Bounds for MatLab: Draft Version 1.0

Arie Beresteanu and Charles F. Manski
Department of Economics, Northwestern University
June 21, 2000

1. Uses of the Package

The MatLab routines bundled in this package implement many of the methods for nonparametric analysis of treatment response developed in Manski (1990, 1994, 1995, 1997), and Manski and Pepper (2000). The most basic of these methods yields sharp bounds on average treatment effects and other quantities of interest in the absence of maintained structural assumptions. Tighter bounds are obtained when various weak structural assumptions are maintained. This version of the package implements the bounds that hold under instrumental variable assumptions as well as those that hold under monotone and concave-increasing treatment response assumptions. The package also generates nonparametric point estimates of treatment effects under the assumption that treatment selection is exogenous. It is envisioned that future versions of the package will implement additional procedures, including ones for estimation of bounds under monotone instrumental variable assumptions.

A further use of the package is to perform nonparametric analysis of regressions with missing outcome data or jointly missing outcome and covariate data, implementing the methods of Manski (1989) and Horowitz and Manski (1998, 2000). These methods yield sharp bounds on regressions in the absence of assumptions on the nature of the missing data.

2. Structure of the Package

Bounds is designed to serve two types of users. Some users may wish to use *Bounds* routines as elements within MatLab programs of their own device. Others may wish to use *Bounds* as a menu-driven stand-alone package. Users of the first type should focus their attention on the *Bounds* Elementary Routines and Core Procedures. Users of the second type will employ the *Bounds* Shell Program. In what follows, the names of the elementary routines, subroutines, and core procedures recognized by *Bounds* are in *italics*.

Elementary Routines

The elementary operations used by *Bounds* are nonparametric estimation of regressions and bootstrap estimation of sampling distributions. The Elementary Routines are Matlab commands performing these operations. They are

kern – a routine performing kernel estimation of regressions

silverman – a routine computing Silverman's "rule of thumb" bandwidth for use in *kern*.

empirical – a routine using the empirical distribution of the data to generate bootstrap pseudo-samples and the resulting bootstrap sampling distribution of the estimates of a Core Procedure.

Core Procedures

Version 1.0 of *Bounds* contains these core procedures for analysis of treatment effects and regressions with missing data. Each procedure is a Matlab command:

treatment – estimates "worst-case" bounds on average treatment effects; that is, bounds imposing no structural assumptions

iv – estimates bounds on average treatment effects with a specified subset of the covariates used as instrumental variables.

monotone – estimates bounds with treatment response assumed to be monotone or concave-increasing.

exogenous – estimates average treatment effects under the assumption that treatment selection is exogenous

outcen – estimates worst-case bounds on regressions when outcome observations have missing outcome data

jointcen – estimates worst-case bounds on regressions when some observations have missing outcome and covariate data.

Bootstrap sampling distributions for the estimates produced by these procedures and by *kern* can be obtained by calling routine *empirical*.

The core procedures and *kern* share a common input-output format, this being

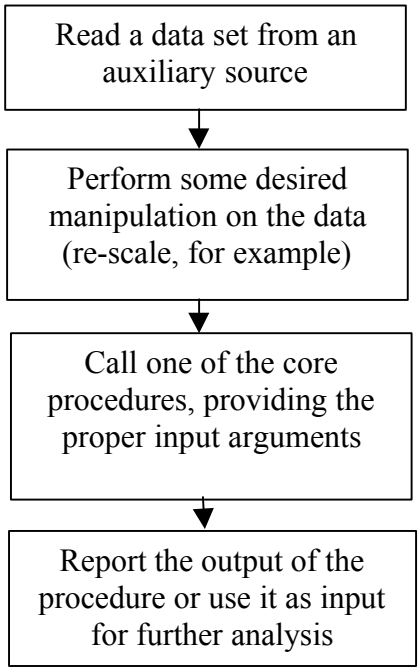
procedure(outcome data, treatment data, covariate data, instruments, covariate value of interest, procedure-specific parameters).

The order of the input arguments is the same for all procedures, but some procedures only require a subset of the arguments. Similarly, the ordering of outputs is common across procedures, but some procedures generate only a subset of the outputs. All of the procedures perform their calculations at the specified "covariate value of interest." To produce output at multiple covariate values, the user should write an outer loop program to update this covariate value and repeat the calculations.

