Appendix A
BaseLocator®

We have operationalized the cartel screen that has been developed in Chapter 5 in a simple topographic detection routine in Delphi 5®.[1] Here we illustrate the logic of the algorithmic steps taken in the software. Below we give the main program with the key subroutines *TracingBP*, which traces the base using function *SumOfSquares*, and *EstimateLstatistic*, which calculates the value of *LoC*.[2]

A.1 Steps to Trace the Base

Input is a structured data file (in Notepad) that has a column of individual mill locations, and a column of individual customer project locations with the volume of trade and the transaction price per project. Base tracing consists of three main steps: data sorting, base tracing, and calculation of the *LoC*-measure.

In the first step, the data are sorted. All transaction data (P_i, q_i) are grouped by base group, using the information on project site locations. Those combinations of project locations that are aligned are disregarded as not independent—but this is a rare occasion. All base groups with less independent observations than the number of unknowns, which is four in the system developed in the text, are ignored—this small sample problem would normally not need to appear. What remains is N sets of independent observations, $N \leq J$.

[1]Excellent programming assistance has been provided by Eelko Ubels.

[2]The other subroutines called in the program are less insightful and lenghty. They are available upon request from the author.

In the second step, each constructed set of observations $l = 1, \ldots N$ is used to recover $\widetilde{c}$, $\widetilde{F}$ and base location used for the base group considered, $\left(\widehat{a}_l, \widehat{b}_l\right)$. For this, the specification of $T\left(q_i, d_{li}\right)$ is crucial. The software would in principle allow for a variety of specifications of $T\left(q_i, d_{li}\right)$—for the user to choose from, or for the program to find the best fit amongst. In the present version, transportation costs are linear in distance and volume, and including a fixed component, as in equation (5.15) in the text.

In a bounded area—that we determine as the size of the convex hull area of customer projects locations, extended with twice that size in all directions—the program step-searches in a grid for the specification of $\left(\widetilde{c}, \widetilde{F}, \left(\widehat{a}_l, \widehat{b}_l\right)\right)$ that returns the lowest S value. To be computationally efficient, we first use the information of the partial first-order conditions to problem (5.19):

$$\begin{aligned}
\frac{\partial S}{\partial \widetilde{c}} &= -2\sum_{i=1}^{I_{G_v}} q_i \left(P_i - \widetilde{c} q_i - \widetilde{F} - \sqrt{\left(a_l - a_{x_i}\right)^2 + \left(b_l - b_{x_i}\right)^2}\right) = 0, \text{ and} \\
\frac{\partial S}{\partial \widetilde{F}} &= -2\sum_{i=1}^{I_{G_v}} \left(P_i - \widetilde{c} q_i - \widetilde{F} - \sqrt{\left(a_l - a_{x_i}\right)^2 + \left(b_l - b_{x_i}\right)^2}\right) = 0,
\end{aligned}$$

to obtain

$$\begin{aligned}
\widetilde{c} &= \frac{\sum_{i=1}^{I_{G_v}} q_i \left(P_i - \widetilde{F} - d_{li}\right)}{\sum_{i=1}^{I_{G_v}} q_i^2}, \text{ and} \\
\widetilde{F} &= \frac{\sum_{i=1}^{I_{G_v}} \left(P_i - \widetilde{c} q_i - d_{li}\right)}{I_{G_v}}.
\end{aligned}$$

where $d_{li} = \sqrt{\left(a_l - a_{x_i}\right)^2 + \left(b_l - b_{x_i}\right)^2}$.

Using the averages $\overline{P} = \frac{\sum_{i=1}^{I_{G_v}} P_i}{I_{G_v}}$, $\overline{q} = \frac{\sum_{i=1}^{I_{G_v}} q_i}{I_{G_v}}$ and $\overline{d}_l = \frac{\sum_{i=1}^{I_{G_v}} d_{li}}{I_{G_v}}$, some manipulation yields

$$\begin{aligned}
\widetilde{c} &= \frac{\frac{1}{I_{G_v}} \sum_{i=1}^{I_{G_v}} q_i \left(P_i - d_i\right) - \overline{q}\left(\overline{P} - \overline{d}_l\right)}{\frac{1}{I_{G_v}} \sum_{i=1}^{I_{G_v}} q_i^2 - \overline{q}^2}, \text{ and} \\
\widetilde{F} &= \overline{P} - \left(\frac{\frac{1}{I_{G_v}} \sum_{i=1}^{I_{G_v}} q_i \left(P_i - d_i\right) - \overline{q}\left(\overline{P} - \overline{d}_l\right)}{\frac{1}{I_{G_v}} \sum_{i=1}^{I_{G_v}} q_i^2 - \overline{q}^2}\right)\overline{q} - \overline{d}_l.
\end{aligned}$$

