Piecewise smooth regression by bootstrapped binary segmentation

Kate McDaid[1] and Florian Pein[1]

[1]Department of Pure Mathematics and Mathematical Statistics, University of Cambridge

January 27, 2022

We provide non-parametric regression estimators for piecewise smooth signals. The main function *BinSegBstrap* estimates a piecewise smooth signal by applying a bootstrapped test recursively (binary segmentation approach). A single bootstrapped test for the hypothesis that the underlying signal is smooth versus the alternative that the underlying signal contains at least one change-point can be performed by the function *BstrapTest*. A single change-point is estimated by the function *estimateSingleCp*. Parts of this work were inspired by [Gijbels and Goderniaux, 2004].

1 Model

We are considering an equally spaced fixed-design non-parametric regression model given by

$$Y_i = f(x_i) + \epsilon_i,$$

for $i = 1, \dots, n$. Here f is an unknown regression function defined on the unit interval $[0, 1]$, x_i are equally spaced fixed-design points, that is $x_i = i/n$, ϵ_i are independent and identically distributed regression errors with mean 0 and variance σ^2, $\sigma^2 > 0$, and Y_i is the noisy observation of f at x_i. The regression function f is defined to be piecewise smooth, i.e.

$$f(x) = \sum_{k=0}^{K} f_k(x) \mathbb{1}_{\tau_k \leq x < \tau_{k+1}}.$$

Here K is the number of change-points, $0 := \tau_0 < \tau_1 < \cdots < \tau_K < \tau_{K+1} := 1$ are the change-point functions and f_k are smooth signals with $f_{k-1}(\tau_k) \neq f_k(\tau_k)$.

2 *estimateSingleCp*: Estimation of a single change-point

Let us assume for the moment that we just want to estimate a single change-point, this means $K = 1$. Let h be a given bandwidth (see the following subsection for how to choose h by crossvalidation). We estimate the change-point location τ_1 by the maximum of the differences of left and right sided

running means. More precisely, let $b(h) := \lfloor nh \rfloor$ be the window size for the running means. We then define

$$\hat{t}_1(h) := \underset{t=b(h)+1,\ldots,n-b(h)}{\operatorname{argmax}} \frac{1}{b(h)} \sum_{i=1}^{b(h)} Y_{t+i-1} - \frac{1}{b(h)} \sum_{i=1}^{b(h)} Y_{t-i}$$

and $\hat{\tau}_1(h) := \hat{t}_1(h)/n$. We then estimate f_0 and f_1 by kernel smoothers with bandwidth h. And the jump size is defined by $\hat{f}_1(\hat{\tau}_1) - \hat{f}_0(\hat{\tau}_1)$.
This is implemented in the function *estimateSingleCp* and the resulting estimation is shown in Figure 1.

```
set.seed(1)
n <- 100
signal <- sin(2 * pi * 1:n / n)
signal[51:100] <- signal[51:100] + 5

y <- rnorm(n) + signal

# call of estimateSingleCp with fixed bandwidth 0.1
est <- estimateSingleCp(y = y, bandwidth = 0.1)

# estimated location
est$cp

## [1] 51

# estimated jump size
est$size

## [1] 4.496773

# plot of observations, true and estimated signal
plot(y, pch = 16, col = "grey30")
lines(signal)
lines(est$est, col = "red")
```

2.1 Bandwidth selection by crossvalidation

The bandwidth and hence the window sizes of the running means are selected by crossvalidation (unless the user input is a single bandwidth as above). Let $h_1, \ldots, h_m$ be potential bandwidths. For each bandwidth h_j we compute window size $b(h_j)$ and the estimated change-point location $\hat{\tau}_1(h_j)$. The bandwidth is then evaluated by the crossvalidation quantity

$$CV(h) = \sum_{i=1}^{i_0} \{\hat{g}_1^{-i}(x_i) - Y_i\}^2 + \sum_{i=i_0+1}^{n} \{\hat{g}_2^{-i}(x_i) - Y_i\}^2,$$

with $i_0 = \max\{i : x_i \leq \tau_1(h_j)\}$ and where $\hat{g}_1^{-i}(\cdot)$ and $\hat{g}_2^{-i}(\cdot)$ denote one sided kernel estimators on the intervals $[0, \hat{\tau}_1(h_j)]$ and $(\hat{\tau}_1(h_j), 1]$, respectively, using bandwidth h_j and disrecarding the i-th

