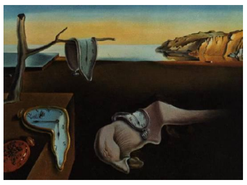

Journal of Statistical Software

March 2023, Volume 106, Issue 6. *doi: 10.18637/jss.v106.i06*

gfpop: An R Package for Univariate Graph-Constrained Change-Point Detection

Vincent Runge
Université d'Évry

Toby Dylan Hocking
Northern Arizona University

Gaetano Romano
Lancaster University

Fatemeh Afghah
Clemson University

Paul Fearnhead
Lancaster University

Guillem Rigaill
INRAE – Université d'Évry

Abstract

In a world with data that change rapidly and abruptly, it is important to detect those changes accurately. In this paper we describe an R package implementing a generalized version of an algorithm recently proposed by Hocking, Rigaill, Fearnhead, and Bourque (2020) for penalized maximum likelihood inference of constrained multiple change-point models. This algorithm can be used to pinpoint the precise locations of abrupt changes in large data sequences. There are many application domains for such models, such as medicine, neuroscience or genomics. Often, practitioners have prior knowledge about the changes they are looking for. For example in genomic data, biologists sometimes expect peaks: up changes followed by down changes. Taking advantage of such prior information can substantially improve the accuracy with which we can detect and estimate changes. Hocking *et al.* (2020) described a graph framework to encode many examples of such prior information and a generic algorithm to infer the optimal model parameters, but implemented the algorithm for just a single scenario. We present the **gfpop** package that implements the algorithm in a generic manner in R/C++. **gfpop** works for a user-defined graph that can encode prior assumptions about the types of changes that are possible and implements several loss functions (Gauss, Poisson, binomial, biweight, and Huber). We then illustrate the use of **gfpop** on isotonic simulations and several applications in biology. For a number of graphs the algorithm runs in a matter of seconds or minutes for 10^5 data points.

Keywords: change-point detection, constrained inference, maximum likelihood inference, dynamic programming, robust losses.

1. Introduction

1.1. Multiple change-point R packages

In the last decade there has been an increasing interest in algorithms for detecting changes in mean. There are a variety of approaches to detecting such change-points, see Truong, Oudre, and Vayatis (2020) for a recent review of the area. Many of these recursively apply a test for a single change-point. These include binary segmentation (Scott and Knott 1974) and its variants (Olshen, Venkatraman, Lucito, and Wigler 2004; Fryzlewicz 2014), multiscale methods (Frick, Munk, and Sieling 2014) and MOSUM (moving sum) methods (Eichinger and Kirch 2018). R (R Core Team 2022) packages that implement these and related methods include **wbs** (Baranowski and Fryzlewicz 2019), **not** (Baranowski, Chen, and Fryzlewicz 2019), **breakfast** (Anastasiou, Chen, Cho, and Fryzlewicz 2021), **stepR** (Pein, Hotz, and Sieling 2022), and **mosum** (Meier, Kirch, and Cho 2021). See Fearnhead and Rigaill (2020) for a comparison of many of these methods.

Alternatively one can try to jointly estimate the location of all change-points by maximizing a penalized likelihood or, equivalently, minimizing a penalized cost. Dynamic programming was originally proposed in the change-point literature in the context of the "segment neighborhood" (SN) and "optimal partitioning" (OP) algorithms (Auger and Lawrence 1989; Jackson *et al.* 2005). More recently Killick, Fearnhead, and Eckley (2012) proposed the PELT (pruned exact linear time) pruning rules, which reduce time complexity from quadratic to linear in asymptotic regimes where the number of change-points increases as we observe more data. This work has stimulated a new interest in these problems. The R package **changepoint** (Killick and Eckley 2014) and **changepoint.np** (Haynes and Killick 2021) are based on these PELT *inequality* pruning rules. A new *functional* pruning rule was independently discovered by Johnson (2013) and Rigaill (2015). When comparing these two pruning rules, the *functional* pruning always prunes more than PELT *inequality* pruning (see Theorem 2 and Figures 4 and 5 in Maidstone, Hocking, Rigaill, and Fearnhead 2017). Furthermore functional pruning empirically shows reduced time complexity in many situations. For example, when we have data with no changes, PELT pruning algorithms have a quadratic complexity, but functional pruning algorithms can have a log-linear complexity (see Section 7 in Maidstone *et al.* 2017). Functional pruning algorithms are implemented in the R packages **fpop** and **fpopw** (Maidstone *et al.* 2017; Rigaill, Hocking, Maidstone, and Fearnhead 2019; Rigaill 2022), **Segmentor3IsBack** (Cleynen, Koskas, Lebarbier, Rigaill, and Robin 2014) and **jointseg** (Pierre-Jean, Rigaill, and Neuvial 2019).

Besides time efficiency, recent efforts have been made to extend the class of change-point models considered by adding constraints. For example, in applications which involve detecting peaks in genomic data, the inference is constrained to return a sequence of down and up segments by packages **PeakSegDP** (Hocking, Rigaill, and Bourque 2015), **PeakSegOptimal** (Hocking *et al.* 2020), **PeakSegDisk** (Hocking, Rigaill, Fearnhead, and Bourque 2022), and **PeakSegJoint** (Hocking and Bourque 2020). Including such constraints, when appropriate for the application, has been shown to substantially improve the accuracy of change-point detection (Hocking *et al.* 2020). Whereas these previous packages only implement up/down constraints and the Poisson loss, the proposed **gfpop** package (Runge *et al.* 2023) is the first to implement inference for a wide range of constraints, allows specifying models that can mix different types of changes through a graph, and also allows for other loss functions. For that reason we named our package **gfpop** as an abbreviation for "generalized functional pruning optimal partitioning". Our intention is to provide a user-friendly package which popularizes these recent discoveries about the functional pruning method, by allowing the user to specify a wide variety of constraints and loss functions, using prior information about their data and application domain. Package **gfpop** is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=gfpop`.

1.2. Standard multiple change-point model

Multiple change-point models are designed to find abrupt changes in a signal. In the standard Gaussian noise model, we have data $Y_{1:n} = (Y_1, \dots, Y_n)$ where each data point, Y_t, is an independent random variable with $Y_t \sim \mathcal{N}(\mu_t, \sigma^2)$ and $t \mapsto \mu_t$ is piecewise constant. The goal is to estimate the number and position of the changes, that is to find all t such that $\mu_t \neq \mu_{t+1}$ from the observed data $(y_t)_{t=1,\dots,n}$. A classical way to proceed is to optimize the log-likelihood by fixing the number of changes. It is also possible to penalize each change by a positive penalty β and minimize in $\mu = (\mu_1, \dots, \mu_n)^\top \in \mathbb{R}^n$ the following least-ssquares criterion:

$$Q_n^{std}(\mu) = \sum_{t=1}^{n} (y_t - \mu_t)^2 + \beta \sum_{t=1}^{n-1} I_{\mu_t \neq \mu_{t+1}} \ ,$$

where $I \in \{0, 1\}$ is the indicator function ($I_x = 1$ if x is true, and zero otherwise). In both cases, fast dynamic programming algorithms can solve the related optimization problem exactly (Killick *et al.* 2012; Rigaill 2015; Maidstone *et al.* 2017). In Section 3.2 we derive the explicit form of the simpler FPOP (functional pruning optimal partitioning) update-rule for our more general graph framework.

1.3. Constrained multiple change-point model

In many applications, it is desirable to constrain the parameters of successive segments (Hocking *et al.* 2015; Maidstone *et al.* 2017; Jewell, Hocking, Fearnhead, and Witten 2020; Baranowski and Fryzlewicz 2019). This means that the μ parameter is restricted by inequalities to a subset of $\mathbb{R}^n$. Arguably, the simplest and most studied case is isotonic regression (Barlow, Bartholomew, Bremner, and Brunk 1972). In this case the goal is to minimize in μ the constrained least-squares criterion:

$$Q_n^{\text{iso}}(\mu) = \sum_{t=1}^{n} (y_t - \mu_t)^2 \ , \text{ subject to the constraint } \mu_t \leq \mu_{t+1}, \ \forall\, t \in \{1, \dots, n-1\}.$$

The obtained estimator is piecewise constant, which establishes the link with the multiple change-point problem. Several efficient inference algorithms have been proposed (Best and Chakravarti 1990; Johnson 2013; Gao, Han, and Zhang 2020), and the **isotone** package provides an implementation (De Leeuw, Hornik, and Mair 2010).

More generally, we may want to impose more complex patterns, such as unimodality (Stout 2008) or a succession of up and down changes (Hocking *et al.* 2015) to detect peaks. There are very efficient algorithms for the isotonic and unimodal cases (Best and Chakravarti 1990; Stout 2008) at least if the number of changes is not penalized. For more complex constraints like the up-down pattern, Hocking *et al.* (2020) proposed an exact algorithm. This algorithm is a generalization of the functional dynamic programming algorithm of Rigaill (2015) and

Maidstone *et al.* (2017). Variants of this algorithm allow penalizing or constraining the number of changes. Other variants allow robust losses, including the biweight loss, instead of the least-squares criterion for assessing fit to the data. In the case of non-constrained (standard) multiple change-point detection, the biweight loss has good statistical properties (Fearnhead and Rigaill 2019). The simulations of Bach (2018) in the context of isotonic regression also show the benefit of such losses.

1.4. Contributions

Hocking *et al.* (2020) described a graph-based framework to encode prior constraints on how parameters change at each change-point, and a generic algorithm to infer the optimal model parameters. However, they only implemented the algorithm for a single scenario (Poisson loss and up/down constraints). The **gfpop** package implements their algorithm in a generic manner in R/C++, for user-defined graphs and several loss functions.

1.5. Outline

In Section 2 we formally define the graphs and explain their connection to hidden Markov model (HMM). We also provide numerous graph examples to illustrate the versatility of our framework. In Section 3 we present the optimization problem solved by our package. In Section 4 we go through the main functions of the package. We illustrate in Section 5 the use of our package on various real data sets. Finally, using simulations we compare in Section 6 the results of our package with those of standard isotonic packages and show the benefit of using robust losses and penalizing the number of segments.

2. Constraint graphs and change-points model as a HMM

We begin by recasting the standard and constrained multiple change-point problem as a continuous HMM with a particular transition kernel represented as a graph (Johnson 2013). At each time t the signal can be in a number of states, which are nodes of the graph. Possible transitions between states at time t and $t+1$ are represented by the edges of the graph. Each edge has three properties: a constraint (e.g., $\mu_t \leq \mu_{t+1}$), a penalty (possibly null) and a loss function (cost associated to a data point). In **gfpop** the set of transitions is constant over time, leading to a collapsed representation of the graph. We then formalize the concept of a valid signal or path, that is one satisfying all constraints, and finally present a number of examples.

2.1. Transition kernel and graph of constraints

Standard multiple change-point model as a HMM. It is possible to recast the classic multiple change-point model as a hidden Markov model with a continuous state space. Precisely, we define random variables $Z_1, \ldots, Z_n$ in $\mathbb{R}$ or some interval $[a, b]$. We consider a transition kernel $k(x, y) \propto I_{x=y} + e^{-\beta} I_{x \neq y}$. Finally, in the Gaussian case, observations are obtained as $(Y_i | Z_i = \mu) \sim \mathcal{N}(\mu, \sigma^2)$. The Bayesian network of this model is given in Figure 1.

Here the state space is $\mathbb{R}$, the set of values that the mean can take. The `gfpop` algorithm of Hocking *et al.* (2020) allows one to consider a more complex state space in $\mathcal{S} \times \mathbb{R}$ where $\mathcal{S}$ is

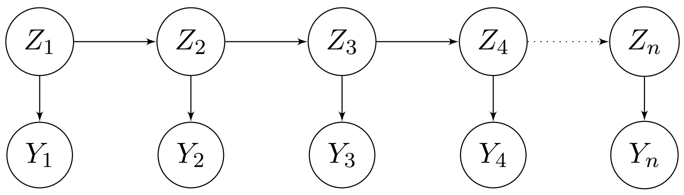

Figure 1: Multiple change-point model as a hidden Markov model.

a finite set. In that case the transition kernel is more complex and can be described using a graph. Below, we first present the graph and then explain how it is linked to the transition kernel.

Graph of constraints. The graph of constraints $\mathcal{G}_n$ is an acyclic directed graph defined as follows.

1. Nodes are indexed by time $t \in \{1, \dots, n\}$ and a state $s \in \mathcal{S} = \{1, \dots, S\}$;
2. We include two undefined states $\#, \emptyset$ for the starting nodes, $v_0 = (0, \#)$, and arrival nodes, $v_{n+1} = (n+1, \emptyset)$;
3. Edges are transitions between consecutive "time" nodes of type $v = (t, s)$ and $v' = (t+1, s')$. Edges e are then described by a triplet $e = (t, s, s')$ for $t \in \{0, \dots, n\}$;
4. Each edge $e = (t, s, s')$ is associated with:
    - An indicator function $\mathcal{I}_e : \mathbb{R} \times \mathbb{R} \to \{0, 1\}$ constraining successive means[1] μ_t and μ_{t+1}. For example an edge e with the corresponding indicator function $\mathcal{I}_e(\mu_t, \mu_{t+1}) = I_{\mu_t \leq \mu_{t+1}}$ ensures that means are non-decreasing; while an edge with indicator function $\mathcal{I}_e(\mu_t, \mu_{t+1}) = I_{\mu_t = \mu_{t+1}}$ would correspond to no change.
    - 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).
    - A loss function γ_e for data point y_{t+1}[2].

Transition kernel. Coming back to our HMM representation of change-point models, the transition from state (s, μ_t) to (s', μ_{t+1}) (up to proportionality) is

- $k((s, \mu_t), (s', \mu_{t+1})) = \exp(-\beta_e)\mathcal{I}_e(\mu_t, \mu_{t+1})$, 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.

Some simple examples. In Figures 2 and 3 we provide the corresponding graphs for the standard and isotonic models. Notice that the only difference is that the transitions between nodes $(t, 1)$ and $(t+1, 1)$ are restricted to non-decreasing means in the isotonic case.

[1]We call this parameter a mean for convenience but some models consider changes in variance or in other natural parameters.

[2]The loss function can be edge-specific; see Figure 17 and the graph construction in Section 4.1.

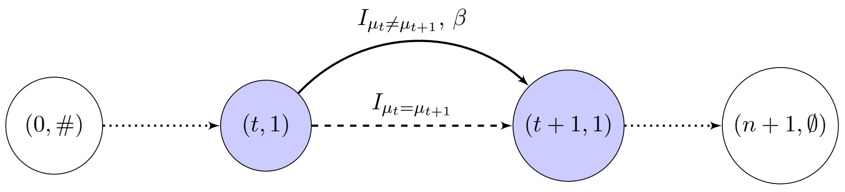

Figure 2: Graph of constraints for the standard multiple change-point model. We have $\mathcal{S} = \{1\}$, the loss function is always the ℓ_2 (Gaussian loss). The penalty is omitted when equal to zero.

