Appendix II

Pascal program for the bilinear imputation

The program `bootsws` generates, from a dataset containing missing values, bootstrap replicas of the dataset and imputes them using the bilinear scheme introduced in Efron (1994) and adapted in Chapter 3. Details are in Serrat and Gómez (1995).

Name	:	**Short description of the procedures**
`llegir_dades()`	:	Reads the dataset
`llegir_par`	:	Reads parameters
`llegeix_noms`	:	Reads the file containing the target names of the bootstrap+imputed dataset
`mostrav(), mostram()`	:	Show a vector or matrix variable
`genera_replica`	:	Performs a bootstrap replication of the dataset
`indicadors`	:	Computes the response indicator matrix
`totals`	:	Computes the marginal counts
`imputacio`	:	Generates the imputed dataset
`txoleski`	:	Applies a Cholesky decomposition
`sol_sim_pos`	:	Solves a symmetric and positive definite linear system
`sol_tri_l()`	:	Solves a lower triangular linear system
`sol_tri_u()`	:	Solves an upper triangular linear system
`escriu_imputacio()`	:	Writes the imputed dataset on a file

```
program bootsws;

const
  num_reg=494;
  num_var=3;

type
  tdades=array[1..num_reg,1..num_var] of real;
  matriu=array[1..num_reg+num_var-1,1..num_reg+num_var-1] of real;
  vector=array[1..num_reg+num_var-1] of real;
  totpreg=array[1..num_reg] of real;
  totpvar=array[1..num_var] of real;
  nom_arxiu=packed array[1..12] of char;

var
  hessia, low, up:  matriu;
  tihessia,x,y,sol:  vector;
  num_files, num_colum, num_boots, cont, i,j,k:  integer;
  codimis:array[1..num_var] of real;
  dades,dad,dadesimp,inddad,indmis:  tdades;
  difsumcr:array[1..num_reg-1,1..num_var-1] of real;
  nomisreg,sumareg, mort, mortdad:totpreg;
  nomisvar,sumavar, mit, sd :totpvar;
  totnomis,totsuma:real;
  numtext:packed array[1..5]of char;
  nom:  array[1..2000] of nom_arxiu;

{*********************************************************************}
{****************** PROCEDIMENT "LLEGIR_DADES" (text) ************}
{*********************************************************************}

procedure llegir_dades(arxiu:  nom_arxiu);

var
  f:  text;
  i:  integer;

begin
  {$I-}
  reset(f,arxiu);
  {$I+}
  i:=1;
```

```
  for i:=1 to num_reg do
  readln(f,dades[i,1],dades[i,2],dades[i,3],mort[i]);
  close(f);
end;

{*********************************************************************}
{****************** PROCEDIMENT "ESCRIU_IMPUTACIO" ***************}
{*********************************************************************}

procedure escriu_imputacio(arxiu:nom_arxiu);

var
  g:  text;
  i,j:  integer;

begin
  {$I-}
  rewrite(g,arxiu);
  {$I+}
  for i:=1 to num_reg do
  begin
    for j:=1 to num_var do write(g, dadesimp[i,j]:10:4);
    writeln(g, mortdad[i]:10:4);
  end;
  close(g);
end;

{*********************************************************************}
{****************** PROCEDIMENT "LLEGIR_PAR" *********************}
{*********************************************************************}

procedure llegir_par;

var
  j :  integer;

begin
  for j:=1 to num_var do
  begin
    write('Variable ',j:4,' codi de missing --> ');
    readln(codimis[j]);
    write('Variable ',j:4,' Mitjana --> ');
```

```
    readln(mit[j]);
    write('Variable ',j:4,' Standar desv.  --> ');
    readln(sd[j]);
  end end;

{*********************************************************************}
{****************** PROCEDIMENT "INDICADORS" ***********************}
{*********************************************************************}

procedure indicadors;

var
  i:integer;

begin
  for i:=1 to num_reg do
  for j:=1 to num_var do
  if dad[i,j]=codimis[j] then
  begin
    inddad[i,j]:=0;
    indmis[i,j]:=1;
  end   else
  begin
    inddad[i,j]:=1;
    indmis[i,j]:=0;
  end;

  for i:=1 to num_reg-1 do
  for j:=1 to num_var-1 do
  difsumcr[i,j]:=inddad[i,j]+inddad[num_reg,num_var]-
         (inddad[i,num_var]+inddad[num_reg,j]);
end;

{*********************************************************************}
{****************** PROCEDIMENT "TOTALS" ***************************}
{*********************************************************************}

procedure totals;

var
  i,j :  integer;
```

```
begin
