Adaptive kernel density estimation in Stata

Philippe Van Kerm
CEPS/INSTEAD, G.-D. Luxembourg

9th London Stata User Group Meeting, 19-20 May 2003

Kernel density function estimation

- Official command: `kdensity`

$$\hat{f}_f(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K\left(\frac{x-x_i}{h}\right) \tag{1}$$

  - 'Point mass' of sample data diffused around x_i's, and averaged at x

Kernel density function estimation

- Official command: `kdensity`

$$\hat{f}_f(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K\left(\frac{x - x_i}{h}\right) \tag{1}$$

  - ‘Point mass’ of sample data diffused around x_i’s, and averaged at x
  - Fixed/constant/global bandwidth h: ‘degree of diffusion’ constant for all x_i’s

Adaptive kernel density function estimation

- `akdensity`

$$\hat{f}_v(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i}K\left(\frac{x-x_i}{h_i}\right) \tag{2}$$

  - Different bandwidths for different x_i's

Adaptive kernel density function estimation

- `akdensity`

$$\hat{f}_v(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i}K\left(\frac{x - x_i}{h_i}\right) \tag{2}$$

  - Different bandwidths for different x_i's
  - Degree of diffusion varies inversely with $f(x_i)$

Adaptive kernel density function estimation

- `akdensity`

$$\hat{f}_v(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h_i} K\left(\frac{x - x_i}{h_i}\right) \tag{2}$$

  - Different bandwidths for different x_i's
  - Degree of diffusion varies inversely with $f(x_i)$
  - Greater precision where data are abundant and ...
  - ... greater smoothness where data are sparse

Adaptive kernel density function estimation

- `akdensity`

$$\hat{f}_v(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i}K\left(\frac{x - x_i}{h_i}\right) \tag{2}$$

  - Different bandwidths for different x_i's
  - Degree of diffusion varies inversely with $f(x_i)$
  - Greater precision where data are abundant and ...
  - ... greater smoothness where data are sparse

- Adaptive two-stage estimator (Abramson 1982):

  - $h_i = h \times \lambda_i$;

Adaptive kernel density function estimation

- `akdensity`

$$\hat{f}_v(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i}K\left(\frac{x-x_i}{h_i}\right) \tag{2}$$

  - Different bandwidths for different x_i's
  - Degree of diffusion varies inversely with $f(x_i)$
  - Greater precision where data are abundant and ...
  - ... greater smoothness where data are sparse

- Adaptive two-stage estimator (Abramson 1982):
  - $h_i = h \times \lambda_i;\ \ \lambda_i = \left(G/\tilde{f}(x_i)\right)^{0.5}$

Adaptive kernel density function estimation

- `akdensity`

$$\hat{f}_v(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i}K\left(\frac{x - x_i}{h_i}\right) \tag{2}$$

  - Different bandwidths for different x_i's
  - Degree of diffusion varies inversely with $f(x_i)$
  - Greater precision where data are abundant and ...
  - ... greater smoothness where data are sparse

- Adaptive two-stage estimator (Abramson 1982):
  - $h_i = h \times \lambda_i$; $\lambda_i = \left(G/\tilde{f}(x_i)\right)^{0.5}$
  - First step: compute pilot estimate (fixed bandwidth h) to generate λ_i

Adaptive kernel density function estimation

- `akdensity`

$$\hat{f}_v(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i}K\left(\frac{x-x_i}{h_i}\right) \tag{2}$$

  - Different bandwidths for different x_i's
  - Degree of diffusion varies inversely with $f(x_i)$
  - Greater precision where data are abundant and ...
  - ... greater smoothness where data are sparse

- Adaptive two-stage estimator (Abramson 1982):
  - $h_i = h \times \lambda_i;\ \ \lambda_i = \left(G/\tilde{f}(x_i)\right)^{0.5}$
  - First step: compute pilot estimate (fixed bandwidth h) to generate λ_i
  - Second step: compute density estimate with h_i local bandwidths

Variability bands as a bonus

- `akdensity` estimates variability bands:
  - $\hat{f}_v(x) \pm b \times SE(x)$

Variability bands as a bonus

- `akdensity` estimates variability bands:
  - $\hat{f}_v(x) \pm b \times SE(x)$
  - Pointwise variability bands

Variability bands as a bonus

- `akdensity` estimates variability bands:
  - $\hat{f}_v(x) \pm b \times SE(x)$
  - Pointwise variability bands
  - Not confidence intervals (accounts for sample variability but not bias!)

Variability bands as a bonus

- `akdensity` estimates variability bands:
  - $\hat{f}_v(x) \pm b \times SE(x)$
  - Pointwise variability bands
  - Not confidence intervals (accounts for sample variability but not bias!)
- Also for fixed bandwidth kernel estimates!

Syntax extract

- 'High-level' command (mimicks `kdensity`):

  `akdensity` *varname* `...` [`, noadaptive width(#)`

  [`epan | gauss`] `stdbands(#) ...`]