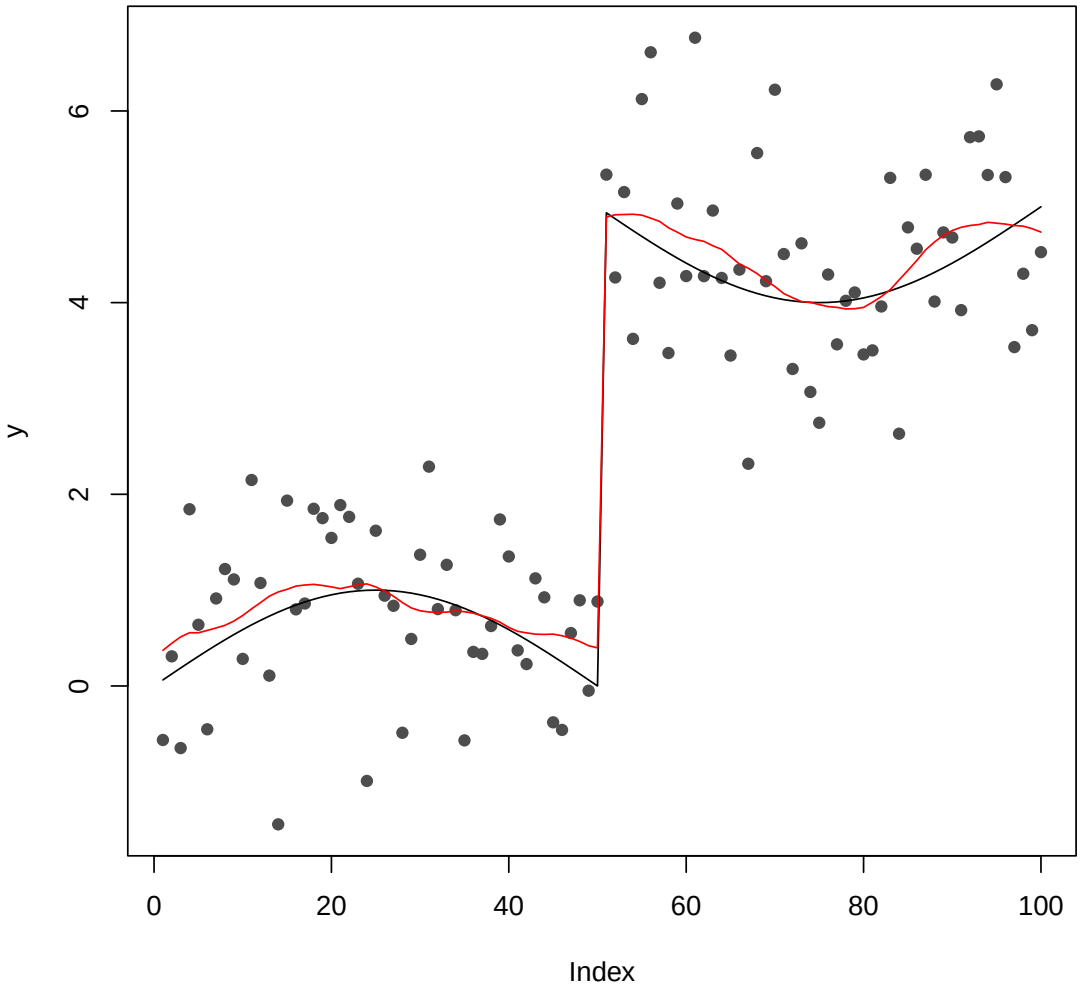

Figure 1: Observations (grey points), underlying signal (black line) and estimated signal (red line).

