Economics 675: Applied Microeconometrics
Fall 2018 - Assignment 2

Paul R. Organ

October 11, 2018

Contents

1 Question 1: Kernel Density Estimation

1.1 Expectation and Variance

First, note that $\hat{f}^{(s)}(x) = \frac{1}{nh_n^{s+1}} \sum_{i=1}^{n} (-1)^s K^{(s)}\left(\frac{x_i - x}{h_n}\right)$.

Then we have:

$$
\begin{aligned}
\mathbb{E}[\hat{f}^{(s)}(x)] &= \mathbb{E}\Big[\frac{1}{nh_n^{s+1}}\sum_{i=1}^{n}(-1)^s K^{(s)}\Big(\frac{x_i - x}{h_n}\Big)\Big] \\
&= (-1)^s \frac{1}{h_n^{s+1}}\int K^{(s)}\Big(\frac{z-x}{h_n}\Big)f(z)dz \\
&= (-1)^s \frac{1}{h_n^{s+1}}\int K^{(s)}(u)f(x+uh_n)h_n du \\
&= (-1)^s \frac{1}{h_n^{s}}\int K^{(s)}(u)f(x+uh_n) du \\
&= (-1)^{s-1} \frac{1}{h_n^{s-1}}\int K^{(s-1)}(u)f^{(1)}(x+uh_n) du \\
&= \text{... (keep rolling back to } s) \\
&= (-1)^{s-s} \frac{1}{h_n^{s-s}}\int K^{(s-s)}(u)f^{(s)}(x+uh_n) du \\
&= \int K(u)f^{(s)}(x+uh_n) du \\
&= \int K(u)\Big[f^{(s)}(x) + ... + \frac{u^P h_n^P}{P!}f^{(s+P)}(x) + \frac{u^{P+1}h_n^{P+1}}{(P+1)!}f^{(s+P+1)}(\tilde{x})\Big]du \\
&= f^{(s)}(x) + h_n^P \mu_P(K)\frac{f^{(P+s)}(x)}{P!} + O(h_n^{P+1})
\end{aligned}
$$

In the third line, using change of variables with $u = \frac{z-x}{h_n}$, so $du = dz * 1/h_n$; in the fifth line using integration-by-parts; in the second-to-last line, using a Taylor approximation; and in the final line using the definition of $\mu_\ell(K)$.

Then turning to the variance, we have:

$$
\begin{aligned}
\mathbb{V}[\hat{f}^{(s)}(x)] &= \mathbb{V}\Big[\frac{1}{nh_n^{s+1}}\sum_{i=1}^{n}(-1)^s K^{(s)}\Big(\frac{x_i - x}{h_n}\Big)\Big] \\
&= \frac{n}{n^2 h_n^{2(s+1)}}\mathbb{V}\Big[K^{(s)}\Big(\frac{x_i - x}{h_n}\Big)\Big] \\
&= \frac{1}{nh_n^{2(s+1)}}\bigg(\mathbb{E}\Big[K^{(s)}\big(\frac{x_i - x}{h_n}\big)^2\Big] - \mathbb{E}\Big[K^{(s)}\big(\frac{x_i - x}{h_n}\big)\Big]^2\bigg) \\
&= \frac{1}{nh_n^{2(s+1)}}\bigg(\mathbb{E}\Big[K^{(s)}\big(\frac{x_i - x}{h_n}\big)^2\Big]\bigg)\big(1+o(1)\big) \\
&= \frac{1}{nh_n^{2(s+1)}}\bigg(\int K^{(s)}\Big(\frac{z-x}{h_n}\Big)^2 f(z)dz\bigg)\big(1+o(1)\big) \\
&= \frac{1}{nh_n^{2(s+1)}}\bigg(\int K^{(s)}(u)^2 f(x+uh_n)h_n du\bigg)\big(1+o(1)\big) \\
&= \frac{1}{nh_n^{2s+1}}\bigg(\int K^{(s)}(u)^2 f(x+uh_n) du\bigg)\big(1+o(1)\big) \\
&= \frac{1}{nh_n^{2s+1}} \cdot v_s(K) \cdot f(x)\big(1+o(1)\big) \\
&= \frac{1}{nh_n^{2s+1}} \cdot v_s(K) \cdot f(x) + o\Big(\frac{1}{nh_n^{2s+1}}\Big)
\end{aligned}
$$

Using the same change-of-variables as above.

1.2 Mean-Squared Error

Start with the given definition of AIMSE$[h]$, then plug in the definition for $\mu_\ell(K)$ and $v_\ell(K)$:

$$
\begin{aligned}
\text{AIMSE}[h] &= \int \left[\left(h_n^P \cdot \mu_P(K) \cdot \frac{f^{(P+s)}(x)}{P!} \right)^2 + \frac{1}{nh_n^{1+2s}} \cdot v_s(K) \cdot f(x) \right] dx \\
&= h_n^{2P} \cdot \mu_P(K)^2 \cdot \int \left(\frac{f^{(P+s)}(x)}{P!} \right)^2 dx + \frac{1}{nh_n^{1+2s}} \cdot v_s(K) \\
&= h_n^{2P} \cdot \mu_P(K)^2 \cdot \frac{1}{(P!)^2} \cdot v_{s+P}(f) + \frac{1}{nh_n^{1+2s}} \cdot v_s(K)
\end{aligned}
$$

From here, we can take the derivative wrt h_n to find the optimal bandwidth:

$$
\frac{\partial}{\partial h_n} \text{AIMSE}[h] = 2P \cdot h_n^{2P-1} \cdot \mu_P(K)^2 \cdot \frac{1}{(P!)^2} \cdot v_{s+P}(f) + \frac{-1-2s}{nh_n^{2+2s}} \cdot v_s(K) = 0
$$

Combining terms and solving for h_n yields the optimal bandwidth as desired:

$$
h_n^* = \left[\frac{(2s+1)(P!)^2}{2P} \frac{v_s(K)}{v_{s+P}(f) \cdot \mu_P(K)^2} \frac{1}{n} \right]^{\frac{1}{2s+2P+1}}
$$

My proposed data-driven bandwidth selection procedure:

We choose P and s, we can easily calculate $\mu_P(K)^2$, we can calculate $v_s(K)$, and we know n. Hence to implement this we only need to estimate $v_{s+P}(f)$, but this requires estimating the $s+P$th derivative of the density function f, which we do not know.

To estimate $f^{(s+P)}$, begin with a generic f (the normal distribution, say). Use this to estimate an initial guess for $h_{n,0}$. Using this initial estimate, we can construct $\hat{f}(x; h_{n,0})$ as defined at the outset of this question, as well as the necessary derivative of $\hat{f}$. Using this updated density $\hat{f}$, we can then determine the optimal bandwidth $h_{n,*}$ as desired.

1.3 Implementation

(a) See the code in Appendix A for R, and Appendix B for Stata. Note that I mainly focused on implementing these questions in R, and relied heavily on my classmates to attempt implementation in Stata.

Using R, I compute theoreticcaly the AIMSE-optimal bandwidth for $s = 0$, $n = 1000$, using the Epanechnikov kernel, as 0.8199. Using Stata I compute this as [].

(b) See Figures 1 and [Stataref]. Using R, I estimate $h_{\widehat{\text{IMSE}},\text{LI}} = h_{\text{AIMSE}} = 0.9019$ and the same for $h_{\widehat{\text{IMSE}},\text{LO}}$. I estimate the same values, respectively, using Stata as *blah* and *blah*.

[Stata figure here]

(c): True.

(d) Using R I estimate $\bar{h}_{\text{AIMSE}} = 0.9883$ and using Stata I estimate $\bar{h}_{\text{AIMSE}} = blah$.