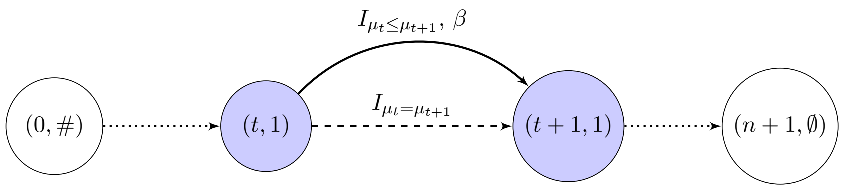

Figure 3: Graph of constraints for the isotonic change-point model. We have $\mathcal{S} = \{1\}$, the loss function is always the ℓ_2. The penalty is omitted when equal to zero.

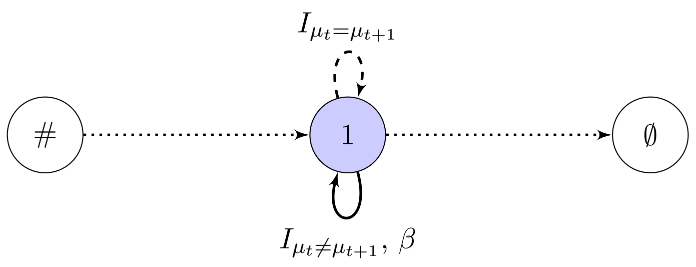

Figure 4: Collapsed graph of constraints for the standard multiple change-point model. We have $\mathcal{S} = \{1\}$. The loss function is always the ℓ_2. The penalty is omitted when equal to zero.

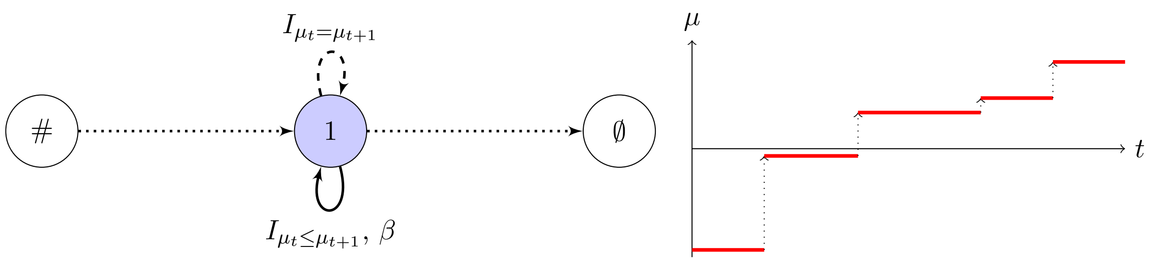

Figure 5: Left: Collapsed graph of constraints for the isotonic change-point model. We have $\mathcal{S} = \{1\}$, the loss function is always the ℓ_2. The penalty is omitted when equal to zero. Right: In red, a piecewise constant function validating the graph of constraints.

2.2. Collapsed graph of constraints

In **gfpop** we only consider transitions that do not depend on time. We then collapse the previous graph structure. To be specific, we have a single node for each s and a transition from node s to s' if there is a transition from (t, s) to $(t+1, s')$ in the full graph structure. In Figures 4 and 5 we provide the corresponding collapsed graphs for the standard and isotonic models.

Path and constraints validation. In Section 3 we show how we can estimate the changes by minimizing a penalized loss equal to the loss associated with the path of μ_t values plus the sum of the penalties for the edges used. This can be viewed as a maximum a-posteriori estimate based on the kernel associated with each edge and the likelihood associated with each observation. To define this maximum properly we formalize the notion of a signal validating our constraints through the concept of a valid path in the collapsed graph.

A path p of the collapsed graph $\mathcal{G}_n$ is a collection of $n+2$ nodes $(v_0, \ldots, v_{n+1})$ with $v_0 = (0, \#)$, $v_{n+1} = (n+1, \emptyset)$ and $v_t = (t, s_t)$ for $t \in \{1, \ldots, n\}$ and $s_t \in \{1, \ldots, S\}$. In addition, the path is made of $n+1$ edges named $e_0, \ldots, e_n$. Recall that each edge e_t is associated to a penalty β_{e_t}, a loss γ_{e_t} and a constraint $\mathcal{I}_{e_t}$. A vector $\mu \in \mathbb{R}^n$ validates the path p if for all $t \in \{1, \ldots, n-1\}$, we have $\mathcal{I}_{e_t}(\mu_t, \mu_{t+1}) = 1$ (true). We write $p(\mu)$ to say that the vector μ checks the path p.

The starting and arrival edges e_0 and e_n are exceptions. The starting edge is associated with a loss (γ_{e_0}) for the first observation but is not associated with a penalty or a constraint and the arrival edge is not associated with a loss, a penalty, or a constraint.

Definition. From now on when we use the word "graph" we mean the collapsed graph of constraints. In this graph, the triplet notation (t, s, s') for edges is replaced by (s, s'). We remove the time dependency also for edges associated with starting and arrival nodes to simplify notations (even if in that case, there is a time dependency).

2.3. A few examples

We present a few constrained models and their graphs. Some models have been already proposed in the literature, but not necessarily using our HMM formalism.

- (Up-down) To model peaks Hocking *et al.* (2015) proposed an up-down constraint using two states $\mathcal{S} = \{Up, Dw\}$. Transitions from Dw to Up are forced to go up $I_{\mu_t \leq \mu_{t+1}}$. Transitions from Up to Dw are forced to go down $I_{\mu_t \geq \mu_{t+1}}$. The graph of this model is given in Figure 6.

- (Up-exponentially down) To model pulses Jewell *et al.* (2020) proposed a model where the mean decreases exponentially between positive spikes. In that case a unique state with two transitions is sufficient. The first transition corresponds to an up change $I_{\mu_t \leq \mu_{t+1}}$ and the second to an exponential decay $I_{\alpha\mu_t = \mu_{t+1}}$ with $0 < \alpha < 1$. The graph of this model is given in Figure 7.

- (Segment neighborhood) One often considers a known number of segments, say D (Auger and Lawrence 1989). This is encoded by a graph with D states, $\mathcal{S} = \{1, \ldots, D\}$. From any $d \in \mathcal{S}$ there are two transitions to consider. One from d to d with constraint $I_{\mu_t = \mu_{t+1}}$ and one from d to $d+1$ with constraint $I_{\mu_t \neq \mu_{t+1}}$. The graph of this model for $D = 3$ is given in Figure 8.

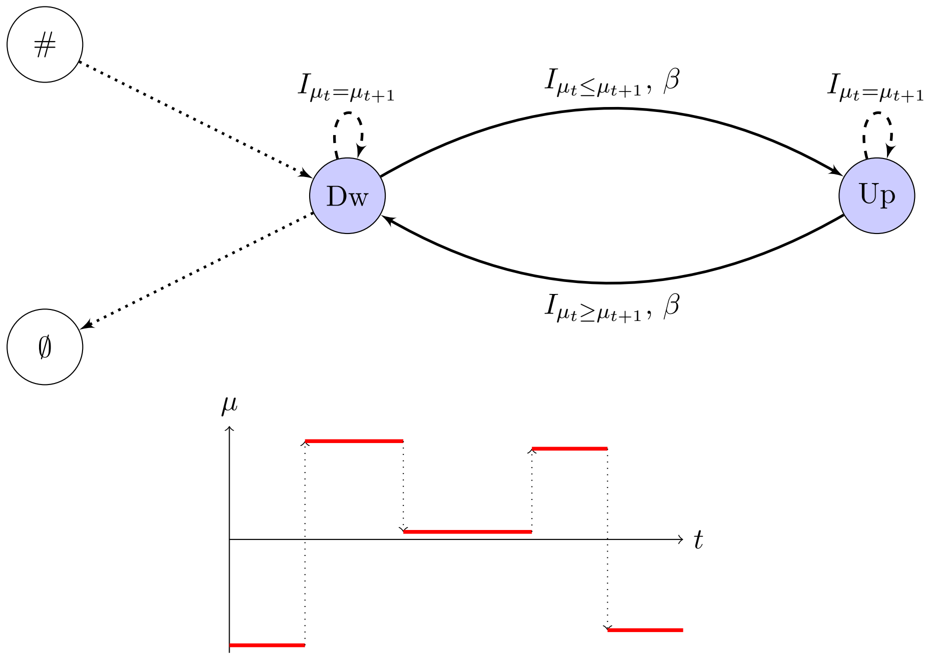
#
Dw
Up
∅
$I_{\mu_t=\mu_{t+1}}$
$I_{\mu_t\le\mu_{t+1}}$, β
$I_{\mu_t=\mu_{t+1}}$
$I_{\mu_t\ge\mu_{t+1}}$, β
μ
t

Figure 6: Top: Graph for the up-down change-point model proposed in Hocking *et al.* (2015). We have $\mathcal{S} = \{Up, Dw\}$, the loss function is always the ℓ_2. The penalty is omitted when equal to zero. Bottom: In red, a piecewise constant function validating the graph of constraints. The penalty is omitted when equal to zero.

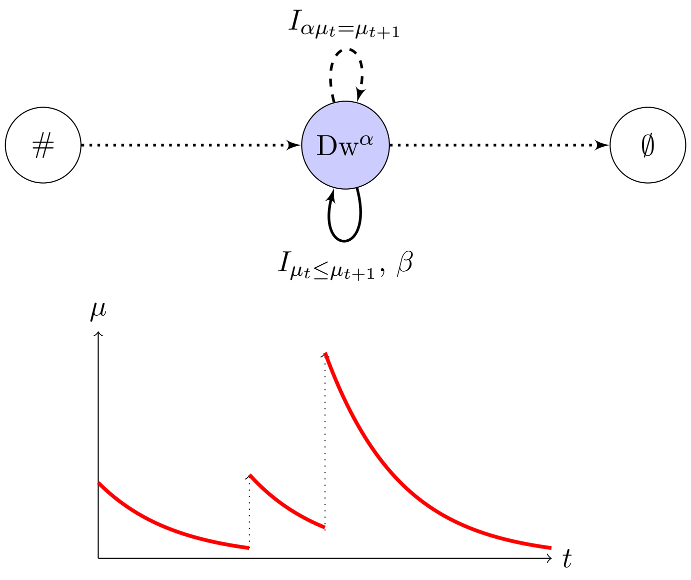

Figure 7: Top: Graph for the up-exponential decrease change-point model proposed in Jewell *et al.* (2020). We have $\mathcal{S} = \{Dw^\alpha\}$, the loss function is always the ℓ_2. The penalty is omitted when equal to zero. Bottom: In red, a function validating the graph of constraints.

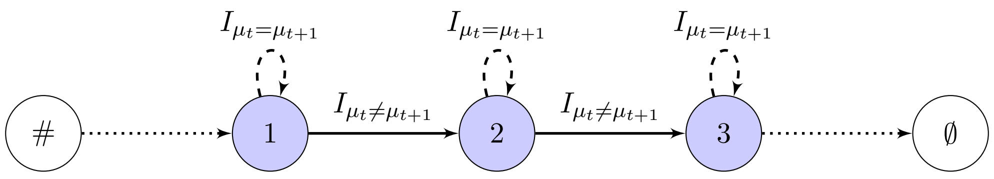

Figure 8: Graph for the 3-segment change-point model. We have $\mathcal{S} = \{1, 2, 3\}$, the loss function is always the ℓ_2. The penalty is always equal to zero.

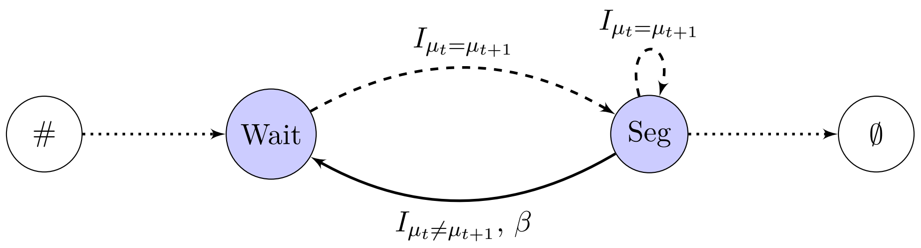

Figure 9: Graph for the at least 2 data points per segment change-point model. We have $\mathcal{S} = \{Wait, Seg\}$, the loss function is always the ℓ_2. The penalty is omitted when equal to zero.

- (At least 2 data points per segment) It is often desirable to impose a minimum segment length. For at least 2 data points one should consider two states $\mathcal{S} = \{Wait, Seg\}$. There are 3 transitions to consider: one from *Seg* to *Wait* with the constraint $I_{\mu_t\neq\mu_{t+1}}$, one from *Seg* to *Seg* with $I_{\mu_t=\mu_{t+1}}$ and one from *Wait* to *Seg* with $I_{\mu_t=\mu_{t+1}}$. The graph of this model is given in Figure 9. This can be extended to p data points. The graph for at least 3 data points per segment is given in Appendix A (Figure 18).

  In Appendix A we provide a few more examples. In particular, we reformulate the collective anomaly model of Fisch, Eckley, and Fearnhead (2022) as a constrained model.

3. Optimization problem solved by gfpop

3.1. Penalized maximum likelihood

We now present the constrained change-point optimization problem. The goal is to minimize the negative log-likelihood over all model parameters that validate the constraints (see Section 2.2):

$$Q_n = \min_{\substack{p=(v,e)\,\in\,\mathcal{G}_n \\ \mu|p(\mu)}} \left\{ \sum_{t=1}^{n} (\gamma_{e_t}(y_t, \mu_t) + \beta_{e_t}) \right\} .$$

This is a discrete optimization problem. A naive exploration of the 2^{n-1} change-point positions is not feasible in practice. Due to the constraints, segments are dependent and Q_n cannot be written as a sum over all segments. Therefore the algorithms of Auger and Lawrence (1989); Jackson *et al.* (2005) and Killick *et al.* (2012) are not applicable.

Hocking *et al.* (2020) have shown that it is possible to optimize Q_n using functional dynamic programming techniques. The idea is to consider the quantity Q_n as a function of the mean and the state of the last data point:

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

where we use the subscript n to denote the number of data points analyzed, s to denote the state of the most recent transition, and θ the mean of the last data point.

By construction, each Q_n^s is a piecewise function and can be defined as the pointwise minimum of a finite number of functions, with the form of these functions depending on the loss used. In the package three analytical decompositions for the pieces of Q_n^s are implemented:

L2 decomposition. $f_1 : \theta \mapsto 1$, $f_2 : \theta \mapsto \theta$ and $f_3 : \theta \mapsto \theta^2$. This decomposition allows one to consider Gaussian (least-squares), biweight and Huber loss functions;

Lin-log decomposition. $f_1 : \theta \mapsto 1$, $f_2 : \theta \mapsto \theta$ and $f_3 : \theta \mapsto \log(\theta)$. This decomposition allows one to consider loss functions for Poisson and exponential models. It is also possible to consider a change in the variance of a Gaussian distribution of mean 0[3];

Log-log decomposition. $f_1 : \theta \mapsto 1$, $f_2 : \theta \mapsto \log(\theta)$ and $f_3 : \theta \mapsto \log(1-\theta)$. This decomposition allows one to consider loss functions for the binomial and negative binomial likelihoods.

As in the Viterbi algorithm for finite state space HMMs, it is possible to define an update formula linking the set of functions $\{\theta \mapsto Q_{n-1}^s(\theta)\,,\, s \in \mathcal{S}\}$ to $\theta \mapsto Q_n^{s'}(\theta)$ for all states s'. Computationally, the update is applied per interval using some edge-dependent operators described in Section 3.2 (Rigaill 2015; Maidstone *et al.* 2017; Hocking *et al.* 2020).