Outputs are produced in two formats, an ASCII file and a MatLab structure. The ASCII file is a text file serving as the log of the session. This output format is optional and is available only when using the *Bounds* Shell Program.

The main output format is a MatLab structure. A Matlab structure is a named list of output variables, called fields. The user may choose the name for the structure, but the fields within the structure have predetermined names that are specific to each procedure. Knowledge of the field names, which are specified in Section 4 within the procedure descriptions, is needed only by users who wish to manipulate the bounds output within their own Matlab programs.

To use a Bounds procedure, one writes a MatLab program with the following general structure:



The *Bounds* Shell Program - The shell program is a set of pre-designed menus that enables stand-alone use of the *Bounds* procedures, with estimates accompanied by bootstrap sampling distributions. To initiate the shell program, simply call the Matlab program *Bounds*. Then choose the desired procedure on the menu that appears. This done, a sub-menu for that procedure will appear, requesting that relevant data file and parameter information be specified. The names of the shell subprograms for the various procedures are as follows:

treat_menu (procedure *treatment*)
iv_menu (procedure *iv*)
mono_menu (procedure *monotone*)
exog_menu (procedure *exogenous*)
out_menu (procedure *outcen*)
joint_menu (procedure *jointcen*)

The user can call these subprograms directly without passing through the main shell program *Bounds*. However, no log file is produced in this case.

3. Installation of the Package

The *Bounds* software is available as a zip file on these webpages:

Charles Manski's homepage:

http://www.econ.faculty.northwestern.edu/faculty/manski/

Arie Beresteanu's homepage: http://pubweb.northwestern.edu/~abe267/bounds

The file name is: bounds_matlab.zip

Unzip the file to a dedicated folder. For example, one might create the folder c:\matlab\bounds. These file names will appear in the folder: *kern.m, silverman.m, empirical.m, treatmen.m, iv.m, monotone.m, exogenous.m, outcen.m, jointcen.m, bounds.m, treat_menu.m, iv_menu.m, mono_menu.m, exog_menu.m, out_menu.m, joint_menu.m*

To use the software, you must "tell" Matlab the name of the folder within which the files are stored. Write the following command in the Matlab command window:

path(path, 'c:\matlab\bounds')

(replace c:\matlab\bounds by your chosen folder name). Matlab will then add c:\matlab\bounds to its active path. To avoid entering this command every time you open a Matlab session, you can add the folder name permanently to the active path of Matlab. To do this, choose "Path set" from the file toolbar. In the path browser window, choose "add to path" from the Path toolbar and write c:\matlab\bounds in the dialog box.

This completes the installation.

4. Elementary Routines

Kernel estimation of mean regressions

Purpose: use the Nadaraya-Watson kernel smoothing method to estimate the conditional expectation E(y|x), where y is a scalar random variable and x is a vector random variable. See Hardle (1990) for the theory of kernel estimation.

Usage: yhat = *kern*(y, x, h, x_0, kernel)

y- outcome data, an (n x 1) vector of real numbers, n being the sample size.

x- regressor data, an (n x k) matrix of real numbers.

h- bandwidth(s), a scalar or a (1 x k) vector, depending on the bandwidth option chosen (see below)

x_0– the value of x at which estimation is to be performed. x_0 is a (1 x k) vector.

kernel - string naming the kernel used (see below for options)

yhat – the estimate of E(y|x = x_0)

(Note: when an input is a string variable, the input need to be written inside quatation marks. See examples bellow.)

Method: For the specified value x_0 and each of the n observations in the data, the routine calculates the bandwidth-normalized Euclidean distance $z_j = \sqrt{\sum_{l=1}^{k}\left(\frac{x_{0l} - x(j)_l}{h_l}\right)^2}$ where j goes from 1 to n and l stands for the l^{th} element of the vector x. If the bandwidth h is specified as a scalar, then the same bandwidth is used for each component of the x-vector. If the bandwidth is a (1 x k) vector, then the bandwidth may vary across components of x.