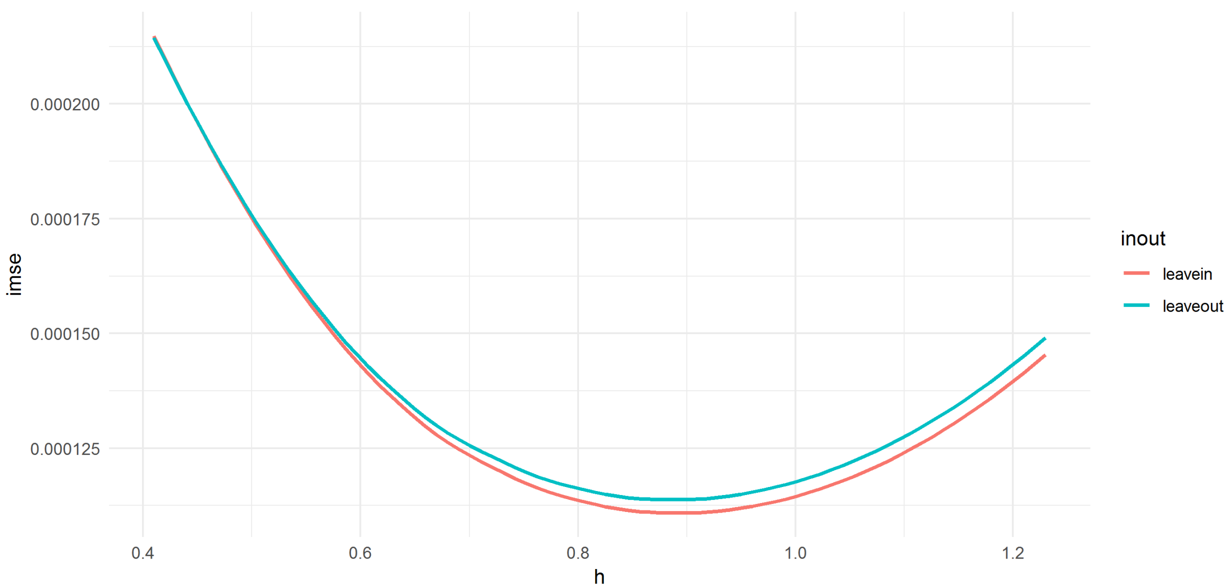

Figure 1: Estimated IMSE (Leave In and Leave Out) as a function of h: R

2 Question 2: Linear Smoothers, Cross-validation, and Series

2.1 Local Polynomial Regression and Series Estimators

For local polynomial regression, I rely on lecture notes available from the University of Manchester here.

Define the following:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 - x & (x_1 - x)^2 & \dots & (x_1 - x)^p \\ 1 & x_1 - x & (x_2 - x)^2 & \dots & (x_2 - x)^p \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n - x & (x_n - x)^2 & \dots & (x_n - x)^p \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}$$

With weighting/kernel matrix $\mathbf{W} = \text{diag}\{K_h(x_i - x), i = 1, ..., n\}$.

Also define the vector $\mathbf{e}_1$ of length $p+1$, with a 1 in the first position and 0 elsewhere.

We can write the estimator in the linear smoother form as:

$$\begin{aligned} \hat{e}(x) &= \mathbf{e}_1'(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}(\mathbf{X}'\mathbf{W}\mathbf{Y}) \\ &= \sum_{i=1}^{n} \mathbf{e}_1'(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1} \begin{bmatrix} 1 \\ (x_i - x) \\ (x_i - x)^2 \\ \dots \\ (x_i - x)^p \end{bmatrix} K_h(x_i - x) y_i \\ &= \sum_{i=1}^{n} w_{n,i}(x) y_i \end{aligned}$$

For series estimators I rely on Bruce Hansen's lecture notes here.

Consider an arbitrary series basis $\mathbf{z}(\cdot) : \mathbb{R}^{d_x} \to \mathbb{R}^K$ and define the regressor matrix $\mathbf{Z}$ as:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}(x_1)' \\ \mathbf{z}(x_2)' \\ \dots \\ \mathbf{z}(x_n)' \end{bmatrix}$$

Then we can write the estimator in the linear smoother form as:

$$\begin{aligned} \hat{e}(x) &= \mathbf{z}(x)'(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{Y} \\ &= \sum_{i=1}^{n} \mathbf{z}(x)'(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{z}(x_i) y_i \\ &= \sum_{i=1}^{n} w_{n,i}(x) y_i \end{aligned}$$

2.2 Cross-Validation

First focusing on series estimators, and using the hint given in the footnote, we have:

$$
\begin{aligned}
\hat{e}(x_i) &= \mathbf{p}(x_i)'(\mathbf{P}'\mathbf{P})^{-1}\mathbf{P}\mathbf{Y} \\
&= \sum_j \mathbf{p}(x_i)'(\mathbf{P}'\mathbf{P})^{-1}\mathbf{p}(x_j)y_j \\
&= \sum_j \mathbf{p}(x_i)'\Big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)} + \mathbf{p}(x_i)\mathbf{p}(x_i)'\Big)^{-1}\mathbf{p}(x_j)y_j \\
&= w_{n,i}(x_i)y_i + \sum_{j\neq i} \mathbf{p}(x_i)'\Big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)} + \mathbf{p}(x_i)\mathbf{p}(x_i)'\Big)^{-1}\mathbf{p}(x_j)y_j \\
&= w_{n,i}(x_i)y_i + \sum_{j\neq i} \mathbf{p}(x_i)'\left(\big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)}\big)^{-1} - \frac{\Big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)}\Big)^{-1}\mathbf{p}(x_i)\mathbf{p}(x_i)'\Big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)}\Big)^{-1}}{1+\mathbf{p}(x_i)'\Big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)}\Big)^{-1}\mathbf{p}(x_i)}\right)\mathbf{p}(x_j)y_j \\
&= w_{n,i}(x_i)y_i + \left(1 - \frac{\mathbf{p}(x_i)'\big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)}\big)^{-1}\mathbf{p}(x_i)}{1+\mathbf{p}(x_i)'\big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)}\big)^{-1}\mathbf{p}(x_i)}\right)\sum_{j\neq i}\mathbf{p}(x_i)'\big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)}\big)^{-1}\mathbf{p}(x_j)y_j \\
&= w_{n,i}(x_i)y_i + \left(1 - \frac{\mathbf{p}(x_i)'\big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)}\big)^{-1}\mathbf{p}(x_i)}{1+\mathbf{p}(x_i)'\big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)}\big)^{-1}\mathbf{p}(x_i)}\right)\hat{e}_{(i)}(x_i)
\end{aligned}
$$

And again using the hint, we have

$$
\begin{aligned}
w_{n,i}(x_i) &= \mathbf{p}(x_i)'\left(\big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)}\big)^{-1} - \frac{\Big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)}\Big)^{-1}\mathbf{p}(x_i)\mathbf{p}(x_i)'\Big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)}\Big)^{-1}}{1+\mathbf{p}(x_i)'\Big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)}\Big)^{-1}\mathbf{p}(x_i)}\right)\mathbf{p}(x_i) \\
&= \frac{\mathbf{p}(x_i)'\big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)}\big)^{-1}\mathbf{p}(x_i)}{1+\mathbf{p}(x_i)'\big(\mathbf{P}'_{(i)}\mathbf{P}_{(i)}\big)^{-1}\mathbf{p}(x_i)}
\end{aligned}
$$

Thus we have $\hat{e}(x_i) = w_{n,i}(x_i)y_i + (1 - w_{n,i}(x_i))\hat{e}_{(i)}(x_i)$, and after rearranging we have:

$$
y_i - \hat{e}_{(i)}(x_i) = \frac{y_i - \hat{e}(x_i)}{1 - w_{n,i}(x_i)}
$$

as desired.

2.3 Standard Errors

We know that $\mathbb{E}[\hat{e}(x)] \to_p e(x)$, so once we understand the variance term, we can apply Slutsky, LLN, and CLT to show the result.

2.4 Confidence Intervals

Using the conclusion of part (3) above, we have the following:

$$
\mathrm{CI}_{95\%}(x) = \left[\hat{e}(x) \pm 1.96 \cdot \sqrt{\hat{\mathbb{V}}[\hat{e}(x)|x_1, x_2, ..., x_n]}\right]
$$

Pointwise valid requires $\forall x, \liminf_n \mathbb{P}[e(x) \in \mathrm{CI}(x)] \geq 0.95$.
Uniformly valid has the stronger requirement that $\liminf_n \mathbb{P}[\forall x : e(x) \in \mathrm{CI}(x)] \geq 0.95$.