3.2. Operators and update-rule for `gfpop`

Let us consider a transition from s to s' at step n. Its edge is (s, s') and its associated constraint is $I_{(s,s')}$. The `gfpop` algorithm involves calculating the best (θ, s) to reach state (θ', s'), i.e., minimizing the functional cost while satisfying the constraint $I_{(s,s')}$. Formally this is defined as an operator:

$$O_n^{s,s'}(\theta') = \min_{\theta | I_{(s,s')}(\theta,\theta')} \{Q_n^s(\theta)\}\,.$$

Operator calculation. For a general constraint I and a general function Q_n^s it is not easy to compute $O_n^{s,s'}$. Recall that $Q_n^s(\theta)$ are piecewise analytical, i.e., they can be exactly represented by a finite set of real-valued coefficients. For algorithmic simplicity Hocking *et al.* (2020) requires that $O_n^{s,s'}(\theta)$ has the same analytical decomposition per interval (L2, lin-log or log-log).

In practice here are the constraints we can accommodate.

[3]To be clear, in that case the log-likelihood is $\frac{1}{2}\log(\frac{1}{\sigma^2}) - \frac{y_t}{2\sigma^2}$ and we get the lin-log decomposition by taking $\theta = \frac{1}{\sigma^2}$.

L2 decomposition: any linear constraint, e.g., $a\mu_t + b\mu_{t+1} + c \leq 0$ or $a\mu_t + b\mu_{t+1} + c = 0$;

Lin-log decomposition: any proportional constraint, e.g., $a\mu_t \leq \mu_{t+1}$ or $a\mu_t = \mu_{t+1}$;

Log-log decomposition: only the two inequalities $\mu_t \leq \mu_{t+1}$ or $\mu_t \geq \mu_{t+1}$.

Note that constraints can be combined by considering more than one edge from one state to another. In particular, for L2 decomposition the constraint $|\mu_{t+1} - \mu_t| \geq c$ can be implemented using $\mu_t + c \leq \mu_{t+1}$ or $\mu_t \geq \mu_{t+1} + c$. This constraint encodes the idea of detecting sufficiently large changes (also called relevant changes) described in Dette and Wied (2016).

Computationally, it is possible to compute $O_n^{s,s'}(\theta)$ by scanning from left to right or from right to left all intervals which correspond to a different functional form of $Q_n^s(\theta)$ (see examples in Hocking *et al.* 2020).

Update-rule. Given this operator function we can now define the update-rule used by the `gfpop` algorithm.

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

For simplicity, we do not describe the update for initial and final steps. The proof of this update-rule is very similar to the proof of the Viterbi algorithm and is given in Appendix B. It follows the strategy of Hocking *et al.* (2020). Notice also that recovering the optimal set of change-points from all $Q_1^s, \dots, Q_n^s$ by backtracking is not straightforward because of the need to validate the constraints between consecutive segments. We provide some details in Appendix C.

An example with `fpop`. With the standard multiple change-point model (see Section 1.2 and Figure 4) we have only one vertex (1) and two edges denoted here 0 (no change) and 1 (a change) replacing the notation (s, s'). We get with Equation 2:

$$Q_{n+1}^1(\theta) = \min \left\{ O_n^0(\theta) + \gamma_0(y_{n+1}, \theta) + \beta_0,\, O_n^1(\theta) + \gamma_1(y_{n+1}, \theta) + \beta_1 \right\} .$$

Only edge 1 is penalized, so that $\beta_0 = 0$ and $\beta_1 = \beta > 0$. As we have no robust loss or parameter constraint on the cost function: $\gamma_0(\cdot,\cdot) = \gamma_1(\cdot,\cdot) = \gamma(\cdot,\cdot)$. For edge 0, $O_n^0(\theta') = \min\limits_{\theta|\theta=\theta'} \{Q_n^1(\theta)\} = Q_n^1(\theta')$ and for edge 1, $O_n^1(\theta') = \min\limits_{\theta|\theta\neq\theta'} \{Q_n^1(\theta)\} = \min\limits_{\theta}\{Q_n^1(\theta)\}$. Removing the state index, we eventually obtain the well-known FPOP update-rule:

$$Q_{n+1}(\theta) = \min \left\{ Q_n(\theta),\, \min_\theta \{Q_n(\theta)\} + \beta \right\} + \gamma(y_{n+1}, \theta) .$$

If we assume a Gaussian loss for change in mean, we have $\gamma(y_{n+1}, \theta) = (y_{n+1} - \theta)^2$, quadratic in θ. The update consists in reconstructing the optimal cost by finding for all θ the minimum between $Q_n(\theta)$ and the $\min\limits_\theta\{Q_n(\theta)\}$ constant line leading to a function $Q_{n+1}(\cdot)$ piecewise quadratic in θ. Ways to deal efficiently with this update-rule have been presented in Maidstone *et al.* (2017). For implementing the constraints included in the package, see Hocking *et al.* (2020) and Hocking *et al.* (2022); for other loss functions see also Fearnhead and Rigaill (2019) and Jewell *et al.* (2020).

3.3. How to choose the loss function and penalty

The choice of the loss function γ is linked to the choice of the noise model. This choice is not necessarily easy. For example for continuous data it might make sense to consider the least-squares error (Picard, Robin, Lavielle, Vaisse, and Daudin 2005); in the presence of outliers considering a robust loss is natural (Fearnhead and Rigaill 2019); and for count data a Poisson loss is often used (Hocking *et al.* 2020). It is our experience that visualizing the data beforehand is a good way to avoid simple modeling mistakes.

The choice of the penalty β is critical to select the number of change-points. In the absence of constraints several penalties have been proposed. For detecting a change in mean with independent Gaussian data, a penalty of $\beta = 2\sigma^2 \log(n)$ was proposed by Yao and Au (1989). It tends to work well when the number of changes is small. More complex penalties exist, e.g., Zhang and Siegmund (2007); Lebarbier (2005); Baraud, Giraud, and Huet (2009). For penalties that are concave in the number of segments one can run the operators and update-rule for the `gfpop()` algorithm for various values of β and recover several segmentations (with a varying number of change-points; Killick *et al.* 2012). This can be done efficiently using the CROPS (changepoints for a range of penalties) algorithm (Haynes, Eckley, and Fearnhead 2017). In labeled data sets, supervised learning algorithms can be used to infer an accurate model for predicting penalty values β (Rigaill, Hocking, Vert, and Bach 2013; Hocking *et al.* 2015, 2020).

For models with constraints, to the best of our knowledge there is very little statistical literature available. The paper of Gao *et al.* (2020) describes a penalty in the isotonic case but it was not calibrated. It is our experience that the penalties proposed for the unconstrained case tend to work reasonably well, although they are probably sub-optimal from a statistical perspective.

4. The gfpop package

4.1. Graph construction

Our **gfpop** package deals with collapsed graphs for which all the cost functions γ have the same decomposition (L2, lin-log or log-log). All other characteristics are local and fixed per edge. The graph $\mathcal{G}_n$ (see Section 2.2) is defined in the **gfpop** package by a collection of edges.

Edge parameters. An edge is a list of four main elements:

- `state1`: the starting node defined by a string;
- `state2`: the arrival node defined by a string;
- `type`: a string equal to `"null"`, `"std"`, `"up"`, `"down"` or `"abs"` defining the type of constraints between successive nodes respectively corresponding to indicators $I_{\mu_t=\mu_{t+1}}$, $I_{\mu_t\neq\mu_{t+1}}$, $I_{\mu_t+c\leq\mu_{t+1}}$, $I_{\mu_t\geq\mu_{t+1}+c}$ and $I_{|\mu_{t+1}-\mu_t|\geq c}$;
- `penalty`: the penalty β_e associated to this edge (it can be zero);

and some optional elements:

- `decay`: a number between 0 and 1 for the mean exponential decay (in case `type` is `"null"`) corresponding to the constraint $I_{\mu_{t+1}=\alpha\mu_t}$;
- `gap`: the gap c between successive means of the `"up"`, `"down"` and `"abs"` types;
- `K`: the threshold for the biweight and Huber losses ($K > 0$);
- `a`: the slope for the Huber robust loss ($a \geq 0$).

An example of an edge. We can define an edge `e1` with the function `Edge` as:

```
R> e1 <- Edge(state1 = "Dw", state2 = "Up", type = "up", penalty = 10,
+    gap = 0.5)
```

which is an edge from node `Dw` to node `Up` with an up constraint, penalty $\beta = 10$ and a minimal jump size of 0.5.

An example of a graph. We provide an example of a graph for collective anomalies detection with the **gfpop** package given in Figure 17 (see Fisch *et al.* 2022):

```
R> graph(Edge(state1 = "mu0", state2 = "mu0", penalty = 0, K = 3),
+    Edge(state1 = "mu0", state2 = "Coll", penalty = 10, type = "std"),
+    Edge(state1 = "Coll", state2 = "Coll", penalty = 0),
+    Edge(state1 = "Coll", state2 = "mu0", penalty = 0, type = "std", K = 3),
+    StartEnd(start = "mu0", end = c("mu0", "Coll")),
+    Node(state = "mu0", min = 0, max = 0))
```

```
  state1 state2  type parameter penalty   K  a min max
1    mu0    mu0  null         1       0   3  0  NA  NA
2    mu0   Coll   std         0      10 Inf  0  NA  NA
3   Coll   Coll  null         1       0 Inf  0  NA  NA
4   Coll    mu0   std         0       0   3  0  NA  NA
5    mu0   <NA> start        NA      NA  NA NA  NA  NA
6    mu0   <NA>   end        NA      NA  NA NA  NA  NA
7   Coll   <NA>   end        NA      NA  NA NA  NA  NA
8    mu0    mu0  node        NA      NA  NA NA   0   0
```

Notice that the graph is encoded into a data-frame.

Note 1. Most graphs (such as the previous one) contain recursive edges, that is edges with the same starting and arrival node. The absence of this edge forces a change and is useful to enforce a minimal segment length (see Figures 9 and 18).

Note 2. In the **gfpop** graph definition, abusing our mathematical notations, we call a starting (resp. arrival) state, a state directly connected to the starting node $v_0 = (0, \#)$ (resp. arrival node $v_{n+1} = (n+1, \emptyset)$). In other words, the first (resp. last) point can only be in state s if this state is a "starting" (resp. "arrival") state for **gfpop**. These specific states are defined using function `StartEnd`. If not specified, all nodes are starting and arrival nodes. The range of values for parameter inference at each node can be constrained using function `Node`.

In this example we have two states, `mu0` and `coll`. Both states can be arrival states, but we have fixed the starting node to be `mu0`. This node `mu0` is restricted by `min = 0` and `max = 0` using the `Node` function, such that only the zero value can be inferred for any segment in that state.

Some default graphs. We included in function `graph()` the possibility to directly build some standard graphs. Here is an example for the isotonic case corresponding to Figure 5:

```
R> graph(type = "isotonic", penalty = 12)

  state1 state2 type parameter penalty   K a min max
1    Iso    Iso null         1       0 Inf 0  NA  NA
2    Iso    Iso   up         0      12 Inf 0  NA  NA
```

Three other standard graph types are: `"std"`, `"updown"` and `"relevant"` corresponding to Figures 4, 6 and 19. All graphs presented in this paper are available in our package through the function `paperGraph()`, where its first parameter is the figure number.

4.2. The `gfpop()` function

The `gfpop()` function takes as an input the data and the graph and runs the algorithm. It returns a set of change-points and the non-penalized cost (that is the value of the fit to the data ignoring the penalties for adding changes). It also returns the mean value and the state of each segment. The boolean `forced` value indicates whether a linear inequality constraint is active, which means that the μ_t and μ_{t+1} values lie on the frontier defined by the inequality constraint. Below we illustrate the use of the `gfpop()` function for various graphs and loss functions.

We first simulate data. To do this we use the `dataGenerator()` function provided by the **gfpop** package. The function generates `n` data points using a distribution of `type` `"mean"` (by default), `"poisson"`, `"exp"`, `"variance"` or `"negbin"` following a change-point model given by relative change-point positions (a vector of increasing values in $(0, 1]$). Standard deviation parameter `sigma` and decay `gamma` are specific to the Gaussian mean model, whereas `size` is linked to function `rnbinom` from the base R package **stats**.

Gaussian model with an up-down graph. Here is an example with a Gaussian cost and a standard penalty of $2\log(n)$ for the up-down graph. We simulate data from a change in mean model with Gaussian observations.

```
R> set.seed(75)
R> n <- 1000
```

```
R> myData <- dataGenerator(n, c(0.1, 0.3, 0.5, 0.8, 1),
+    c(1, 2, 1, 3, 1), sigma = 1)
```

This data has 5 segments, with the end of segments at relative positions 0.1, 0.3, 0.5, 0.8 and 1 along the $n = 1000$ data points; and with segment means being respectively $1, 2, 1, 3$ and 1.

The call to the `gfpop()` function requires specifying the data, the graph that encapsulates the changepoint model, and the type of loss function. Here `type = "mean"` specifies the use of the L2 or biweight loss.

```
R> myGraph <- graph(penalty = 2 * log(n), type = "updown")
R> gfpop(data = myData, mygraph = myGraph, type = "mean")

$changepoints
[1]  108  295  500  800 1000
$states
[1] "Dw" "Up" "Dw" "Up" "Dw"
$forced
[1] FALSE FALSE FALSE FALSE
$parameters
[1] 1.044920 2.047202 1.017550 2.916826 1.030938
$globalCost
[1] 963.0278
```

The response contains four vectors. A vector `changepoints` contains the last index of each segment, a vector `states` gives the nodes in which lie the successive parameter values of the `parameters` vector. The vector `forced` is a vector of booleans of size "number of segments − 1" with entry `TRUE` when the transition between two states (nodes) has been forced. The `globalCost` is the non-penalized cost.

Gaussian robust biweight model with an up-down graph. Below we illustrate the use of the biweight loss on data where 10% of the data points are outliers. We shift these data by ±5 using function `rbinom` (4th line of code below). We use the biweight loss with $K = 3$ and an up-down graph with a difference of at least 1 between consecutive means.

```
R> n <- 1000
R> chgtpt <- c(0.1, 0.3, 0.5, 0.8, 1)
R> myData <- dataGenerator(n, chgtpt, c(0, 1, 0, 1, 0), sigma = 1)
R> myData <- myData + 5 * rbinom(n, 1, 0.05) - 5 * rbinom(n, 1, 0.05)
R> beta <- 2 * log(n)
R> myGraph <- graph(
+    Edge("Dw", "Up", type = "up", penalty = beta, gap = 1, K = 3),
+    Edge("Up", "Dw", type = "down", penalty = beta, gap = 1, K = 3),
+    Edge("Dw", "Dw", type = "null", K = 3),
+    Edge("Up", "Up", type = "null", K = 3),
+    StartEnd(start = "Dw", end = "Dw"))
R> gfpop(data = myData, mygraph = myGraph, type = "mean")
```

```
$changepoints
[1]  102  311  500  806 1000
$states
[1] "Dw" "Up" "Dw" "Up" "Dw"
$forced
[1] TRUE FALSE FALSE FALSE
$parameters
[1] -0.02296768  0.97703232 -0.03434534  1.00246359 -0.03334062
$globalCost
[1] 1097.364
```

The difference between this graph and the one for the previous example is the specification `K = 3` for each edge. This enforces the use of the biweight loss (with $K = 3$) as opposed to the L2 loss.

