mkern: a Stata routine for estimating a local multivariate kernel regression

Giovanni Cerulli

IRCrES-CNR, Research Institute on Sustainable Economic Growth

Stata meeting 2016
Rome, Italy
November 17-18

Introduction

- **mkern** is a Stata routine for estimating a **local multivariate kernel regression**.
- Non-parametric regression is becoming a popular methodology in many disciplines and research contexts.
- However, a multivariate version of such an approach was not available in Stata yet, neither as built-in, nor as user-written command.
- Stata Corp and users have so far implemented only *bivariate* versions of local kernel regressions (as, in particular, the popular command **lpoly**).

The method

- **mkern** employs a **radial local weighted mean** approach, by using as weighting scheme various *Kernel* functions (at user's choice).
- By default, **mkern** provides also the **optimal bandwidth** by means of a (computational) **cross-validation** approach.
- The user can however provide also his own **choice of the bandwidth**, thus producing estimation for both *over-smoothing* and *under-smoothing* cases.
- Finally, as option, **mkern** offers a graphical plot of the row data against the predicted values, in order to assess also visually the goodness-of-fit of the provided estimation.

The intuition behind mkern

Consider the following regression of Y on the exogenous covariates $[X_1, ..., X_M]$:

$$Y_i = m(X_{1i}, ..., X_{Mi}) + e_i \tag{1}$$

where $i = 1, ..., N$, e is an error term that embodies all heterogeneity across individuals. We thus have that:

$$E(Y_i | X_{1i}, ..., X_{Mi}) = m(X_{1i}, ..., X_{Mi}) \tag{2}$$

Here, the problem is that $m(\cdot)$ is **unknown**, thus we cannot use standard parametric estimation methods, such as OLS, ML, and GMM.

We have to rely on **non-parametric approches**, which generally devided into *global* and *local* methods. **mkern** uses a *local* Kernel approach.

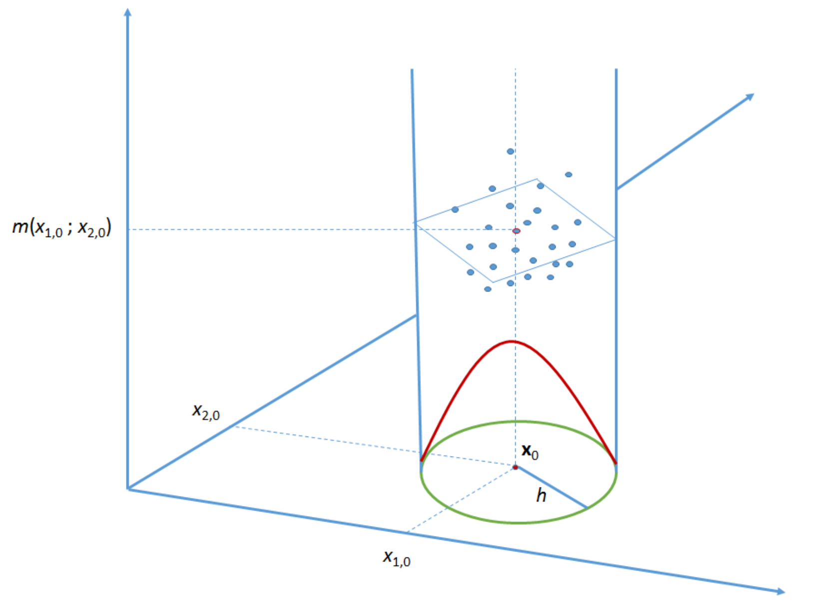
The intuition behind mkern

The *local* Kernel estimation adopts a **pointwise imputation** of the **unknown** function $m(\cdot)$, using local weighted mean (or local weighted polynomials). In the case of mean imputation at point $\mathbf{X}_0 = [X_{10}, ..., X_{M0}]$:

$$\hat{m}(X_{1,0}, ..., X_{M,0}) = \sum_{i=1}^{N} w_{i,0,h} \cdot Y_i \tag{3}$$

where: $w_{i,0,h}$ is a weight depending on units i and unit 0 characteristics with $\sum_{i=1}^{N} w_{i,0,h} = 1$, and on the bandwidth h. By using a specific *kernel function* (i.e., *kernel weights*) $K(\cdot)$ we have:

$$\hat{m}(X_{1,0}, ..., X_{M,0}) = \sum_{i=1}^{N} \left[\frac{K\left(\frac{\mathbf{X}_i - \mathbf{X}_0}{h}\right)}{\frac{1}{N_h} \sum_{i=1}^{N} K\left(\frac{\mathbf{X}_i - \mathbf{X}_0}{h}\right)} \right] \cdot Y_i \tag{4}$$



