Estimating the conditional variance by local linear regression

Caballero Vergés Biel, Menzenbach Svenja and Reyes Illescas Kleber Enrique

2023-11-18

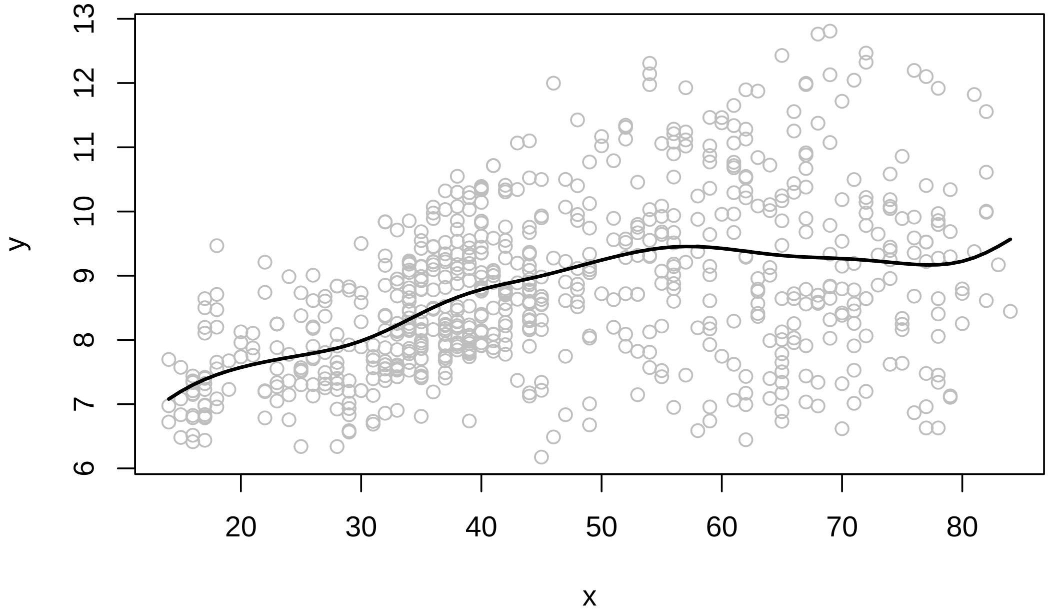
```
x <- Yr
y <- lg_weight
n <- length(x)
```

Estimating the conditional variance (Using loc.pol.reg)

- First, use the function loc.pol.reg that you can find in ATENEA and choose all the bandwidth values you need by leave-one-out cross-validation

1.Fit a nonparametric regression to data (x_i, y_i) and save the estimated values $\hat{m}(x_i)$.

```
fit.lpr <- fit.loc_pol_reg(x, y)
```



```r
fit_y <- fit.lpr$lpr
mtgr <- fit_y$mtgr # \hat_{m}(x_i)
```

Looking at the plot of the data we can see that the variance is not constant for all x. The variance at the beginning is lower and it increases as the value of x increases, until almost a value of x equal to 60, as it decreases until a value of x of 80 that it starts to increase again.