Using the Nadaraya-Watson method, the conditional expectation is calculated as follows:

$$E[y \mid x = x0)] = \frac{\sum_{j=1}^{n} y(j) \cdot k(z_j)}{\sum_{j=1}^{n} k(z_j)}.$$

Here $k(\cdot)$ is the kernel function. The user may specify these kernels:

epa - The Epanechnikiv density. $k(u) = \frac{3}{4}(1-u^2)$ for $|u|<1$ and zero otherwise.

gau - The Gaussian density. $k(u) = \frac{1}{\sqrt{2p}} e^{-\frac{1}{2}u^2}$ for all u.

tri - The triangular density. $k(u) = |1-u|$ for $|u|<1$ and zero otherwise.

uni - The uniform density. $k(u) = \frac{1}{2}$ for $|u|<1$ and zero otherwise.

qua - The quartic density. $k(u) = \frac{15}{16}(1-u^2)^2$ for $|u|<1$ and zero otherwise.

The Epanechnikov kernel is used if the user does not specify a kernel or enters an invalid kernel type. Silverman's rule-of-thumb bandwidth (see below) is used if no bandwidth is specified.

The product of the routine is the estimate yhat.

Example: yhat = *kern*(y, x, [2.3 1.8], [12 12], '*gau*'). Here x should be an (n x 2) matrix. The bandwidths applied to the first and second components of x are 2.3 and 1.8. The estimate is obtained at the covariate value x = [12 12]. The Gaussian kernel is used.

Kernel functions

Purpose: These routines are called by *kern* to generate the desired kernel functions.

Usage: weight=*epa*(x,x_0,h); (and the same with *gau*, *tri*, *uni*, *qua*)
x- (1 x k) vector of sample covariate data.
x0 - (1 x k) vector, the covariate value of interest.
h - bandwidth(s), a scalar or a (1 x k) vector, depending on the bandwidth option chosen (see *kern* above)

Method: If h is a scalar then $u = \frac{\|x - x_0\|}{h}$ where $\|\cdot\|$ is the euclidean norm. If h is a (1 x k) vector then $u = \left(\sum_{i=1}^{k} \left(\frac{x_i - x_{0,i}}{h_i} \right)^2 \right)^{\frac{1}{2}}$. Then weight will be equal to $k(u)$ for the kernel functions as described in *kern*.

Example: *epa*([3 3], [2 1], [1.5 3]) will give 0.0833 .
tri([3 3], [2 1], [1.5 3]) will give 0.0572 .
uni([3 3] ,[1 0], [1 2]) will give 0 .
gau([3 3] ,[1 0], [1 2]) will give 0.0175 .

Silverman's rule-of-thumb bandwidth

Purpose: This routine computes Silverman's rule-of-bandwidth for users who prefer to employ automated bandwidths rather than to select their own. See Silverman (1986) and Hardle (1990). The bandwidth computed is a vector.

Usage: bandwidth = *silverman*(x);
x – (n x k) matrix of regressor data.

Method: The following formula describes the computation of the bandwidth: Let

$$std = std(x_i)_{i=1..k}$$
$$q25 = (25^{th}\ quantile\ of\ x_i)_{i=1..k}$$
$$q75 = (75^{th}\ quantile\ of\ x_i)_{i=1..k}$$

Then the bandwidth vector is

$$bandwidth_i = 0.9 * \min(std_i, \frac{q75 - q25_i}{1.349}) * n^{-0.2} \qquad \text{for i=1..k}$$

Empirical Bootstrap

Purpose: This routine uses the empirical distribution of the sample to draw bootstrap pseudo-samples, which are then analyzed using a user-specified core procedure (see Efron and Tibshirani, 1993). The routine in *Bounds* is a modification of the existing MatLab bootstrap procedure, *bootstrp*. The routine in *Bounds* differs from *bootstrp* in two aspects. First, the routine can distinguish between sample data that should be re-sampled and input arguments that should be kept invariant in each bootstrap iteration (see below). Second, the routine is designed to accept a Matlab structure from the function simulated and to break it into a vector.

Usage: [bootstat,bootsam] = *empirical*(nboot, procedure, data, parameters,...)
nboot - number of bootstrap iterations, a positive integer.
procedure - the *Bounds* procedure to be bootstrapped
data – an (n x k) matrix of sample data. The empirical distribution of these data form the basis for bootstrap re-sampling.
parameters- a matrix or vector or scalar of parameters used in "procedure."
bootstat – an (nboot x p) matrix of the procedure's output on the nboot iterations, where p is the length of the procedure's output vector
bootsam – an (n x nboot) matrix of integers. The (i, j) element of bootsam specifies the row of "data" that forms pseudo-observation i in bootstrap iteration j. (see explanation below).

