Machine Learning for Time Series - MVA 2023/2024
gfpop: an R Package for Univariate Graph-Constrained

Simon Queric simon.queric@telecom-paris.fr
Vincent Herfeld vincent.herfeld@telecom-paris.fr

December 18, 2023

1 Introduction and contributions

- We both worked on understanding the graph modelling for change point detection. As a toy model, we propose the Up-Up-Down graph of constraints. In the application part, Vincent focused on the isotonic regression for synthetic corrupted signals and on multi-modal regression for hospitalisation count signals. Simon worked on ECG signals for finding QRS complex and on public light energy consumption signals.
- We used the gfpop package described in the article. The source code is available here : `https://github.com/vrunge/gfpop/tree/master`. We conducted some experiment with both synthetic signals and real signals we find on public available databases.
- We reused the available code on github for QRS detection, and applied what was explained in the article for isotonic regression. We also conduct qualitative discussion of the influence of the penalty β.

Classical change point detection aims to find regions of a signal where it is stationary (see [TOV18]). With a dynamic programming approach, the problem has a solution. However the computational cost, equals to $O(NK^2)$ is expansive and some pruning algorithms can improve the efficiency.

Although the general problem seems intuitive, it's somehow ill-posed. Expert knowledge can make the task of finding change points clearer to understand. For instance, a physician knows how a blood pressure signal behaves while an economist should understand how stock market prices behave and when the signal is abnormal (does he ?). In particular we're interested in patterns of signals in order to distinguish between different signal regimes. This prior knowledge of patterns in signals can be represented in a general framework with graphs of constraints. This idea was introduced recently by [Hoc+20]. The authors of [Run+22] implemented the idea of [Hoc+20] in a more general case.

2 Method

2.1 HMM Model and Collapsed Graphs

The multiple change-point model as a Hidden Markov Model

We can consider the multiple change-point model as a HMM with a continuous state space, where we consider n latent variables $Z_1, ..., Z_n$ of $\mathbb{R}$ or of an interval I that selects the mean value of the observed variables $Y_1, ..., Y_n$.

In the gaussian case we would have :

$\forall t \in \{1, ..., n\}$, $Y_t = Z_t + \varepsilon_t$ where $\varepsilon = (\varepsilon_t)_t$ is gaussian noise of variance σ^2. And so $(Y_t | Z_t = \mu_t) \sim \mathcal{N}(\mu_t, \sigma^2)$. Figure 1 is a graph representation of the hierarchical model.

Working with a HMM means that we also need a transition kernel that holds the property of homogeneous Markov chains :

$\mathbb{P}(Z_t \in A) = K(Z_{t-1}, A)$, where A is a measurable set and K the Markov Kernel.

We can consider a kernel of the form $k(x, y) \propto I_{x=y} + \exp(-\beta) I_{x \neq y}$. This supposes that the state space is $\mathbb{R}$, but we would like to define more abstract state spaces of the form $S \times \mathbb{R}$ such as $\{Up, Down\} \times \mathbb{R}$.

To do so we can describe the transition kernel as a graph, this is the work of [Hoc+20].

Collapsed graphs as a representation of transition kernels

For **gfpop** the authors implemented collapsed graphs, which are time invariant graphs of constraints that aim to characterize the transitions present in a time dependant process. This is where we reach the current limitations of **gfpop**, it is unable to study graphs that don't have seemingly periodic behaviour, but it is brought forward by the authors as a future advancement for their project.

A collapsed graph of constraints $\mathcal{G}$ is a directed graph defined by the following components:

- Nodes indexed by a state s of an abstract finite state space S. We denote the nodes as the family $v = (v_i)$.
- Edges which represent transitions between nodes this equivalently represents transitions between states, for instance we write the transition from the state s to s' as (s, s') . Additionally, each edge e has attached to it :
  - An indicator function $\mathcal{I}_e : \mathbb{R} \times \mathbb{R} \to \{0, 1\}$, that describes a constraint on successive mean values μ_t, μ_{t+1}. An example would be $\mathcal{I}_e(\mu_t, \mu_{t+1}) = I_{\mu_t > \mu_{t+1}}$ which constrains the means to decrease.
  - A penalty $\beta_e \geq 0$ which is used to regularize the model (larger penalty values result in more costly change-points and thus fewer change-points in the optimal model).
  - γ_e a loss function.
- Special begin and end nodes denoted respectively by # and $\emptyset$.