2.Transform the estimated residuals $\hat{\epsilon} = y_i - \hat{m}(x_i)$

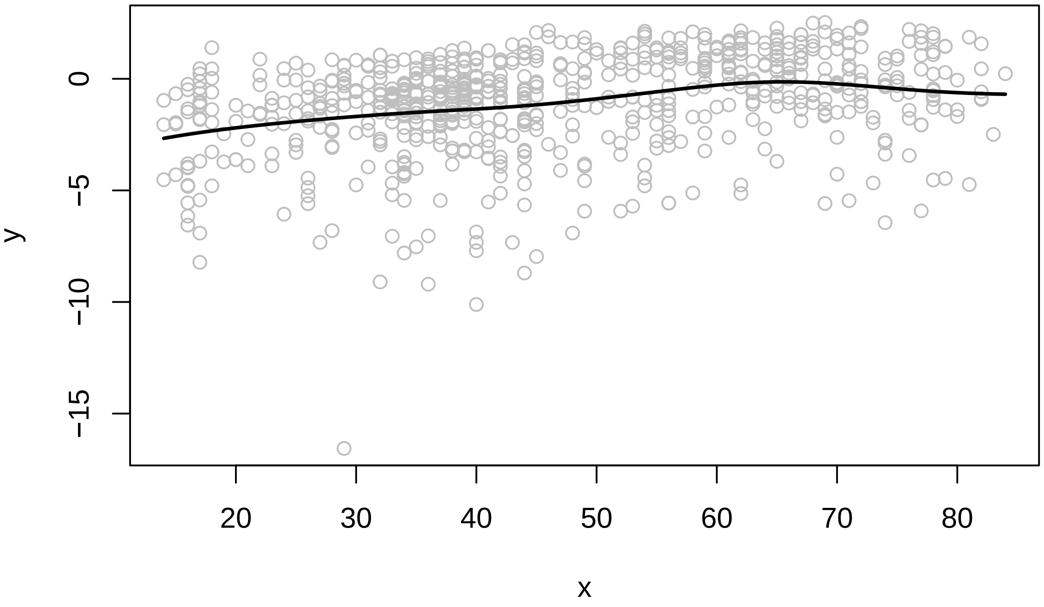
$$z_i = \log \epsilon_i^2 = log((y_i - \hat{x}_i))^2)$$

```r
hat_e <- y - mtgr
z <- log((hat_e)^2)
```

3. Fit a nonparametric regression to data (x_i, z_i) and call the estimated function $\hat{q}(x)$. Observe that $\hat{q}(x)$ is an estimate of log $\sigma^2(x)$.

In order to estimate a function that helps us to explain the variable variance present in our data, we can also use the local linear regression in order to fit a nonparametric regression line that estimates the variance in each point of x.

```r
fit_z <- fit.loc_pol_reg(x, z)$lpr
```



```
qtgr <- fit_z$mtgr
```

We can see the small variations in the curved line. It increases slightly as the value of x increase, but at x=65 more or less, it seems to decrease a little bit. (Probably this is explained because we have less density of point at the end of our dataset).