Syntax extract

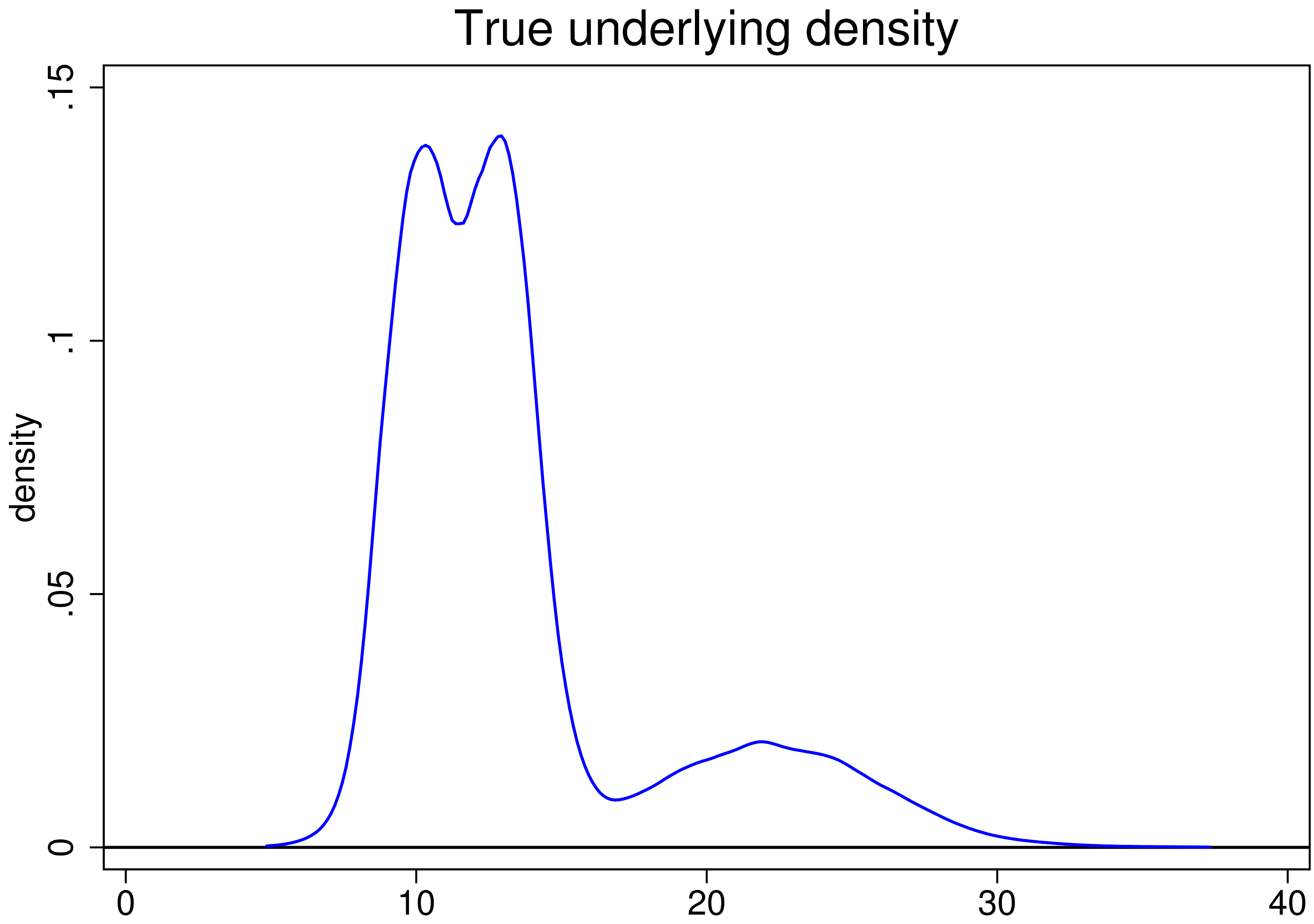
- 'High-level' command (mimicks `kdensity`):

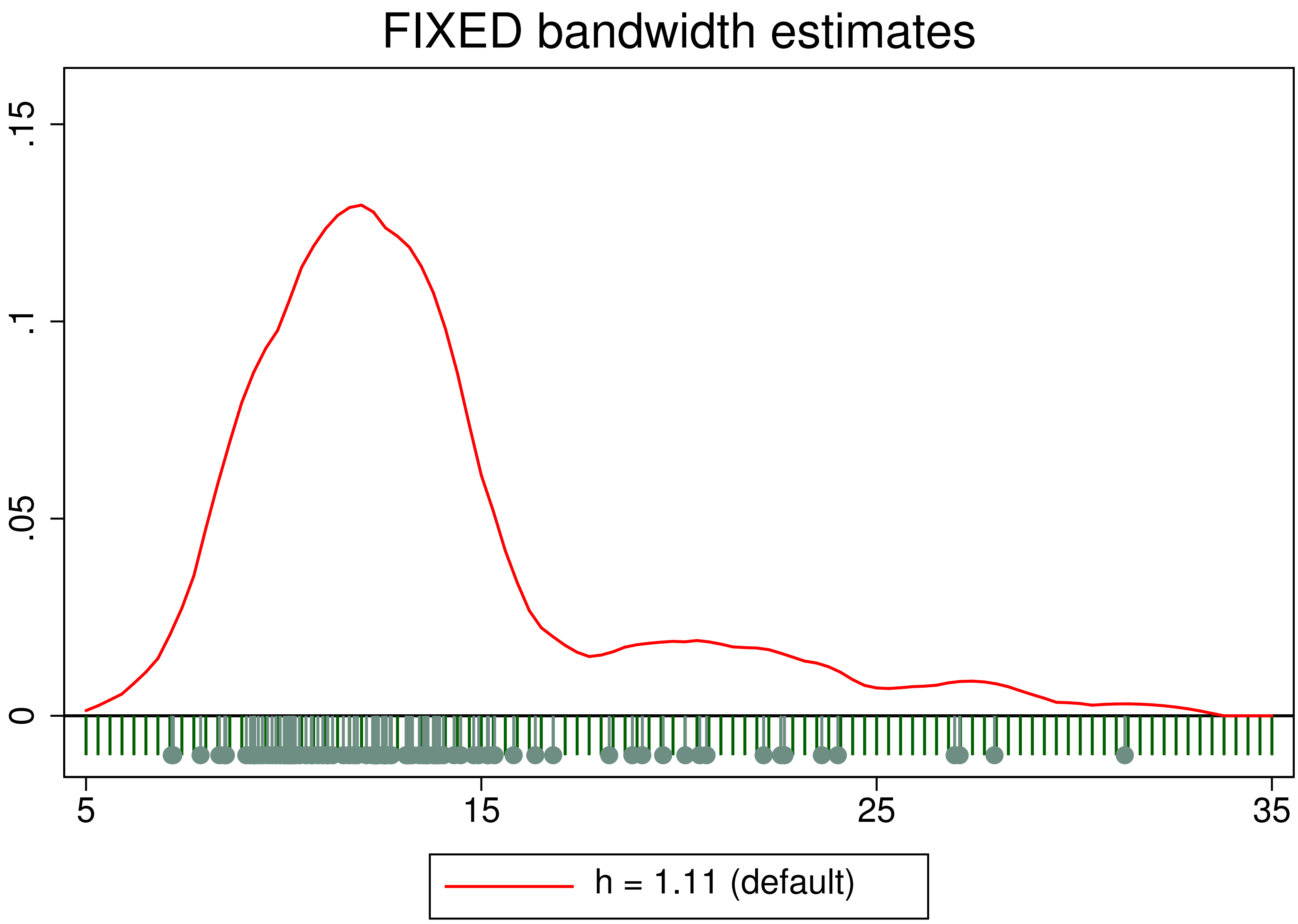
  `akdensity` *varname* `...` [`, noadaptive width(#)` [`epan | gauss`] `stdbands(#) ...`]