2.5 Implementation

(a) See the code in Appendix A for R, and Appendix B for Stata. Note that I mainly focused on implementing these questions in R, and relied heavily on my classmates to attempt implementation in Stata.

(b) See Figures 2 and 3. Based on my simulations, the optimal CV estimator is $\hat{K}_{CV} = 7$.

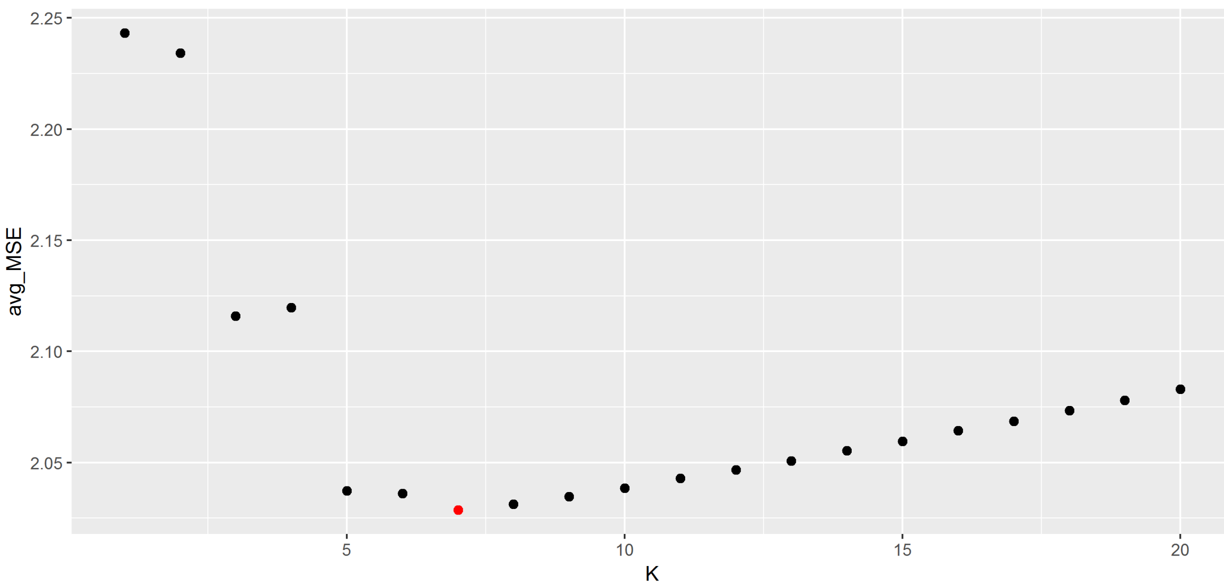

Figure 2: Cross-Validation Simulation Results: R

(c) See Figures 4 and [Stataref].
[Stata figure here]

(d) See Figures 5 and [Stataref].
[Stata figure here]