Poisson model with isotonic up graph. We provide an example with a lin-log cost decomposition with Poisson data constrained to up changes, with the mean at least doubling at each change.

```
R> n <- 1000
R> chgtpt <- c(0.1, 0.3, 0.5, 0.8, 1)
R> myData <- dataGenerator(n, chgtpt, c(1, 3, 5, 7, 12), type = "poisson")
R> beta <- 2 * log(n)
R> myGraph <- graph(type = "isotonic", gap = 2)
R> gfpop(data = myData, mygraph = myGraph, type = "poisson")
```

```
$changepoints
[1]    2   99  297  796 1000
$states
[1] "Iso" "Iso" "Iso" "Iso" "Iso"
$forced
[1] TRUE FALSE TRUE TRUE
$parameters
[1]  0.4693878  0.9387755  2.9840954  5.9681909 11.9363817
$globalCost
[1] -5832.845
```

The use of the Poisson loss is enforced by `type = "poisson"` in the call to `gfpop()`. The graph that describes our change-point model is the default one for isotonic changes, but with the additional constraint on means at least doubling being specified by `gap = 2`.

Negative binomial model with 3-segment graph. The parameters to find are probabilities and we restrict the inference to 3 segments. The optional parameter `all.null.edges` in the `graph()` function automatically generates for all nodes a "null" type edge that is an edge with constraint $I_{\mu_t=\mu_{t+1}}$ and with a penalty equal to 0 (see also bottom of page 12).

```
R> myGraph <- graph(Edge("1", "2", type = "std", penalty = 0),
+    Edge("2", "3", type = "std", penalty = 0),
+    StartEnd(start = "1", end = "3"), all.null.edges = TRUE)
R> myData <- dataGenerator(n = 1000, changepoints = c(0.3, 0.7, 1),
+    parameters = c(0.2, 0.25, 0.3), type = "negbin")
R> gfpop(myData, myGraph, type = "negbin")

$changepoints
[1]  300  714 1000
$states
[1] "1" "2" "3"
$forced
[1] FALSE FALSE
$parameters
[1] 0.2117808 0.2652162 0.3212748
$globalCost
[1] 2193.216
```

Each data point can be weighted using argument `weights` in the `gfpop()` function. It can be useful to gather consecutive identical values for count data time series in order to speed-up the change-point analysis (Cleynen *et al.* 2014).

4.3. Some additional useful functions in gfpop

Standard deviation estimation. For many examples using real data sets, we are obliged to estimate the standard deviation from the observed data. This value is then used to normalize the data or to be included in edge penalties. The function `sdDiff()` returns such an estimation with a difference-based estimator of the variance (Hall, Kay, and Titterington 1990) (denoted HALL) well suited for time series with change-points.

A plotting function. We defined a plotting function `plot()`, which shows data points and the results of the `gfpop()` function by using inferred segment parameters and change-points. The user can plot the result in two graphs or only one for `"mean"` and `"poisson"` types (see parameter `multiple`) and has to explicitly use the `data` argument as in the following examples.

Example 1:

```
R> set.seed(86)
R> myData <- dataGenerator(1000, c(0.3, 0.4, 0.7, 0.95, 1),
+    c(1, 3, 1, -1, 4), "mean", sigma = 3)
R> s <- sdDiff(myData)
R> g <- gfpop(myData,
+    graph(type = "relevant", gap = 0.5, penalty = 2 * s ^ 2 * log(1000)),
+    type = "mean")
R> plot(x = g, data = myData, multiple = FALSE)
```

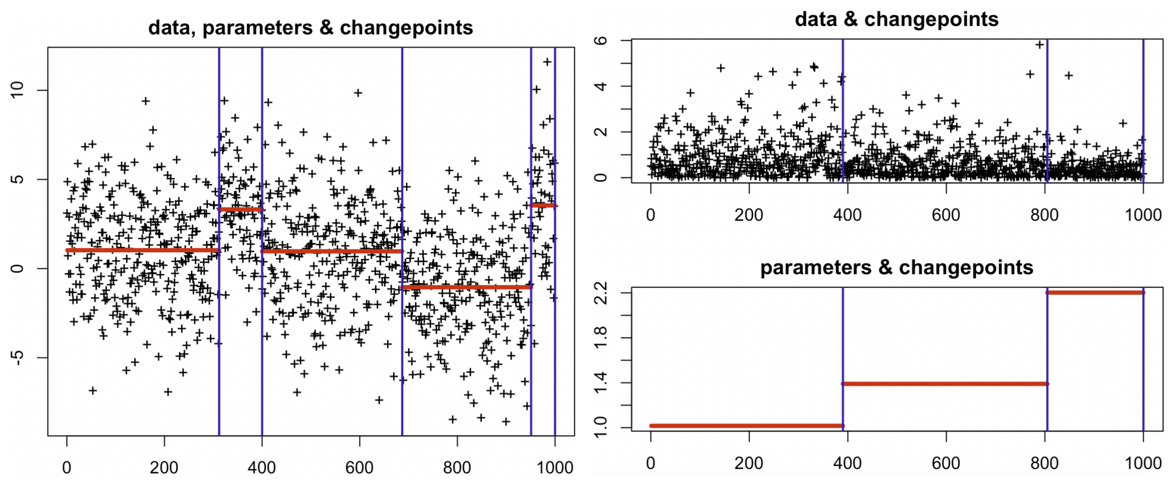

Figure 10: The red piecewise constant signal is the μ vector found by the `gfpop()` function, the blue vertical lines indicate the change-point positions. They are built using response vectors `changepoints` and `parameters`. The left graph presents the result of Example 1, the right graphs of Example 2.

Example 2:

```
R> set.seed(86)
R> myData <- dataGenerator(1000, c(0.4, 0.8, 1), c(1, 1.3, 2.3), "exp")
R> s <- sdDiff(myData)
R> g <- gfpop(myData, type = "exp",
+    graph(type = "isotonic", penalty = 2 * s ^ 2 * log(1000)))
R> plot(x = g, data = myData, multiple = TRUE)
```

5. Modeling real data with graph-constrained models

In this section we illustrate the use of our package on several real data sets. For each application we illustrate several possible sets of constraints and briefly discuss their relative advantages.

5.1. Gaussian model for DNA copy number data

We consider DNA copy number data, which are biological measurements that characterize the number of chromosomes in cell samples. Abrupt changes along chromosomes in these data are important indicators of severity in cancers such as neuroblastoma (Schleiermacher *et al.* 2010). The non-constrained Gaussian segmentation model has been shown to have state-of-the-art change-point detection accuracy in these data (Hocking *et al.* 2013).

However, in some high-density copy number data sets, this model incorrectly detects small changes in mean which are not relevant (Hocking and Rigaill 2012). One such data set is shown in Figure 11, which also has positive and negative labels from an expert genomic scientist that indicated regions with (1breakpoint) or without (0breakpoints) relevant change-points.

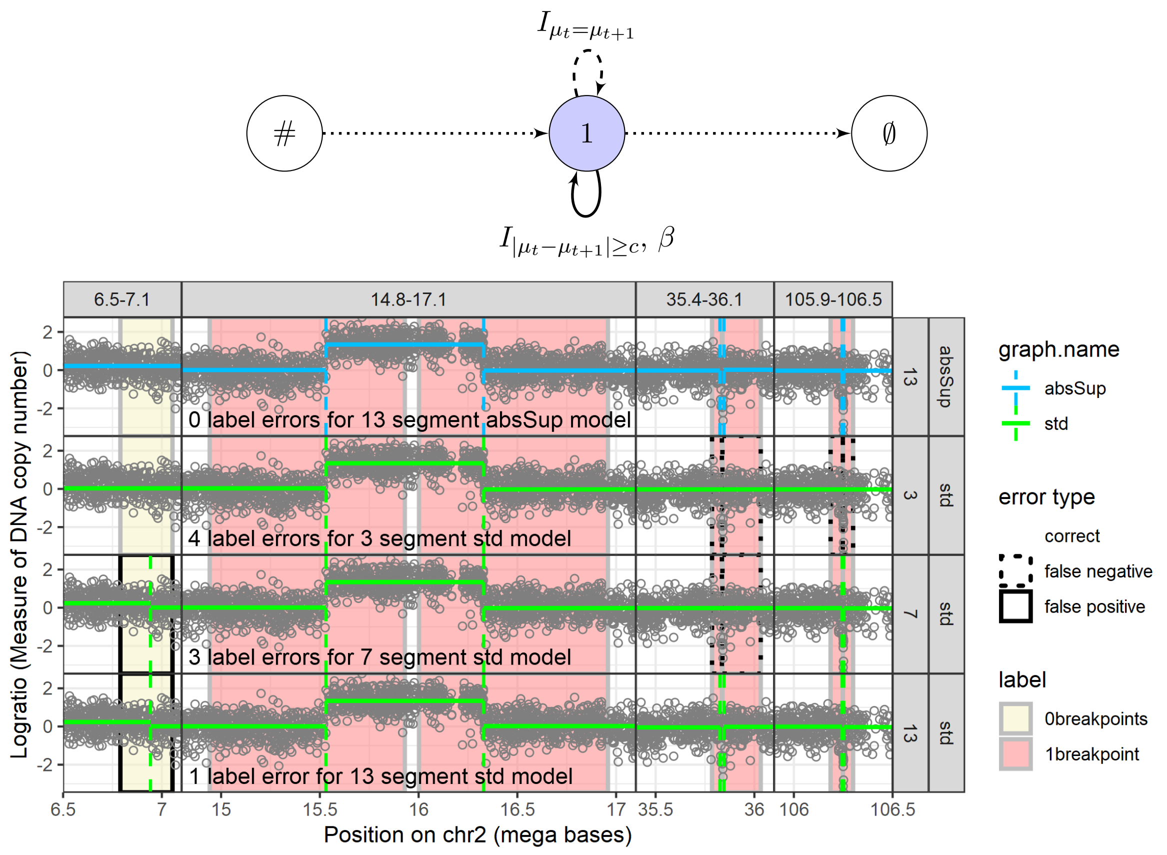

Figure 11: Top graph: Relevant change-point model; all changes are forced to be greater than a in absolute value. Below: Four subsets/windows of a DNA copy number profile (panels from left to right) and four change-point models (panels from top to bottom); rectangles show expert-provided labels which are assumed to be a gold standard. Top panel with blue model: abs model with 13 segments enforces the constraint that the absolute value of each change must be at least 1, $|\mu_{t+1} - \mu_t| \geq 1$, which achieves zero label errors in these data. Bottom panels with green models: each model with no constraints between adjacent segment means has label errors (4 false positives for 3 segments, 2 false positives and 1 false negative for 7 segments, 1 false positive for 13 segments).

We used these labels to quantify the accuracy of three unconstrained Gaussian change-point models with several different penalties β.

- The model with 13 segments predicts a change-point in each positive label (0/6 false negatives), but predicts one change-point in the negative label (1 false positive), for a total of 1 incorrectly predicted label (bottom panel).
- The model with 7 segments predicts a change-point in four positive labels (2/6 false negatives), and also predicts the false positive change-point in the negative label, for a total of 3 incorrectly predicted labels (second panel from bottom).
- The model with 3 segments predicts a change-point in only two positive labels (4/6 false negatives), and predicts no change-point in the negative label (0/1 false positive), for a total of 4 incorrectly predicted labels (second panel from top).

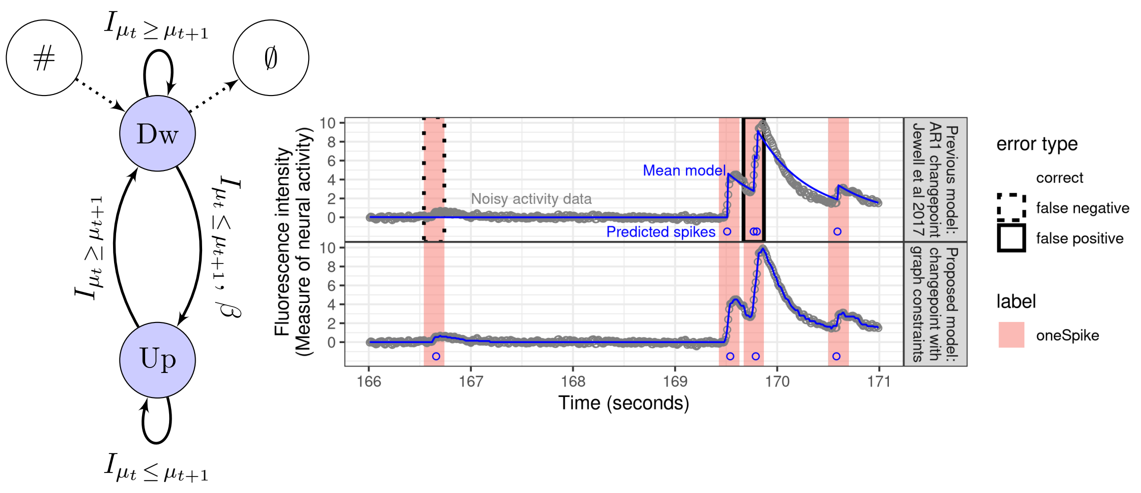

Figure 12: Left: Graph for multi-modal regression. Right top: In these data the previously proposed AR1 model misses a spike in the left label (false negative) and predicts two spikes where there should be only one in the right label (false positive). Right bottom: The proposed multi-modal regression model correctly detects one spike in all the labeled regions.

We computed all non-constrained Gaussian models from 1 to 20 segments for these data, and none of them were able to provide change-point predictions that perfectly match the expert-provided labels (each model had at least one false positive or false negative). It is thus problematic to use the unconstrained change-point model in this context, because none of the unconstrained models achieves zero label errors.

To solve this problem we propose a graph (Figure 11, top graph) which enforces only "relevant" change-points $|\mu_{t+1} - \mu_t| \geq c$, for some relevant threshold $c > 0$. For the DNA copy number data set, we set $c = 1$ and choose β such that the algorithm returns 13 segments (Figure 11, top panel with blue model). The proposed model predicts a change-point in each of the positive labels, but does not predict a change-point in the negative label. The proposed graph-constrained change-point model is therefore able to predict change-points that perfectly match the expert-provided labels. If overfitting is a concern with this procedure, we can consider using the two labels in the last region as a test set (105.9–106.5 mega bases), and the other labels as a train set. In that case we chose the penalty with minimal errors with respect to the train set labels, and we observed that the test set label error was also minimized.

5.2. Gaussian multi-modal regression for neuro spike train data

The so-called AR1 (autoregressive of order 1) change-point model, where the mean decreases exponentially within each segment has been proposed for detecting spikes in calcium imaging data from neuroscience (Jewell *et al.* 2020). We fit this model to one calcium imaging data set (Figure 12, right top) and observed that it is difficult to find a parameter that detects both labeled spikes. Red rectangles in Figure 12 indicate labels provided by an electrophysiological method which is taken as ground-truth in order to emphasize the qualitative difference between the two algorithms. Part of the difficulty of the AR1 model is the fact that there are two parameters to tune, the penalty β and also the exponential decay parameter γ. It is more difficult to tune two parameters using grid search because of its quadratic time complexity.