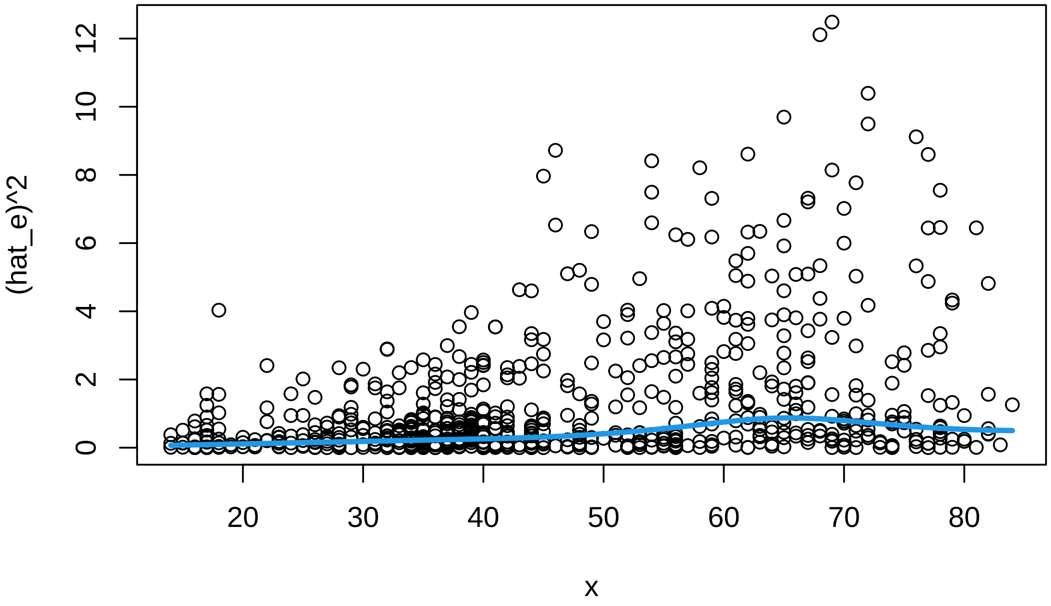
4. Estimate $\sigma^2(x)$ by

$$\hat{\sigma}^2(x) = e^{\hat{q}(x)}$$

```
sig.sqr <- exp(qtgr)
```

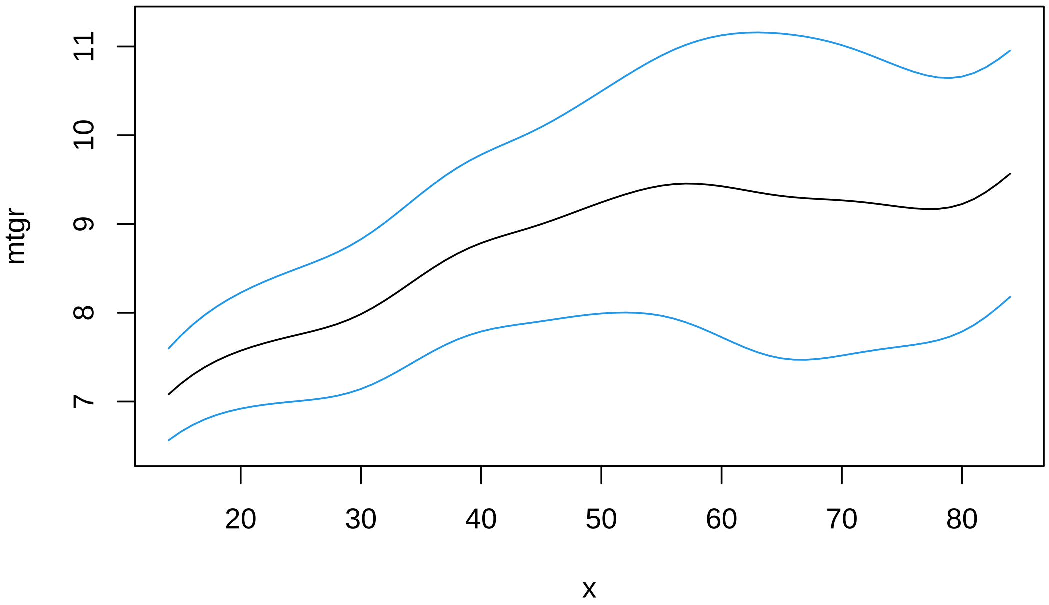
Plots

```
plot(x, (hat_e)^2)
lines(x, sig.sqr, col=4, lwd=3)
```



Here we can see that the residual of the estimation and the data points have a high variance that increases over x until about x=68 and decreases again from there. This also shows the changing variance which is also shown in the blue line. While the variance is close to zero at the beginning it rises to about 1.

```
y.min <- min(mtgr - 1.96 * sqrt(sig.sqr)) - 0.1
y.max <- max(mtgr + 1.96 * sqrt(sig.sqr)) + 0.1
plot(x,mtgr, type = 'l', ylim=c(y.min, y.max))
lines(x,mtgr + 1.96 * sqrt(sig.sqr), col=4)
lines(x,mtgr - 1.96 * sqrt(sig.sqr), col=4)
```



Here we can see very well that the variance changes over x as the blue boundaries don't have the same distance to the estimate at all times. They represent an uncertainty of our estimation.