$m(x_{1,0}\,;\,x_{2,0})$
$x_{2,0}$
$\mathbf{x}_0$
h
$x_{1,0}$

Optimal bandwidth using Cross-Validation (CV)

In order to determine the **optimal bandwidth**, the routine **mkern** uses a CV-approach. This is a *computational* method aimed at *minimizing* the following **objective funcion** over the bandwidth h:

$$\mathrm{CV}(h) = \sum_{i=1}^{N} [Y_i - \hat{m}_{-i}(\mathbf{x}_i)]^2 \tag{5}$$

that is:

$$h^* = \mathrm{argmin}_h \sum_{i=1}^{N} [Y_i - \hat{m}_{-i}(\mathbf{x}_i)]^2 \tag{6}$$

where $\hat{m}_{-i}(\mathbf{x}_i)$ is the **leave-one-out** estimate of $m(\mathbf{x}_i)$.

Stata implementation using `mkern`

The syntax of mkern (1)

```
mkern dep_var indep_vars [if] [in] [pweights]
      [, h(bandwidth) cvfile(file_name) graph]
      k(kernel_function)

-------------------------------------------------
  kernel_function
-------------------------------------------------
  epan               Epanechnikov weighting scheme
  normal             Normal weighting scheme
  biweight           Biweight (or Quartic) scheme
  uniform            Uniform weighting scheme
  triangular         Triangular weighting scheme
  tricube            Tricube weighting scheme
-------------------------------------------------
```

The fitted values of **mkern** are stored in the generated variable **"_kern"**

The syntax of mkern (2)

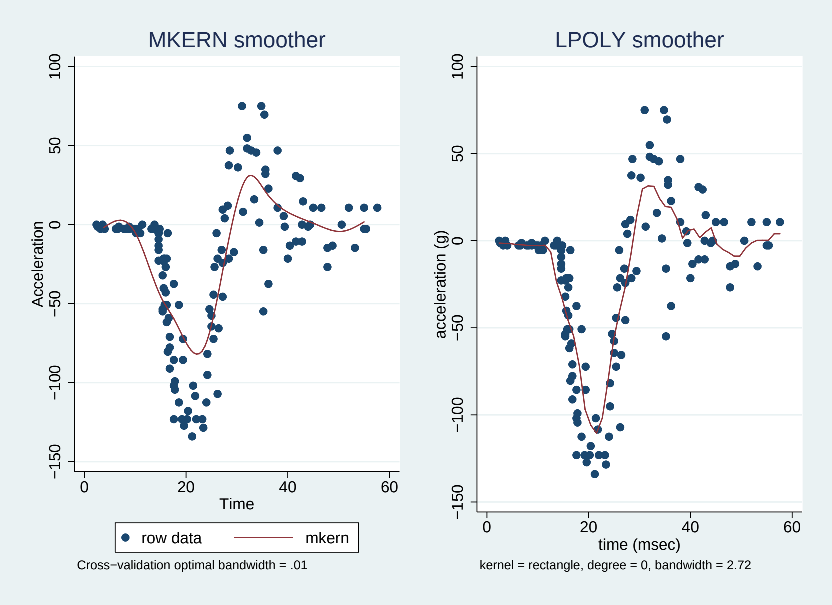
```
mkern varlist [if] [in] [pweights]
      [, h(bandwidth) cvfile(file_name) graph]
      k(kernel_function)

-----------------------------------------------------
h(bandwidth):         provides the user's declared
                      bandwidth. By default -
                      mkern provides the CV bandwidth.
-----------------------------------------------------
cvfile(file_name): is the name of the file where the
                      Cross-Validation results are stored.
-----------------------------------------------------
graph:                provides a joint plot of the outcome
                      and the model's predictions
-----------------------------------------------------
```

Application 1 (1)