Another issue is that a visual inspection of the data suggests that the rate of decay of the mean between spikes may not be constant as assumed by the AR1 model.

We therefore propose a new multi-modal regression model (isotonic up-isotonic down graph shown in Figure 12, left) with only one parameter, the penalty β. We can view this model as detecting modes in the data. Each mode consists of a period before-hand where the mean increases followed by a period where then mean decreases. The period where the mean increases can be interpreted as a period of time where a spike occurs, with the periods where the mean decreases modeling the decay in the data after the spike end. The number of detected spikes is equal to the number of regions where the mean increases, and is controlled by the penalty β. We observed that it is easy to find a penalty β which detects both labeled spikes. Overall these results indicate that the proposed multi-modal regression model (isotonic up-isotonic down) is promising for spike detection in calcium imaging data. We leave a more extensive quantitative comparison to future work.

5.3. Gaussian nine-state model for electrocardiogram data

In the context of monitoring hospital patients with heart problems, electrocardiogram (ECG) analysis is one of the most common non-invasive techniques for diagnosing several heart arrhythmia (Afghah, Razi, and Najarian 2015).

A preliminary and fundamental step in ECG analysis is the detection of the QRS complex (the combination of the waves Q, R and S seen on a typical ECG, see `https://en.wikipedia.org/wiki/QRS_complex`) that leads to detecting the heartbeat and classifying the rhythms (Mousavi and Afghah 2019).

Here, we utilize the proposed change-point detection method to locate the QRS complex in ECG waveforms. The ECG signals used in this study are extracted from the publicly available Physionet Challenge 2015 database (PhysioNet 2015; Clifford *et al.* 2016) that includes measurements for three physiological signals (including ECG) for 750 patients. The resolution and frequency of each signal are 12bit and 250Hz, respectively. Also, each signal has been filtered by a finite impulse response (FIR) notch filter with band pass 0.05 to 40Hz.

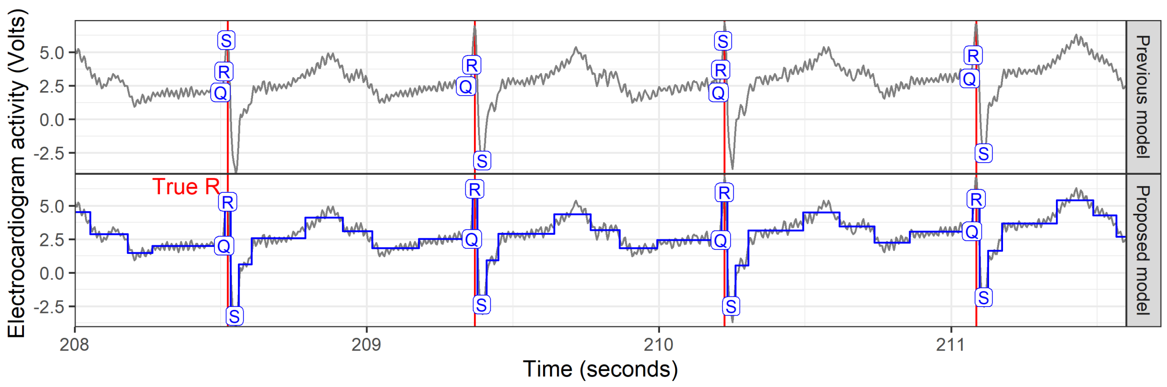

Figure 13: In these electrocardiogram data, it is important for models (blue) to accurately detect the QRS complex (Q is before the peak, R is the peak marked in red, S is the local minimum after the peak, other states o1–o6). Top: Previous model of Pan and Tompkins (1985) mistakenly predicts S at the peak. Bottom: Proposed constrained change-point model accurately predicts R at each peak.

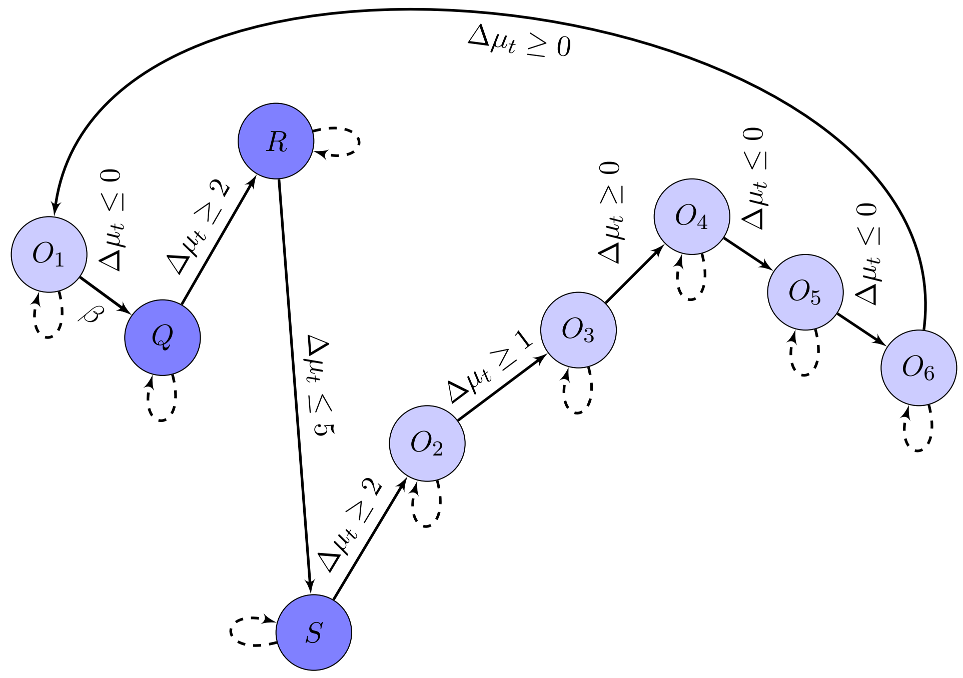

Figure 14: Graph structure of proposed nine-state constrained change-point model. The graph is cyclic: the last node O_1 is the first node O_1. Only one transition (from O_1 to Q) to enter the QRS complex is penalized by a positive penalty $\beta = 8 \times 10^3$. We used the notation $\Delta\mu_t = \mu_{t+1} - \mu_t$. Transitions from state Q to state O_3 are constrained with a minimal gap size of 2, 5, 2 and 1. Due to lack of space, we removed the indicator function I on this graph. Dashed arrows correspond to $I_{\mu_t=\mu_{t+1}}$ transitions. The vertical position of the states gives information on the direction of the constrained changes.

The Pan-Tompkins algorithm is one of the most common segmentation methods used for ECG analysis (Pan and Tompkins 1985; Agostinelli *et al.* 2017). This method uses a patient-specific threshold-based approach for real-time detection of the QRS complex in ECG signals, which represents the ventricular depolarization. In this algorithm, after a pre-processing step by a band-pass filter, the signal is passed through differentiation and squaring blocks to determine and amplify the slope of QRS, followed by a moving window integration step with an adaptive set of thresholds to determine the peaks. The detection thresholds are learned at the beginning of the algorithm and are calibrated periodically to follow the variations of the ECG signal.

Figure 13 (top) shows four seconds of ECG data for which we predicted the QRS complex using the well-known Pan-Tompkins method. The peak of each heartbeat should be predicted as R, but the algorithm incorrectly predicts S in two cases. In contrast the peak is correctly classified as R (bottom) using our proposed model with nine states (see Figure 14), which were determined using prior knowledge about the expected sequence of changes. In this model, the QRS complex is modeled by an up-spike followed by a down-spike with the maximum amplitude difference related to adjacent spikes. The graph model considers a vertex for each main waveform (i.e., P, Q, R, S, T) as well as three baselines, which are intermediate states (Fotoohinasab, Hocking, and Afghah 2021).

6. Isotonic regression with a constraint graph and robust loss

Our package can be used with robust loss functions which have been shown to be useful in the presence of outliers (Fearnhead and Rigaill 2019) and in particular in the context of isotonic regression (Bach 2018). Here we illustrate this on simulations inspired by those of (Bach 2018, see Figure 15 with corrupted data). We compare our package using the isotonic model described in Figure 5 with several implementations of the Pool Adjacent Violators Algorithm (PAVA) (Best and Chakravarti 1990; De Leeuw *et al.* 2010).

Relative to the very fast $O(n)$ PAVA algorithm, our dynamic programming algorithm is slower. However, PAVA only works for the squared loss and the non-penalized model (maximum number of changes). In contrast, `gfpop()` can handle non-convex losses (such as the biweight loss) and can include a positive penalty in order to reduce the number of changes.

6.1. Parametrization

gfpop. In all simulations we use **gfpop** with the graph of Figure 5 and a quadratic (L2) or a biweight loss (bw). We consider two different values for the penalty β: 0 and $2\sigma^2\log(n)$, with σ^2 the true variance. Thus, we have 4 different algorithms of the `gfpop()` function: `gfpop1` ($\beta = 0$, $K = 0$), `gfpop2` ($\beta = 2\sigma^2\log(n)$, $K = 0$), `gfpop3` ($\beta = 0$, $K = 3\sigma^2$) and `gfpop4` ($\beta = 2\sigma^2\log(n)$, $K = 3\sigma^2$).

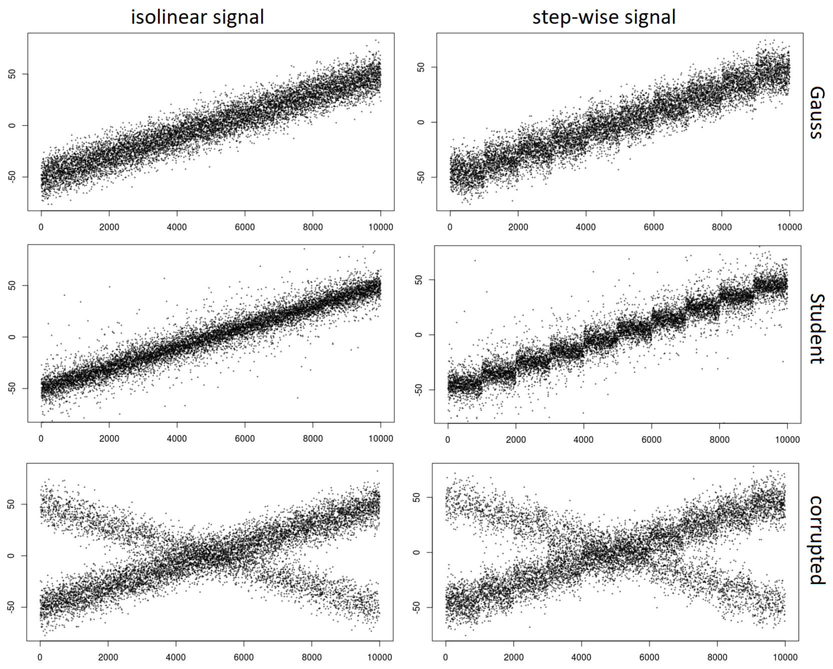

Figure 15: For the two types of signal, we show simulated data with $n = 10^4$ data points and $\sigma = 10$ for the three different noises (Gauss, Student, corrupted). Note that, in order to display the same slope on all of the 6 panels, about 0.15% of the data-points are out of the plot for the student case.

Competitors. We compare the output of **gfpop** with those of 2 isotonic regression package functions:

- `isoreg()` function of the **stats** package which is based on the very fast Pool adjacent violators algorithm for the ℓ_2 loss (Best and Chakravarti 1990);
- `reg_1d()` function developed in package **UniIsoRegression** which solves the isotonic regression problem for the ℓ_2 and ℓ_1 losses (Stout 2008).

We also include a simple linear regression approach (`lm()` function of the **stats** package) as a reference. In total we have 4 competitors (`lm()`, `isoreg()`, `reg_1d()` with the ℓ_2 and ℓ_1 losses).

6.2. Simulated data

We focus on two types of increasing signals:

linear: as in Bach (2018) we consider linearly increasing time series with a signal

$$s_i = \alpha(i - \frac{n}{2}) \quad i = 1, \dots, n\,;$$

step-wise: as our package is devoted to change-point inference we also consider a step-wise increasing series (with 10 steps) with a signal

$$s_i = \lfloor \frac{10(i-1)}{n} \rfloor - \frac{n}{2}\,, \quad i = 1, \dots, n\,.$$

We consider three ways to corrupt the data.

Gaussian noise: here we simply add a Gaussian noise, with a variance σ^2 to the signal (i.e., $Y_i = s_i + \varepsilon_i$).

Student noise: we also consider a Student noise with degrees of freedom equal to 3.

Corrupted noise: in the most difficult scenario, suggested by Bach (2018), we randomly select a proportion p of data points and multiply them by -1 and then add a Gaussian noise, i.e., $Y_i = X_i s_i + \varepsilon_i$, where $X_i \sim \mathcal{B}(p)$ is a Bernoulli trial with probability p to get -1 and probability $1 - p$ to get 1. We fix $p = 0.3$ for all simulations.

In total we have 6 scenarios (2 signals and 3 ways to corrupt the data). In Figure 15 we illustrate those 6 scenarios with $n = 10^4$ and $\sigma = 10$.

Criteria. To assess the quality of the results, we compute the mean-squared error (MSE) as well as the ability to recover the true number of changes when there are changes in the data in the step-wise scenario.

6.3. A simple illustration

We illustrate our results on a step-wise increasing signal with corrupted data. In Figure 16, we represent the data and the results of various approaches. We see that using a biweight loss our package in blue is closer to the true signal in black than other approaches.

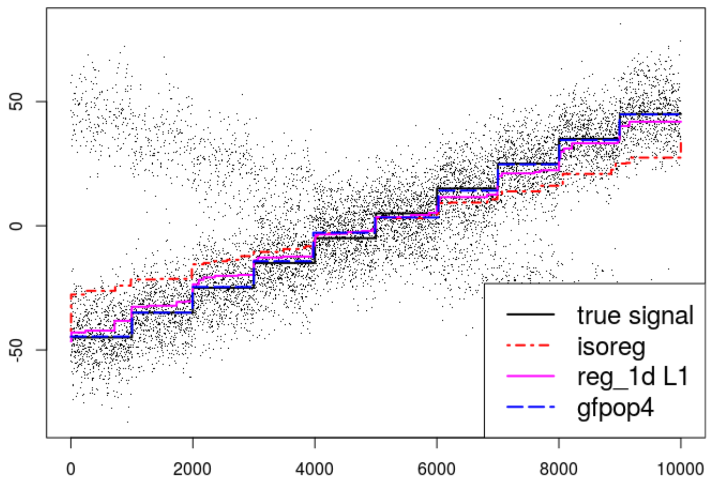

Figure 16: Isotonic regression with 30% of corrupted data in the step-wise scenario with 10 steps. We have 10^4 data points and $\sigma = 10$. `gfpop4()` is close to the signal and with a number of segments equal to 10.