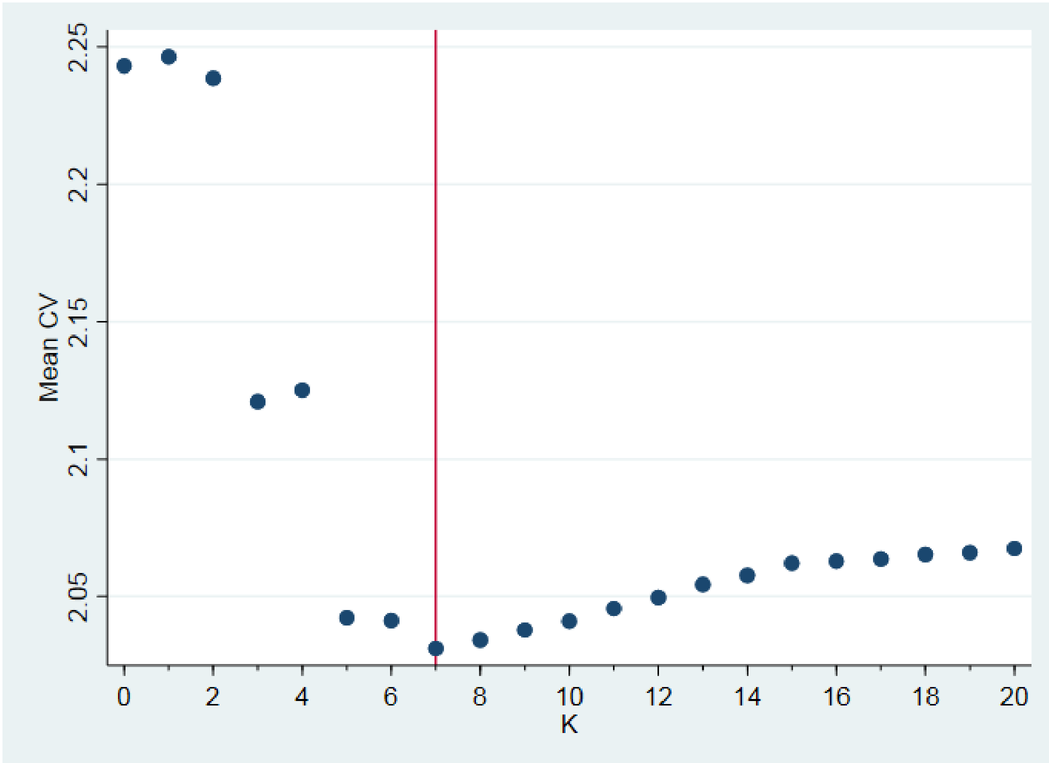

Figure 3: Cross-Validation Simulation Results: R

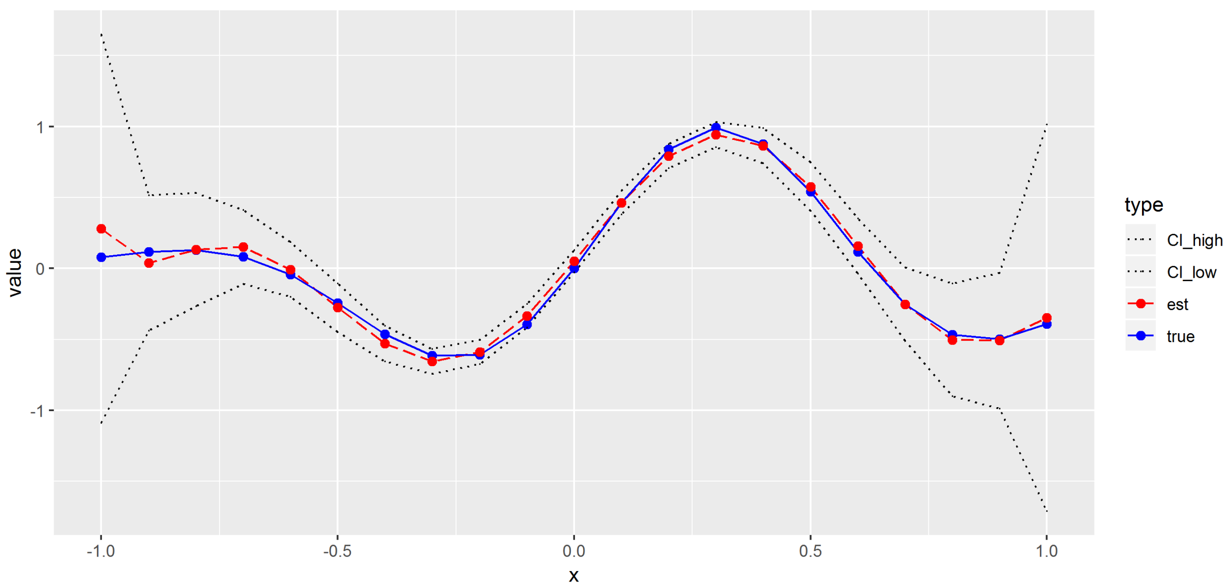

Figure 4: True and Estimated Regression Functions: R

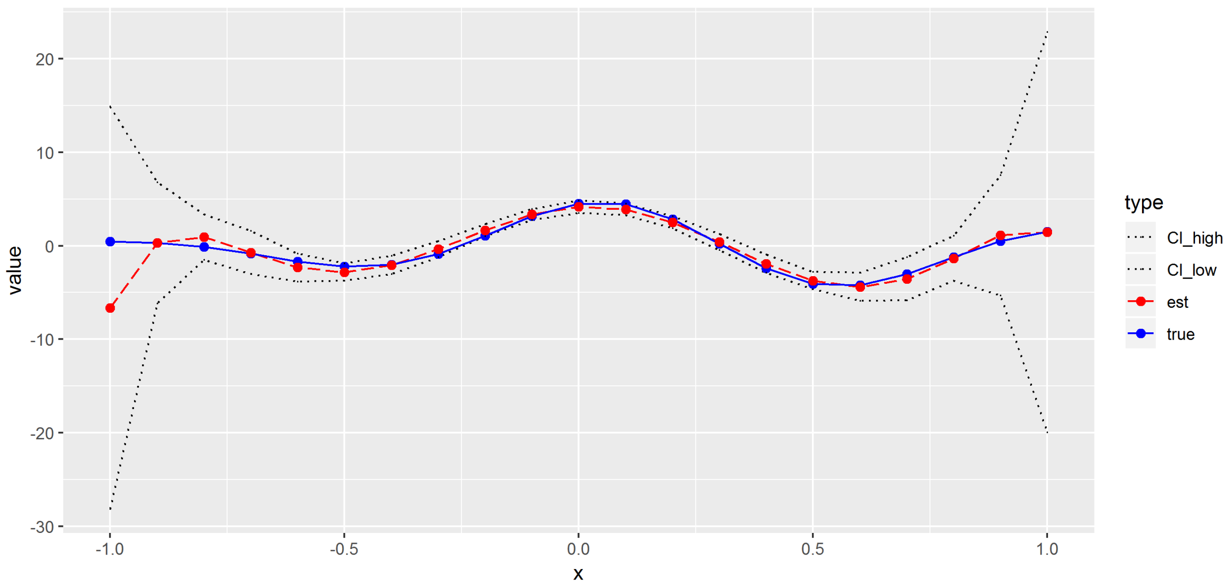

Figure 5: True and Estimated First Derivatives of the Regression Function: R

3 Question 3: Semiparametric Semi-Linear Model

3.1 Identification

We apply the Law of Iterated Expectations and substitute in the definition of y_i to split out the inside of the initial expectation, yielding terms that are assumed to equal zero:

$$\begin{aligned}
\mathbb{E}[(t_i - h_0(\mathbf{x}_i))(y_i - t_i\theta_0)] &= \mathbb{E}\Big[\mathbb{E}[(t_i - h_0(\mathbf{x}_i))(y_i - t_i\theta_0)|\mathbf{x}_i]\Big] \\
&= \mathbb{E}\Big[\mathbb{E}[(t_i - h_0(\mathbf{x}_i))(g_0(\mathbf{x}_i) + \varepsilon_i)|\mathbf{x}_i]\Big] \\
&= \mathbb{E}\Big[\mathbb{E}[(t_i - h_0(\mathbf{x}_i))g_0(\mathbf{x}_i)|\mathbf{x}_i]\Big] + \mathbb{E}\Big[\mathbb{E}[(t_i - h_0(\mathbf{x}_i))\varepsilon_i|\mathbf{x}_i]\Big] \\
&= \mathbb{E}\Big[g_0(\mathbf{x}_i)\mathbb{E}[(t_i - h_0(\mathbf{x}_i))|\mathbf{x}_i]\Big] + \mathbb{E}\Big[\mathbb{E}[(t_i - h_0(\mathbf{x}_i))\varepsilon_i|\mathbf{x}_i, t_i]\Big] \\
&= \mathbb{E}\Big[g_0(\mathbf{x}_i)\mathbb{E}[(t_i - h_0(\mathbf{x}_i))|\mathbf{x}_i]\Big] + \mathbb{E}\Big[(t_i - h_0(\mathbf{x}_i))\mathbb{E}[\varepsilon_i|\mathbf{x}_i, t_i]\Big] \\
&= \mathbb{E}\Big[g_0(\mathbf{x}_i)\mathbb{E}[(t_i - \mathbb{E}[t_i|\mathbf{x}_i])|\mathbf{x}_i]\Big] + \mathbb{E}\Big[(t_i - h_0(\mathbf{x}_i))\mathbb{E}[\varepsilon_i|\mathbf{x}_i, t_i]\Big] \\
&= \mathbb{E}\Big[g_0(\mathbf{x}_i)\cdot 0\Big] + \mathbb{E}\Big[(t_i - h_0(\mathbf{x}_i))\cdot 0\Big] = 0
\end{aligned}$$

Then taking the initial expecation, splitting it, and pulling out θ_0 we have:

$$\mathbb{E}[(t_i - h_0(\mathbf{x}_i))y_i] - \mathbb{E}[(t_i - h_0(\mathbf{x}_i))t_i]\cdot\theta_0 = 0 \implies \theta_0 = \frac{\mathbb{E}[(t_i - h_0(\mathbf{x}_i))y_i]}{\mathbb{E}[(t_i - h_0(\mathbf{x}_i))t_i]}$$

Given this expression for θ_0, we can see that it will be identified so long as the denominator is non-zero, i.e., $\mathbb{E}[(t_i - h_0(\mathbf{x}_i))t_i] \neq 0$.

For an IV interpretation, consider the reduced form expression $y_i = t_i\theta_0 + g_0(\mathbf{x}_i) + \varepsilon_i = t_i\theta_0 + u_i$. Here, u_i is uncorrelated with t_i so we can define an instrument $z_i = t_i - h_0(\mathbf{x}_i)$; we have $\mathbb{E}[z_iu_i] = 0$ and $\mathbb{E}[t_iz_i] \neq 0$, and hence a valid instrument.

3.2 Series Estimation

(a) Define matrices as follows (similar to Subsection 2.1 above):

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}^K(\mathbf{x}_1)' \\ \mathbf{p}^K(\mathbf{x}_2)' \\ \cdots \\ \mathbf{p}^K(\mathbf{x}_n)' \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} t_1 \\ t_2 \\ \cdots \\ t_n \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}$$

Then we can define the annihilator matrix $\mathbf{M_P} = \mathbf{I} - \mathbf{P}(\mathbf{P}'\mathbf{P})^{-1}\mathbf{P}'$, and regress y_i on t_i and $\mathbf{p}^{K_n}(\mathbf{x}_i)$, yielding:

$$\hat{\theta}(K) = (\mathbf{T}'\mathbf{M_P}\mathbf{T})^{-1}(\mathbf{T}'\mathbf{M_P}\mathbf{Y})$$

(b) We have $h_0(\mathbf{x}) = \mathbb{E}[t_i|\mathbf{x}_i] \approx \mathbf{p}^K(\mathbf{x}_i)'\delta_K$. Thus if we regress t_i on $\mathbf{p}^K(\mathbf{x}_i)'$ to estimate $\hat{\delta}_K$, we can then estimate $\hat{h}(\mathbf{x}_i)$:

$$\hat{h}(\mathbf{x}_i) = \mathbf{p}^K(\mathbf{x}_i)'\hat{\delta}_K = \mathbf{p}^K(\mathbf{x}_i)'(\mathbf{P}'\mathbf{P})^{-1}\mathbf{P}'\mathbf{T}$$

Then we can construct the residuals, and show that with these we yield the same estimate of θ_0 as in part (a):

Residual of the t_i regression is $t_i - \mathbf{p}^K(\mathbf{x}_i)'(\mathbf{P}'\mathbf{P})^{-1}\mathbf{P}'\mathbf{T} = \mathbf{e}_1'\mathbf{M}_P\mathbf{T}$, with $\mathbf{e}_i$ a vector of zeros with a 1 in the i-th element.

