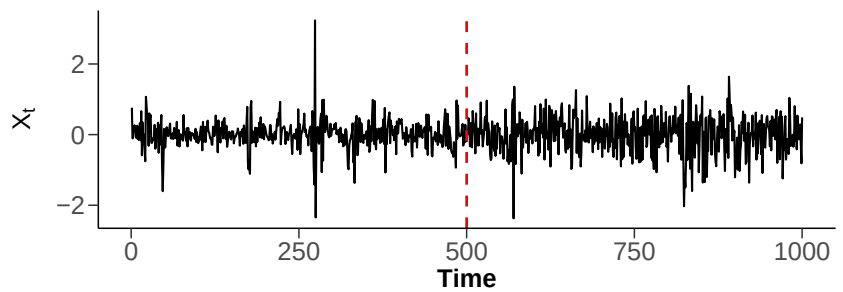
Nonparametric approaches

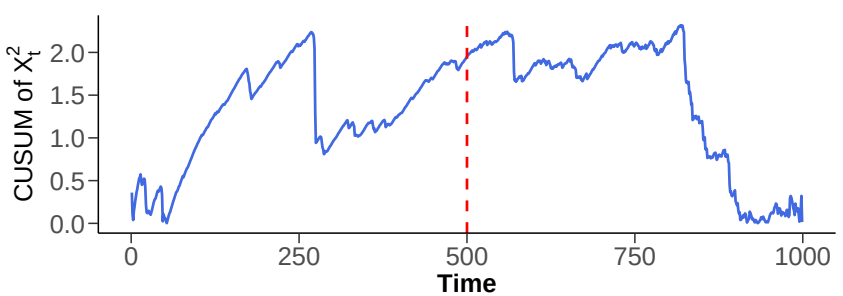
What if we're not sure what type of parameters/model we want to pick?

We can miss changes if we don't know what we're looking for, or falsely detect changes, if we assume the wrong model:









In this case, we can use nonparametric methods.

Nonparametric approaches

- We can do a kernel version of the least-squares/cost function approach: for given q, minimise

$$L(q) = \frac{1}{n}\sum_{i=1}^{n} k(X_i, X_i) - \frac{1}{n}\sum_{j=1}^{q}\left[\frac{1}{\theta_j - \theta_{j-1}}\sum_{k=\theta_{j-1}+1}^{\theta_j}\sum_{\ell=\theta_{j-1}+1}^{\theta_j} k(X_k, X_\ell)\right],$$

  where $k(\cdot,\cdot)$ is a kernel function.