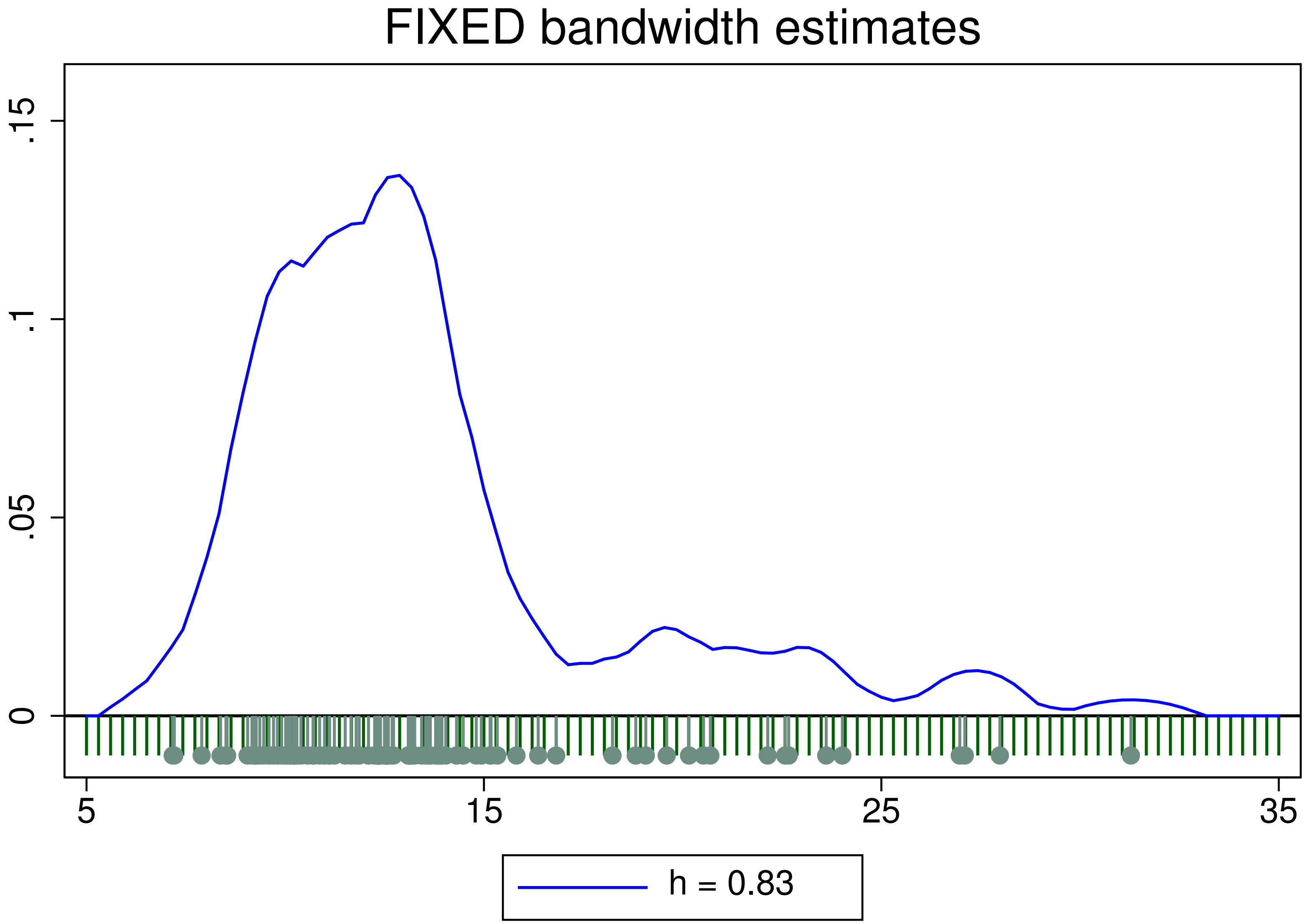
- 'Low-level' command (rarely used, but full control):

  `akdensity0` *varname* `... , width(#` | *varname*`) ...` [`lambda(`*string*`) ...`]