Then we can show that the numerator and denominator we derived in part (a) are numerically equivalent to that we find using this method:

$$\hat{\mathbb{E}}[(t_i - h_0(\mathbf{x}_i))y_i] = \frac{1}{n}\sum_{i=1}^{n}\mathbf{e}_i'\mathbf{M_P}\mathbf{T}y_i = \frac{1}{n}\mathbf{T}'\mathbf{M_P}\sum_{i=1}^{n}\mathbf{e}_i y_i = \frac{1}{n}\mathbf{T}'\mathbf{M_P}\mathbf{Y}$$

$$\hat{\mathbb{E}}[(t_i - h_0(\mathbf{x}_i))t_i] = \frac{1}{n}\sum_{i=1}^{n}\mathbf{e}_i'\mathbf{M_P}\mathbf{T}t_i = \frac{1}{n}\mathbf{T}'\mathbf{M_P}\sum_{i=1}^{n}\mathbf{e}_i t_i = \frac{1}{n}\mathbf{T}'\mathbf{M_P}\mathbf{T}$$

Finally, dividing the numerator by the denominator, we have the same result as in (a) for estimating θ_0:

$$\hat{\theta}(K) = \frac{\hat{\mathbb{E}}[(t_i - h_0(\mathbf{x}_i))y_i]}{\hat{\mathbb{E}}[(t_i - h_0(\mathbf{x}_i))t_i]} = (\mathbf{T}'\mathbf{M_P}\mathbf{T})^{-1}(\mathbf{T}'\mathbf{M_P}\mathbf{Y})$$

3.3 Asymptotics

(a) We can show that $\hat{\theta}(K) - \theta_0 \to_d \mathcal{N}(0, V)$, and then describe V.

$$\begin{aligned}
\hat{\theta}(K) - \theta_0 &= (\mathbf{T}'\mathbf{M_P}\mathbf{T})^{-1}(\mathbf{T}'\mathbf{M_P}\mathbf{Y}) - \theta_0 \\
&= (\mathbf{T}'\mathbf{M_P}\mathbf{T})^{-1}(\mathbf{T}'\mathbf{M_P}(\mathbf{T}\theta_0 + g_0(\mathbf{x}_i) + \varepsilon)) - \theta_0 \\
&= (\mathbf{T}'\mathbf{M_P}\mathbf{T})^{-1}(\mathbf{T}'\mathbf{M_P}(g_0(\mathbf{x}_i) + \varepsilon))
\end{aligned}$$

(b) This is a straightforward application of the distribution noted in (a):

$$\text{CI}_{95\%} = \left[\hat{\theta}(K) \pm 1.96 \cdot \sqrt{\hat{V}_{\text{HCO}}}\right]$$

3.4 Implementation

(a) See the code in Appendix A for R, and Appendix B for Stata. Note that I mainly focused on implementing these questions in R, and relied heavily on my classmates to attempt implementation in Stata.

(b) Results are in the following tables; code in the appendices.

Table 1: Monte Carlo Results: R

K	Avg $\hat{\theta}(K)$	Avg Bias	Sample Variance	Avg $\hat{V}_{\text{HCO}}$	Coverage Rate
6	3.044	2.044	0.392	0.346	0.109
11	0.645	-0.355	0.108	0.107	0.997
21	0.654	-0.346	0.108	0.105	0.998
26	0.655	-0.345	0.108	0.105	0.994
56	0.695	-0.305	0.102	0.094	1.000
61	0.727	-0.273	0.095	0.086	1.000
126	1.019	0.019	0.030	0.022	1.000
131	1.018	0.018	0.031	0.022	1.000
252	1.001	0.001	0.033	0.019	1.000
257	0.999	-0.001	0.033	0.019	1.000
262	0.999	-0.001	0.034	0.019	1.000
267	1.001	0.001	0.035	0.019	1.000
272	1.000	0.000	0.036	0.019	1.000
277	1.000	-0.000	0.036	0.019	1.000

[Stata table here]

(c) Using R, I estimate the following:

- Average $K_{CV} = 127.6$, Median $K_{CV} = 126$
- Average $\hat{\theta}(K_{CV}) = 1.018$
- Average Bias of $\hat{\theta}_{K_{CV}} = .0180$
- Sample Variance of $\hat{\theta}_{K_{CV}} = .0300$
- Average of $\hat{\mathbb{V}}_{\text{HCO}} = .0224$
- Average Coverage Rate $= 90.4\%$

I also plot some of my results in Figure 6:

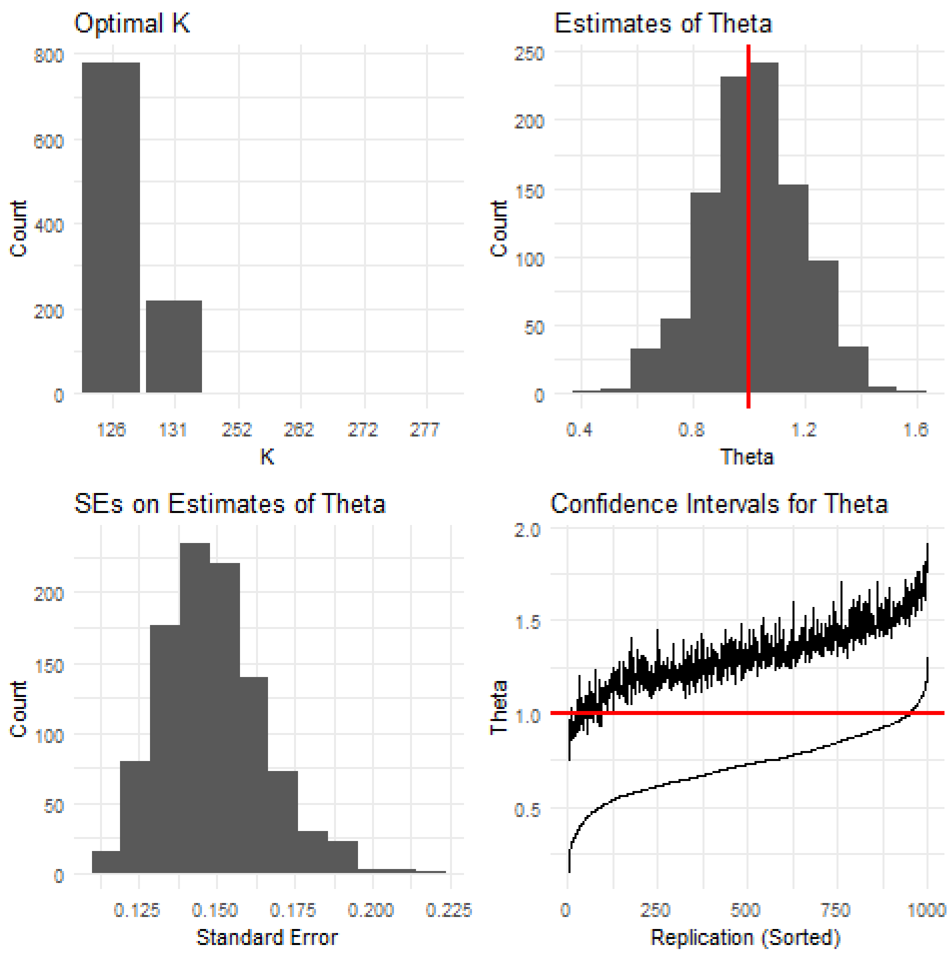

Figure 6: Monte Carlo Results using Cross Validation: R

Using Stata, I estimate the following:

- Average $K_{CV} =$, Median $K_{CV} =$
- Average $\hat{\theta}(K_{CV}) =$
- Average Bias of $\hat{\theta}_{K_{CV}} =$
- Sample Variance of $\hat{\theta}_{K_{CV}} =$

- Average of $\hat{\mathbb{V}}_{\text{HCO}} =$
- Average Coverage Rate =

I also plot some of my results in Figure [Stataref]:
[Stata figure here]