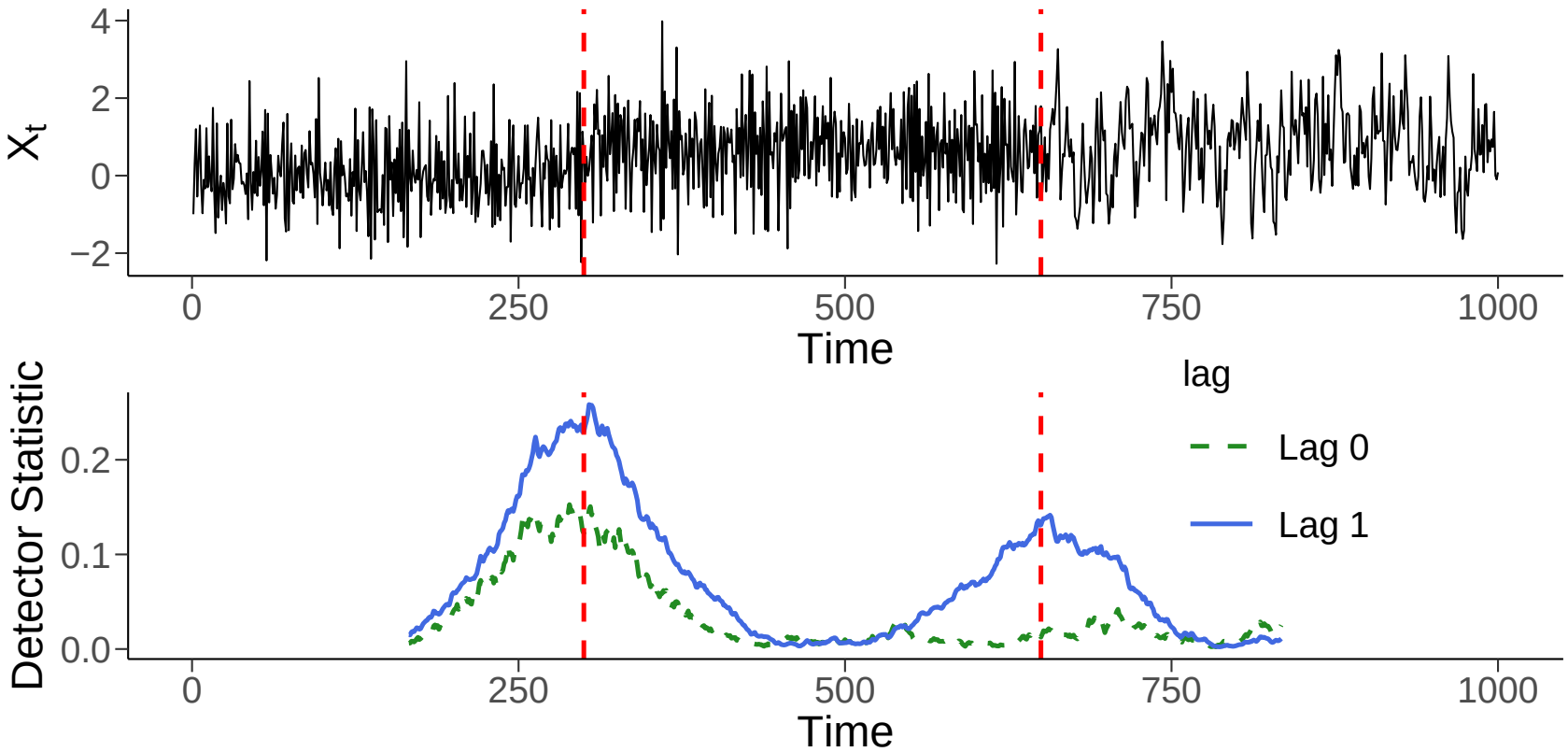
- Many choices of kernel: most common is the Gaussian $k(x, y) = \exp(-||x - y||^2/(2h^2))$.

- If $k(x, y) = xy$, this becomes the same as the PELT method for changes in mean.

- We are basically just turning the problem into mean change point detection via kernels.