Method: The empirical distribution of the data is used to draw nboot pseudo-samples. The specified procedure is run on each pseudo-sample, the output being stored in bootstat. The matrix bootsam logs the generation of pseudo-observations from the real data file "data" and so permits reconstruction of a set of bootstrap samples after *empirical* has been run, should further analysis of these samples be desired.

The routine distinguishes two kinds of inputs. The sample data are stored in "data" while parameters of the procedure are stored in "parameters." For example, to produce a bootstrap sampling distribution of the *kern* estimate, write:

empirical(25, '*kern*', y, x, h, '*gau*');

The sample data (y, x) will be used to create 25 pseudo samples, while (h, '*gau*') will be transferred to *kern* as the required bandwidth and kernel parameters.

How does the bootstrap routine know what constitutes the sample data and what constitutes the parameters? The routine interprets the first variable following the specified procedure to be a data variable and interpret the number of rows in this variable to be the sample size. The routine then interprets any other variable having the same number of rows to be a data variable and interprets variables with different numbers of rows to be parameters. Thus, in the above example, the (n x 1) vector y is the first variable following the specified procedure '*kern*.' The variable x has n rows and so is interpreted to be a data variable along with y. The variables h and '*gau*' do not have n rows and so are interpreted as parameters.

5. Core Procedures for Analysis of Treatment Response

The framework/notation for the analysis of treatment response used in this section follows Manski (1995, 1997). There is a population J of persons. Each member j of population J has observable covariates $x_j \in X$ and a response function $y_j(\cdot): T \to Y$ mapping the mutually exclusive and exhaustive *treatments* $t \in T$ into *outcomes* $y_j(t) \in Y$. Person j has a realized treatment $z_j \in T$ and a realized outcome $y_j = y_j(z_j)$, both of which are observable. The latent outcomes $y_j(t), t \neq z_j$ are not observable.

An empirical researcher learns the distribution P(x, z, y) of covariates, realized treatments, and realized outcomes by observing a random sample of the population. The researcher's problem is to combine this empirical evidence with assumptions in order to learn about the distribution of response functions within the subpopulation defined by a specified value of the covariates x. In particular, the researcher may want to learn an average treatment effect of the form E[y(t)| x] – E[y(s)|x], where s and t are specified treatments.

With one exception, all of the core procedures for the analysis of treatment response assume that there are two treatments, coded as 0 and 1. The exception is procedure *monotone*, which permits treatments to be real-valued.

Worst-Case Bounds

Purpose: This procedure estimates the worst-case bounds introduced in Manski (1990). The outcome variable y is assumed bounded. Before using this procedure, one should scale y so that its lower bound equals zero and upper bound equals one.

Usage: [results] = *treatment*(y, z, x, x_0, cont, h, 'kernel')

y- outcome data, an (n x 1) vector of real numbers, n being the sample size

z- treatment data, an (n x 1) vector of binary variables

x- covariate data, an (n x k) matrix of real numbers

x_0- covariate value defining the subpopulation of interest, a (1 x k) vector

cont – a scalar equal to 1 if nonparametric smoothing is to be performed and 0 if not. (see below).

h - scalar or (1 x k) vector, bandwidth to be used if cont = 1 (optional).

kernel - string indicating the kernel function to be used if cont = 1 (optional).

Method: For each treatment t, the worst-case bounds on E[y(1)|x] and E[y(0)|x] are

$$\begin{aligned} &E[y(1)\mid x, z=1]\Pr(z=1\mid x) \\ &\quad \le E[y(1)\mid x] \le \\ &E[y(1)\mid x, z=1]\Pr(z=1\mid x)+\Pr(z=0\mid x) \end{aligned}$$

and

$$\begin{aligned} &E[y(0)\mid x, z=0]\Pr(z=0\mid x) \\ &\quad \le E[y(0)\mid x] \le \\ &E[y(0)\mid x, z=0]\Pr(z=0\mid x)+\Pr(z=1\mid x) \end{aligned}$$

The resulting bound on the average treatment effect is

$$\begin{aligned} &E[y(1)\mid x, z=1]\Pr(z=1\mid x)-E[y(0)\mid x, z=0]\Pr(z=0\mid x)+\Pr(z=1\mid x) \\ &\quad \le E[y(1)\mid x]-E[y(0)\mid x] \le \\ &E[y(1)\mid x, z=1]\Pr(z=1\mid x)+\Pr(z=0\mid x)-E[y(0)\mid x, z=0]\Pr(z=0\mid x) \end{aligned}$$

