Advanced Statistical Modeling

Non-parametric models - Alternative estimations of conditional variance

Haoran Mo, Alexandra Yamaui

November 22, 2017

The first approach is Rice. We would like to make it more clear: The variance of a random variable tells us something about the spread of the possible values of the variable. For a discrete random variable $y_i - y_{i-1}$, the variance of $y_i - y_{i-1}$ is written as $\text{Var}(y_i - y_{i-1})$, and $\text{Var}(y_i - y_{i-1}) = E[((y_i - y_{i-1})\breve{}m)^2]$ where m is $E(y_i - y_{i-1})$, it can be written as $Var(y_i - y_{i-1}) = E[(y_i - y_{i-1})^2] - m^2$. After further more steps as shown in the chapter 2.4, we can obtain the function $\hat{\sigma}^2 = \frac{1}{2(n-1)} \sum_{i=2}^{n} (y_i - y_{i-1})^2$

The second approach is the method proposed by Gasser, Stroka and Jennen-Steinmetz in 1986, which consists in linear interpolation of every point (x_i, y_i) with the previous and following observations (x_{i-1}, y_{i-1}) and (x_{i+1}, y_{i+1}), respectively. The observations are sorted based on the value of x_i in ascending order. The idea behind this approach is that $\hat{y_i}$ $(\hat{m_i})$ is approximetly equal to $(x_{i-1}, m(x_{x-i}))$ and $(x_{x+1}, m(x_{x+i}))$ if the function m is smooth and x_i, x_{i-1} and x_{i+1} are close enough.

The linear interpolation for x_i is defined by:

$$\hat{y_i} = \frac{x_{i+1} - x_i}{x_{i+1} - x_{i-1}} y_i + \frac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}} y_{i+1} = a_i y_{i-1} + b y_{i+1}$$

The estimation of the residuals would be the difference between the estimated value $\hat{y_i}$ from the interpolation and the real value y_i with an expected value $E(\tilde{\varepsilon}) \approx 0$:

$$\tilde{\varepsilon} = \hat{y_i} - y_i = a_i y_{i-1} + b y_{i+1} - y_i$$

The residuals can be seen as the deviation from the true value, therefore:

$$E(\tilde{\varepsilon}^2) \approx V(\tilde{\varepsilon_i}) = (a_i^2 + b_i^2 + 1)\sigma^2$$

and the residual variance $\hat{\sigma}^2$ can be approximated as:

$$\hat{\sigma}^2 = \frac{1}{n-2} \sum_{i=2}^{n-1} \frac{1}{a_i^2 + b_i^2 + 1} \tilde{\varepsilon}_i^2$$

Below we present the implementation of a function that calculates the estimation of the residual variance through both approaches. For the second method, in the cases when the previous (x_{i-1}) and following (x_{i+1}) observations are equal, the maximum of lower observations and the minimum of higher observations are taken instead.

```r
calculate_residual_variance <- function(X,Y) {
  XY <- data.frame(X,Y)
  XY <- XY[order(XY$X),]
  rownames(XY) <- 1:nrow(XY)
  n <- length(Y)

  # Rice
  t2 = XY[-c(length(X)),"Y"]
  t1 = XY[-c(1),"Y"]
  rice.sigma_2 <- (sum((t2-t1)^2))/(2*(n-1))
```

```r
  # Gasser, Sroka, and Jennen-Steinmetz
  summation <- 0

  for (i in 2:(n-1)) {
    xi <- XY[i, 'X']

    x.previous = XY[i-1,'X'] # x_i-1
    x.following = XY[i+1,'X'] # x_i+1
    y.previous = XY[i-1,'Y'] # y_i-1
    y.following = XY[i+1,'Y'] # y_i+1

    if (x.previous == x.following) {
      x.following <- min(XY[XY$X > xi,]$X)
      x.previous <- max(XY[XY$X < xi,]$X)
    }

    a_i <- (x.following - xi)/(x.following - x.previous)
    b_i <- (xi - x.previous)/(x.following - x.previous)

    y.hat_i <- a_i*y.previous + b_i*y.following

    residual.hat_i <- y.hat_i - XY[i, 'Y']
    summation <- summation + (residual.hat_i^2/(a_i^2 + b_i^2 + 1))
  }

  gasser.sigma2 <- summation/(n - 2)
  return(list(rice.sigma_2, gasser.sigma2))
}

X <- boston.c$LSTAT
Y <- boston.c$RM
(result <- calculate_residual_variance(X,Y))
```