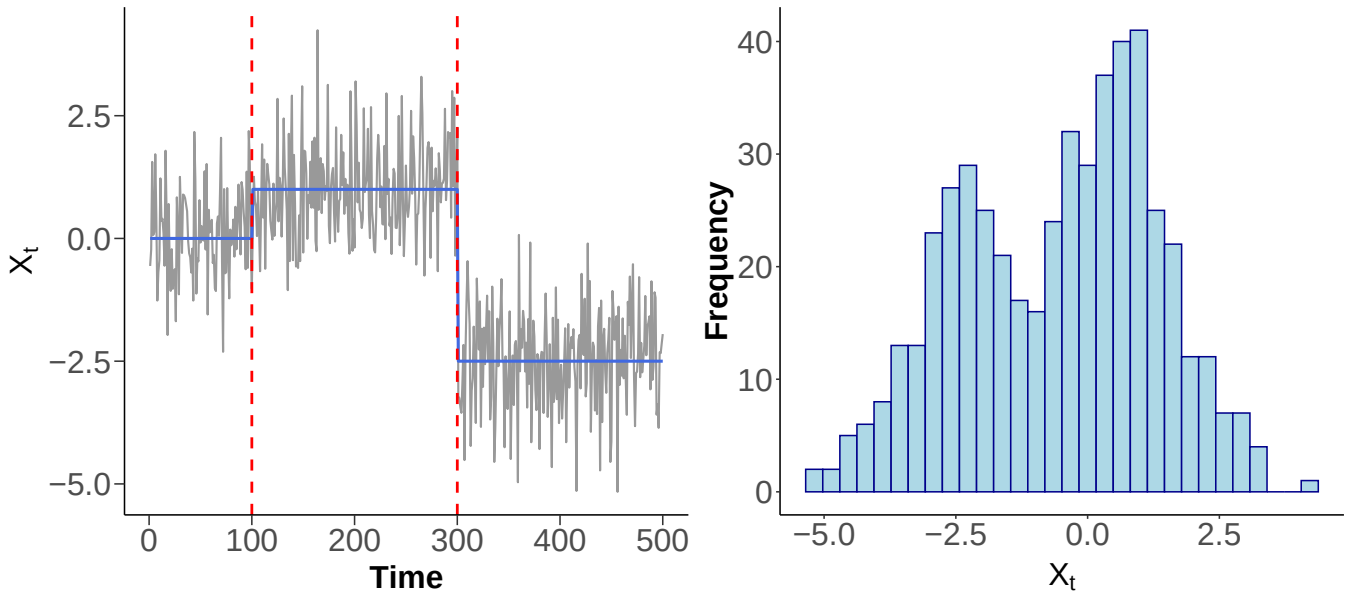
Nonparametric approaches

Example approach: a single test statistic/cost function can detect multiple types of changes.



Model validation/checking assumptions

- Often we want to validate the assumptions of our model, e.g. normality, independence of data.
- Everything becomes more difficult in the presence of change points:



- Can use basic diagnostics like checking residuals, e.g., QQ or ACF plots, with caution.
- We can use model selection approaches to compare competing change point models, e.g., with information criterion.

Multivariate data

All of the above discussion considered univariate data. What about multivariate data?

- Each variable of the data might be unconnected to the rest: univariate methods could work well.
- There might be structure/dependence across variables. For example,
    1. Changes occur at the same time across all variables,
    2. changes occur only in a subset of variables,
    3. changes in some variables cause changes in others at a later time.
- Many possible other scenarios: different types of change, different noise levels, ...
- Generally, we can take ideas from univariate methods and extend them to multivariate data.

Aside: multivariate change point software

Many packages doing multivariate change point detection.

In R:

- `ecp`: nonparametric CPD using "energy" distance.
- `InspectChangepoint`: high-dimensional mean CPD using projections.
- `fastcpd`: many settings, including several from `ruptures`.

In Python:

- `ruptures`: already has multivariate functionality.
- `gchangepoint`: graph-based nonparametric CPD.
- `changeforest`: nonparametric CPD via random forests.

Complications/considerations

- Speed of method: sequential nature means efficient implementations often possible.
- High-dimensionality: how do we aggregate information/find relevant structure?
- Outliers: "instantaneous" change points? Many methods not robust.
- Threshold selection: how to balance over/under detection?
- Missing data: how to impute when there are change points?
- Autocorrelation: harder to distinguish changes from noise.

Aside: some alternative approaches

Whilst I'm contractually obliged to advertise CPD, it's only 1 of many approaches:

- Mathematical/physical model: something that already captures the evolution of behaviour of interest.
- Local stationarity: statistical properties evolve slowly.
- Just ignore it (!): for some applications, e.g. forecasting, it might be better to just fit a stationary model.
- Data pre-processing: remove nonstationarity first, e.g. via a transformation.

Some things to think about

- What types of time series might contain change points?
- What types of change points might you expect to see in the type of data you look at.
- What parameters you might model that have nonstationary behaviour.
- Benefits and drawbacks to a change point approach.
- Homogeneity vs inhomogeneity – over space, time, ...

Further reading

Eichinger, B. and Kirch, C. (2018). A mosum procedure for the estimation of multiple random change points. *Bernoulli*, 24(1):526–564.

Fearnhead, P. and Fryzlewicz, P. (2022). Detecting a single change-point. *arXiv preprint arXiv:2210.07066*.

Fryzlewicz, P. (2014). Wild binary segmentation for multiple change-point detection. *The Annals of Statistics*, 42(6):2243–2281.

Killick, R. and Eckley, I. A. (2014). changepoint: An r package for changepoint analysis. *Journal of Statistical Software*, 58:1–19.

Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598.

Meier, A., Kirch, C., and Cho, H. (2021). mosum: A package for moving sums in change point analysis. *Journal of Statistical Software*, 97(8):1–42.

Truong, C., Oudre, L., and Vayatis, N. (2018). ruptures: change point detection in python. *arXiv preprint arXiv:1801.00826*.