A R Code

```
################################################################################
# Author: Paul R. Organ
# Purpose: ECON 675, PS2
# Last Update: Oct 10, 2018
################################################################################
# Preliminaries
options(stringsAsFactors = F)

# packages
require(tidyverse) # data cleaning and manipulation
require(magrittr)  # syntax
require(ggplot2)   # plots
require(kedd)      # kernel bandwidth estimation
require(car)       # heteroskedastic robust SEs
require(xtable)    # tables for LaTeX

setwd('C:/Users/prorgan/Box/Classes/Econ_675/Problem_Sets/PS2')

################################################################################
## Question 1: Kernel Density Estimation
## Q1.3a

# sample size
n      <- 1000

# data generating process
dgp <- function(n){
  # equally weight two distributions
  comps <- sample(1:2,prob=c(.5,.5),size=n,replace=T)

  # Normal density specs
  mus <- c(-1.5, 1)
  sds <- sqrt(c(1.5, 1))

  # generate sample
  samp <- rnorm(n=n,mean=mus[comps],sd=sds[comps])

  return(samp)
}

# true dgp
f_true <- function(x){.5*dnorm(x,-1.5,sqrt(1.5))+.5*dnorm(x,1,1)}

# second deriv of normal dist
```

```r
norm_2d <- function(u,meanu,sdu){dnorm(u,mean=meanu,sd=sdu)*
                        (((u-meanu)^2/(sdu^4))-(1/(sdu^2)))}

# f for integration (theoretical)
f_int <- function(x){(.5*norm_2d(x,-1.5,1.5) + .5*norm_2d(x,1,1))^2}

# function to calculate (theoretically or empirical) optimal h
optimal_h <- function(x,mu,sd){
  k1 <- .75^2 * (2- 4/3 + 2/5)
  k2 <- .75 * (2/3 - 2/5)

  if(x=='theoretical'){
    f <- f_int
    k3 <- integrate(f,-Inf,Inf)$val
  } else{
    f <- function(x,mu,sd){norm_2d(x,mu,sd)^2}
    k3 <- integrate(f,-Inf,Inf,mu=mu,sd=sd)$val
  }

  h <- (k1/(k3*k2^2)*(1/n))^(1/5)
  return(h)
}

# theoretically optimal h
h_aimse <- optimal_h('theoretical',NA,NA)

######################################################################################
## Q1.3b
# define Kernel function: K(u)=.75(1-u^2)(ind(abs(u)<=1))
K0 <- function(u){
  out <- .75 * (1-u^2) * (abs(u) <= 1)
}

# function to calculate IMSE
imse <- function(h,X){
  # empty vectors to fill with results
  e_li <- rep(NA,n)
  e_lo <- rep(NA,n)

  # loop over each i to do leave one out
  for(i in 1:n){
    # repeat observation for each simulation
    Xi_n <- rep(X[i], n)
    # apply kernel function to (x-x_i)/h
    df <- K0((Xi_n-X)/h)

    # fhat with i in
    fhat_li <- mean(df)/h
    # fhat with i out
    fhat_lo <- mean(df[-i])/h

    # f(x_i)
    f_xi <- f_true(X[i])
```

```
    # mse with i in
    e_li[i] <- (fhat_li-f_xi)^2
    # mse with i out
    e_lo[i] <- (fhat_lo-f_xi)^2
  }
  out <- c(mean(e_li),mean(e_lo))
  return(out)
}

# simulate 1000 times
M <- 1000

# sequence of h's to test
hs <- seq(.5,1.5,.1) * h_aimse
nh <- length(hs)

# empty matrices to fill
imse_li <- matrix(NA, nrow=M, ncol=nh)
imse_lo <- matrix(NA, nrow=M, ncol=nh)

# generate matrix with M rows of sampled data
set.seed(22)
for(m in 1:M){
  X <- dgp(n)
  for(j in 1:nh){
    temp <- imse(hs[j],X)
    imse_li[m,j] <- temp[1]
    imse_lo[m,j] <- temp[2]
  }
}

df <- data.frame(h = hs, leavein = colMeans(imse_li),
                 leaveout = colMeans(imse_lo)) %>%
  gather(key = inout, value = imse, -h)

# plot
p <- ggplot(df, aes(x=h,y=imse,color=inout)) + geom_smooth(se=F) +
  theme_minimal()
p
ggsave('q1_3b_R.png')

# averages
h_hat_li <- df %>% filter(inout == 'leavein') %>%
  filter(imse == min(imse)) %>% select(h) %>% as.numeric
h_hat_lo <- df %>% filter(inout == 'leaveout') %>%
  filter(imse == min(imse)) %>% select(h) %>% as.numeric

##############################################################################
## Q1.3d

opt_hs <- rep(NA,M)

set.seed(22)
for(m in 1:M){
```

```r
  X <- dgp(n)
  mu <- mean(X)
  sd <- sd(X)
  opt_hs[m] <- optimal_h(n,mu,sd)
}

hbar <- mean(opt_hs)

##########################################################################################
## Question 2: Linear Smoothers, Cross-Validation, and Series
# clean up
rm(list = ls())
gc()

##########################################################################################
## Q2.5a
# define datagenerating process
set.seed(22)
dgp <- function(n){
  # input:  sample size n
  # output: n draws of X and Y according to DGPs as specified in PSet

  # X ~ Uniform(-1,1)
  X <- runif(n,-1,1)

  # Epsilon ~ x^2 * (Chisq_5 - 5)
  E <- (X^2) * (rchisq(n,5)-5)

  # Y ~ exp(-.1*(4x-1)^2) * sin(5x) + Eps
  Y <- exp(-0.1 * (4*X-1)^2) * sin(5*X) + E

  out = data.frame(X=X,Y=Y)
}

# sample size
n <- 1000

# replications
M <- 1000

##########################################################################################
## Q2.5b
# series truncation
K  <- 1:20
nK <- length(K)

# define series cross-validation function
series_cv <- function(n, X, Y, nK, K){
  # input:  n draws of X and Y, series truncation K of length nK
  # output: nK prediction errors

  # start with polynomial basis of X
  X_poly <- cbind(rep(1,n), poly(X,degree = nK))
```

```r
  # QR decomposition
  X_qr <- qr(X_poly)

  # coefficients
  coefs <- qr.coef(X_qr,Y)

  # cycle up through each power and store prediction errors
  out <- rep(NA,nK)
  for(k in 1:nK){
    X_poly_k <- X_poly[,1:(K[k]+1)]
    coefs_k  <- matrix(coefs[1:(K[k]+1)],nrow=K[k]+1)
    Y_hat_k  <- X_poly_k %*% coefs_k

    # w (see part 1 of question 2)
    w_k <- diag(X_poly_k %*% solve(t(X_poly_k) %*% X_poly_k) %*% t(X_poly_k))

    # prediction error
    cv_k <- mean( ((Y-Y_hat_k)/(1-w_k))^2 )

    # save
    out[k] <- cv_k
  }

  return(out)
}

# run series cv formula 1000 times, save results for plotting
# we want to capture the MSE for each K for each rep
MSEs <- matrix(NA, ncol=nK, nrow=M)
set.seed(22)
for(m in 1:M){
  df <- dgp(n)
  MSEs[m,] <- series_cv(n=n, X=df$X, Y=df$Y, nK=nK, K=K)
}

# average CV(K) across simulations:
averages <- data.frame(K = 1:20, avg_MSE = colMeans(MSEs))

# identify optimal CV estimator
averages %<>% mutate(Group = (avg_MSE == min(avg_MSE)))
K_CV <- averages %>% filter(Group) %>% select(K) %>% as.numeric

# plot average CV(K), highlight the optimal one
plot <- ggplot(data = averages, aes(x = K, y = avg_MSE, color = Group)) +
  geom_point(size = 2) + scale_color_manual(values=c('black','red')) +
  theme(legend.position='none')
plot
ggsave('q2_5b_R.png')

#################################################################################