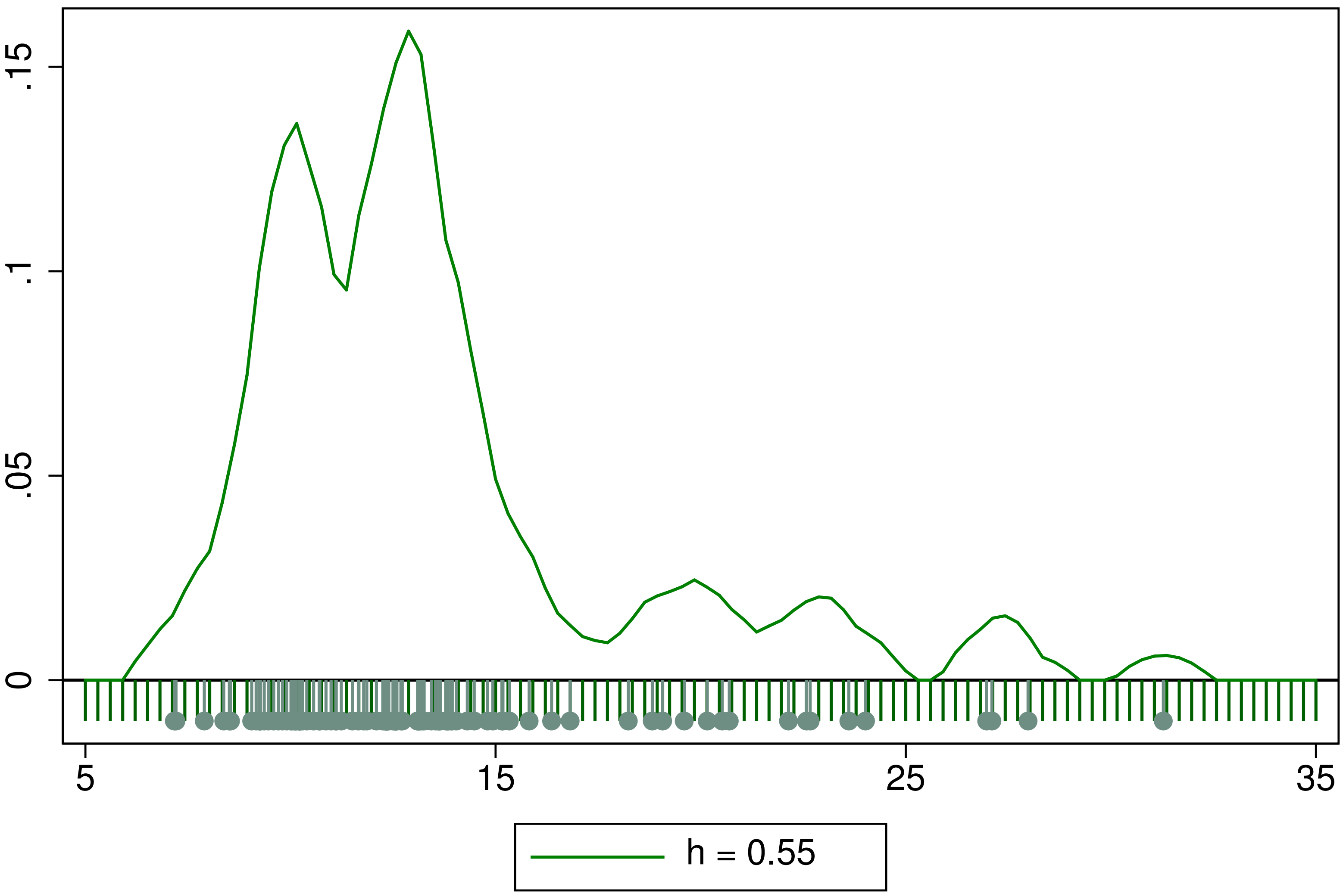
A simulated example

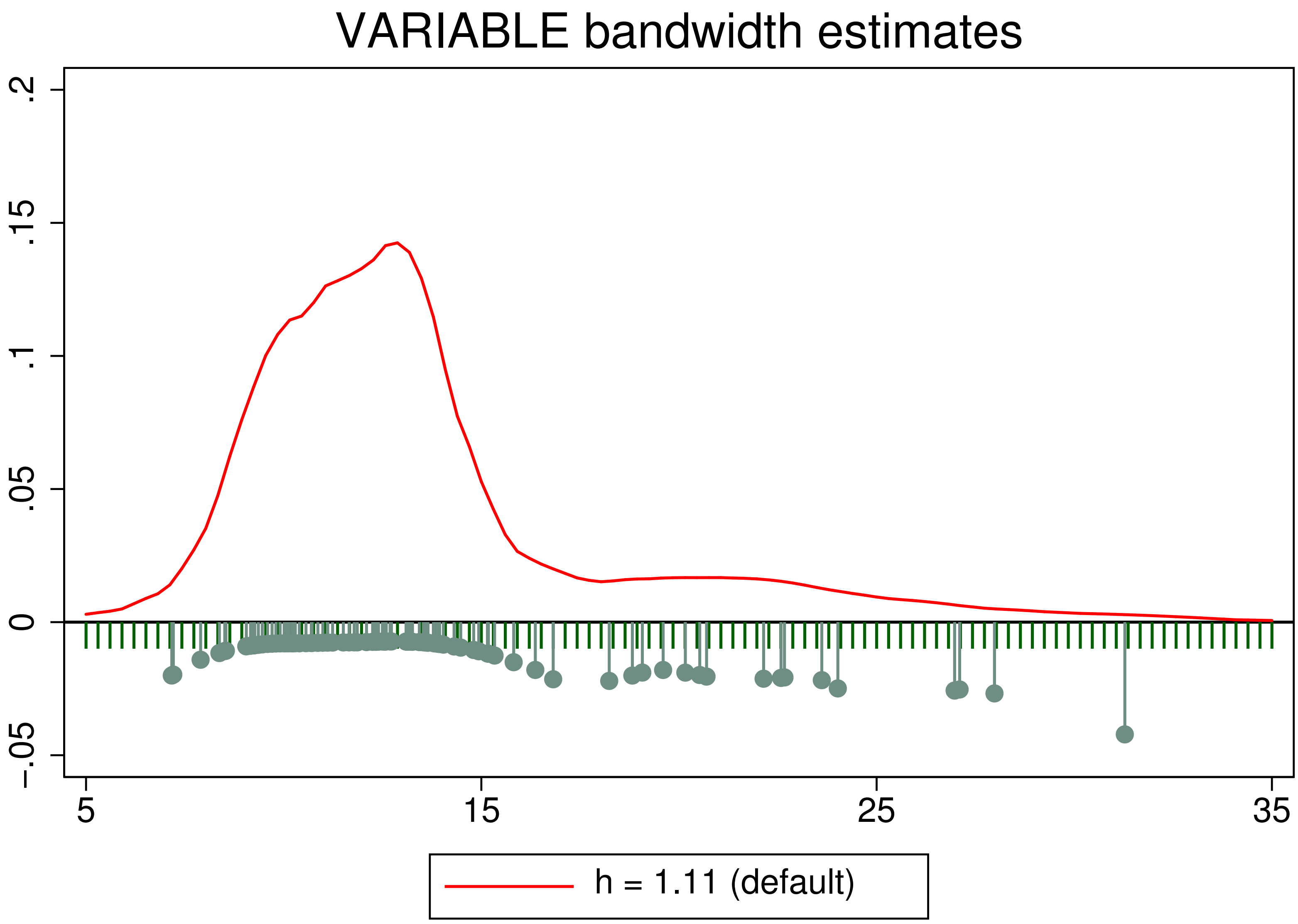


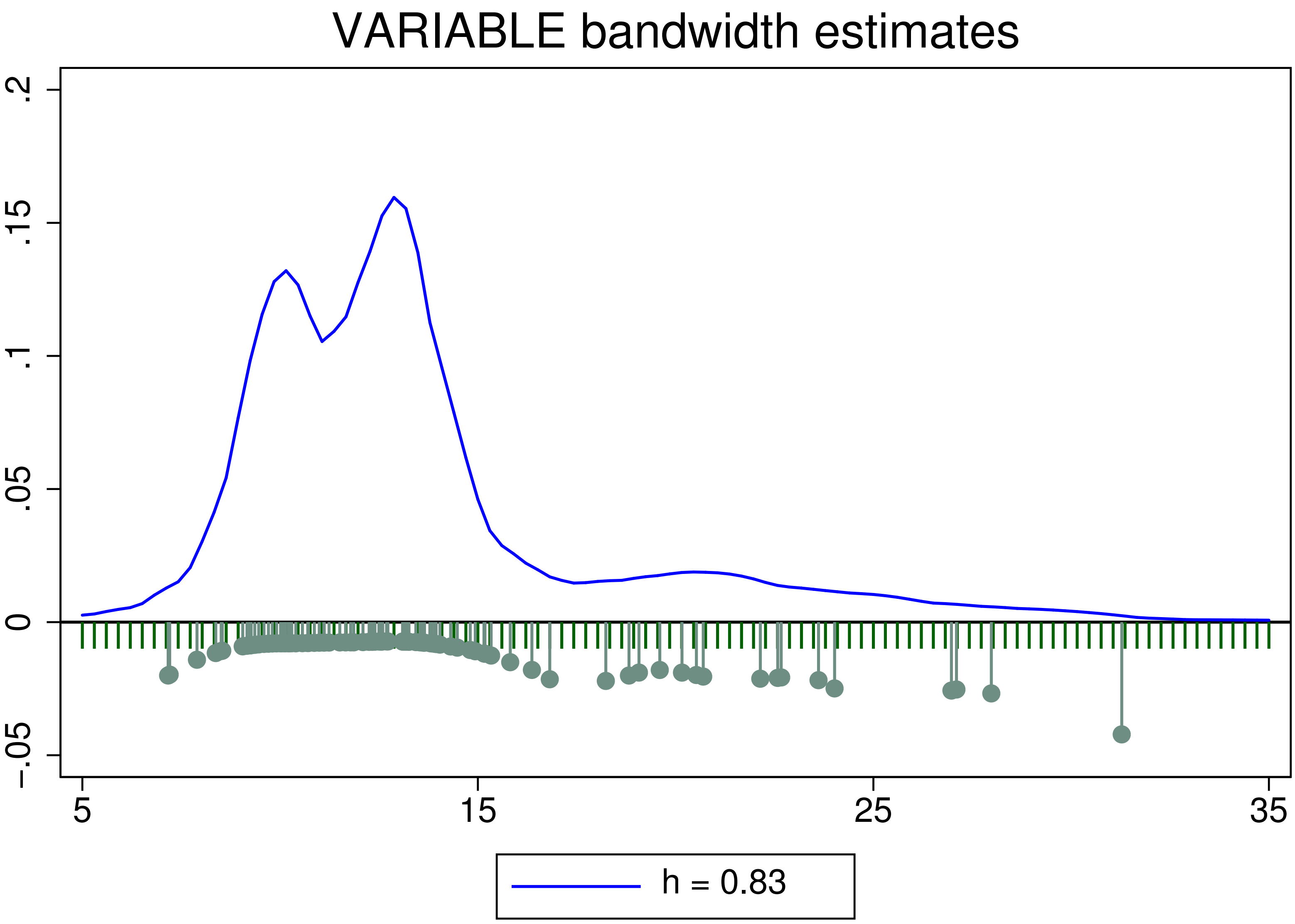

FIXED bandwidth estimates
0
.05
.1
.15
5
15
25
35
h = 1.11 (default)