The procedure uses elementary routine *kern* to estimate these bounds if the parameter "cont" is set equal to zero, and uses cell means otherwise. In the former case, the user may input the bandwidth and kernel to be used. If no values are input, the defaults are employed (see the section on *kern*). The procedure outputs a Matlab structure with the following fields (in each case, the output is the estimate of the quantity specified):

- *result.prop0* = $\Pr(z = 0 \mid x = x_0)$
- *result.prop1* = $\Pr(z = 1 \mid x = x_0)$
- *result.yhat0* = $E[y(0) \mid x = x_0, z = 0]$
- *result.yhat1* = $E[y(1) \mid x = x_0, z = 1]$
- *result.bound0L* = lower bound on E[y(0)|x]
- *result.bound0U* =upper bound on E[y(0)|x]
- *result.bound1L* = lower bound on E[y(1)|x]
- *result.bound1U* = upper bound on E[y(1)|x]
- *result.treatL* = lower bound on treatment effect
- *result.treatU* = upper bound on treatment effect

Examples: r = *treatment*(y, z, x, [12 12], 0);
r = *treatment*(y, z, x, [12,12],1, [2.3 1.8]);
r = *treatment*(y, z, x, [12,12], 1, [2.3 1.8], '*gau*')

In all cases, the value of x defining the subpopulation of interest is [12 12]. In the first case, x is discrete and cell means are used to compute the estimates. In the second case, nonparametric smoothing is used with bandwidth [2.3 1.8] and the default kernel. In the third case, the Gaussian kernel is used instead of the default Epanechnikov kernel.

Estimates Assuming Exogenous Treatment Selection

Purpose: This procedure estimates the treatment effect under the exogenous selection assumption.

Usage: result = *exogenous*(y, z, x, x_0, cont, h, kernel)
Variables (y, z, x, x_0, cont, h, kernel) are as in the *treatment* procedure.

Method: Under the exogenous selection assumption, the following holds for each treatment t:

$$E[y(1) \mid x = x0, z = 0] = E[y(1) \mid x = x0, z = 1]$$
$$E[y(0) \mid x = x0, z = 0] = E[y(0) \mid x = x0, z = 1]$$

The assumption, which is not testable, implies that

$$E[y(1) \mid x = x0, z = 0] - E[y(0) \mid x = x0, z = 1] = E(y \mid x = x0, z = 1) - E(y \mid x = x0, z = 0)$$

Thus the treatment effect is identified.

The procedure outputs a Matlab structure with the following fields:

- *result.prop0* = $\Pr(z = 0 \mid x = x_0)$
- *result.prop1* = $\Pr(z = 1 \mid x = x_0)$
- *result.yhat0* = $E[y(0) \mid x = x_0, z = 0]$
- *result.yhat1* = $E[y(1) \mid x = x_0, z = 1]$
- *result.treat* = $E[y(1) \mid x = x_0, z = 1] - E[y(1) \mid x = x_0, z = 0]$

Examples: r = *exogenous*(y, z, x, [12 12], 0);
r = *exogenous*(y, z, x, [12,12],1, [2.3 1.8]);
r = *exogenous*(y, z, x, [12,12], 1, [2.3 1.8], '*gau*')

In all cases, the value of x defining the sub-population of interest is [12 12]. In the first case, x is discrete and cell means are used to compute the estimates. In the second case, nonparametric smoothing is used with bandwidth [2.3 1.8] and the default kernel. In the third case, the Gaussian kernel is used instead of the default Epanechnikov kernel.

Instrumental Variable (IV) Bounds

Purpose: This procedure estimates the instrumental variable bound developed in Manski (1990, 1994). The outcome variable y is assumed bounded. Before using this procedure, one should scale y so that its lower bound equals zero and upper bound equals one.

Usage: results = *iv*(y, z, w, v, w_0, grid, cont, h, kernel)

Variables (y, z, cont, h, kernel) are as in procedure *treatment*

v – instrumental variable data, an (n x k) matrix of real numbers

w– covariates not used as instruments, an (n x p) matrix of real numbers

w_0- covariate value defining the sub-population of interest, an (1 x p) vector

grid – an optional (m x k) real matrix specifying m values of the instrumental variable v.

