A Vignette for Package Polydect

Zhihua Su

March 27, 2009

1 Background

This package is an implement to the paper "On Jump Detection In Regression Curves Using Local Polynomial Kernel Estimation" by B. Zhang, Z. Su and P.Qiu.

1.1 Difference Polynomial Kernel Estimation

In the one-dimensional case, we formulate a jump regression model as follows:

$$f(x) = g(x) + \sum_{i=1}^{p} d_j I(x > s_j), \text{for } x \in [0, 1],$$

where g is a continuous function, p is the number of jump points, $\{s_j, j = 1, 2, ..., p\}$ are the jump positions, and $\{d_j, j = 1, 2, ..., p\}$ are the corresponding jump magnitudes.

But in reality, our observations will also contain noises. And here for simplicity, we assume the noises ϵ_i's are independent and identically distributed random errors with $E(\epsilon_i) = 0$ and $Var(\epsilon_i) = \sigma^2(X_i)$. But for simplicity, here we just assume that the noise level for all the design points are the same. So we have

$$Y_i |\ X_i = f(X_i) + \epsilon_i, \qquad i = 1, 2, \cdots, n,$$

where Y_i's are data we can observe. And X_i's, which are called the design points, are the locations we take observations.

One approach to detect jumps is to use a difference polynomial kernel estimation procedure. This procedure estimate the left limit of a design point and also the right limit of a design point by a local polynomial kernel estimator. For an example, if we use a local kernel estimator (polynomial order equals to 0), the estimator will be:

$$M_{DKE}(x) = \frac{1}{nh_n}\sum_{i=1}^{n} Y_i K_1\left(\frac{x_i - x}{h_n}\right) - \frac{1}{nh_n}\sum_{i=1}^{n} Y_i K_2\left(\frac{x_i - x}{h_n}\right)$$

where h_n is a positive bandwidth parameter, K_1 and K_2 are two one-sided kernel functions.

If the design point is a continuous point in the model, then the difference may expected to be small. However, if it is a jump point, then this different is expected to be large. And as shown in the paper, as n goes large, the difference will approach the true jump magnitude for a jump point and vanish for a continuous points.

Therefore, to detect the jump points, we estimate the variance of our detector first, then we may compare it with a threshold value, using a significance level α. For example, the variance for the difference kernel estimator is $2.4\sigma^2/nh_n$, so the threshold value will be $z_\alpha\sqrt{2.4\sigma^2/nh_n}$, given the significance level α. If the value of the detector is greater than the threshold value, we may think that the point is a jump candidate, if not, we may consider it as a continuous point.

Polynomial estimators with higher order will reduce bias in estimation, but their variances will increase. In practice, the most commonly used difference polynomial kernel estimators are kernel and linear estimators (polynomial order equals to 0 and 1 respectively). The paper shows that actually, we only need polynomial order up to 2 for most of the cases. Polynomials with higher order will not improve the result of detection a lot, while they will increase the computing time a lot. More specifically, when the function is flat, we may use the kernel estimator. When the function has large slope but not large curvature, we may use the linear estimator, and when we have a function with large curvature, we may use the quadratic estimator. However, if we do not have a substantial amount of design points, the quadratic estimator does not work well either since it has a larger variance. In this case, use the linear estimator.

1.2 Some technical details

1.2.1 Bandwidth and significance level Selection

In order to use the detector, we need to select a bandwidth h_n which decides how many points should be used in estimating the one-sided limits. And also, in order to reduce false detection, we need a significance level α. Our method is based on the bootstrap procedure described in Gijbels, I. and Goderniaux, A.-C. (2004). For a given observed dataset $\mathcal{D} = \{(x_1, Y_1), (x_2, Y_2), \ldots, (x_n, Y_n)\}$ and given parameters h_n and α_n in (3)–(7), assume that the estimated jumps are $\widehat{S} = \{\widehat{s}_j, j = 1, 2, \ldots, \widehat{J}\}$ and the estimated jump magnitudes are $\{\widehat{d}_j, j = 1, 2, \ldots, \widehat{J}\}$. The Hausdorff distance between $\widehat{S}$ and the set of true jumps S is

$$d_H(S, \widehat{S}; h_n, \alpha_n) = \max \left\{ \sup_{s_1 \in S} \inf_{s_2 \in \widehat{S}} |s_1 - s_2|, \sup_{s_1 \in \widehat{S}} \inf_{s_2 \in S} |s_1 - s_2| \right\}.$$

Bandwidth Selection Procedure

- **Step 1:** Define new observations

$$\tilde{Y}_i = Y_i - \sum_{j=1}^{\widehat{J}} \widehat{d}_j I(x_i > \widehat{s}_j), \text{ for } i = 1, 2, \ldots, n.$$

