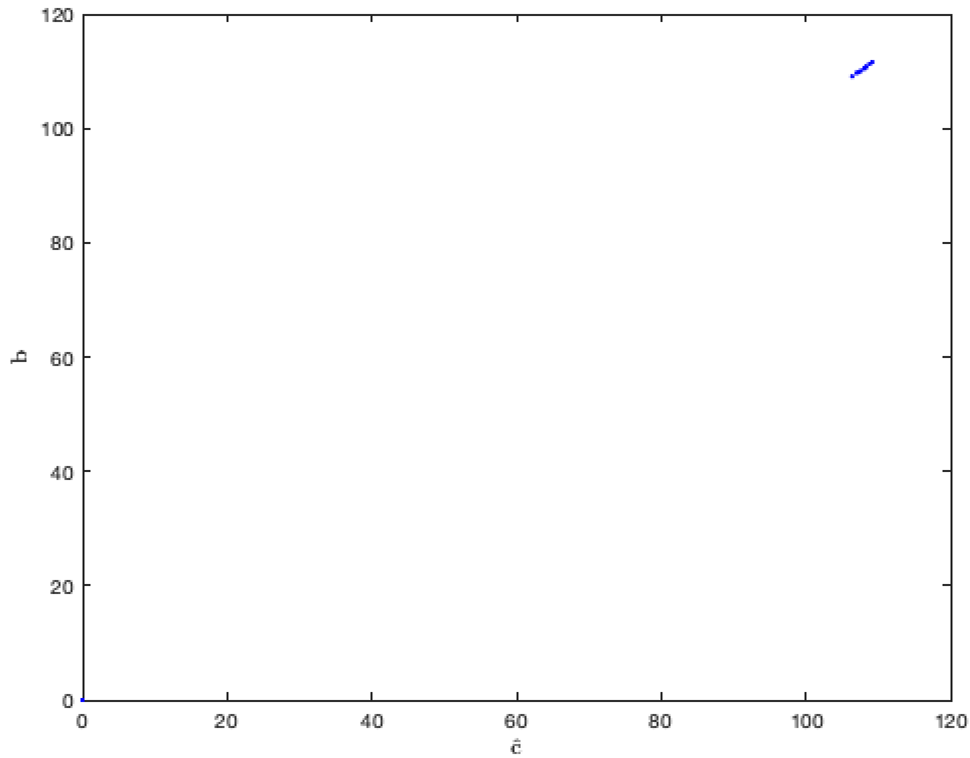
```
% Code for "Validate Structural Analysis"

% Reference:
% "Brent R. Hickman and Timothy P. Hubbard (2015). Replacing Sample
% Trimming with Boundary Correction in Nonparametric Estimation of
% First-Price Auctions. Journal of Applied Econometrics"

clear all
clc

% parameters for figure properties
lwidth = 2;
fsize = 14;
set(0,'defaulttextinterpreter','latex','Defaulttextfontsize',fsize);

% use hardle bandwidth transformation constant for non-gaussian kernel
usehardle = 1;

% indicator to turn on plotting
plotind = 0;

% indicator to turn on simulation
simulind = 1;

% choice of kernel---everything else is automated
kernel = 'triweight';

% b0 is needed for h0 bandwidth---see KZ (2008) bottom of page 501 and
 the
% K0 term in (4.9) of ZK (1998) which is K0 = 6 + 18t + 12t^2, t
\in[-1,0];
% this will compute b0 for any kernel choice provided by user
num_t1 = quad(@evalkspdf_num,-1,1,1e-14,[],kernel);
num_t2 = 0 - (-144/5 + 432/4 - 468/3 + 216/2 - 36);
denom_t1 = 0 - (-2 + 18/4 - 12/5);
denom_t2 = quad(@evalkspdf_denom,-1,1,1e-14,[],kernel);
b0 = ((num_t1^2*num_t2)/(denom_t1^2*denom_t2))^(1/5);

data = readtable('../data/DataProcessed.csv');
data.PercentOfEstimates = data.PercentOfEstimates .* 100;

% parameters
v = [];
bids = [];
vmin = 999;
vmax = -999;
participant_count = unique(data.ParticipantsCount);


% % step 1: recover pseudo-values using boundary correction on left
 boundary
for i = 1:length(participant_count)
```

```
    N = participant_count(i); % number of participants
    data_subset = data(data.ParticipantsCount == N, :);
    b = data_subset.PercentOfEstimates;

    if length(b) < 32
        continue
    end

    bmax = max(b);
    bmin = min(b);
    [gB_bc,hb_l] = kspdf_bc(b,kernel,b,b0,usehardle);
    % right-boundary correction: reflect bids over zero and do left-
boundary
    % correction
    [gB_bc_right,hb_r] = kspdf_bc(-b,kernel,-b,b0,usehardle);
    bind = find(b >= bmax - hb_r);
    gB_bc(bind) = gB_bc_right(bind);
    GB = kscdf(b,'edf',b);
    v_bc = b - (gB_bc)./((1-GB)*(N - 1));

    v = vertcat(v, v_bc);
    bids = vertcat(bids, b);

end

v_bc = v;


% remove inf rows
inf_idx = find(isinf(v_bc));
v_bc(inf_idx) = [];
bids(inf_idx) = [];

vmax_bc = max(v_bc);


% optional valuation points to evaluate each estimator at
neval = 1000;
evalpts = linspace(min(v_bc),max(v_bc),neval);

% step 2: recover valuation densities using boundary correction
[fV_bc,hv_l] = kspdf_bc(v_bc,kernel,evalpts,b0,usehardle);
% right-boundary correction: reflect pseudo-valuations over zero and
 do
% left-boundary correction
[fV_bc_right,hv_r] = kspdf_bc(-v_bc,kernel,-evalpts,b0,usehardle);
vind = find(evalpts >= vmax_bc - hv_r);
fV_bc(vind) = fV_bc_right(vind);