FIXED bandwidth estimates
0
.05
.1
.15
5
15
25
35
h = 0.83

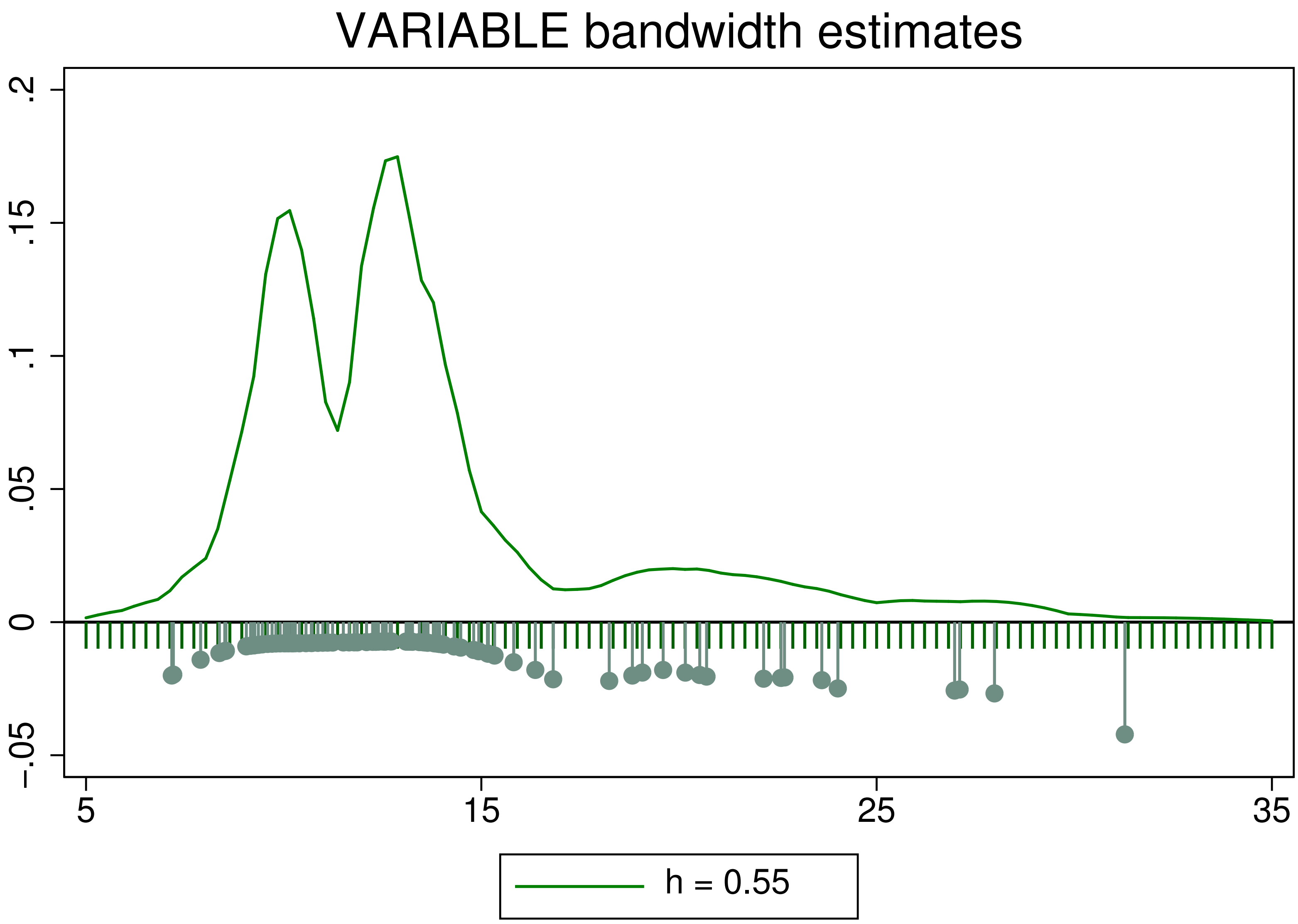
FIXED bandwidth estimates