Plugging these expressions for $\widetilde{c}$ and $\widetilde{F}$ into the criterion function (5.19), we obtain

$$S = \sum_{k=1}^{I_{G_v}} \left(P_k - \overline{P} + \overline{d}_l - d_{lk} + \left[\overline{q} - q_k\right]\left(\frac{\frac{1}{I_{G_v}} \sum_{i=1}^{I_{G_v}} q_i \left(P_i - d_i\right) - \overline{q}\left(\overline{P} - \overline{d}_l\right)}{\frac{1}{I_{G_v}} \sum_{i=1}^{I_{G_v}} q_i^2 - \overline{q}^2}\right)\right)^2,$$

for which we are to find the value(s) for $\left(\widehat{a}_l, \widehat{b}_l\right)$ that return(s) the lowest S-value.

In the search area, the value of S is determined for each combination of (a, b). The program stores the S-value and overwrites it when further grid-point yields a lower value. This is our candidate basing point.

In the third step, the base locations found are translated into the *LoC*-measure. The program determines the convex hull of firm locations and its surface. It determines λ on the basis of the theoretical competitive mean base point and variance, using project and mill locations only. Since the data are sorted per base group, in competition each mill would be found only once. The theoretical competitive mean base point therefore is the unweighted mean of the mill locations. The program subsequently calculates the mean recovered base location and the 'distance spread-circle' around it. This returns the surface $S^{\mathcal{L}}$. The intersection of these two areas gives the *LoC*-measure, a number between zero and one.

As output, the program returns the name of data set used, the location l^*, which is referred to as the center of the convex hull for reference, the value of λ, the sample mean base, the sample mean variance, the parameters of the bid structure estimated (normalized on $t_d = 1$), and the value of the *LoC*-measure. High values of *LoC* are indicative of collusion, in particular when supported by a small sample variance.