FV = kscdf(v_bc, 'edf', evalpts);

if plotind == 1
    % scatter plot of bid function
    figure
```

```
    set(gcf,'DefaultLineLineWidth',lwidth)
    set(gca,'FontSize',fsize)
    scatter(v_bc, bids,'.b')
    xlabel('$\mathbf{\hat{c}}$')
 ylabel('$\mathbf{b}$')
    box on

    % bid density
%     figure
%     set(gcf,'DefaultLineLineWidth',lwidth)
%     set(gca,'FontSize',fsize)
%     plot(evalpts_b,gB_bc_global)
%  xlabel('$\mathbf{b}$')
%  ylabel('$\mathbf{\hat{G}_B(b)}$')
%  box on

    % valuation density
    figure
    set(gcf,'DefaultLineLineWidth',lwidth)
    set(gca,'FontSize',fsize)
    plot(evalpts,fV_bc)
    xlabel('$\mathbf{\hat{c}}$')
    ylabel('$\mathbf{\hat{f}_C(c)}$')
    box on

    % valuation distribution
    figure
    set(gcf, 'DefaultLineLineWidth', lwidth)
    set(gca, 'FontSize', fsize)
    plot(evalpts, FV)
    xlabel('$\mathbf{\hat{c}}$')
    ylabel('$\mathbf{\hat{F}_C(c)}$')
    box on
end


evalpts = linspace(min(v_bc), max(v_bc), max(v_bc)-min(v_bc));
FV = kscdf(v_bc, 'edf', evalpts);

if simulind == 1
    N = 16;
    b = zeros(N, 1);
    c = zeros(N, 1);
    run = 1000;

    % not vectorized here because we may need to draw a different
 random number
    for i = 1:run
        for j = 1:10

            while true
                rand_number = rand;
                idx = find(rand_number == FV);
```

```
                if isempty(idx)
                    [minValue, closestIdx] = min(abs(FV -
 rand_number));
                    cost = evalpts(closestIdx);

                    if length(FV(closestIdx:end)) < 2 % need at least
 two points to evaluate integral
                        continue
                    end

                    b(j) = b(j) + cost + ...
                        trapz(evalpts(closestIdx:end), (1-
FV(closestIdx:end)).^(N-1)) / (1-FV(closestIdx))^(N-1);
                else
                    cost = mean(evalpts(idx));

                    if length(FV(idx(1):end)) < 2 % need at least two
 points to evaluate integral
                        continue
                    end

                    b(j) = b(j) + cost + ...
                        trapz(evalpts(idx(1):end), (1-
FV(idx(1):end)).^(N-1)) / (1-prob(i, j))^(N-1);
                end
                c(j) = c(j) + cost;

                break
            end
        end
    end

    b = b ./ run;
    c = c ./ run;

    figure
    set(gcf,'DefaultLineLineWidth',lwidth)
    set(gca,'FontSize',fsize)
    scatter(c, b,'.b')
    xlabel('$\mathbf{\hat{c}}$')
    ylabel('$\mathbf{b}$')
    box on
end
```

Warning: Variable names were modified to make them valid MATLAB identifiers. The original names are saved in the VariableDescriptions property.



Published with MATLAB® R2018b