data point. The cross-validated bandwidth selector is then defined as

$$\hat{h}_{\mathrm{CV}} = \underset{h \in \{h_1, \ldots, h_m\}}{\operatorname{argmin}} \; CV(h).$$

This criterion was suggested in [Gijbels and Goderniaux, 2004].
Note that the test has almost no power when the bandwidth for the kernel smoother is too small, since then a change-point can be approximated well by a quickly changing smooth function.
We now recall *estimateSingleCp* without a user given bandwidth. Instead, the bandwidth will be determined by crossvalidation. The resulting fit is shown in Figure 2.

```
set.seed(1)
n <- 100
signal <- sin(2 * pi * 1:n / n)
signal[51:100] <- signal[51:100] + 5

y <- rnorm(n) + signal

# call of estimateSingleCp with crossvalidated bandwidth
est <- estimateSingleCp(y = y)

# crossvalidated bandwidth
est$bandwidth

## [1] 0.1482495

# estimated location
est$cp

## [1] 51

# estimated jump size
est$size

## [1] 4.37739

# plot of observations, true and estimated signal
plot(y, pch = 16, col = "grey30")
lines(signal)
lines(est$est, col = "red")
```

3 *BstrapTest*: Bootstrap test for a single change-point

The function *BstrapTest* tests whether the underlying signal is smooth or contains at least one change-point, i.e.

$$H_0 : K = 0 \;\; vs. \;\; H_1 : K \neq 0.$$

As test statistic we simply use absolute value of the jump size of the previous estimator, i.e.