  Estimate g by local linear kernel smoothing with bandwidth h_{est} from data $\{(x_i, \tilde{Y}_i^*), i = 1, 2, \ldots, n\}$, and the estimator is denoted as $\widehat{g}$. Then, define residuals

$$\widehat{\epsilon}_i = Y_i - \widehat{g}(x_i) - \sum_{j=1}^{\widehat{J}} \widehat{d}_j I(x_i > \widehat{s}_j), \text{ for } i = 1, 2, \ldots, n.$$

- **Step 2:** Obtain B batches of resampled residuals from $\{\widehat{\epsilon}_i, i = 1, 2, \ldots, n\}$; each batch has n values. For the b-th batch of resampled residuals, denoted as $\{\widehat{\epsilon}_i^*, i = 1, 2, \ldots, n\}$, define pseudo-data as follows.

$$Y_i^* = \widehat{g(x_i)} + \sum_{j=1}^{\widehat{J}} \widehat{d}_j I(x_i > \widehat{s}_j) + \widehat{\epsilon}_i^*, \text{ for } i = 1, 2, \ldots, n.$$

- **Step 3:** Apply the jump detection procedure (3)–(7) with parameters h_n and α_n to the b-th pseudo-data, and the set of detected jumps is denoted as $\widehat{S}_b$. Then, the Hausdorff distance $d_H(S, \widehat{S}; h_n, \alpha_n)$ is estimated by

$$\widehat{d}_H(S, \widehat{S}; h_n, \alpha_n, h_{est}) = \frac{1}{B}\sum_{b=1}^{B} d_H(\widehat{S}, \widehat{S}_b; h_n, \alpha_n, h_{est}),$$

where $d_H(\widehat{S}, \widehat{S}_b; h_n, \alpha_n, h_{est})$ denotes the Hausdorff distance between $\widehat{S}$ and $\widehat{S}_b$, which depends on parameters h_n, α_n, and h_{est}.

- **Step 4:** Parameters h_n and α_n are approximated by the solution of

$$\min_{h_n>0}\min_{\alpha_n\in[0,1]}\left[\min_{h_{est}>0}\widehat{d}_H(S, \widehat{S}; h_n, \alpha_n, h_{est})\right].$$

1.2.2 Estimating variance

In order to estimate the variance of the detector, we need to estimate the noise level of the data. For estimating the noise level, we may use the local linear estimator. Suppose we have two one-sided local linear estimators M_{DLK-} and M_{DLK+} for a particular design point, then the variance of the noise at this point is estimated by

$$\min\left\{\frac{\sum(Y_i - M_{\mathrm{DLK}-})^2 K_1(\frac{x-x_i}{h_n})}{\sum K_1(\frac{x-x_i}{h_n})}, \frac{\sum(Y_i - M_{\mathrm{DLK}+})^2 K_2(\frac{x-x_i}{h_n})}{\sum K_2(\frac{x-x_i}{h_n})}\right\}$$

1.2.3 Modification procedure

Since the design points near the true jump points have high probability to be detected, in order to reduce false detection, we apply the modification procedure described in Qiu (1994). The basic idea is that we define a sequence of jump candidates which are close to each other as a *tie*, and we replace a tie by its middle point. More specifically, let us first assume that $a_{i_1} < a_{i_2} < ... < a_{i_r}$ are the points flagged as jump candidates, then we call $\{a_{i_j}, j = r_1, r_1+1, ..., r_2\}$ a tie if

$$\begin{cases} a_{i_j+1} - a_{i_j} \le h_n, j = r_1, r_1+1, ..., r_2-1 \\ a_{i_{r_1}} - a_{i_{r_1-1}} > h_n \\ a_{i_{r_2+1}} - a_{i_{r_2}} > h_n \end{cases}$$

For the jump candidates in a tie, we replace all of them by a new candidate which is defined by the middle point $(a_{i_{r_2}} + a_{i_{r_1}})/2$ of the tie. After this

modification, the current candidates consist of two types of points: those do not belong to any ties and the middle points of all ties.

2 About the package

The package contains the following stuffs:

- The one-sided local polynomial kernel estimators: kernel, linear, quadratic, cubic. They give the estimation of the difference between the left and right limits at a certain point. The kernel functions used are the one-sided kernel function: $K_1 : y = 1.5(1 - x^2), x \in (-\infty, 0)$ and $K_2 : y = 1.5(1 - x^2), x \in (0, \infty)$.
- The function for selecting bandwidth and significance level using bootstrap procedure.
- The function to implement the modification procedure described in Qiu (1994).
- The function for estimating the noise level using one-sided linear kernel estimators.
- The function to calculate Hausdorff distance.
- Some real data-sets for illustration.