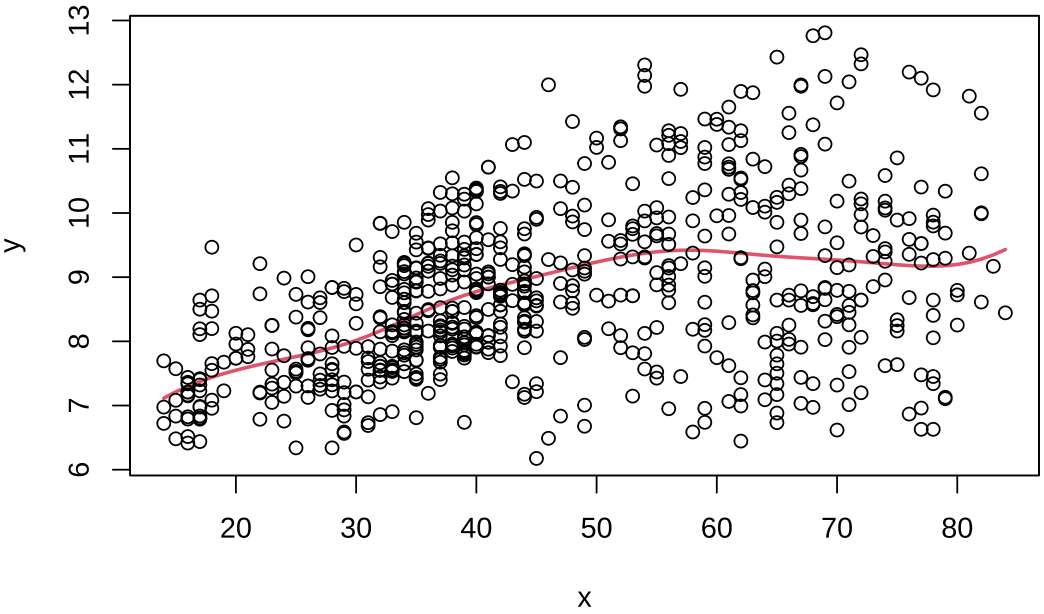
Estimating the conditional variance (Using sm.regression)

- Second, use the function **sm.regression** from library **sm** and choose all the bandwidth values you need by *direct plug-in*

1.Fit a nonparametric regression to data (x_i, y_i) and save the estimated values $\hat{m}(x_i)$.

(In order to simplify the calculations and facilitate comparison with the previous section, we use **eval.points=x**)

```
h.dpi <- dpill(x=x, y=y, range.x=range(x))
sm_regression <- sm.regression(x=x, y=y, eval.points=x, h=h.dpi, pch=1, cex=1, col=2, lwd=2)
```



```
mtgr_sm <- sm_regression$estimate
```

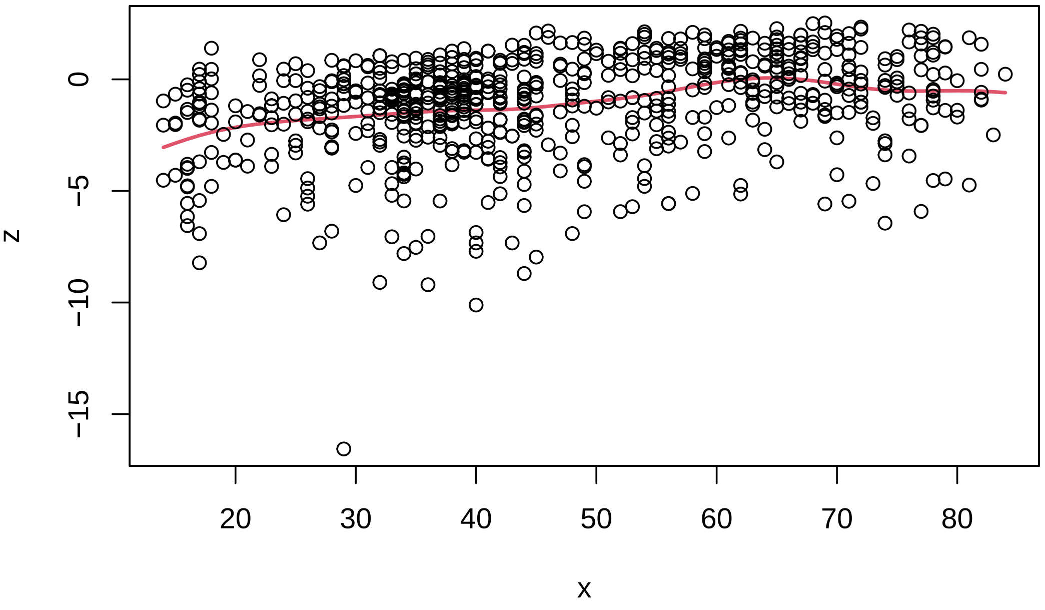
2.Transform the estimated residuals $\hat{\epsilon} = y_i - \hat{m}(x_i)$