To see an example check 2 in the appendix. The dashed edges are used when the state doesn't change between to successive time stamps, we detect a change-point when passing through full edges.

We recall that this collapsed graph is time invariant, but when we study a time serie, we consider a path p of length n through the graph that consists of a finite sequence of $n+2$ nodes $(v_0, ..., v_{n+1})$ (with $v_0 = \#$ and $v_{n+1} = \emptyset$), and $n+1$ edges $(e_0, ..., e_n)$, this time indexed by time. Since we study the mean values (μ_t) of the variables (Y_t) that generate the observed trajectory (y_t), we make sure that the vector $\mu = (\mu_1, ..., \mu_n) \in \mathbb{R}^n$ validates the path, meaning that $\forall t$, $\mathcal{I}_{e_t}(\mu_t, \mu_{t+1}) = 1$. We write $p(\mu)$, when μ checks the path p, and $\mathcal{G}_n$ the collection of paths of length n.

In this setting we can then write the implicit transition kernel :

- $k((s,\mu_t),(s',\mu_{t+1})) = \exp(-\beta_{e_t})\mathcal{I}_{e_t}(\mu_t,\mu_{t+1})$, if there is an edge $e_t = (s,s')$ in the graph;
- $k((s,\mu_t),(s',\mu_{t+1})) = 0$ if there is no edge $e_t = (s,s')$ in the graph.

For instance we can interpret $\exp(-\beta_{e_t}) \in]0,1]$ as the probability of passing from the state s to s' at time t. But this is not exactly the case since we have no obligation of summing the penalties exiting a node to 1.

2.2 Optimization problem

Once we have defined properly the HMM model with a proper transition kernel represented by a collapsed graph of constraints we wish to solve the multiple-change point detection problem. This consists of estimating when there is a state change (in the graph) when studying the signal and estimating the mean values. For this the authors share an optimization problem that can be solved by dynamical programming as seen in the course with other change-point detection methods. We thus consider a function Q_n^s of the mean and state of the last studied observation :

$$Q_n^s(\theta) = \min_{\substack{p=(v,e)\in\mathcal{G}_n \\ \mu|p(\mu) \\ \mu_n=\theta, v_n=s}} \sum_{t=1}^{n} (\gamma_{e_t}(y_t,\mu_t) + \beta_{e_t}) \tag{1}$$

This expression is well suited for dynamical programming, and consists of an optimization over the path and the mean values, giving us when solved an array of the time indexes of the change points, as well as the mean values at each index and finally the corresponding path of states.

Update rule

To solve this problem we recall that we rely on dynamical programming which is a setting where when solving at step $n+1$ we have at disposal the optimal solutions up to step n. So we consider a step by step approach with an update rule given by :

$$Q_{n+1}^{s'}(\theta) = \min_{s|\exists edge(s,s')} O_n^{s,s'}(\theta) + \gamma_{(s,s')}(y_{n+1},\theta) + \beta_{(s,s')} \tag{2}$$

where $O_n^{s,s'}(\theta)$ is the best (θ', s) to reach (θ, s'), expressed as $O_n^{s,s'}(\theta') = \min_{\theta'|\mathcal{I}_{(s,s')}(\theta',\theta)=1} Q_n^s(\theta')$.

2.3 gfpop functionalities and usage

In this section we will share how to gt started with the gfpop package in R. We will explain and show how to build your own collapsed graphs and execute the method to find the change points of a signal given the chosen graph.

Graph construction

In an intuitive way we can add nodes and edges to a graph complex in gfpop. To do so we just have to add edges, gfpop will add the nodes on it's own, edges have the following format :

```
R> e <- Edge(state1, state2, type, penalty, decay, gap)
```

where we give, a string for the **state** names, a string in {"null", "std", "up", "down" or "abs"}for the **type** that gives the constraint type (e.g "std" corresponds to $I_{\mu_{t+1}\neq\mu_t}$), a float for the **penalty** value. Additionally we can add a float in [0,1] for the **decay** value corresponding to α in the constraint $I_{\mu_{t+1}=\alpha\mu_t}$ (we take "null" as the **type** for this), a **gap** value between the successive mean, a threshold **K** for the biweight and Huber losses and finally a slope **a** for the Huber robust loss.