3 An example

Now we use the data of the weekly Dow Jones Industry Average open price (from September 2000 to August 2002) to demonstrate the usage of the package. First, we will load the data and let us take a look at the data in the plot. As shown in the plot, it has the typical characteristic of financial data, which jumps up and down dramatically during the time. There is a big jump around the 56th observation, which corresponds to the week following 9.11.

```
> library(polydect)
> data(DJIA)
> head(DJIA)

      Open
1 11219.54
```

```
2 11221.76
3 11219.54
4 10926.42
5 10847.37
6 10659.06

> attach(DJIA)

> ts.plot(Open, type = "l")
```

We suppose that the design points are within 0 and 1. Therefore the design points and other variables are defined as follows:

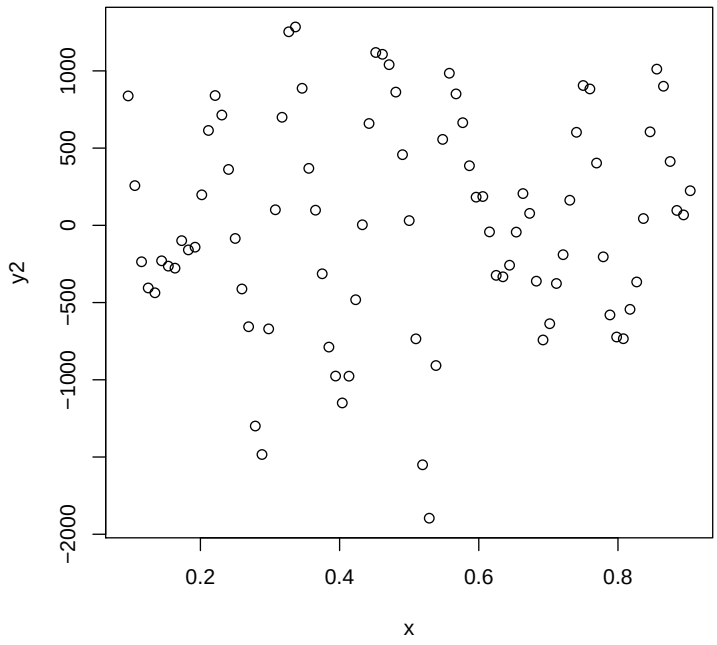
```
> n <- length(Open)
> X <- c(1:n)/n
> Y <- Open
> x <- X[floor(0.1 * n):ceiling(0.9 * n)]
```

To estimate the noise level of the data, we may first specify the bandwidth and then use the sigma estimation function. Since here we are mainly demonstrating the use of the function *sig.est*, the bandwidth is not chosen by bootstrap described before and is just picked up based on experience.

```
> h2 = 0.15
> variance <- sig.est(x, X, Y, h2)
> sigma <- sqrt(variance)
> sigma

[1] 350.7547
```

As we may notice from the plot that the data has quite large curvature, therefore, we may use the one-sided local quadratic kernel estimator for detecting the jumps, and then we will use the modification procedure used in Qiu (1994) to identify the jump candidates:

```
> C = sqrt(2 * sigma^2 * 4.44178/n/h2) * qnorm(0.999999995)
> C

[1] 1516.857

> y2 <- numeric(length(x))
> y2 <- m.det2(h2, x, X, Y)
```


y2
1000
500
0
−500
−1000
−2000
0.2
0.4
0.6
0.8
x

```
> plot(x, y2)
> abline(C, 0)
```

```
> tot <- det.jmp(x, abs(y2), h2, C, n)[[1]]
> pos <- det.jmp(x, abs(y2), h2, C, n)[[2]]
> tot
```

```
[1] 1
```

```
> pos
```

```
[1] 0.5278846
```

The position of the jump position just corresponds to the week after 9.11 which causes a big change in the open price of the stocks.

Here we also did not choose the bandwidth by rigorous bootstrap procedure, but just by experience. To choose a good bandwidth, we need to use the function *bds*. To use the boostrap procedure, there are actually two bandwidths involved, one for detection and the other for estimating the continuous part. We may try different combinations of the two, and find a "best" combination which gives the smallest Hausdorff distance. For example, we can try combination as follow:

```
> h = 0.15
> h2 = 0.2
> B = 100
> order = 2
> bds(h, B, h2, X, Y, x, order)
```

```
[1] 0.68
```

Having the bandwith, we may do the jump detection and the modification procedure in one step, using the function *polydect*. But you have to specify the order:

```
> h = 0.15
> polydect(h, X, Y, x, 2)
```

```
[1] 0.5326923
```

It gives the same result as we have before.

4 Reference

Zhang, B., Su, Z., and Qiu, P. (2009), *On Jump Detection In Regression Curves Using Local Polynomial Kernel Estimation.*

Qiu, P. (2005), *Image Processing and Jump Regression Analysis*, New York: John Wiley & Sons.

Gijbels, I., and Goderniaux, A.-C. (2004), "Bandwidth selection for change point estimation in nonparametric regression," *Technometrics*, **46**, 76–86.

Qiu, P. (1999), "Comparisons of several local smoothing jump detectors in one-dimensional nonparametric regression," *The ASA Proceedings of the Statistical Computing Section*, 150–155.