$$z_i = \log \epsilon_i^2 = log((y_i - \hat{x}_i))^2)$$

```
hat_e_sm <- y - mtgr_sm
z_sm <- log((hat_e_sm)^2)
```

3. Fit a nonparametric regression to data (x_i, z_i) and call the estimated function $\hat{q}(x)$. Observe that $\hat{q}(x)$ is an estimate of log $\sigma^2(x)$.

```
sm_regression_z <- sm.regression(x=x, y=z, pch=1, cex=1, col=2, lwd=2,
                                 eval.points=x,
                                 h=dpill(x=x, y=z, range.x=range(x)))
```



```
qtgr_sm <- sm_regression_z$estimate
```

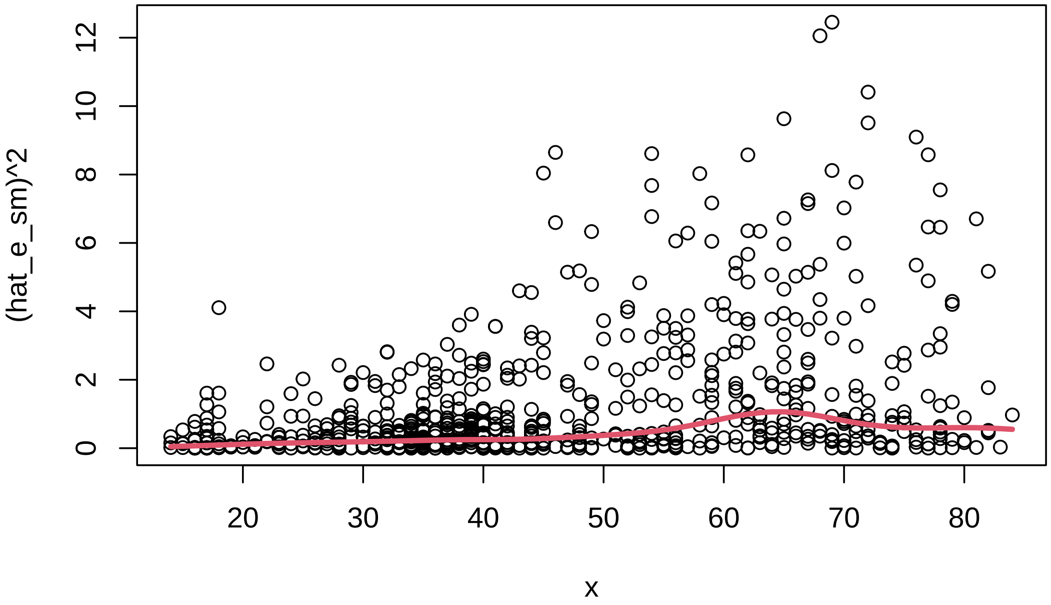
4. Estimate $\sigma^2(x)$ by

$$\hat{\sigma}^2(x) = e^{\hat{q}(x)}$$

```
sig.sqr_sm <- exp(qtgr_sm)