$$T := \left| \hat{f}_1(\hat{\tau}_1) - \hat{f}_0(\hat{\tau}_1) \right|.$$

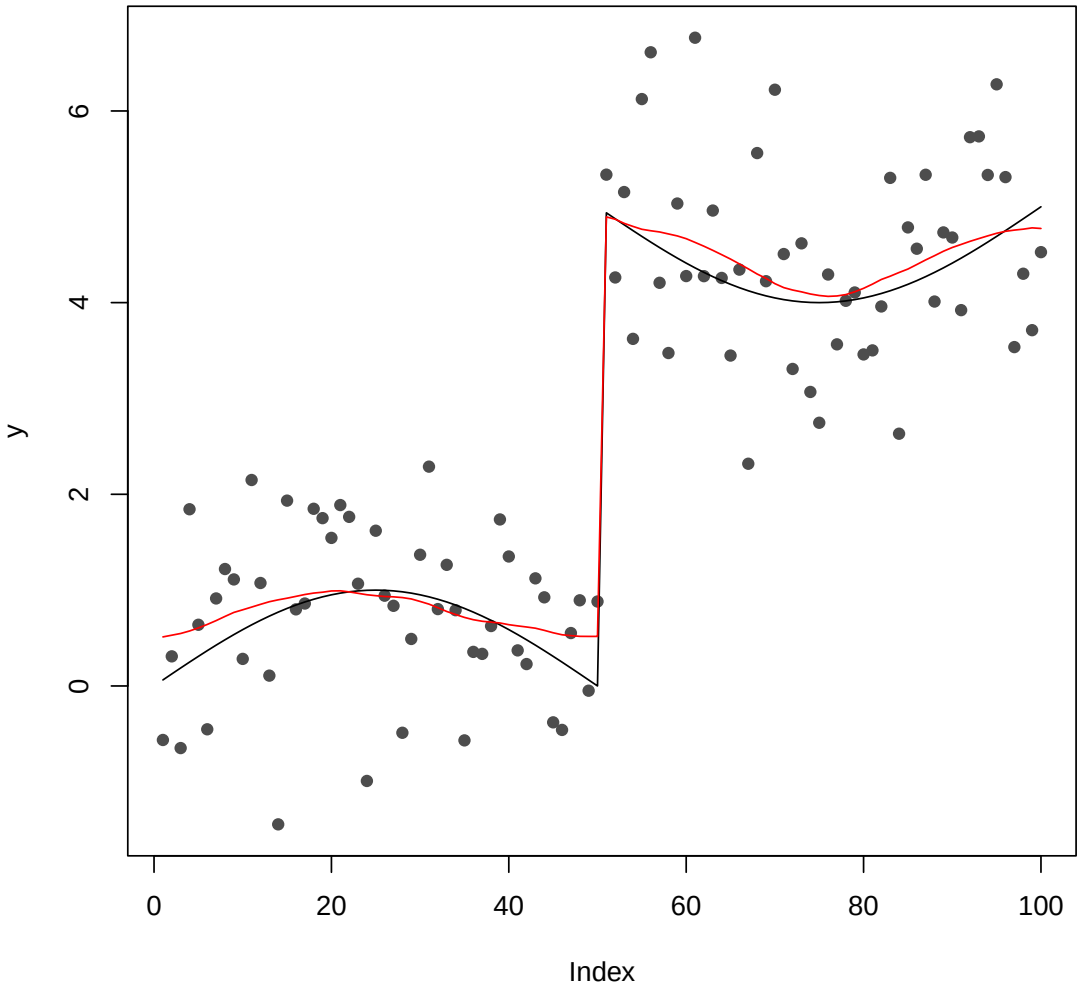

Figure 2: Observations (grey points), underlying signal (black line) and estimated signal (red line).

Critical value and p-value are obtained by boostrapping: We estimate the errors by subtracting the previous estimate $\hat{f}$ from the observations. From these estimated errors we resample with replacement $\epsilon_1^*, \ldots, \epsilon_n^*$. And defined our bootstrapped observations as these errors plus a usual kernel estimate of the observations with a crossvalidated bandwidth. Finally, we compute for these observations our test statistic and repeat the procedure B times. This approach was proposed in [Gijbels and Goderniaux, 2004] and more details can be found there as well.

```
set.seed(1)
n <- 100
signal <- sin(2 * pi * 1:n / n)
signal[51:100] <- signal[51:100] + 5

y <- rnorm(n) + signal

test <- BstrapTest(y = y)

# whether the test rejected
test$outcome

## [1] TRUE

# p-Value
test$pValue

## [1] 0
```

4 *BinSegBstrap*: Estimates a piecewise smooth signal

To estimate a signal with arbitary many changes we use binary segmentation and call the previous test recursively. The final estimator is given by kernel smoothing on each segment separately. To this end, we use a bandwidth that is jointly selected by crossvalidation.
More precisely, binary segmentation is a generic technique for multiple change-point detection in which we initially search the entire data-set for one change-point. Once a change-point is detected the data are split into two subsegments defined by the detected change point. A similar search is performed on both sub-segment possibly resulting in further splits. Recursion on a given segment continues until the null hypothesis that the underlying signal is smooth on the considered subsegment is accepted. A pseudocode for the method of standard binary segmentation is given in [Fryzlewicz, 2014].
This methodology can be applied by the function *BinSegBstrap*. Figure 3 shows the estimated signal for a function with three true change-points.

```
set.seed(1)
n <- 200
signal <- sin(2 * pi * 1:n / n)
signal[51:100] <- signal[51:100] + 5
signal[151:200] <- signal[151:200] + 5
```

```r
y <- rnorm(n) + signal

est <- BinSegBstrap(y = y)

# estimated change-points
est$cps

## [1]  51 101 151

# plot of observations, true and estimated signal
plot(y, pch = 16, col = "grey30")
lines(signal)
lines(est$est, col = "red")
```

References

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

[Gijbels and Goderniaux, 2004] Gijbels, I. and Goderniaux, A.-C. (2004). Bootstrap test for change-points in nonparametric regression. *Journal of Nonparametric Statistics*, 16(3-4):591–611.