We consider first a **univariate** case, using the dataset "motorcycle" reporting data on ACCELERATION against TIME for $N = 133$ motorcycles, and compare **mkern** with the Stata built-in command **lpoly**.

```
set more off
webuse motorcycle , clear
global y accel  //  dependent variable
global x time   //  observed covariate

mkern $y $x , k(uniform) cvfile(cv_res)

* Result using "mkern"
global h=round(e(opt_bandw),0.01)
tw (scatter $y $x ) (mspline _kern $x) , ///
legend(order(1 "row data" 2 "mkern")) xtitle(Time) ytitle(Acceleration) ///
note(Cross-validation optimal bandwidth = $h) name(mk, replace) title(MKERN smoother)

* Result using "lpoly"
lpoly  $y $x , kernel(rectangle) degree(0) name(lpol,replace) title(LPOLY smoother)

* Combine "mkern" and "lpoly"
graph combine mk lpol
```

Application 1 (2)



Comments

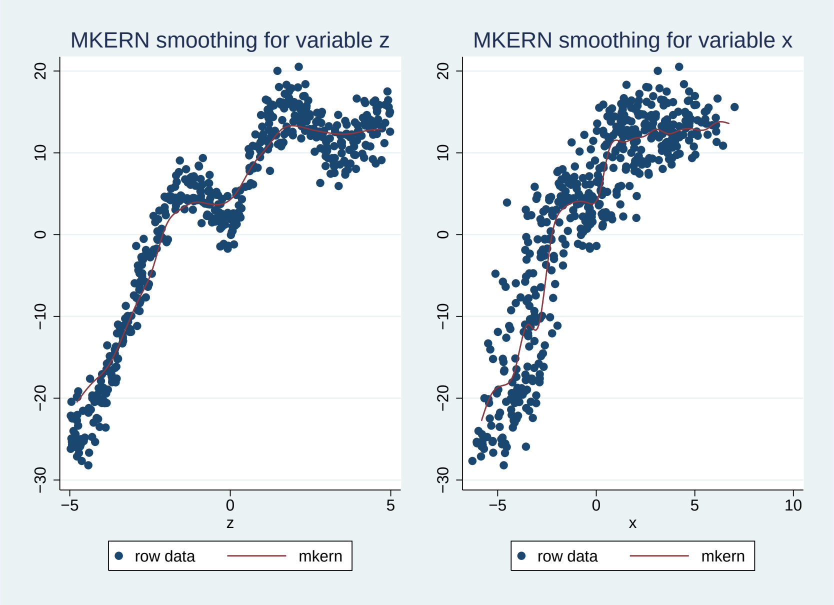
- **mkern** seems to perform rather well, especially if one considers that **lpoly** uses an *analytical formula* to calculate the *optimal bandwidth*, while **mkern** uses a computational one (Cross-Validation).
- The **Cross-Validation** approach has been proved to converge to the right optimal bandwidth at a very slow rate of $O(N^{-0.1})$.
- This means that when N is small, the CV approach should be instable. Fortunately, we saw that such instability is not too strong in our example, although we rely on just 133 observations.

Application 2 (1)

Now we consider a **multivariate** case using a simulated dataset produced by an “odd” data generating process.

```
*** EXAMPLE 2 ***************************************************************
clear
set more off
set seed 101
set obs 500

* Generate an odd function
drawnorm e
generate z=(uniform()-0.5)*10
generate x=z+invnorm(uniform())
generate x2=z+invnorm(uniform())
generate x3=z+invnorm(uniform())
generate y=x+x2+x3+e
replace y=(10*sin(abs(z)))*(z<_pi)+y

* Use mkern for estimating it, with the "graph" option
mkern y x z , k(normal) cvfile(cv_res) graph

tw (scatter y z ) (mspline _kern z) , ///
legend(order(1 "row data" 2 "mkern")) ///
title(MKERN smoothing for variable z) name(gz)

tw (scatter y x ) (mspline _kern x) , ///
legend(order(1 "row data" 2 "mkern")) ///
title(MKERN smoothing for variable x) name(gx)

graph combine gz gx
```

Application 2 (2)



Conclusions

- **mkern** seems to behave rather well both in univariate and multivariate regressions.
- A next step is to provide also *local linear* smoothing (rather straightforward to do), and a correction for the (typical) asymptotic bias of local kernel regressions.
- Unfortunately, Cross-Validation has slow rate of convergence by increasing N. This provides imprecision of bandwidth estimates.
- Cross-Validation is computationally intensive, thereby it takes a lot of time to get results when N is large.