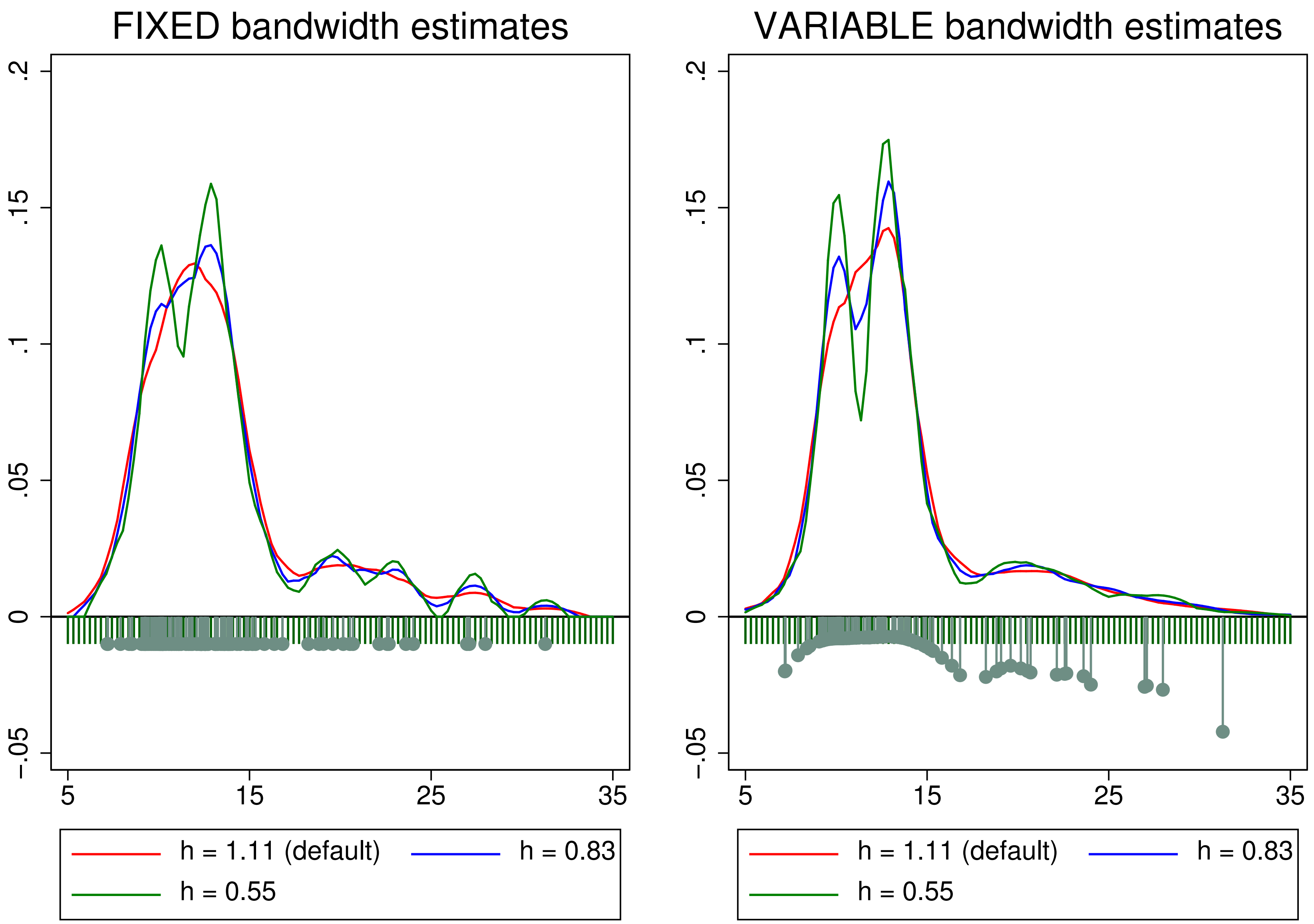


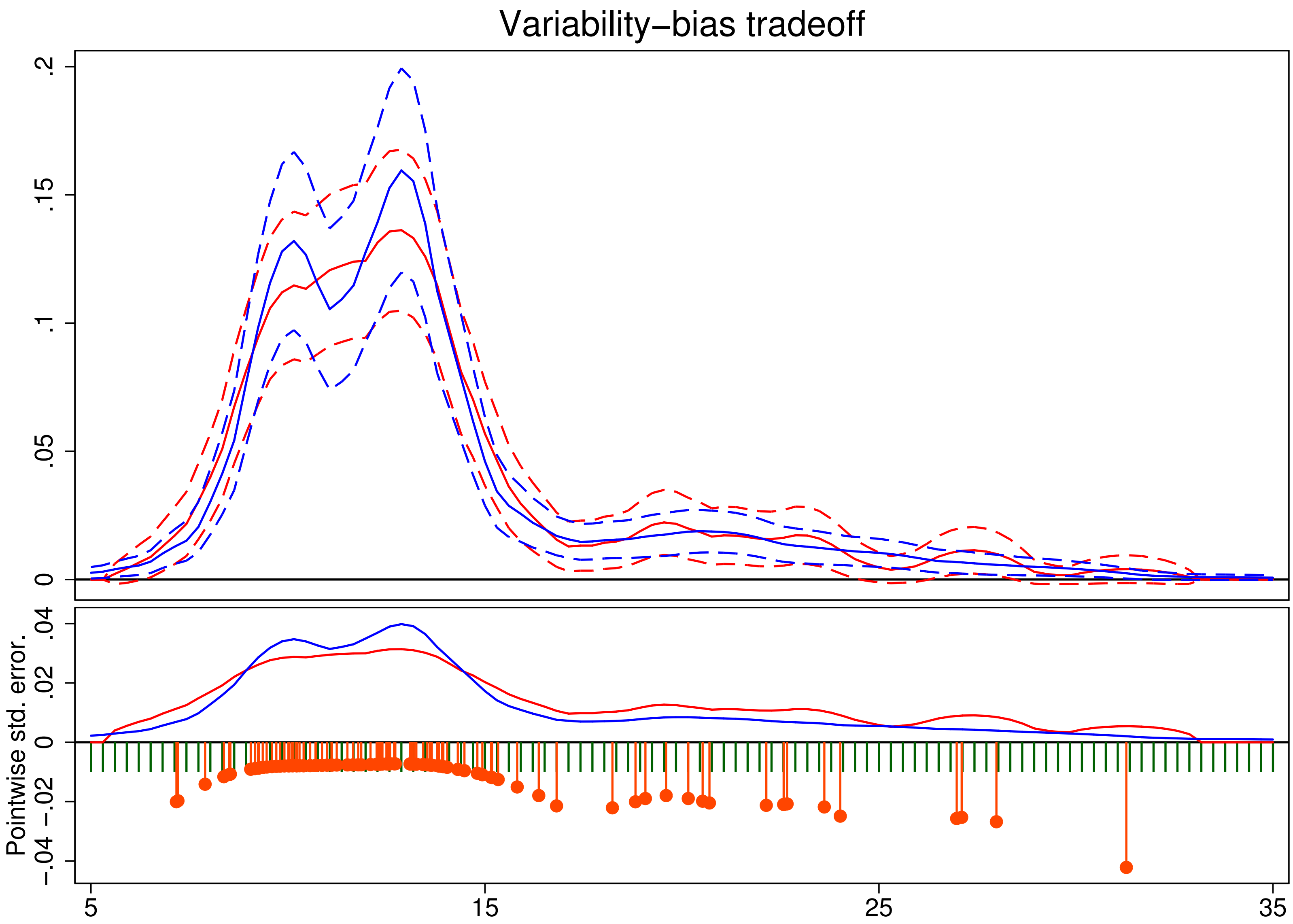

VARIABLE bandwidth estimates
.2
.15
.1
.05
0
-.05
5
15
25
35
h = 1.11 (default)