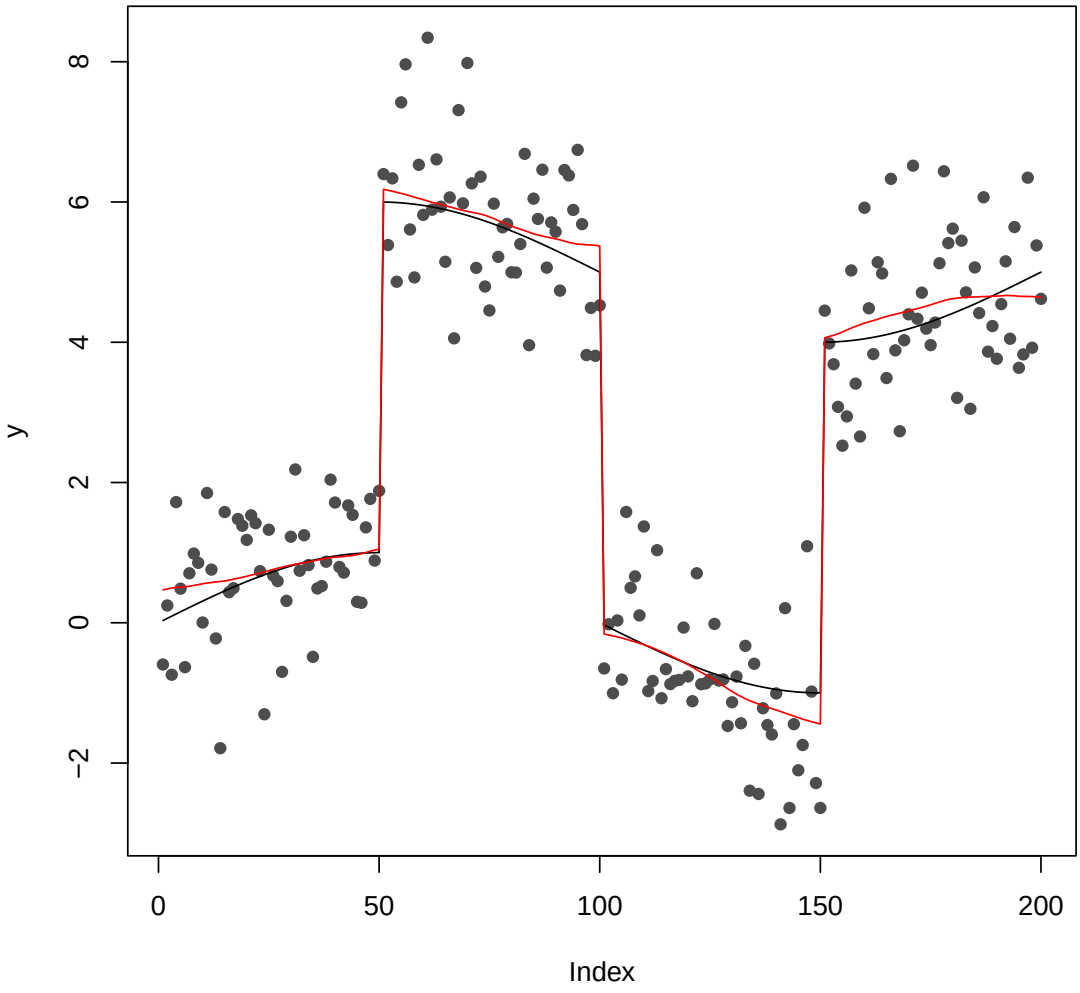

Figure 3: Observations (grey points), underlying signal (black line) and estimated signal (red line).