```

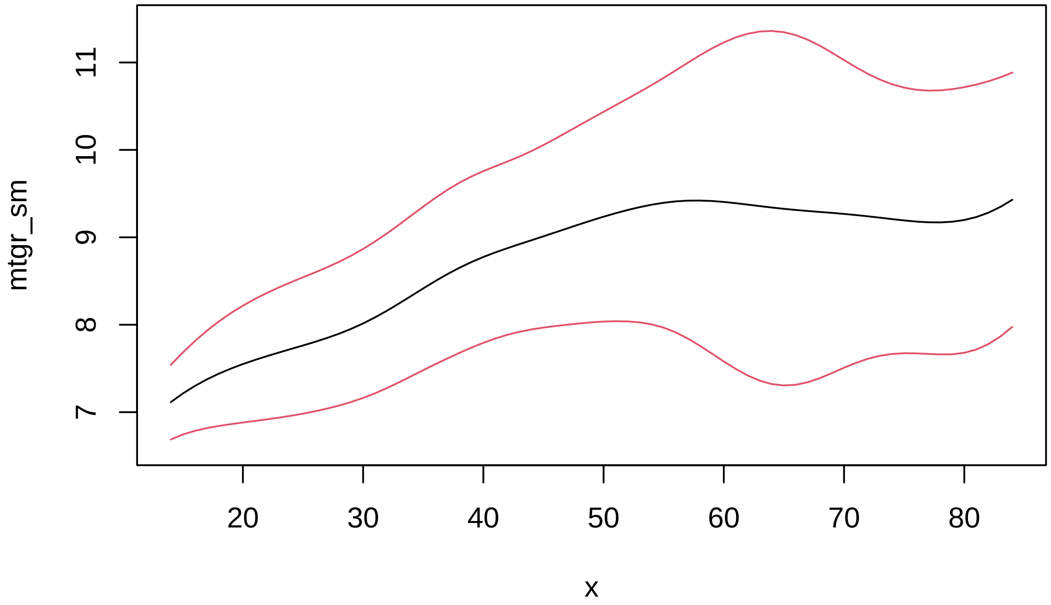
Plots

```
plot(x, (hat_e_sm)^2)
lines(x, sig.sqr_sm, col=2, lwd=3)
```



The result looks almost the similar to the result of the former Local polynomial regression function. The only difference is that it seems that about x=68 the variance decreases more than in the previous result (Local polynomial regression function).

```
y.min_sm <- min(mtgr_sm - 1.96 * sqrt(sig.sqr_sm)) - 0.1
y.max_sm <- max(mtgr_sm + 1.96 * sqrt(sig.sqr_sm)) + 0.1
plot(x,mtgr_sm, type = 'l', ylim=c(y.min_sm, y.max_sm))
lines(x,mtgr_sm + 1.96 * sqrt(sig.sqr_sm), col=2)
lines(x,mtgr_sm - 1.96 * sqrt(sig.sqr_sm), col=2)
```