VARIABLE bandwidth estimates
.2
.15
.1
.05
0
-.05
5
15
25
35
h = 0.83

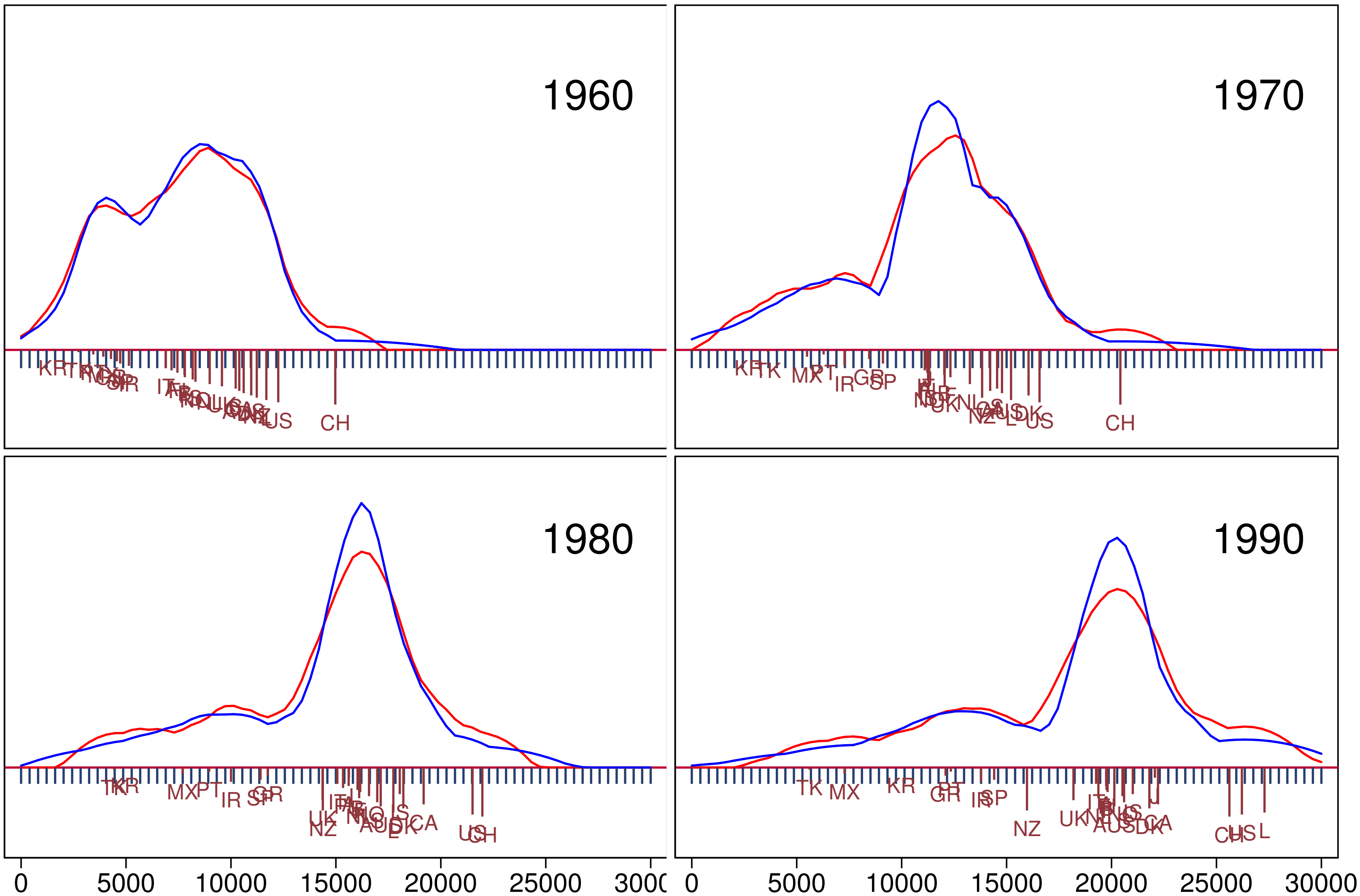

VARIABLE bandwidth estimates
.2
.15
.1
.05
0
-.05
5
15
25
35
h = 0.55

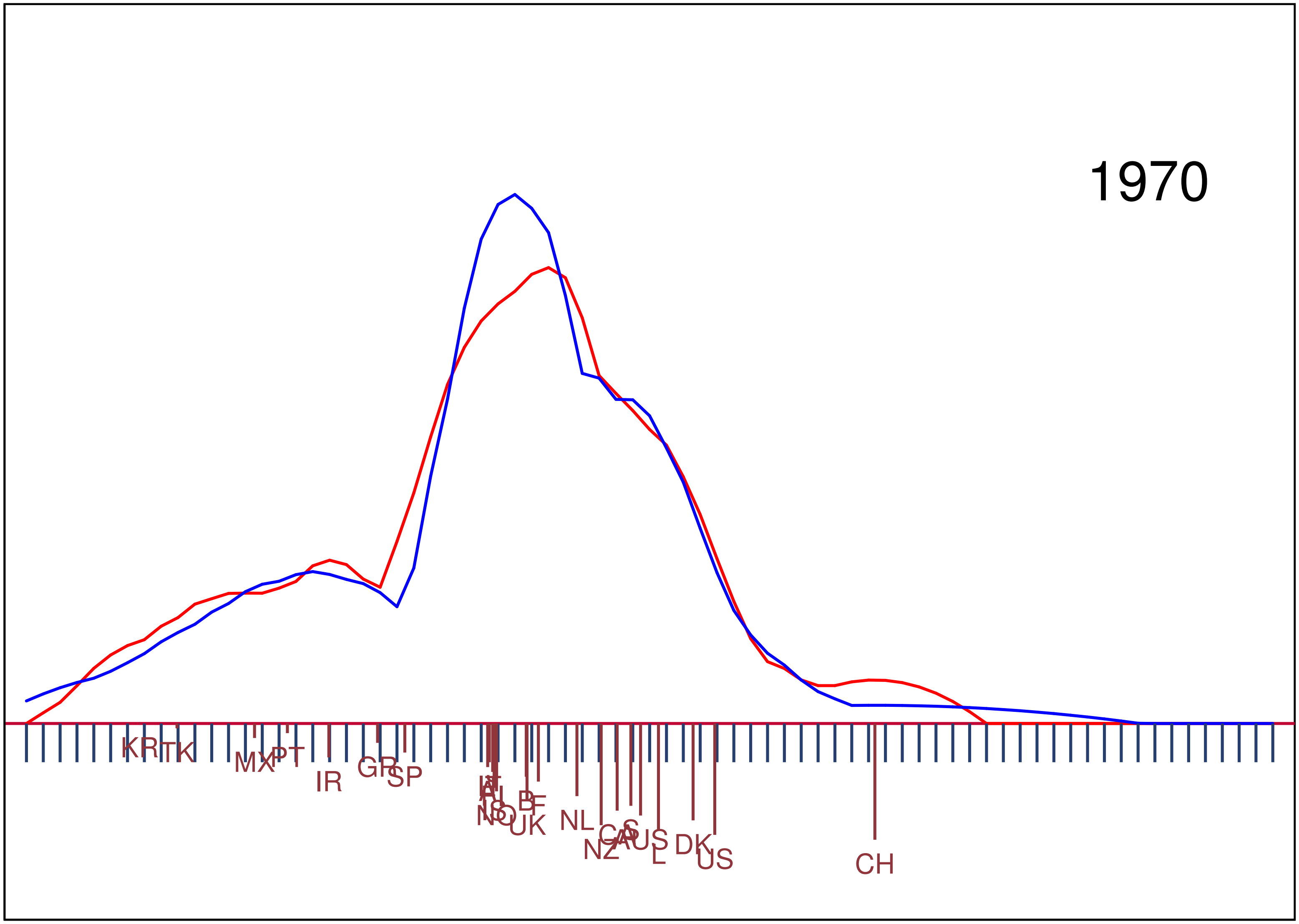

FIXED bandwidth estimates
VARIABLE bandwidth estimates
−.05
0
.05
.1
.15
.2
5
15
25
35
h = 1.11 (default)
h = 0.83
h = 0.55