Method: The IV bound on E[y(1)|w] is

$$\sup_v \{E(y \mid w = w0, V = v, z = 1)\Pr(z = 1 \mid w = w0, V = v)\}$$
$$\leq E[y(1) \mid w] \leq$$
$$\inf_v \{E(y \mid w = w0, V = v, z = 1)\Pr(z = 1 \mid w = w0, V = v) + \Pr(z = 0 \mid w = w0, V = v)\}$$

The IV bound on E[y(0)|w] is defined analogously. The lower (upper) bound on the treatment effect is the lower (upper) bound on E[y(1)|w] minus the upper (lower) bound on E[y(0)|w].

If "grid" is not specified, the procedure performs the above sup and inf operations over all values of v that occur in the sample. If "grid" is specified, the procedure performs the sup and inf on the m points specified in "grid."

The program estimates a Matlab structure with these fields:

- *result.LB0* = lower bound on E[y(0)|w = w_0]
- *result.UB0* = upper bound on E[y(0)|w = w_0]
- *result.LB1* = lower bound on E[y(1)|w = w_0]
- *result.UB1* = upper bound on E[y(1)|w = w_0]
- *results.treatLB* - lower bound on E[y(1)|w = w_0] – E[y(0)|w = w_0].
- *results.treatUB* - upper bound on E[y(1)|w = w_0] – E[y(0)|w = w_0].

Monotone Treatment Response Bounds

Purpose: This procedure estimates the monotone and concave-increasing treatment response bounds developed in Manski (1997). The outcome variable need not be bounded. If only monotonicity is assumed, the treatment may be real-valued. If response is assumed to be increasing-concave, the treatment is assumed to be bounded from below, with t = 0 being the smallest possible value.

The current version of the procedure estimates the bound on E[y(t)|x] for a specified treatment t. Under construction is an extension of the procedure to estimate bounds on treatment effects E[y(t)|x] - E[y(s)|x] for specified treatments t and s.

Usage: results = *monotone*(y, z, x, t, x_0, k_0, k_1, cont, h, kernel, inc, conv);
Variables (y, x, x_0, cont, kernel) are as in *treatment*.
z – the (n x 1) vector of realized real-valued treatments
t – the treatment value of interest
k_0 – lower bound on the outcome y; if y is unbounded from below, enter *–Inf*.
k_1– upper bound on the outcome y; if y is unbounded from above, enter *Inf*.
h – bandwidths; may be a scalar or a (1 x k) vector. Enter 0 to use Silverman's rule of thumb.
inc – 1 if the outcome is assumed increasing with the treatment, 0 if it is decreasing.
conv – 1 if the outcome is assumed concave-increasing in the treatment. Do not enter a value otherwise.

Method: Consider the case in which the outcome is assumed increasing in the treatment. The bound on E[y(t)|x = x_0] is

$$K_0 P(t < z) + E(y \mid t \geq z) \cdot P(t \geq z) \leq E[y(t)] \leq K_1 P(t > z) + E(y \mid t \leq z) \cdot P(t \leq z)$$

If the outcome is assumed to be a concave-increasing function of the treatment, the bound is

$$E(yt/z \mid t<z)\cdot P(t<z)+E(y \mid t\geq z)P(t\geq z)\leq E[y(t)]$$

$$\leq E(yt/z \mid t>z)\cdot P(t>z)+E(y \mid t\leq z)\cdot P(t\leq z)$$

The program outputs a Matlab structure with these fields:

- *results.PZ0* = The estimate for Pr(z < t).
- *results.PZ1* = The estimate for Pr(z ≥ t).
- *results.LB* = The estimate for the lower bound of E[y(t)|x = x_0].
- *results.UB* = The estimate for the upper bound of E[y(t)|x = x_0].

6. Core Procedures for Analysis of Regressions with Missing Data

The framework/notation used in this section follows Manski (1989) and Horowitz and Manski (1998). There is a population J of persons. Each member j of population J has observable covariates x_j 0 X and a bounded outcome y_j. Outcomes should be scaled so as to have lower bound 0 and upper bound 1. The researcher wants to learn the conditional expectation E(y|x). The binary variable z_j takes the value 1 if (y_j, x_j) is observed and 0 if data are missing. The nature of the missing data differs across the procedures below.

Censoring of outcome variables

Purpose: This procedure estimates the worst-case bound when some outcome data are missing, but covariate data are always observed. See Manski (1989).