A.2 Kernel of the Software

```
function SumOfSquares(const p,q:Vector; const x:Matrix; const ag,bg:integer):Vector;

  {uses criterion function to calculate sum of squares, marginal cost and fixed
cost
  for given basepoint candidate}
  var s_i,s_j,s_jt,s_jn,pm,qm,dm,h:extended;
   i,j,len:integer;
   d,bp,s:Vector;
  begin
   len:=Length(p);
   s_i:=0;
   pm:=Mean(p);
   qm:=Mean(q);
   SetLength(d,len);
   SetLength(bp,2);
   bp[0]:=ag; bp[1]:=bg;
   for i:=0 to len-1 do begin
   h:=0;
   for j:=0 to 1 do
   h:=h+Sqr(bp[j]-x[i][j]);
   d[i]:=Sqrt(h);
   end;
   dm:=Mean(d);
   s_jt:=0; s_jn:=0;
   for j:=0 to len-1 do begin
   s_jt:=s_jt+q[j]*(p[j]-d[j])-qm*(pm-dm);
   s_jn:=s_jn+Sqr(q[j])-Sqr(qm);
   end;
   s_j:=s_jt/s_jn;
   for i:=0 to len-1 do begin
   s_i:=s_i+Sqr(p[i]-pm+dm-d[i]+(qm-q[i])*s_j)
   end;
   SetLength(s,3);
   s[0]:=s_i;
   s[1]:=s_j;
   s[2]:=pm-(s[1]*qm)-dm;
   SumOfSquares:=s;
  end; {SumOfSquares}


  procedure TracingBP(const base:integer; const x:Matrix; const p,q:Vector; var
BP:Matrix;
   const i0,j1,i1,j0:integer);
  {determines location with lowest value of sum of squares}
  var step,len_BP,intm,a,b,a0,a1,b0,b1:integer;
   sum,c,fc:extended;
   sos:Vector;

  begin {TracingBP}
   step:=50;
   len_BP:=Length(BP);
   SetLength(BP,len_BP+1);
   SetLength(BP[len_BP],4);
   intm:=Max(j1-j0,i1-i0);
   a0:=i0-0*intm;
   a1:=i1+0*intm;
   b0:=j0-0*intm;
   b1:=j1+0*intm;
   sum:=SumOfSquares(p,q,x,a0,b0-1)[0];
   c:=SumOfSquares(p,q,x,a0,b0-1)[1];
   fc:=SumOfSquares(p,q,x,a0,b0-1)[2];
   SetLength(sos,3);
   for a:=a0 to a1 do begin
   if a mod step=0 then begin
   for b:=b0 to b1 do begin
   if b mod step=0 then begin
   sos:=SumOfSquares(p,q,x,a,b);
   if sos[0]<sum then begin
   sum:=sos[0];
   BP[len_BP][0]:=a;
   BP[len_BP][1]:=b;
   c:=sos[1];
   fc:=sos[2];
   BP[len_BP][2]:=c;
   BP[len_BP][3]:=fc;
   end;
   end;
   end;
   end;
   end;

  end; {TracingBP}

  procedure EstimateLstatistic(const h:Matrix; const sq,v:extended; const m:Vector;
out Ls:extended);
  {determines overlap of circle in hull}
  var i,j,i0,i1,j1,j0,ot,tot:integer;
   p:Vector;
  begin
   SetLength(p,2);
   MinimalRectangle(h,i0,i1,j1,j0);
   ot:=0;
   tot:=0;
   for i:=0 to 100 do begin
   for j:=0 to 100 do begin
   p[0]:=i0+i*(i1-i0)/100;
   p[1]:=j0+j*(j1-j0)/100;
   if PointInHull(h,p,sq) then begin
   tot:=tot+1;
   if PointInCircle(m,p,r) then begin
   ot:=ot+1;
   end;
   end;
   end;
   end;
   Ls:=1-ot/tot;
  end; {EstimateLstatistic}



  begin {main}
   LoadFileName(s);
   s_out:='LoCw1.txt';
   AssignFile(f,s_out);
   Rewrite(f);
```

```
 SetLength(res,2);
 for i:=0 to 1 do begin
 SetLength(res[i],201);
 end;
 for j:=0 to 200 do begin //read 201 files with standard errors on price
 for i:=0 to 1 do begin
 SetLength(res[i][j],10);
 end;
 for k:=0 to 9 do begin //averageing over 10 files with same error
 SetLength(arr_BP,0);
 Writeln(k,Chr(9),j);
 s1:=s+'-'+IntToStr(k)+'-'+IntToStr(j)+'-comp_n.txt';


LoadData(s1,BP,sd,int_c,int_F,int_I,int_J,arr_x,arr_y,arr_p,arr_q,all_x,all_p,all_q);

 thMean:=MeanVector(arr_y);
 ConvexHull(arr_y,arr_h,sqHull);
 len_x:=Length(arr_x);
 MinimalRectangle(all_x,west,east,north,south);
 for i:=0 to len_x-1 do begin
 if Length(arr_x[i])>3 then begin

TracingBP(i,arr_x[i],arr_p[i],arr_q[i],arr_BP,west,east,north,south);
 end;
 end;
 thSE:=SigmaBar(arr_y);
 ConvexHull(arr_y,arr_h,sqHull);
 lambda:=Labda(Radius(thMean,arr_h),thSE);
 r:=lambda*SigmaBar(arr_BP);
 SetLength(mC,2);
 mC:=MeanVector(arr_BP);
 EstimateLstatistic(arr_h,sqHull,r,mC,LStat);
 res[0][j][k]:=LStat;

 SetLength(arr_BP,0);
 s1:=s+'-'+IntToStr(k)+'-'+IntToStr(j)+'-col_n.txt';

LoadData(s1,BP,sd,int_c,int_F,int_I,int_J,arr_x,arr_y,arr_p,arr_q,all_x,all_p,all_q);

 len_x:=Length(arr_x);
 MinimalRectangle(all_x,west,east,north,south);
 for i:=0 to len_x-1 do begin
 if Length(arr_x[i])>3 then begin

TracingBP(i,arr_x[i],arr_p[i],arr_q[i],arr_BP,west,east,north,south);
```

```
 end;
 end;
 r:=lambda*SigmaBar(arr_BP);
 SetLength(mC,2);
 mC:=MeanVector(arr_BP);
 EstimateLstatistic(arr_h,sqHull,r,mC,LStat);
 res[1][j][k]:=LStat;
 end;Writeln(f,Mean(res[0][j]):4:2,Chr(9),Mean(res[1][j]):4:2,Chr(9),sd:4:0);


 end;
 Writeln('the end');

GetProfits(all_x,arr_y,BP,int_I,int_J,all_p,all_q,int_c,int_F,pi_col,pi_comp);

 for j:=0 to int_J-1 do begin
 Writeln(f,'firm ',j+1,Chr(9),pi_comp[j]:4:2,Chr(9),Chr(9),Chr(9),pi_col[j]:4:2);

 end;
 CloseFile(f);
 Readln;
end. {main}
```