We can also choose the edges for the Start and End nodes and add to a node a constraint for the mean values to pass into a state, accepting only values in a given interval. As an example look at 3 in the appendix as the implementation of the multi-modal regression graph.

Change point detection

Some words on data structures used to implement gfpop

The optimization problem solved by the gfpop package requires to use smart data structure for representing cost functions. It's impossible to solve this problem in the general case. Cost functions should have nice properties i.e decomposable on a basis. This is why we consider only classical losses such as ℓ^2 and Poisson costs. Both can be represented on a basis of three functions.

$$\begin{aligned}\text{L2 decomposition}: \ & f_1 : \theta \mapsto 1, f_2 : \theta \mapsto \theta, f_3 : \theta \mapsto \theta^2 \\ \text{Lin-log decomposition}: \ & f_1 : \theta \mapsto 1, f_2 : \theta \mapsto \theta, f_3 : \theta \mapsto \log\theta\end{aligned}$$

$$\begin{aligned}\gamma_{\ell^2}(y,\mu) &= (y-\mu)^2 = y^2 - 2y\mu + \mu^2 = y^2 f_1(\mu) - 2y f_2(\mu) + f_3(\mu) \\ \gamma_{\text{Poisson}}(y,\mu) &= -\log\left(e^{-\mu}\frac{\mu^y}{y!}\right) = \log(y!) f_1(\mu) + f_2(\mu) - y f_3(\mu)\end{aligned}$$

Then, the cost functions defined recursively by the dynamic programming algorithm are piecewise functions and can be represented ([Hoc+20]) as a list of tuples $(\underline{u}_n, \overline{u}_n, [c_1^n, c_2^n, c_3^n])_n$ where :

1. $\overline{u}_k = \underline{u}_{k+1}$
2. c_1^k, c_2^k, c_3^k are the coefficient of the cost functions on the interval $[\underline{u}_k, \overline{u}_k]$.

Then it's straightforward to get the minimum. This data structure also allows to add a constant to a cost function and to take the sum of two cost functions.

3 Data

The Data section (indicative length : 1 page) should provide a deep analysis of the data used for experiment. In particular, we are interested here in your capacity to provide relevant and thoughtful feedbacks on the data and to demonstrate that you master some "data diagnosis" tools that have been dealt with in the lectures/tutorials.

Synthesised data with gfpop

gfpop has a framework where we can easily synthesise signals with change-points, where all we need to do is give it a number of points, an array of proportion indexes in $[0, 1]$ that will place the change point locations accordingly, an array representing a distribution of values associated to each interval between the change-points, and a **type** from { "mean"(Gaussian mean model), "poisson",

"exp", "variance" or "negbin"} that determines the behaviour of the signal in these intervals. We can also choose the standard-deviation for the Gaussian mean case with the keyword **sigma**.

See 4 for an example associated to our Up-Up-Down Gaussian mean toy model.

ECG signals from the Physionet database

The paper explains how the gfpop package could be use for detecting the QRS complex on ECG signals. It provides also an example where classical algorithm fails to detect QRS complex while the gfpop algorithm succed. However the QRS detection with gfpop package requires to tune some hyperparameters (gap between jump in the signal).

ECG signals are taken from the Physionet database available at https://www.physionet.org/content/challenge-2015/1.0.0/. They have already been preprocessed with a FIR bandpass filter and resampled at 250 Hz.

Other data from www.data.gouv.fr

We were able to use some real signals available on the French government website : www.data.gouv.fr. In particular we found electrical consumption data (public lighting energy consumption by city) and hospital data (number of hospitalisations for Covid-19 since 2020 by county).

4 Results

Results on synthetic data

As we can see in Figure 10 we are able to detect, when choosing an adapted penalty value, the correct change points with the Up-Up-Down model described in Figure 5. This figure also shows the importance of choosing the right penalty values, to have meaningful results. The hard part here is to determine the appropriate penalties. One can think about the Bayesian Information Criterion (BIC) or simply find the best one with a grid search.

We also synthesised corrupted data following a generation rule given by the authors : $Y_i = X_i s_i + \varepsilon_i$, where $X_i \sim \mathcal{B}(p) \in \{-1, 1\}$, s_i is the signal data point and ε is Gaussian white noise.