Usage: results = *outcen*(y, z, x, x_0, cont, h, kernel);

Variables (x, x_0, cont, h, kernel) are as in procedure *treatment*.

y – outcome data, an (n x 1) vector. In those observations where y is not observed, any value may be entered.

z = 1 if y is observed, 0 otherwise.

Method: The bound on E(y|x) is

$$E(y \mid x, z=1)\Pr(z=1 \mid X) \leq E(y \mid x) \leq E(y \mid x, z=1)\Pr(z=1 \mid x) + \Pr(z=0 \mid x)$$

The program outputs a Matlab structure with these fields:

- *results.prop0* = Pr(z = 0| x = x_0); probability of missing data conditional on x = x_0
- *results.prop1* = Pr(z = 1|x = x_0); probability of observing y, conditional on x = x_0
- *results.yhat1* = E(y|x = x_0, z = 1), estimate of E(y|x) under assumption of exogenous nonresponse
- *Results.boundL* = lower bound on E(y|x = x_0)
- *Results.boundU* = upper bound on E(y|x = x_0)

Joint censoring of Outcomes and Covariates

Purpose: This procedure estimates the worst-case bound when some (outcome, covariate) data are jointly missing. See Horowitz and Manski (1998).

Usage: results = *jointcen*(y, z, x, xl, xu);

y – outcome data, an (n x 1) vector. In those observations where (y, x) is not observed, any value may be entered.

x – covariate data, an (n x k) matrix. In those observations where (y, x) is not observed, any value may be entered.

z = 1 if (y, x) is observed, 0 otherwise.

xl - (1xk) vector, lower bound of the interval of interest.

xu - (1xk) vector, upper bound of the interval of interest.

Method: The worst-case bound under joint censoring has the same form as the one under outcome censoring, except that P(z = 1|x) is replaced by *the effective response probability*

$$Pe(z=1 \mid x_l \le x \le x_u) = \frac{\Pr(x_l \le x \le x_u \mid z=1)\cdot \Pr(z=1)}{\Pr(x_l \le x \le x_u \mid z=1)\cdot \Pr(z=1) + \Pr(z=0)}$$

The cases of continuous and discrete covariates are treated in same way in the *jointcen* procedure. If the covariates are discrete and the probability that x = x0 is positive, then the user may choose xl = xu = x0. If the covariates are continuous, then the user has to choose xl < xu in order to get a positive Pe. In both cases the probability $P(Y \mid x_l \le x \le x_u, z=1)$ is calculated using cell means (and the same when z = 0).

The program outputs a Matlab structure with these fields:

- *results.prop* = the effective response probability $P_e(z = 1|x_l \le x \le x_u)$
- *results.yhatL* = lower bound on $E(y|x_l \le x \le x_u)$
- *results.yhatU* = lower bound on $E(y|x_l \le x \le x_u)$

References

Efron, B. and R. Tibshirani (1993), Introduction to the Bootstrap, London: Chapman & Hall.

Hardle, W. (1990), Applied Nonparametric Regression Analysis, New York: Cambridge University Press.

Horowitz, J. and C. Manski (1998), "Censoring of Outcomes and Regressors due to Survey Nonresponse: Identification and Estimation Using Weights and Imputations," Journal of Econometrics, 84, 37-58.

Horowitz, J. and C. Manski (2000), "Nonparametric Analysis of Randomized Experiments With Missing Covariate and Outcome Data," Journal of the American Statistical Association, forthcoming.

Manski, C. (1989), "Anatomy of the Selection Problem," Journal of Human Resources, 24, 343-360.

Manski, C. (1990), "Nonparametric Bounds on Treatment Effects," American Economic Review Papers and Proceedings, 80, 319-323.

Manski, C. (1994), "The Selection Problem," in C. Sims (editor), Advances in Econometrics, Sixth World Congress, Cambridge, UK: Cambridge University Press.

Manski, C. (1995), Identification Problems in the Social Sciences, Harvard University Press.

Manski, C. (1997), "Monotone Treatment Response," Econometrica, 65, 1311 - 1334.

Manski, C. and J. Pepper (2000), "Monotone Instrumental Variables: With an Application to the Returns to Schooling," Econometrica, forthcoming.

Silverman, B. (1986), Density Estimation for Statistics and Data Analysis, London: Chapman & Hall.