In the appendix, we consider Monte-Carlo simulations to confirm this result: see Appendix D.1 for the linear increasing scenario and Appendix D.2 for the step-wise increasing scenario. As expected, recovering the true number of changes with corrupted data with also a small MSE is a challenging task for all methods excepted for `gfpop4()` using a robust loss, a positive penalty and the isotonic constraint graph. The R code of these simulations can be found on the Github page at `https://github.com/vrunge/gfpop/tree/master/simulations`.

7. Conclusion

In this paper we described the **gfpop** package, which provides a generalized version of an algorithm recently proposed by Hocking *et al.* (2020) for penalized maximum likelihood inference of constrained multiple change-point models. The **gfpop** package implements the algorithm in a generic manner in R/C++ and allows the user to specify the constraint graph in R code. We explained how these constrained multiple change-point models can also be seen as constrained continuous state space HMMs. **gfpop** allows one to encode modeling assumptions on the type of changes using a graph of states and constraints. We illustrated the use of **gfpop** on isotonic simulations and several applications in biology.

For a number of graphs the algorithm runs in a matter of seconds or minutes for 10^5 data points. While **gfpop** can be used to fit simple change-point models, such as the standard change in mean in Gaussian data, it is slower than **fpop** which implements functional programming specifically for that model: For example, for 10^5 points with no change **fpop** runs in 0.040 seconds and **gfpop** in 0.49 seconds. This is because the **gfpop** package is coded in a more generic manner, as it handles constraints and various losses. As we illustrated with numerous examples, the advantage of **gfpop** is that it allows one to include constraints and/or unconventional losses, and thus fit a range of change-point models that cannot be fit by other generic software.

Future work. For future work we are interested to explore generalizations which allow time-dependent constraints. As mentioned in Section 2.2 our implementation only allows inference in models that can be represented by a collapsed graph with transitions that are valid for all time points. We are interested in exploring new frameworks for defining which transitions and/or states are feasible at which time points, in order to efficiently support inference in models such as labeled optimal partitioning (Hocking and Srivastava 2023). There are a number of other extensions of **gfpop** that are possible, including allowing local fluctuations in the parameter between change-points and modeling autocorrelated noise – these can be both incorporated using ideas from Romano, Rigaill, Runge, and Fearnhead (2022). Another extension would be to consider the detection of change-points in trees as proposed in Chapter 3 of the Ph.D. thesis of Thépaut (2019). Furthermore, the underlying `gfpop` algorithm is sequential and thus can be adapted to allow for online change-point detection.

References

Afghah F, Razi A, Najarian K (2015). "A Shapley Value Solution to Game Theoretic-Based Feature Reduction in False Alarm Detection." Neural Information Processing Systems (NIPS), Workshop on Machine Learning in Healthcare. `doi:10.48550/arXiv.1512.01680`.

Agostinelli A, Marcantoni I, Moretti E, Sbrollini A, Fioretti S, Di Nardo F, Burattini L (2017). "Noninvasive Fetal Electrocardiography Part I: Pan-Tompkins' Algorithm Adaptation to Fetal R-Peak Identification." *The Open Biomedical Engineering Journal*, **11**, 17–24. `doi:10.2174/1874120701711010017`.

Anastasiou A, Chen Y, Cho H, Fryzlewicz P (2021). ***breakfast**: Methods for Fast Multiple Change-Point Detection and Estimation.* R package version 2.2, URL `https://CRAN.R-project.org/package=breakfast`.

Auger IE, Lawrence CE (1989). "Algorithms for the Optimal Identification of Segment Neighborhoods." *Bulletin of Mathematical Biology*, **51**(1), 39–54. `doi:10.1007/BF02458835`.

Bach F (2018). "Efficient Algorithms for Non-Convex Isotonic Regression through Submodular Optimization." In *NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 1–10.

Baranowski R, Chen Y, Fryzlewicz P (2019). "Narrowest-Over-Threshold Detection of Multiple Change Points and Change-Point-Like Features." *Journal of the Royal Statistical Society B*, **81**(3), 649–672. `doi:10.1111/rssb.12322`.

Baranowski R, Fryzlewicz P (2019). ***wbs**: Wild Binary Segmentation for Multiple Change-Point Detection.* R package version 1.4, URL `https://CRAN.R-project.org/package=wbs`.

Baraud Y, Giraud C, Huet S (2009). "Gaussian Model Selection with an Unknown Variance." *The Annals of Statistics*, **37**(2), 630–672. `doi:10.1214/07-aos573`.

Barlow RE, Bartholomew DJ, Bremner JM, Brunk HD (1972). "Statistical Inference under Order Restrictions: The Theory and Application of Isotonic Regression." *Technical report*, Defense Technology Information Centre. Report AD0751311.

Best MJ, Chakravarti N (1990). "Active Set Algorithms for Isotonic Regression: A Unifying Framework." *Mathematical Programming*, **47**(1–3), 425–439. doi:10.1007/bf01580873.

Cleynen A, Koskas M, Lebarbier E, Rigaill G, Robin S (2014). "**Segmentor3IsBack**: An R Package for the Fast and Exact Segmentation of Seq-Data." *Algorithms for Molecular Biology*, **9**(1), 1–11. doi:10.1186/1748-7188-9-6.

Clifford GD, Silva I, Moody B, Li Q, Kella D, Chahin A, Kooistra T, Perry D, Mark RG (2016). "False Alarm Reduction in Critical Care." *Physiological Measurement*, **37**(8), 5–23. doi:10.1088/0967-3334/37/8/e5.

De Leeuw J, Hornik K, Mair P (2010). "Isotone Optimization in R: Pool-Adjacent-Violators Algorithm (PAVA) and Active Set Methods." *Journal of Statistical Software*, **32**(5), 1–24. doi:10.18637/jss.v032.i05.

Dette H, Wied D (2016). "Detecting Relevant Changes in Time Series Models." *Journal of the Royal Statistical Society B*, **78**(2), 371–394. doi:10.1111/rssb.12121.

Eichinger B, Kirch C (2018). "A MOSUM Procedure for the Estimation of Multiple Random Change Points." *Bernoulli*, **24**(1), 526–564. doi:10.3150/16-bej887.

Fearnhead P, Rigaill G (2019). "Changepoint Detection in the Presence of Outliers." *Journal of the American Statistical Association*, **114**(525), 169–183. doi:10.1080/01621459.2017.1385466.

Fearnhead P, Rigaill G (2020). "Relating and Comparing Methods for Detecting Changes in Mean." *Stat*, **9**(1), e291. doi:10.1002/sta4.291.

Fisch ATM, Eckley IA, Fearnhead P (2022). "A Linear Time Method for the Detection of Collective and Point Anomalies." *Statistical Analysis and Data Mining*, **15**(4), 494–508. doi:10.1002/sam.11586.

Fotoohinasab A, Hocking TD, Afghah F (2021). "A Greedy Graph Search Algorithm Based on Changepoint Analysis for Automatic QRS Complex Detection." *Computers in Biology and Medicine*, **130**, 104208. doi:10.1016/j.compbiomed.2021.104208.

Frick K, Munk A, Sieling H (2014). "Multiscale Change Point Inference." *Journal of the Royal Statistical Society B*, **76**(3), 495–580. doi:10.1111/rssb.12047.

Fryzlewicz P (2014). "Wild Binary Segmentation for Multiple Change-Point Detection." *The Annals of Statistics*, **42**(6), 2243–2281. doi:10.1214/14-aos1245.

Gao C, Han F, Zhang CH (2020). "On Estimation of Isotonic Piecewise Constant Signals." *The Annals of Statistics*, **48**(2), 629–654. doi:10.1214/18-aos1792.

Hall P, Kay JW, Titterington DM (1990). "Asymptotically Optimal Difference-Based Estimation of Variance in Nonparametric Regression." *Biometrika*, **77**(3), 521–528. doi:10.1093/biomet/77.3.521.

Haynes K, Eckley IA, Fearnhead P (2017). "Computationally Efficient Changepoint Detection for a Range of Penalties." *Journal of Computational and Graphical Statistics*, **26**(1), 134–143. doi:10.1080/10618600.2015.1116445.

Haynes K, Killick R (2021). ***changepoint.np**: Methods for Nonparametric Changepoint Detection.* R package version 1.0.3, URL https://CRAN.R-project.org/package=changepoint.np.

Hocking TD, Bourque G (2020). "Machine Learning Algorithms for Simultaneous Supervised Detection of Peaks in Multiple Samples and Cell Types." In *Proceedings of the Pacific Symposium on Biocomputing*, volume 25, pp. 367–378.

Hocking TD, Rigaill G (2012). "**SegAnnot**: An R Package for Fast Segmentation of Annotated Piecewise Constant Signals." HAL technical report 00759129, URL https://hal.inria.fr/hal-00759129/.

Hocking TD, Rigaill G, Bourque G (2015). "**PeakSeg**: Constrained Optimal Segmentation and Supervised Penalty Learning for Peak Detection in Count Data." In *International Conference on Machine Learning*, pp. 324–332. PMLR.

Hocking TD, Rigaill G, Fearnhead P, Bourque G (2020). "Constrained Dynamic Programming and Supervised Penalty Learning Algorithms for Peak Detection in Genomic Data." *Journal of Machine Learning Research*, **21**(87), 1–40.

Hocking TD, Rigaill G, Fearnhead P, Bourque G (2022). "Generalized Functional Pruning Optimal Partitioning (GFPOP) for Constrained Changepoint Detection in Genomic Data." *Journal of Statistical Software*, **101**(10), 1–31. doi:10.18637/jss.v101.i10.

Hocking TD, Schleiermacher G, Janoueix-Lerosey I, Boeva V, Cappo J, Delattre O, Bach F, Vert JP (2013). "Learning Smoothing Models of Copy Number Profiles Using Breakpoint Annotations." *BMC Bioinformatics*, **14**(164). doi:10.1186/1471-2105-14-164.

Hocking TD, Srivastava A (2023). "Labeled Optimal Partitioning." *Computational Statistics*, **38**(1), 461–480. doi:10.1007/s00180-022-01238-z.

Jackson B, Scargle JD, Barnes D, Arabhi S, Alt A, Gioumousis P, Gwin E, Sangtrakulcharoen P, Tan L, Tsai TT (2005). "An Algorithm for Optimal Partitioning of Data on an Interval." *IEEE Signal Processing Letters*, **12**(2), 105–108. doi:10.1109/lsp.2001.838216.

Jewell S, Hocking TD, Fearnhead P, Witten D (2020). "Fast Nonconvex Deconvolution of Calcium Imaging Data." *Biostatistics*, **21**(4), 709–726. doi:10.1093/biostatistics/kxy083.

Johnson NA (2013). "A Dynamic Programming Algorithm for the Fused Lasso and L_0-Segmentation." *Journal of Computational and Graphical Statistics*, **22**(2), 246–260. doi:10.1080/10618600.2012.681238.

Killick R, Eckley IA (2014). "**changepoint**: An R Package for Changepoint Analysis." *Journal of Statistical Software*, **58**(3), 1–19. doi:10.18637/jss.v058.i03.

Killick R, Fearnhead P, Eckley IA (2012). "Optimal Detection of Changepoints with a Linear Computational Cost." *Journal of the American Statistical Association*, **107**(500), 1590–1598. doi:10.1080/01621459.2012.737745.

Lebarbier É (2005). "Detecting Multiple Change-Points in the Mean of Gaussian Process by Model Selection." *Signal Processing*, **85**(4), 717–736. doi:10.1016/j.sigpro.2004.11.012.

Maidstone R, Hocking TD, Rigaill G, Fearnhead P (2017). "On Optimal Multiple Changepoint Algorithms for Large Data." *Statistics and Computing*, **27**(2), 519–533. doi:10.1007/s11222-016-9636-3.

Meier A, Kirch C, Cho H (2021). "**mosum**: A Package for Moving Sums in Change-Point Analysis." *Journal of Statistical Software*, **97**(8), 1–42. doi:10.18637/jss.v097.i08.

Mousavi S, Afghah F (2019). "Inter- And Intra- Patient ECG Heartbeat Classification for Arrhythmia Detection: A Sequence to Sequence Deep Learning Approach." In *ICASSP 2019 – 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1308–1312. doi:10.1109/icassp.2019.8683140.

Olshen AB, Venkatraman E, Lucito R, Wigler M (2004). "Circular Binary Segmentation for the Analysis of Array-Based DNA Copy Number Data." *Biostatistics*, **5**(4), 557–572. doi:10.1093/biostatistics/kxh008.

Pan J, Tompkins WJ (1985). "A Real-Time QRS Detection Algorithm." *IEEE Transactions on Biomedical Engineering*, **BME-32**(3), 230–236. doi:10.1109/tbme.1985.325532.

Pein F, Hotz T, Sieling H (2022). ***stepR**: Multiscale Change-Point Inference.* R package version 2.1-3, URL https://CRAN.R-project.org/package=stepR.

PhysioNet (2015). *Reducing False Arrhythmia Alarms in the ICU.* Accessed 2016-07-28, URL https://www.physionet.org/challenge/2015/.

Picard F, Robin S, Lavielle M, Vaisse C, Daudin JJ (2005). "A Statistical Approach for Array CGH Data Analysis." *BMC Bioinformatics*, **6**(1), 27. doi:10.1186/1471-2105-6-27.

Pierre-Jean M, Rigaill G, Neuvial P (2019). ***jointseg**: Joint Segmentation of Multivariate (Copy Number) Signals.* R package version 1.0.2, URL https://CRAN.R-project.org/package=jointseg.

R Core Team (2022). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rigaill G (2015). "A Pruned Dynamic Programming Algorithm to Recover the Best Segmentations with 1 to $K_{\max}$ Change-Points." *Journal de la Société Française de Statistique*, **156**(4), 180–205.

Rigaill G (2022). ***fpopw**: Weighted Segmentation using Functional Pruning and Optimal Partioning.* R package version 1.1, URL https://CRAN.R-project.org/package=fpopw.

Rigaill G, Hocking TD, Maidstone R, Fearnhead P (2019). ***fpop**: Segmentation Using Optimal Partitioning and Function Pruning.* R package version 2019.08.26, URL https://CRAN.R-project.org/package=fpop.

Rigaill G, Hocking TD, Vert JP, Bach F (2013). "Learning Sparse Penalties for Change-Point Detection Using Max Margin Interval Regression." In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pp. 172–180.

Romano G, Rigaill G, Runge V, Fearnhead P (2022). "Detecting Abrupt Changes in the Presence of Local Fluctuations and Autocorrelated Noise." *Journal of the American Statistical Association*, **117**(540), 2147–2162. doi:10.1080/01621459.2021.1909598.

Runge V, Hocking TD, Rigaill G, Grose D, Romano G, Afghah F, Fearnhead P (2023). ***gfpop**: Graph-Constrained Functional Pruning Optimal Partitioning.* R package version 1.1.1, URL https://CRAN.R-project.org/package=gfpop.

Schleiermacher G, Janoueix-Lerosey I, Ribeiro A, Klijanienko J, Couturier J, Pierron G, Mosseri V, Valent A, Auger N, Plantaz D, Rubie H, Valteau-Couanet D, Bourdeaut F, Combaret V, Bergeron C, Michon J, Delattre O (2010). "Accumulation of Segmental Alterations Determines Progression in Neuroblastoma." *Journal of Clinical Oncology*, **28**(19), 3122–3130. doi:10.1200/jco.2009.26.7955.