This time we had to use an isotonic graph where we only allow going up. The subtlety here was to consider the biweight loss 9 and take the correct penalty and loss parameter. Knowing the real standard deviation σ behind the signal noise we took : $\beta = 2\sigma^2 \log(n), K = 4\sigma^2$.

Results on ECG signal

Although the database is not annotated with QRS complexes, we can give some qualitative interpretation of the results. Again, the difficulty is to tune the hyperparameters. The graph described in the article works well for their ECG signal but fails to detect QRS on a random ECG signal from Physionet database. Size and amplitudes of QRS and waves vary a lot from one patient to another. We can't take the same graph for every signals.

4.1 Other results

We have also plotted the obtained graphs in 11 and 12 for the www.data.gouv.fr data, as well as given a short description of what is obtained.

References

[Hoc+20] Toby Dylan Hocking et al. "Constrained Dynamic Programming and Supervised Penalty Learning Algorithms for Peak Detection in Genomic Data". In: *Journal of Machine Learning Research* 21.87 (2020), pp. 1–40. URL: http://jmlr.org/papers/v21/18-843.html.

[Run+22] Vincent Runge et al. *gfpop: an R Package for Univariate Graph-Constrained Change-Point Detection*. 2022. arXiv: 2002.03646 [stat.CO].

[TOV18] Charles Truong, Laurent Oudre, and Nicolas Vayatis. "A review of change point detection methods". In: *CoRR* abs/1801.00718 (2018). arXiv: 1801.00718. URL: http://arxiv.org/abs/1801.00718.

Appendix

A HMM hierarchical representation

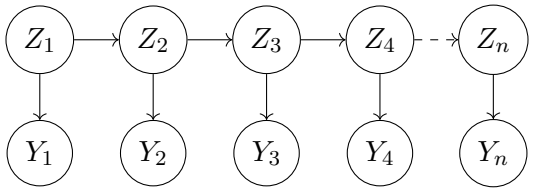

Figure 1: Hierarchical representation of the multiple change-point model

B Examples with the graph for multi-modal regression

As an illustration, this is the graph associated to multi-modal regression :

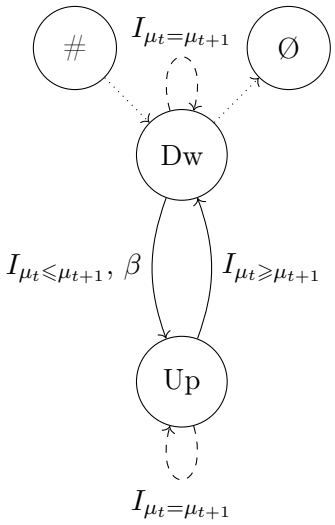

Figure 2: Collapsed graph for multi-modal regression

```
graph_spike <-  graph(
                    Edge(state1 = "Up", state2 = "Dw", type="down"),
                    Edge(state1 = "Dw", state2 = "Up", type="up", penalty=beta),
                    Edge(state1 = "Dw", state2 = "Dw", type="null"),
                    Edge(state1 = "Up", state2 = "Up", type="null"),
                    StartEnd(start="Dw", end=c("Up", "Dw"))
                )
```

Figure 3: **gfpop** implementation of the graph for multi-modal regression

C A toy signal : the Up-Up-Down model

```
n <- 1000
sigma=0.1
myData <- dataGenerator(n, c(0.1,0.2,0.3,0.4,0.5, 0.6, 0.7, 0.8, 0.9, 1),
                        c(1,2,3,1,1.5, 10, -2, -1, 0, -5), type="mean", sigma = sigma)
```

Figure 4: **gfpop** synthesised data for the Up-Up-Down Gaussian model, we can see the generated data points in figure 10.

```
beta = 10
g <-  graph(
             Edge(state1 = "Dw", state2 = "Dw", type="null"),
             Edge(state1 = "Up", state2 = "Up", type="null"),
             Edge(state1 = "Up", state2 = "Up", penalty=beta, type="up"),
             Edge(state1 = "Up", state2 = "Dw", penalty=beta, type="down"),
             Edge(state1 = "Dw", state2 = "Up", penalty=beta, type="up"),
             StartEnd(start="Dw", end=c("Up", "Dw"))
        )
```

Figure 5: **gfpop** representation of the Up-Up-Down Model graph.