## Q2.5c
# define grid of evaluation points
grid      <- seq(-1,1,.1)
eval_pts <- length(grid)
```

```
# we know the true value of the regression function (from dgp defined above)
f_true <- exp(-0.1 * (4*grid-1)^2) * sin(5*grid)

# estimate regression function using polynomial basis (7th degree based on (b))
# generate polynomial basis for grid points
grid_poly <- cbind(1, grid, grid^2, grid^3, grid^4, grid^5, grid^6, grid^7)

# define empty matrices to fill with estimates
ests <- matrix(NA, ncol=eval_pts, nrow=M)
SEs  <- matrix(NA, ncol=eval_pts, nrow=M)

# simulate M times
set.seed(22)
for(m in 1:M){
  # draw data
  df <- dgp(n)
  X <- df$X
  Y <- df$Y

  # create polynomial
  X_poly <- cbind(1,X,X^2,X^3,X^4,X^5,X^6,X^7)

  # run regression using drawn data
  reg   <- lm(Y ~ X_poly - 1)
  betas <- coefficients(reg)
  vars  <- hccm(reg, type='hc0') # heteroskedasticity-corrected using 'car' package

  # calculate and save estimates and SEs using grid evaluation points
  ests[m,] <- grid_poly %*% betas
  SEs[m,]  <- sqrt(diag(grid_poly %*% vars %*% t(grid_poly)))
}

# average across the simulations to create estimated regression function
f_est <- colMeans(ests)

# calculate average confidence intervals across the simulations
f_est_low  <- colMeans(ests)-1.96*colMeans(SEs)
f_est_high <- colMeans(ests)+1.96*colMeans(SEs)

# gather data for plotting
df <- data.frame(x = grid, true = f_true, est = f_est,
                 CI_low = f_est_low, CI_high = f_est_high) %>%
  gather(key = type, value = value, -x)

# plot in one graph, note specification of options is alphabetical by type
# types = CI_high, CI_low, est, true
plot <- ggplot(data = df, aes(x = x, y = value, color = type)) +
  geom_line(aes(linetype=type, color=type)) +
  geom_point(aes(color=type, size=type)) +
  scale_linetype_manual(values = c('dotted', 'dotted', 'longdash', 'solid')) +
  scale_color_manual(values = c('black', 'black', 'red', 'blue')) +
  scale_size_manual(values = c(0,0,2,2))
plot
```

```r
ggsave('q2_5c_R.png')

################################################################################
## Q2.5d
# Now estimating the derivative of the regression function
# This will reuse my code from above, with new polynomials (taking derivs)

#  true value of derivative of regression function (product rule)
f1_true <- exp(-.1*(4*grid-1)^2)*5*cos(5*grid) +
  sin(5*grid)*(-.8*(4*grid-1)*exp(-.1*(4*grid-1)^2))

# estimate regression function using polynomial basis (7th degree based on (b))
# generate polynomial basis for grid points, first derivatives of grid_poly
grid1_poly <- cbind(0, 1, 2*grid, 3*grid^2, 4*grid^3, 5*grid^4, 6*grid^5, 7*grid^6)

# define empty matrices to fill with estimates
ests1 <- matrix(NA, ncol=eval_pts, nrow=M)
SEs1  <- matrix(NA, ncol=eval_pts, nrow=M)

# simulate M times
set.seed(22)
for(m in 1:M){
  # draw data
  df <- dgp(n)
  X <- df$X
  Y <- df$Y

  # create polynomial
  X_poly <- cbind(1,X,X^2,X^3,X^4,X^5,X^6,X^7)

  # run regression using drawn data
  reg   <- lm(Y ~ X_poly - 1)
  betas <- coefficients(reg)
  vars  <- hccm(reg, type='hc0') # heteroskedasticity-corrected using 'car' package

  # calculate and save estimates and SEs using grid evaluation points
  ests1[m,] <- grid1_poly %*% betas
  SEs1[m,]  <- sqrt(diag(grid1_poly %*% vars %*% t(grid1_poly)))
}

# average across the simulations to create estimated regression function
f1_est <- colMeans(ests1)

# calculate average confidence intervals across the simulations
f1_est_low  <- colMeans(ests1)-1.96*colMeans(SEs1)
f1_est_high <- colMeans(ests1)+1.96*colMeans(SEs1)

# gather data for plotting
df1 <- data.frame(x = grid, true = f1_true, est = f1_est,
                  CI_low = f1_est_low, CI_high = f1_est_high) %>%
   gather(key = type, value = value, -x)

# plot in one graph, note specification of options is alphabetical by type
# types = CI_high, CI_low, est, true
```

```
plot <- ggplot(data = df1, aes(x = x, y = value, color = type)) +
  geom_line(aes(linetype=type, color=type)) +
  geom_point(aes(color=type, size=type)) +
  scale_linetype_manual(values = c('dotted', 'dotted', 'longdash', 'solid')) +
  scale_color_manual(values = c('black', 'black', 'red', 'blue')) +
  scale_size_manual(values = c(0,0,2,2))
plot
ggsave('q2_5d_R.png')

##############################################################################
## Question 3: Semiparametric Semi-Linear Model
# clean up
rm(list = ls())
gc()

##############################################################################
## Q3.4a
# define data generating process

# sample size
n <- 500
# replications
M <- 1000

# define data generating process
dgp <- function(n){
  # input:  sample size n
  # output: n draws of X and Y according to DGPs as specified in PSet

  # X is a d(5) by n matrix ~ U(-1,1)
  X <- matrix(runif(n*5,-1,1), ncol=5)

  # V ~ N(0,1) and U ~ N(0,1)
  V <- rnorm(n)
  U <- rnorm(n)

  # Eps = .36...*(1+||X||^2)*V
  E <- 0.3637899*(1+diag(X %*% t(X)))*V

  # g_0(X) = exp(||X||^2)
  G <- exp(diag(X %*% t(X)))

  # T = ind(||x||+u >= 0) (times 1 to convert from Boolean to numeric)
  Tee <- matrix((sqrt(diag(X %*% t(X))) + U >= 0)*1, ncol = 1)

  # Y as defined in problem (assuming \theta_0 = 1)
  Y <- matrix(Tee + G + E, ncol = 1)

  # returning list with matrix X, vector Y, vector T
  out = list(X=X,Y=Y,Tee=Tee)
}

# define polynomial basis
K <- c(6,11,21,26,56,61,126,131,252,257,262,267,272,277)
```

```r
polybasis <- function(X,K){
  # inputs: data matrix X, 'order' K
  # outputs: polynomial basis of 'order' K
  # Note: this gets really ugly at the end,
  # but I was tired and gave up trying to find a more elegant way
  if(K==6){basis <- poly(X, degree=1,raw=T)}
  if(K==11){basis <- cbind(poly(X, degree=1,raw=T),
                          X[,1]^2,X[,2]^2,X[,3]^2,X[,4]^2,X[,5]^2)}
  if(K==21){basis <- poly(X, degree=2,raw=T)}
  if(K==26){basis <- cbind(poly(X, degree=2,raw=T),
                          X[,1]^3,X[,2]^3,X[,3]^3,X[,4]^3,X[,5]^3)}
  if(K==56){basis <- poly(X, degree=3,raw=T)}
  if(K==61){basis <- cbind(poly(X, degree=3,raw=T),
                          X[,1]^4,X[,2]^4,X[,3]^4,X[,4]^4,X[,5]^4)}
  if(K==126){basis <- poly(X, degree=4,raw=T)}
  if(K==131){basis <- cbind(poly(X, degree=4,raw=T),
                           X[,1]^5,X[,2]^5,X[,3]^5,X[,4]^5,X[,5]^5)}
  if(K==252){basis <- poly(X, degree=5,raw=T)}
  if(K==257){basis <- cbind(poly(X, degree=5,raw=T),
                           X[,1]^6,X[,2]^6,X[,3]^6,X[,4]^6,X[,5]^6)}