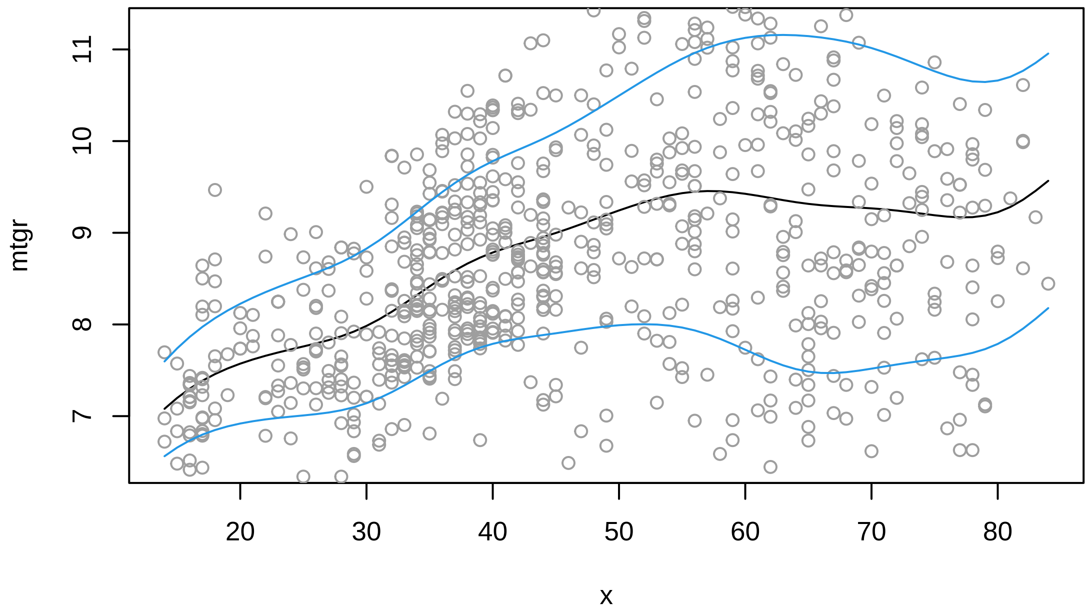


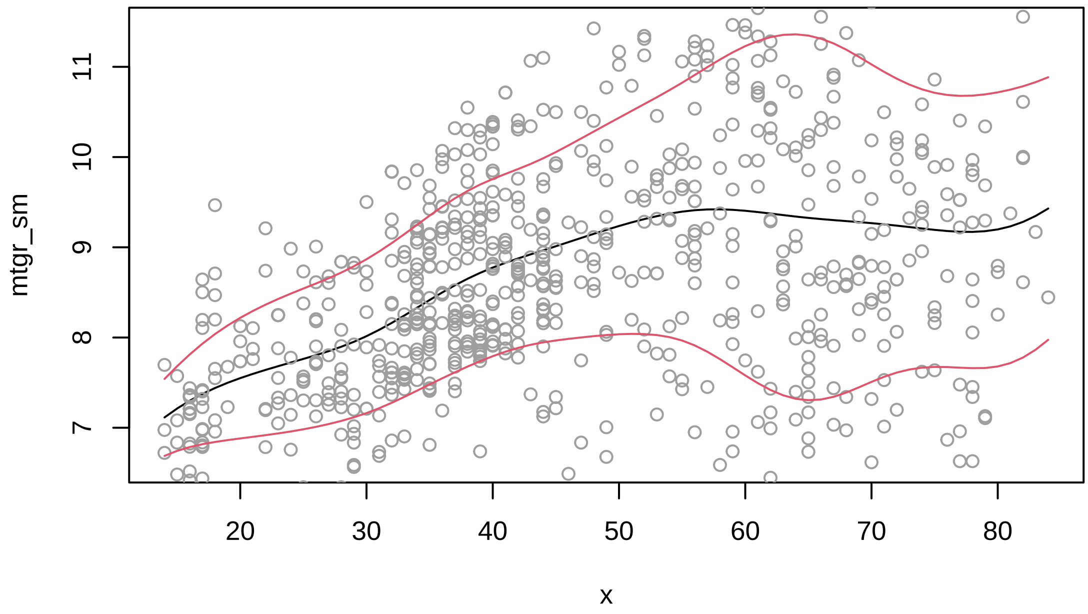
Also this result looks very similar to the result of the former Local polynomial regression function. We can see very well that the variance changes over x as the red boundaries don't have the same distance to the estimate at all times. They represent an uncertainty of our estimation.

Conclusion

We have explored two different approaches to estimate the function $\sigma^2(x)$ through local linear regression, while also practicing how to properly select the bandwidth in each approach.

As we have mentioned earlier, both results are quite similar and we can observe an increase in variance along the values of x.


mtgr
7
8
9
10
11
20
30
40
50
60
70
80
x


mtgr_sm
7
8
9
10
11
20
30
40
50
60
70
80
x