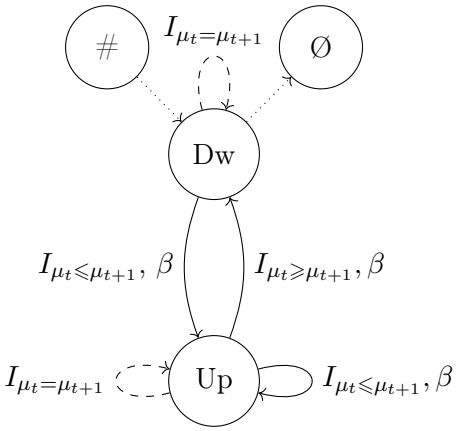

Figure 6: Collapsed graph for the Up-Up-Down model

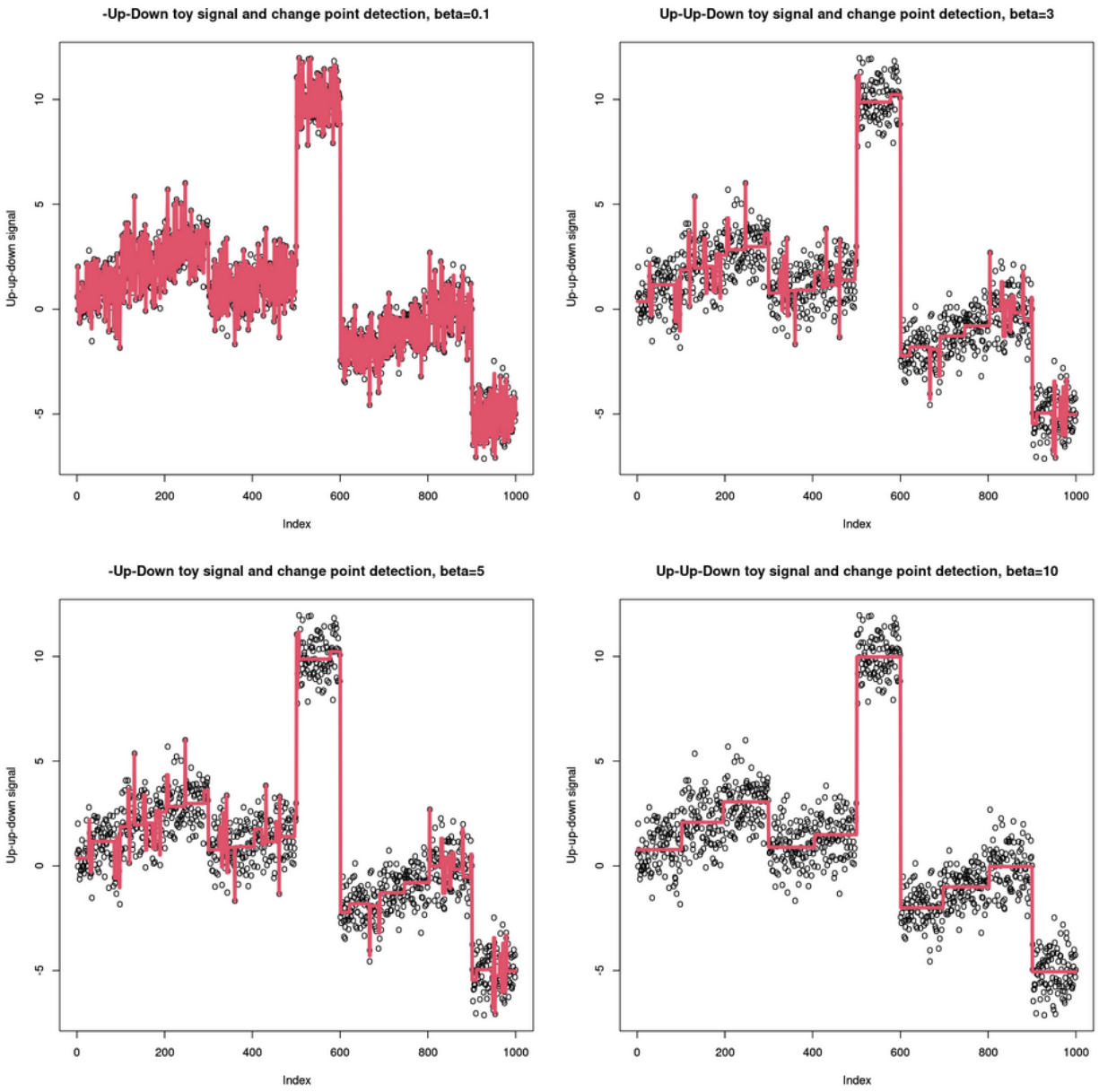

Figure 7: influence of the penalty β on the number of change points

D Finding change points in corrupted data

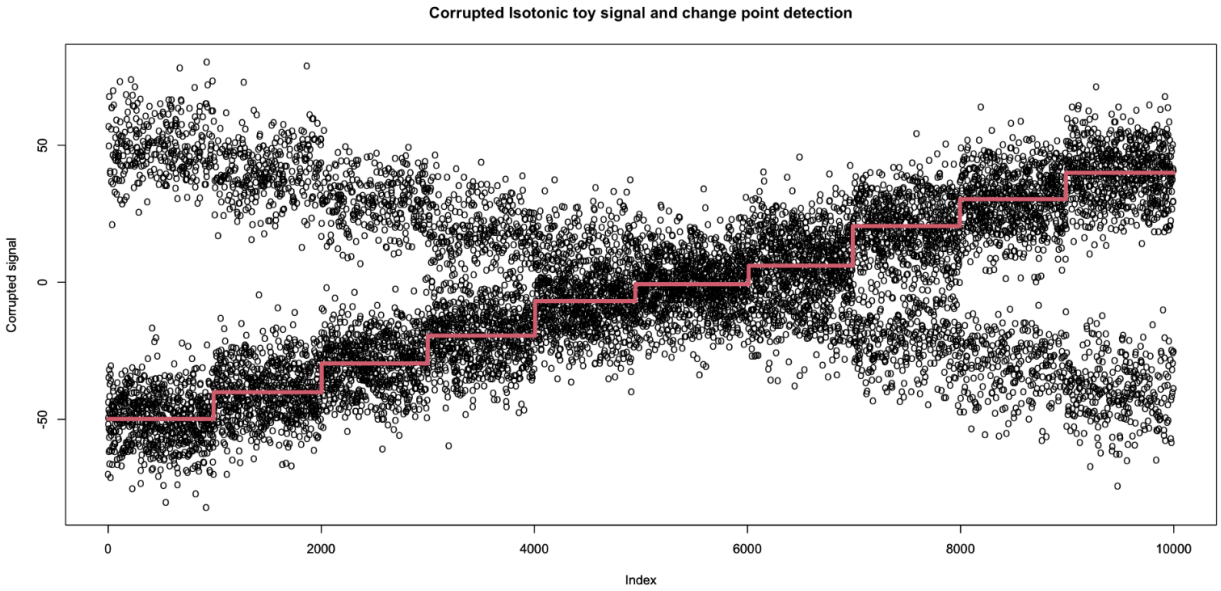

Figure 8: Isotonic regression for finding change points. There are 10^4 data points, the probability of a data point being corrupted is $1 - p = 0.3$, the variance is $\sigma = 10$.

$$\ell(r) = \begin{cases} \frac{K^2}{6}\left(1 - \left[1 - \left(\frac{r}{K}\right)^2\right]^3\right) & \text{if } |r| \leq K \\ \frac{K^2}{6} & \text{otherwise.} \end{cases}$$