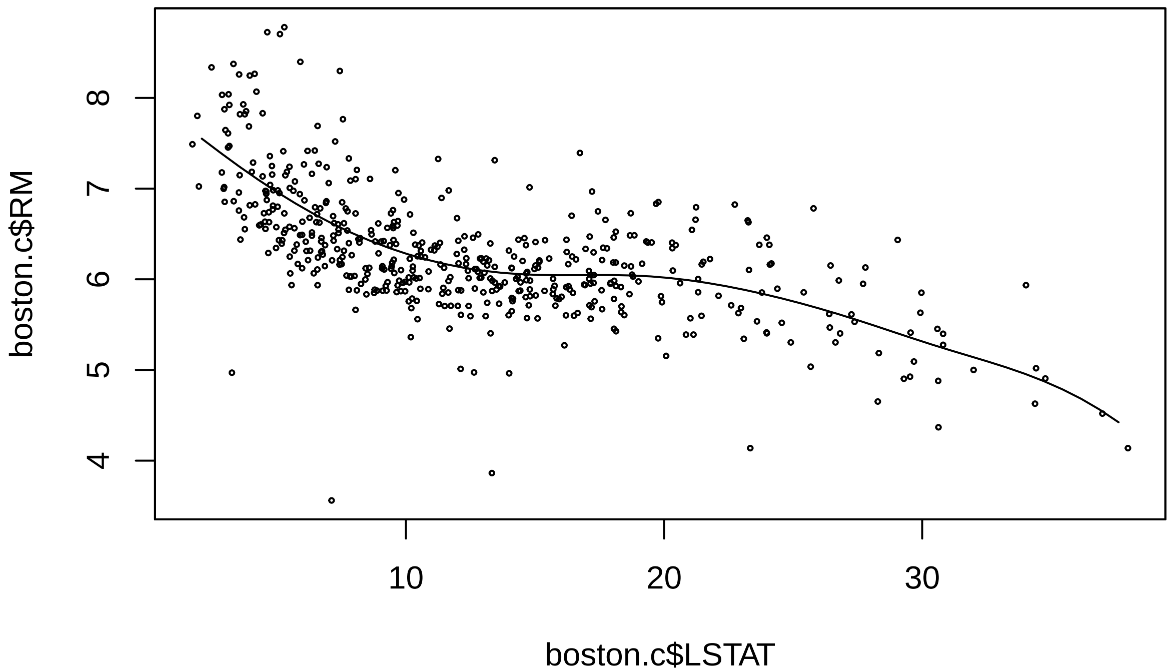
```
## [[1]]
## [1] 0.2825677
##
## [[2]]
## [1] 0.2677084
```

Comparing the estimated values with another R packages methods we see that the values are similar: loess: 0.50088 sm: 0.5097599 approach 1: 0.5315709 approach 2: 0.5174054

```r
loess.fit <- loess(RM~LSTAT, data = boston.c)
residual.variance.loess <- var(loess.fit$residuals)
print(sqrt(residual.variance.loess))
```

```
## [1] 0.50088
```

```r
sm.fit <- sm.regression(boston.c$LSTAT,boston.c$RM)
```




```r
print(sm.fit$sigma)
```

```
## [1] 0.5097599
```

```r
print(sqrt(unlist(result[1])))
```

```
## [1] 0.5315709
```

```r
print(sqrt(unlist(result[2])))
```

```
## [1] 0.5174054
```