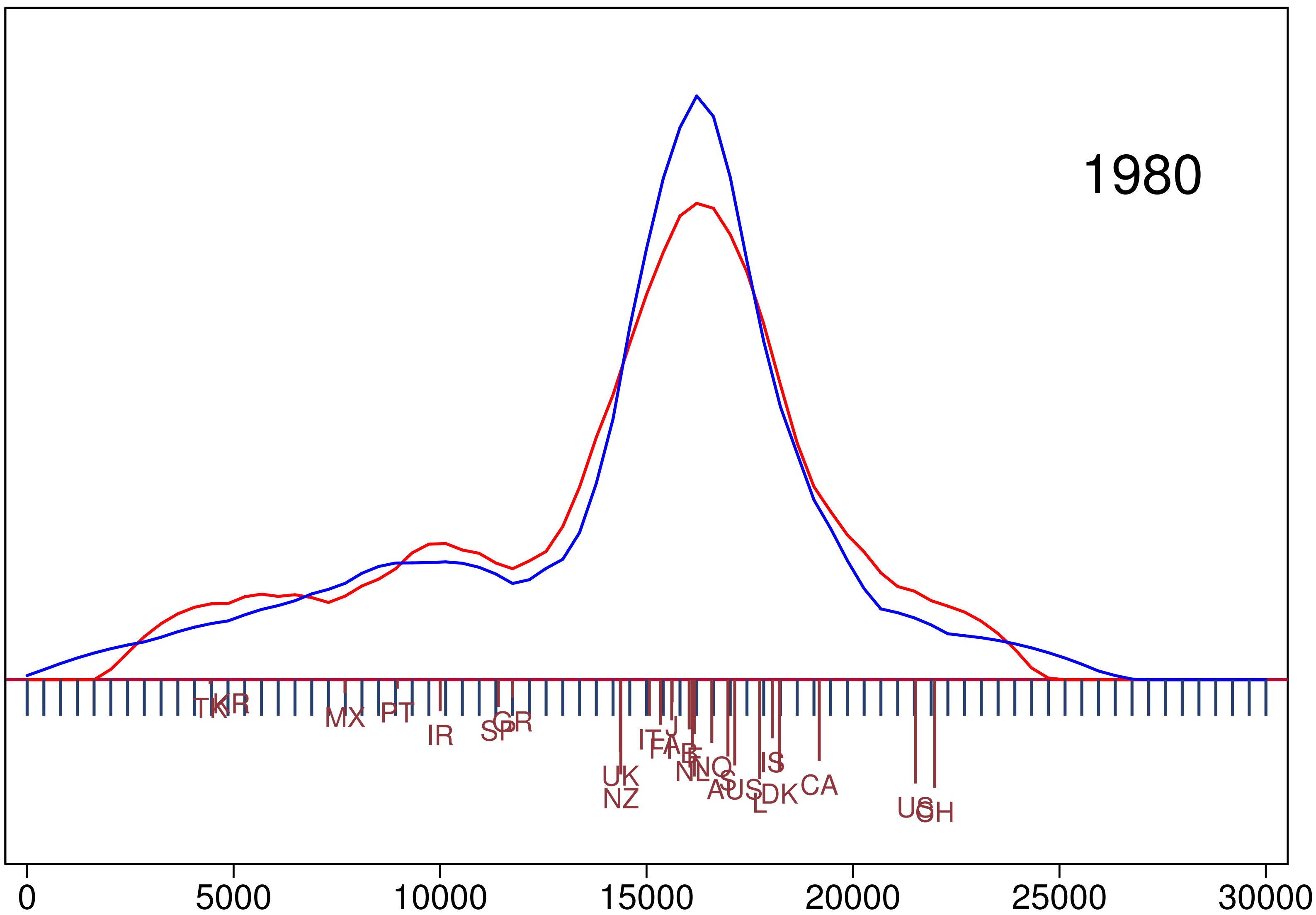

Variability−bias tradeoff
.2
.15
.1
.05
0
Pointwise std. error.
.04
.02
0
−.02
−.04
5
15
25
35

A real example

Real GDP per capita – FIXED and VARIABLE bandwidths




1970
KR
TK
MX
PT
IR
GR
SP
NZ
CA
AUS
L
DK
US
NL
UK
CH


1980
TK
KR
MX
PT
IR
SP
GR
IT
FA
NL
UK
NZ
NO
AUS
DK
IS
CA
US
CH
0
5000
10000
15000
20000
25000
30000

Final remarks

- Fast implementation for large datasets (use of linear interpolation for the pilot estimate)

Final remarks

- Fast implementation for large datasets (use of linear interpolation for the pilot estimate)
- Works under both version 7 and version 8:

Final remarks

- Fast implementation for large datasets (use of linear interpolation for the pilot estimate)
- Works under both version 7 and version 8:
  - use new graphics engine if called from Stata 8, and former engine if called from Stata 7

Final remarks

- Fast implementation for large datasets (use of linear interpolation for the pilot estimate)
- Works under both version 7 and version 8:
  - use new graphics engine if called from Stata 8, and former engine if called from Stata 7
  - graphics options therefore vary (in Stata 8, option `plot(...)` especially useful!)

Final remarks

- Fast implementation for large datasets (use of linear interpolation for the pilot estimate)
- Works under both version 7 and version 8:
  - use new graphics engine if called from Stata 8, and former engine if called from Stata 7
  - graphics options therefore vary (in Stata 8, option `plot(...)` especially useful!)
  - coded using `if _caller()`<8 statements

Final remarks

- Fast implementation for large datasets (use of linear interpolation for the pilot estimate)
- Works under both version 7 and version 8:
  - use new graphics engine if called from Stata 8, and former engine if called from Stata 7
  - graphics options therefore vary (in Stata 8, option `plot(...)` especially useful!)
  - coded using `if _caller()<8` statements
- Available in next *Stata Journal* issue (vol.3(2))