Figure 9: Expression of the Tukey biweight loss of parameter K

E Finding QRS complex in ECG signals

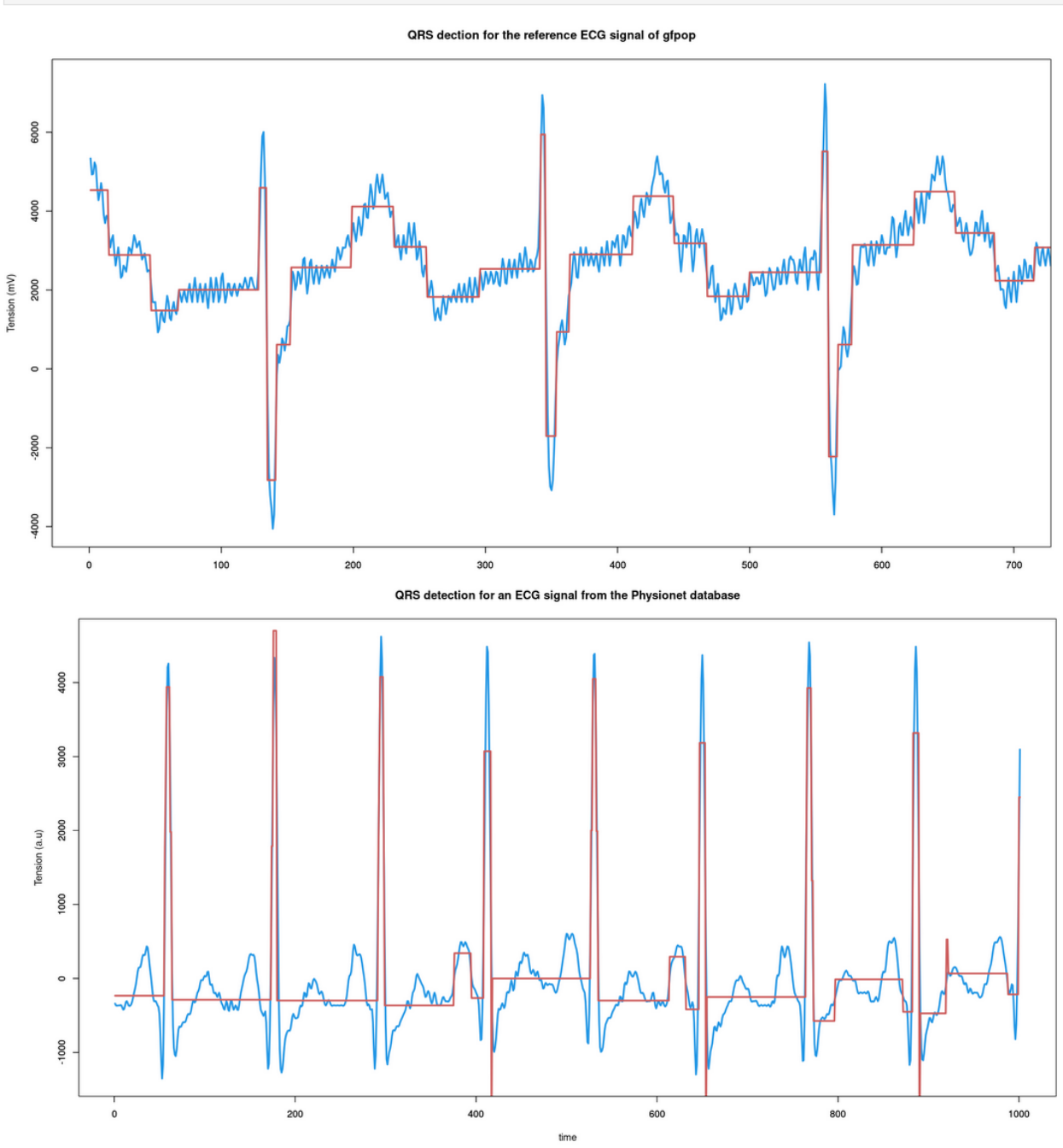

Figure 10: QRS detection in ECG signals. **Above :** the reference ECG signal from gfpop. **Below :** ECG signal from the Physionet database.