Scott AJ, Knott M (1974). "A Cluster Analysis Method for Grouping Means in the Analysis of Variance." *Biometrics*, **30**(3), 507–512. doi:10.2307/2529204.

Stout QF (2008). "Unimodal Regression via Prefix Isotonic Regression." *Computational Statistics & Data Analysis*, **53**(2), 289–297. doi:10.1016/j.csda.2008.08.005.

Thépaut S (2019). *Problèmes de clustering liés à la synchronie en écologie: Estimation de rang effectif et détection de ruptures sur les arbres.* Ph.D. thesis, Université Paris-Saclay (ComUE). 2019SACLS477, URL https://www.theses.fr/2019SACLS477.

Truong C, Oudre L, Vayatis N (2020). "Selective Review of Offline Change Point Detection Methods." *Signal Processing*, **167**, 107299. doi:10.1016/j.sigpro.2019.107299.

Yao YC, Au ST (1989). "Least-Squares Estimation of a Step Function." *Sankhyā A*, **51**(3), 370–381.

Zhang NR, Siegmund DO (2007). "A Modified Bayes Information Criterion with Applications to the Analysis of Comparative Genomic Hybridization Data." *Biometrics*, **63**(1), 22–32. doi:10.1111/j.1541-0420.2006.00662.x.

A. Some other graphs

Here are three graphs for models discussed in the main part of the paper.

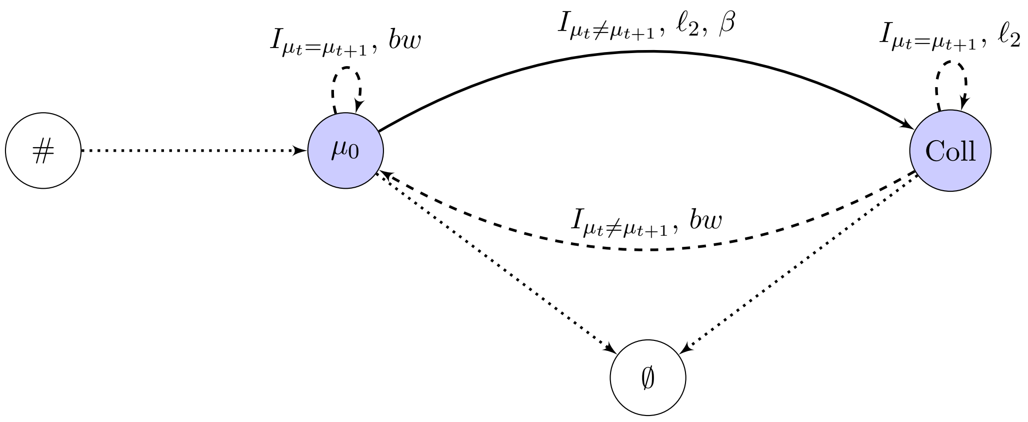
$I_{\mu_t=\mu_{t+1}}$, bw
$I_{\mu_t\neq\mu_{t+1}}$, ℓ_2, β
$I_{\mu_t=\mu_{t+1}}$, ℓ_2
#
μ_0
Coll
$I_{\mu_t\neq\mu_{t+1}}$, bw
$\emptyset$

Figure 17: Graph for the model proposed in Fisch *et al.* (2022). We have $\mathcal{S} = \{\mu_0, Coll\}$. Note that the value of μ_0 is given and that the loss function is either the ℓ_2 or the biweight bw. The penalty is omitted when equal to zero.

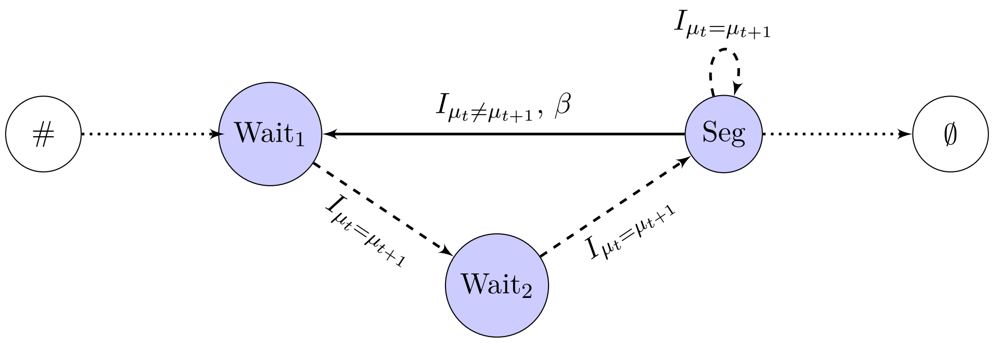

Figure 18: Graph for the at least 3 data points per segment model. We have $\mathcal{S} = \{Wait_1, Wait_2, Seg\}$, the loss function is always the ℓ_2. The penalty is omitted when equal to zero.

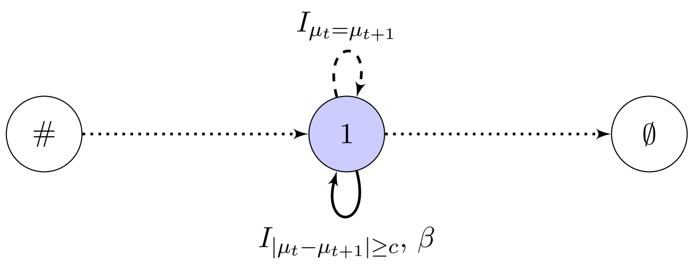

Figure 19: Graph for relevant change-point model. We have $\mathcal{S} = \{1\}$, the loss function is always the ℓ_2. The penalty is omitted when equal to zero.

Below we provide a few other constrained models and their graphs.

- (Up-down relevant) It might make sense to consider sufficiently large changes. This is a simple modification of the up-down model (see Figure 6). The *Dw* to *Up* constraint $I_{\mu_t \leq \mu_{t+1}}$ can be replaced by $I_{c+\mu_t \leq \mu_{t+1}}$ for $c > 0$ or $I_{a\mu_t \leq \mu_{t+1}}$ for $a > 1$ if μ_t is positive. The graph is shown in Figure 20.

- (Up-down with at least two data points) If one wants to detect peaks and is certain that segments are at least of length 2 it suffices to add two waiting states in the up-down graph. The graph of this model is given Figure 21.

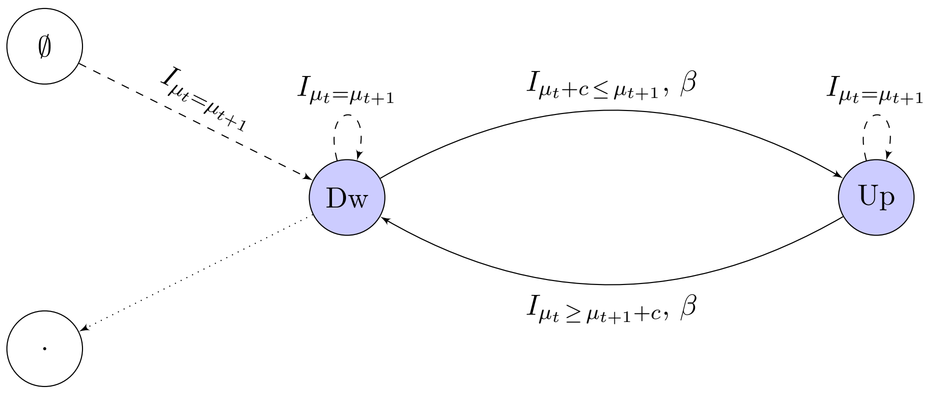

Figure 20: Graph for the up-down relevant model. We have $\mathcal{S} = \{Up, Dw\}$, the loss function is always the ℓ_2. The penalty is omitted when equal to zero.

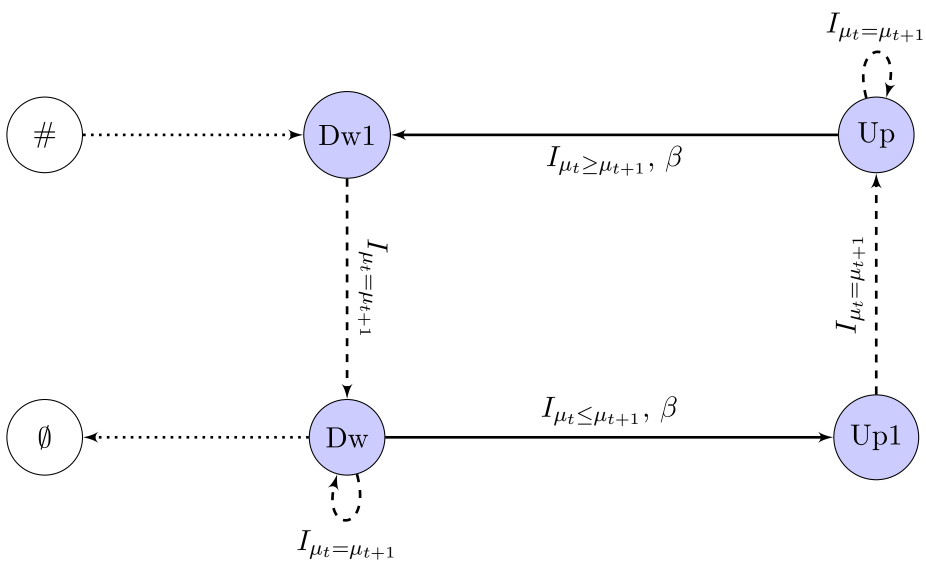

Figure 21: Graph for the up-down model with segments of size at least 2. We have $\mathcal{S} = \{Up1, Up, Dw1, Dw\}$. The loss function is always the ℓ_2 or the Poisson. The penalty is omitted when equal to zero.

- (Up-isotonic down) In the pulse detection example (up-exponentially down model in Figure 7) if one is not sure of the exponential decrease it could make sense to consider an isotonic decrease. For this it suffices to consider two states $\mathcal{S} = \{Up, Dw\}$. Compared to the up-down model, described earlier, we add an additional transition from *Dw* to *Dw* with the constraint $I_{\mu_t \geq \mu_{t+1}}$. The graph of this model is given in Figure 22.

- (Isotonic up-isotonic down) In the previous model one considers a sharp transition up. It might make sense to consider an isotonic increase. For this it suffices to add an edge from *Up* to *Up* in the previous model. Only transitions from *Up* to *Dw* and *Dw* to *Up* are penalized. The graph of this model is given in Figure 23.

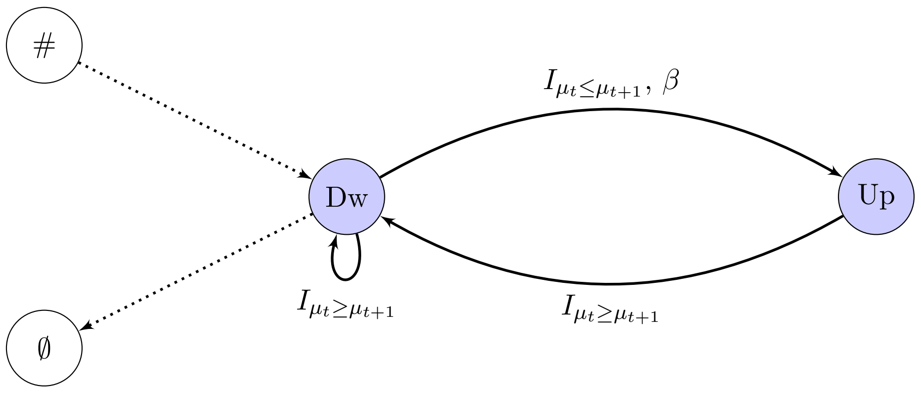
#
Dw
Up
∅
$I_{\mu_t \leq \mu_{t+1}}, \beta$
$I_{\mu_t \geq \mu_{t+1}}$
$I_{\mu_t \geq \mu_{t+1}}$

Figure 22: Top: Graph for the up-down* change-point model. We have $\mathcal{S} = \{Dw, Up\}$, the loss function is always the ℓ_2. The penalty is omitted when equal to zero.

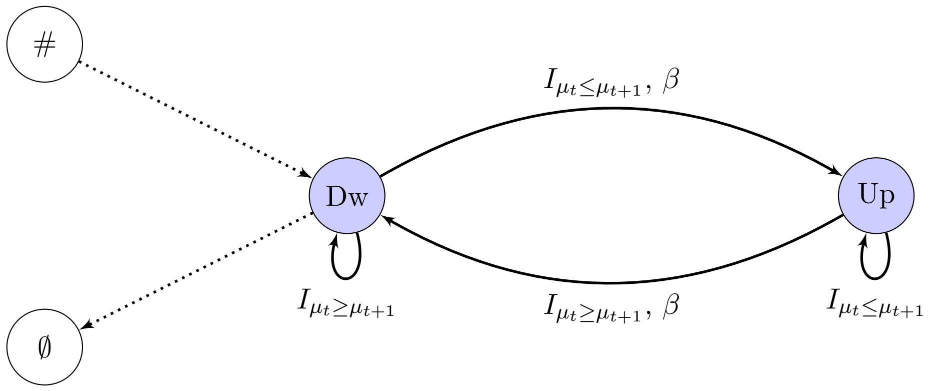
#
Dw
Up
∅
$I_{\mu_t \leq \mu_{t+1}}, \beta$
$I_{\mu_t \geq \mu_{t+1}}$
$I_{\mu_t \geq \mu_{t+1}}, \beta$
$I_{\mu_t \leq \mu_{t+1}}$

Figure 23: Graph for the up*-down* change-point model. We have $\mathcal{S} = \{Dw, Up\}$, the loss function is always the ℓ_2. The penalty is omitted when equal to zero.

B. Update-rule proof

We recall here the update-rule (2) given at the end of Section 3.2.

$$Q^{s'}_{n+1}(\theta) = \min_{s|\exists\, edge\,(s,s')} \left\{ O^{s,s'}_n(\theta) + \gamma_{(s,s')}(y_{n+1},\theta) + \beta_{(s,s')} \right\} .$$

We name the path and vector realizing the best cost $Q^{s'}_{n+1}(\theta)$, defined in Equation 1, p^* and μ^*. We call s^* the corresponding vector of states. We have $s^*_{n+1} = s'$, $\mu^*_{n+1} = \theta$ and

$$Q^{s'}_{n+1}(\theta) = \sum_{t=1}^{n+1} \left(\gamma_{e^*_t}(y_t,\mu^*_t) + \beta_{e^*_t} \right) = \sum_{t=1}^{n} \left(\gamma_{e^*_t}(y_t,\mu^*_t) + \beta_{e^*_t} \right) + \left(\gamma_{(s^*_n,s')}(y_{n+1},\theta) + \beta_{(s^*_n,s')} \right) .$$

We will first show that

$$\sum_{t=1}^{n} \left(\gamma_{e^*_t}(y_t,\mu^*_t) + \beta_{e^*_t} \right) = Q^{s^*_n}_n(\mu^*_n) = O^{s^*_n,s'}_n(\theta).$$

(Proof) Restricting the path p^* and the vector μ^* to their first n elements, by definition of $Q^{s^*_n}_n(\mu^*_n)$ we have $\sum_{t=1}^n \left(\gamma_{e^*_t}(y_t,\mu^*_t) + \beta_{e^*_t} \right) \geq Q^{s^*_n}_n(\mu^*_n)$. Also, given that a move from parameter μ^*_n to θ is a valid transition from state s^*_n to s' and by the definition of $O^{s^*_n,s'}_n(\theta)$, we have $Q^{s^*_n}_n(\mu^*_n) \geq O^{s^*_n,s'}_n(\theta)$.