  if(K==262){basis <- cbind(poly(X, degree=5,raw=T),
                           X[,1]^6,X[,2]^6,X[,3]^6,X[,4]^6,X[,5]^6,
                           X[,1]^7,X[,2]^7,X[,3]^7,X[,4]^7,X[,5]^7)}
  if(K==267){basis <- cbind(poly(X, degree=5,raw=T),
                           X[,1]^6,X[,2]^6,X[,3]^6,X[,4]^6,X[,5]^6,
                           X[,1]^7,X[,2]^7,X[,3]^7,X[,4]^7,X[,5]^7,
                           X[,1]^8,X[,2]^8,X[,3]^8,X[,4]^8,X[,5]^8)}
  if(K==272){basis <- cbind(poly(X, degree=5,raw=T),
                           X[,1]^6,X[,2]^6,X[,3]^6,X[,4]^6,X[,5]^6,
                           X[,1]^7,X[,2]^7,X[,3]^7,X[,4]^7,X[,5]^7,
                           X[,1]^8,X[,2]^8,X[,3]^8,X[,4]^8,X[,5]^8,
                           X[,1]^9,X[,2]^9,X[,3]^9,X[,4]^9,X[,5]^9)}
  if(K==277){basis <- cbind(poly(X, degree=5,raw=T),
                           X[,1]^6,X[,2]^6,X[,3]^6,X[,4]^6,X[,5]^6,
                           X[,1]^7,X[,2]^7,X[,3]^7,X[,4]^7,X[,5]^7,
                           X[,1]^8,X[,2]^8,X[,3]^8,X[,4]^8,X[,5]^8,
                           X[,1]^9,X[,2]^9,X[,3]^9,X[,4]^9,X[,5]^9,
                           X[,1]^10,X[,2]^10,X[,3]^10,X[,4]^10,X[,5]^10)}
  return(basis)
}

##########################################################################################
## Q3.4b

# number of different orders to test
nK <- length(K)

# define blank matrices to fill with simulated results
thetas <- matrix(NA, ncol = nK, nrow = M)
SEs    <- matrix(NA, ncol = nK, nrow = M)

set.seed(22)
ptm <- proc.time()
```

```r
for(m in 1:M){
  # draw data
  data <- dgp(n)
  X   <- data$X
  Y   <- data$Y
  Tee <- data$Tee

  # cycle through K orders
  for(k in 1:nK){
    # generate basis, add intercept
    X_poly <- cbind(1,polybasis(X,K[k]))

    # define M_P (I-P(P'P)^{-1}P)
    M_P <- diag(n) - (X_poly %*% solve((t(X_poly) %*% X_poly)) %*% t(X_poly))

    # estimate theta(K)
    theta <- (t(Tee) %*% M_P %*% Y)/(t(Tee) %*% M_P %*% Tee)

    # sigma (for variance estimate)
    sigma <- diag( as.numeric((M_P %*% (Y - Tee*as.numeric(theta))))^2 )

    # standard error
    bread <- solve((t(Tee) %*% M_P %*% Tee))
    se <- sqrt( bread %*% (t(Tee)%*%M_P%*%sigma%*%M_P%*%Tee) %*% bread)

    # save to matrix
    thetas[m,k] <- theta
    SEs[m,k]    <- se
  }
}
proc.time() - ptm
# 12 minute runtime

# calculate averages, variances, etc. of simulated values
summ <- matrix(NA, ncol=6, nrow = nK)
for(k in 1:nK){
  summ[k,1] <- K[k] # 'order' K
  summ[k,2] <- mean(thetas[,k]) # avergage theta(K)
  summ[k,3] <- summ[k,2]-1 # average bias of theta(K) (assuming theta_0=1)
  summ[k,4] <- sd(thetas[,k])^2 # sample variance of theta(K)
  summ[k,5] <- mean( (SEs[,k])^2 ) # average of vhat
  # check if CIs for each simulation include 1 (here checking the boundaries)
  summ[k,6] <- 1 - mean( thetas[,k]-1.96*SEs[k] > 1 | thetas[,k]+1.96*SEs[k] < 1)
}

# format
summ %<>% as.data.frame
names(summ) <- c('K', 'avg_theta', 'avg_bias',
                 'samp_variance', 'avg_vhat', 'coverage_rate')

# write for inclusion in latex document
print(xtable(summ,digits=c(0,0,3,3,3,3,3)),include.rownames=F)

##############################################################################
```

```r
## Q3.4c

# define crossvalidation function
crossval <- function(X, Y, Tee, nK, K){
  # blank vector to fill with MSE
  MSEs <- rep(NA, nK)

  # loop through each K to identify optimal bandwidth
  for(k in 1:nK){
    # define polynomial basis
    X_poly <- cbind(1, Tee, polybasis(X, K[k]))
    # QR decomposition
    X_poly_Q <- qr.Q(qr(X_poly))
    XX <- X_poly_Q %*% t(X_poly_Q)
    Y_hat <- XX %*% Y
    W <- diag(XX)
    MSEs[k] <- mean( ((Y-Y_hat)/(1-W))^2 )
  }

  # return the optimal K
  return(K[which.min(MSEs)])
}

# define blank vectors to fill with simulated results and optimal Ks
# (not matrices now, since we are using optimal K)
thetas <- rep(NA, M)
SEs    <- rep(NA, M)
Ks     <- rep(NA, M)

# simulate M times
set.seed(22)
ptm <- proc.time()
for(m in 1:M){
  # draw data
  data <- dgp(n)
  X   <- data$X
  Y   <- data$Y
  Tee <- data$Tee

  # given data, estimate optimal K using cross validation
  K_CV <- crossval(X, Y, Tee, nK, K)

  # generate basis, add intercept
  X_poly <- cbind(1,polybasis(X,K_CV))

  # define M_P (I-P(P'P)^{-1}P)
  M_P <- diag(n) - (X_poly %*% solve((t(X_poly) %*% X_poly)) %*% t(X_poly))

  # estimate theta(K)
  theta <- (t(Tee) %*% M_P %*% Y)/(t(Tee) %*% M_P %*% Tee)

  # sigma (for variance estimate)
  sigma <- diag( as.numeric((M_P %*% (Y - Tee*as.numeric(theta))))^2 )
```

```r
  # standard error
  bread <- solve((t(Tee) %*% M_P %*% Tee))
  se <- sqrt( bread %*% (t(Tee)%*%M_P%*%sigma%*%M_P%*%Tee) %*% bread)

  # save to vectors
  thetas[m] <- theta
  SEs[m]    <- se
  Ks[m]     <- K_CV
}
proc.time() - ptm
# runtime 14 minutes

# prep data for plots to show results
df <- data.frame(K = Ks, theta = thetas, se = SEs, rep = 1:M) %>%
  mutate(ci_low = theta - 1.96*se, ci_high = theta + 1.96*se) %>%
  arrange(ci_low) %>% mutate(rep_sorted = 1:M)

# panel 1: histogram of optimal Ks
k_df <- df %>% group_by(K) %>% summarise(Count = n()) %>% mutate(K = as.character(K))
p1 <- ggplot(k_df, aes(x=K,y=Count)) + geom_bar(stat='identity') +
  theme_minimal() + ggtitle('Optimal K')
p1

# panel 2: distribution of theta
p2 <- ggplot(df, aes(x=theta)) + geom_histogram(bins=12) + theme_minimal() +
  geom_vline(xintercept=1,linetype='solid',color='red',size=1) +
  labs(title='Estimates of Theta',x='Theta',y='Count')
p2

# panel 3: distribution of SEs
p3 <- ggplot(df, aes(x=se)) + geom_histogram(bins=12) + theme_minimal() +
  labs(title='SEs on Estimates of Theta',x='Standard Error',y='Count')
p3

# panel 4: confidence intervals, sorted by lower point
p4 <- ggplot(df) +
  geom_line(aes(x = rep_sorted, y=ci_low)) +
  geom_line(aes(x = rep_sorted, y=ci_high)) +
  geom_hline(yintercept=1,linetype='solid',color='red',size=1) +
  theme_minimal() +
  labs(title='Confidence Intervals for Theta',x='Replication (Sorted)',y='Theta')
p4

# combine with multiplot
source('multiplot.R')
png('q3_4c_R.png')
multiplot(p1, p3, p2, p4, cols=2)
dev.off()

# values for latex
avg_K      <- mean(Ks)
median_K   <- median(Ks)
avg_theta  <- mean(thetas)
avg_bias   <- mean(thetas)-1
```

```
samp_var  <- sd(thetas)^2
avg_vhat  <- mean(SEs^2)
coverage  <- 1 - mean(df$ci_low > 1 | df$ci_high < 1)

####################################################################################
```

B Stata Code

Blah