F Other Signals

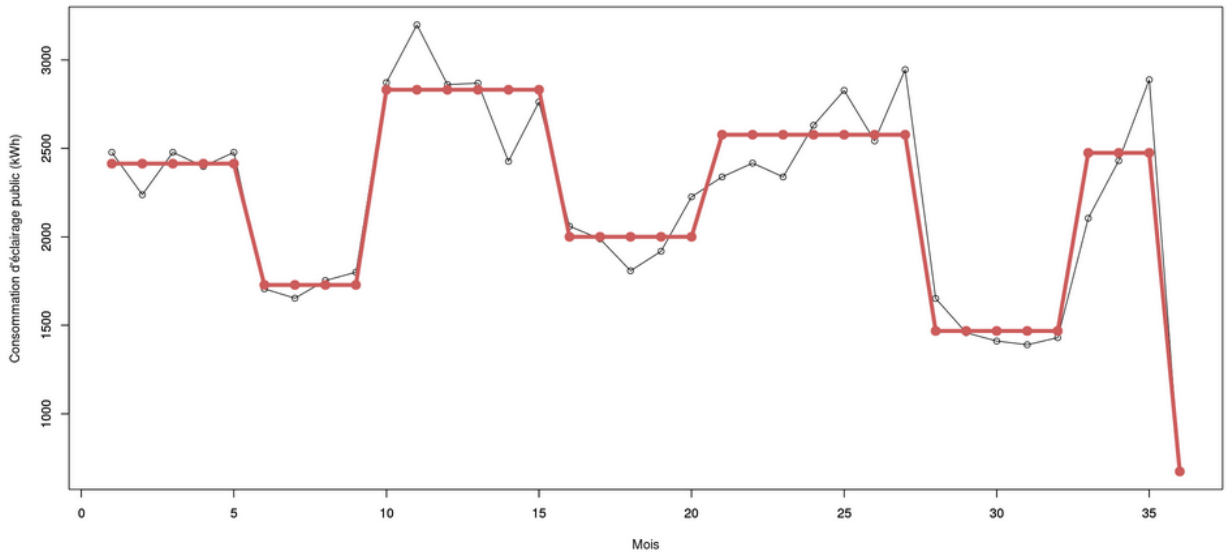

Figure 11: Public lighting energy consumption data by month of Béthune in the street René Coty. The data start in January. We just fit a classical up-down model. In Summer, the consumption is lower.

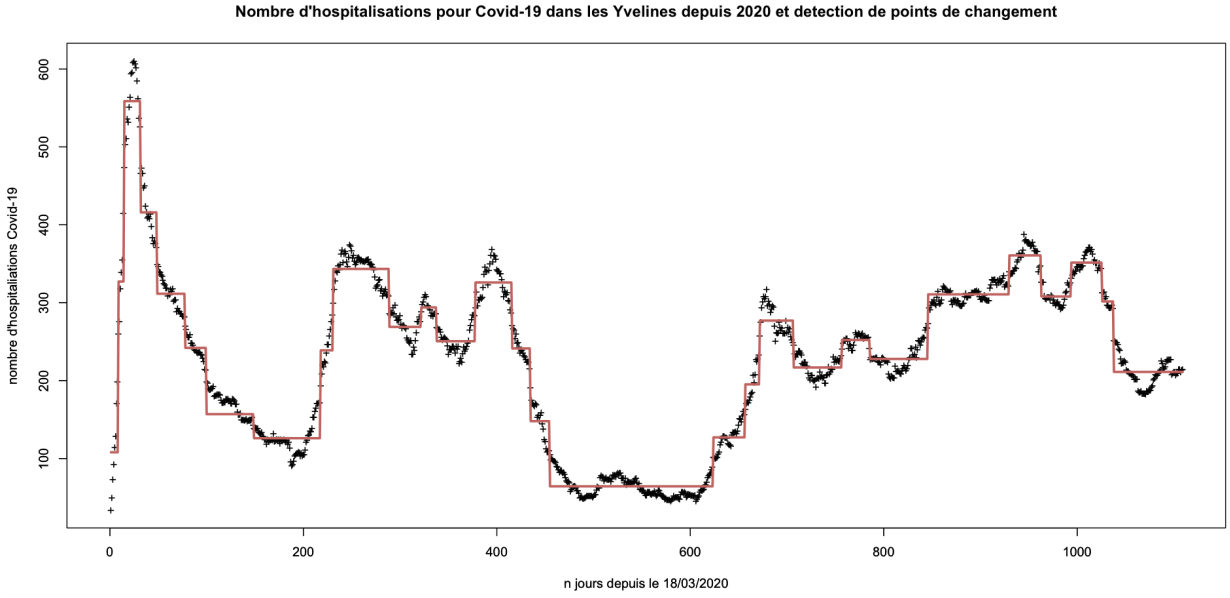

Figure 12: Number of Covid-19 hospitalisations in the Yvelines, France since 2020-18-03. We have used a multi-modal regression graph. We capture well the real spikes, but we also have meaningless change-points. A graph allowing slopes such as exponential growth/decay would be better and not have the stair like behavior imposed by the graph we chose, but **gfpop** does not include exponential growth.