We will now proceed by contradiction. Let us assume: $\sum_{t=1}^n \left(\gamma_{e^*_t}(y_t,\mu^*_t) + \beta_{e^*_t} \right) > O^{s^*_n,s'}_n(\theta)$. We name the path and vector realizing the $O^{s^*_n,s'}_n(\theta)$ p^+ and μ^+. Extending this path and vector to $n+1$ with $s^+_{n+1} = s'$ and $\mu^+_{n+1} = \theta$ we get a better cost than p^* for $Q^{s'}_{n+1}(\theta)$ which is a contradiction.

So we have

$$Q^{s'}_{n+1}(\theta) = O^{s^*_n,s'}_n(\theta) + \gamma_{(s,s')}(y_{n+1},\theta) + \beta_{(s^*_n,s')},$$

and considering all possible states at time n we get the update-rule.

C. Backtracking

After running the Viterbi-like algorithm with update-rule (2), we need a backward procedure called backtracking to return the optimal change-point vector. First, we recover using Algorithm 1 the optimal vector of states $\hat{s} \in \{1,\dots,S\}^n$ and vector of means $\hat{\mu} \in \mathbb{R}^n$. We then find the best change-point vector $\hat{\tau} \subset \{1,\dots,n\}$ with Algorithm 2. The basic idea of Algorithm 1 is that if we knew $\hat{s}_{t+1}$ and $\hat{\mu}_{t+1}$ we could recover first $\hat{s}_t$ and then $\hat{\mu}_t$ taking the argmin of the update-rule (see lines 8 and 9 of Algorithm 1).

The obtained vectors $\hat{s}$ and $\hat{\mu}$ are simplified removing repetitions of consecutive identical states or values: i.e., $\hat{s}_t = \sigma_0$ and $\hat{\mu}_t = m_0\gamma^{t_2-t}$ for $t = t_1,\dots,t_2$ (including the case of exponential decay with parameter γ and $\gamma = 1$ if no decay). In that case, the index t_2 is an element of the change-point vector and m_0 its associated segment parameter. The vector of change-points can be built by a linear-in-time procedure described in Algorithm 2.

Notice that $\hat{\tau}$ is the `changepoints` vector returned by the `gfpop()` function. Restricting $\hat{s}$ and $\hat{\mu}$ vectors to positions in $\hat{\tau}$, these vectors are respectively the `states` and the `parameters` vectors.

Algorithm 1 Backtracking $\hat{s}$ and $\hat{\mu}$.

```
1: procedure BACKTRACK($(Q_1^1, \dots, Q_1^S), \dots, (Q_n^1, \dots, Q_n^S)$)
2: $\hat{\mu} \leftarrow$ empty vector of size $n$
3: $\hat{s} \leftarrow$ empty vector of size $n$
4: $(\hat{s}_n, \hat{\mu}_n) = \underset{(s,\mu)}{\text{argmin}}\{Q_n^s(\mu)\}$
5:                                  ▷ We can impose a subset of arrival states $\tilde{\mathcal{S}} \subset \{1, \dots, S\}$
6:                                  by $(\hat{s}_n, \hat{\mu}_n) \leftarrow \{\underset{(s,\mu)}{\text{argmin}}\{Q_n^s(\mu)\}\ ,\ s \in \tilde{\mathcal{S}}\}$
7: for $t = n - 1$ to $t = 1$ do
8:     $\hat{s}_t = \underset{s \mid \exists\, edge\,(s,\hat{s}_{t+1})}{\text{argmin}} \left\{ O_t^{s,\hat{s}_{t+1}}(\hat{\mu}_{t+1}) + \gamma_{(s,\hat{s}_{t+1})}(y_{t+1}, \hat{\mu}_{t+1}) + \beta_{(s,\hat{s}_{t+1})} \right\}$
9:     $\hat{\mu}_t = \underset{\mu \mid I_{(\hat{s}_t,\hat{s}_{t+1})}(\mu,\hat{\mu}_{t+1})}{\text{argmin}} \left\{ Q_t^{\hat{s}_t}(\mu) \right\}$        ▷ If $\hat{\mu}_t$ is such that the constraint is active,
10:                                 we have 'forced = TRUE' in gfpop() response.
11: end for
12: return $(\hat{s}, \hat{\mu})$
```

Algorithm 2 Change-point vector.

```
1: procedure CHANGE-POINT($\hat{s}, \hat{\mu}$)
2: $\hat{\tau} \leftarrow NULL$
3: $t \leftarrow n + 1$
4: while $t > 1$ do
5:     $\hat{\tau} \leftarrow (t - 1, \hat{\tau})$
6:     while $(\hat{s}_{t-1}, \gamma\hat{\mu}_{t-1}) = (\hat{s}_t, \hat{\mu}_t)$ do
7:         $t \leftarrow t - 1$
8:     end while
9: end while
10: return $\hat{\tau}$
```

D. Simulation results for isotonic regression

D.1. Linear signal

We simulate 100 linearly increasing time series and compute the mean of the MSE for each noise structure. The results are given in Table 1. We highlight in bold the two best results in each row and also give the standard deviation (SD).

In the Gaussian case the ℓ_2 isotonic regression is the best method. For the Student and for the corrupted scenarios the robust biweight loss with $\beta = 0$ is performing better in terms of MSE. Note that it is however much slower than PAVA. Including a penalty for change-points ($\beta_0 = 2\sigma^2 \log(n)$) deteriorates the results. This make sense as there are in fact no change-points in the data.

D.2. Iso-step signal

We simulate 100 step-wise increasing time series with 10 segments and compute the mean of the MSE for each noise structure. The results are given in Table 2. We highlight in bold the two best results in each row and also give the standard deviation (SD).

Isolinear simulations	linear fit	`isoreg` ℓ_2	`reg_1d` ℓ_1	`reg_1d` ℓ_2	`gfpop1` $\beta = 0$ ℓ_2	`gfpop2` $\beta = 0$ $K = 3\sigma^2$	`gfpop3` $\beta = \beta_0$ ℓ_2	`gfpop4` $\beta = \beta_0$ $K = 3\sigma^2$
Gauss								
MSE	0.0176	**0.702**	1.09	**0.699**	0.828	1.03	2.57	3.55
(SD)	(0.018)	(0.082)	(0.17)	(0.082)	(0.14)	(0.14)	(0.23)	(0.37)
Student								
MSE	0.0193	0.682	**0.549**	0.680	0.752	**0.560**	2.56	2.69
(SD)	(0.019)	(0.092)	(0.066)	(0.092)	(0.11)	(0.088)	(0.23)	(0.25)
Corrupted								
MSE	300	299	29.0	299	295	**3.49**	302	**6.99**
(SD)	(8.7)	(8.5)	(2.1)	(8.5)	(14)	(0.59)	(8.7)	(0.79)

Table 1: Mean-squared errors MSE $= \frac{1}{n}\sum_{i=1}^{n}(\hat{s}_i - s_i)^2$ for different algorithms on linear simulations with their empirical standard deviation (SD). We consider three types of noise: Gaussian, Student and corrupted.

Iso-step simulations	linear fit	`isoreg` ℓ_2	`reg_1d` ℓ_1	`reg_1d` ℓ_2	`gfpop1` $\beta = 0$ ℓ_2	`gfpop2` $\beta = 0$ $K = 3\sigma^2$	`gfpop3` $\beta = \beta_0$ ℓ_2	`gfpop4` $\beta = \beta_0$ $K = 3\sigma^2$
Gauss								
MSE	8.27	0.636	1.27	0.634	1.26	0.958	**0.378**	**0.586**
(SD)	(0.021)	(0.14)	(0.26)	(0.14)	(1.9)	(0.19)	(0.12)	(0.22)
Student								
MSE	8.27	0.591	0.588	0.588	1.07	0.443	**0.342**	**0.217**
(SD)	(0.021)	(0.17)	(0.15)	(0.17)	(0.74)	(0.14)	(0.21)	(0.088)
Corrupted								
MSE	302	297	30.4	297	292	**3.20**	299	**2.80**
(SD)	(8.2)	(7.9)	(3.6)	(7.9)	(15)	(0.62)	(8.1)	(0.65)

Table 2: Mean-squared errors MSE $= \frac{1}{n}\sum_{i=1}^{n}(\hat{s}_i - s_i)^2$ for different algorithms on step-wise simulations with their empirical standard deviation (SD). We consider three types of noise: Gaussian, Student and corrupted.

In the Gaussian and Student cases the penalized algorithms `gfpop3` and `gfpop4` with $\beta = \beta_0 = 2\sigma^2 \log(n)$ are better. For the corrupted scenario, we need the robust loss of algorithms `gfpop2` and `gfpop4` to get a much better MSE than other approaches. To confirm the benefit of using a penalized approach in the Student case, we plot the distribution of the MSE for five of the algorithms in Figure 24.

We also compare the ability of the different methods to estimate the number of steps. The average estimated number of steps over 100 simulations is reported in Table 3. Only the penalized algorithms are able to recover the true number of steps (10). The choice of a good penalty in isotonic simulations is an area of ongoing research in statistics (Gao *et al.* 2020).

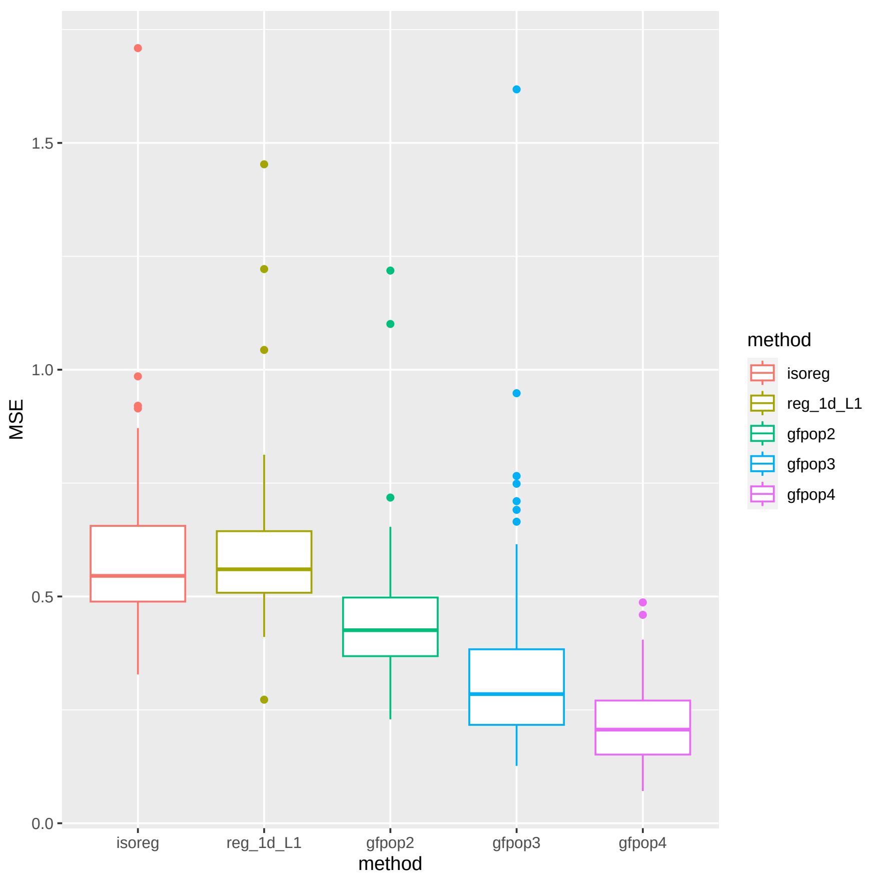

Figure 24: Box plots of the MSE for iso-step simulations with Student noise. The shape of the distribution is very similar for the 5 methods considered.

Iso-step simulations	`isoreg` ℓ_2	`reg_1d` ℓ_1	`reg_1d` ℓ_2	`gfpop1` $\beta = 0$ ℓ_2	`gfpop2` $\beta = 0$ $K = 3\sigma^2$	`gfpop3` $\beta = \beta_0$ ℓ_2	`gfpop4` $\beta = \beta_0$ $K = 3\sigma^2$
Gauss							
$\hat{D}$	67.1	59.1	66.9	70.2	68.3	10.0	10.0
(SD)	(6.7)	(6.1)	(6.6)	(8.3)	(6.7)	(0)	(0)
Student							
$\hat{D}$	68.9	63.9	68.8	70.5	70.8	10.0	10.0
(SD)	(6.7)	(6.7)	(6.7)	(6.8)	(7.6)	(0.1)	(0)
Corrupted							
$\hat{D}$	40.6	49.7	40.4	41.0	63.6	11.2	10.0
(SD)	(5.6)	(6.3)	(5.6)	(5.6)	(7.6)	(1.0)	(0.14)

Table 3: Mean number of segments over 100 simulations with 10^4 data points for different algorithms on step-wise simulations. We consider three types of noise: Gaussian, Student and corrupted.

Affiliation:

Vincent Runge
Université Paris-Saclay, CNRS, Univ Evry, Laboratoire de Mathématiques et Modélisation d'Evry (LaMME)
91037, Evry-Courcouronnes, France
E-mail: `vincent.runge@univ-evry.fr`

Toby Dylan Hocking
Northern Arizona University
School of Informatics, Computing, and Cyber Systems
Building 90, Room 120, 1295 S. Knoles Dr.
Flagstaff, AZ 86011, United States of America
E-mail: `toby.hocking@nau.edu`

Gaetano Romano
Department of Mathematics and Statistics
Lancaster University
LA1 4YF, United Kingdom
E-mail: `g.romano@lancaster.ac.uk`

Fatemeh Afghah
Department of Electrical and Computer Engineering
Clemson University
216 Palmetto BLvd.
Clemson, SC 29634, United States of America
E-mail: `fafghah@clemson.edu`

Paul Fearnhead
Department of Mathematics and Statistics
Lancaster University
LA1 4YF, United Kingdom
E-mail: `p.fearnhead@lancaster.ac.uk`

Guillem Rigaill
Université Paris-Saclay, CNRS, INRAE, Univ Evry
Institute of Plant Sciences Paris-Saclay (IPS2)
Batiment 630, 91405 Orsay, France. Université de Paris
CNRS, INRAE, Institute of Plant Sciences Paris-Saclay (IPS2)
Batiment 630, 91405 Orsay, France
and

Université Paris-Saclay, CNRS, Univ Evry
Laboratoire de Mathématiques et Modélisation d'Evry (LaMME)
91037, Evry-Courcouronnes, France
E-mail: guillem.rigaill@inrae.fr

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
March 2023, Volume 106, Issue 6 *Submitted:* 2021-02-26
doi:10.18637/jss.v106.i06 *Accepted:* 2022-04-01