  for i:=1 to num_reg do
  begin
    nomisreg[i]:=0;
    sumareg[i]:=0;
  end;
  for j:=1 to num_var do
  begin
    nomisvar[j]:=0;
    sumavar[j]:=0;
  end;
  totnomis:=0;
totsuma:=0;

  for i:=1 to num_reg do
  for j:=1 to num_var do
  begin
    nomisreg[i]:=nomisreg[i]+inddad[i,j];
    nomisvar[j]:=nomisvar[j]+inddad[i,j];
    totnomis:=totnomis+inddad[i,j];
    sumareg[i]:=sumareg[i]+dad[i,j]*inddad[i,j];
    sumavar[j]:=sumavar[j]+dad[i,j]*inddad[i,j];
    totsuma:=totsuma+dad[i,j]*inddad[i,j];
  end;
end;

{*********************************************************************}
{******************* PROCEDIMENT "SOL_TRI_L" **********************}
{*********************************************************************}

procedure sol_tri_l(var a:matriu; var b, sol:vector);

var
  i,j:integer;

begin
  for i:=1 to num_files do
  begin
    sol[i]:=b[i];
    for j:=1 to i-1 do sol[i]:=sol[i]-a[i,j]*sol[j];
    sol[i]:=sol[i]/a[i,i];
  end;
```

```
end;

{*********************************************************************}
{******************* PROCEDIMENT "SOL_TRI_U" ***********************}
{*********************************************************************}

procedure sol_tri_u(var a:matriu; var b:vector; var sol:vector);

var
  i,j:integer;

begin
  for i:=num_files downto 1 do
  begin
    sol[i]:=b[i];
    for j:=i+1 to num_files do sol[i]:=sol[i]-a[i,j]*sol[j];
    sol[i]:=sol[i]/a[i,i];
  end;
end;

{*********************************************************************}
{******************* PROCEDIMENT "TXOLESKI" **************************}
{*********************************************************************}

procedure txoleski;

var
  i,j,k:  integer;

begin
  for k:=1 to num_files do
  begin
    low[k,k]:=hessia[k,k];
    for i:=1 to k-1 do low[k,k]:=low[k,k]-low[k,i]*low[k,i];
    low[k,k]:=sqrt(low[k,k]);
    if low[k,k]=0 then writeln('ATENCIO ',k, low[k,k]);
    for i:=k+1 to num_files do
    begin
         low[i,k]:=hessia[i,k];
         for j:=1 to k-1 do low[i,k]:=low[i,k]-low[i,j]*low[k,j];
         low[i,k]:=low[i,k]/low[k,k];
    end;
```

```
  end;
  for i:=1 to num_files do for j:=1 to num_colum do up[i,j]:=low[j,i];
end;

{*********************************************************************}
{******************* PROCEDIMENT "MOSTRAM" ***************************}
{*********************************************************************}

procedure mostram(var a:tdades);

var
  i,j:  integer;

begin
  for i:=1 to num_reg do
  begin
    for j:=1 to num_var do write(a[i,j]:20:4);
    writeln;
  end;
  writeln;
end;

{*********************************************************************}
{******************* PROCEDIMENT "MOSTRAV" ***************************}
{*********************************************************************}

procedure mostrav(v:vector);

var
  i:  integer;

begin
  for i:=1 to num_files do write(v[i]:20:4);
  writeln;
end;

{*********************************************************************}
{******************* PROCEDIMENT "SOL_SIM_POS" ***********************}
{*********************************************************************}

procedure sol_sim_pos;
```

```
begin
  txoleski;
  sol_tri_l(low,tihessia,y);
  sol_tri_u(up,y,sol);
end;

{*********************************************************************}
{****************** PROCEDIMENT "IMPUTACIO" **********************}
{*********************************************************************}

procedure imputacio;

var
  i,j:  integer;
  nu:real;
  alfa :  array[1..num_reg] of real;
  beta :  array[1..num_var] of real ;

begin
  writeln('el nombre total de no-missings es :  ', totnomis);
  hessia[1,1]:=totnomis;
  for j:=2 to num_reg do hessia[1,j]:=nomisreg[j-1]-nomisreg[num_reg];
  for j:=num_reg+1 to num_files do
         hessia[1,j]:=nomisvar[j-num_reg]-nomisvar[num_var];

  for i:=2 to num_reg do hessia[i,i]:=nomisreg[i-1]+nomisreg[num_reg];
  for i:=num_reg+1 to num_files do
         hessia[i,i]:=nomisvar[i-num_reg]+nomisvar[num_var];

  for i:=2 to num_reg do
  for j:=i+1 to num_reg do hessia[i,j]:=nomisreg[num_reg];

  for i:=num_reg+1 to num_files do
  for j:=i+1 to num_files do hessia[i,j]:=nomisvar[num_var];

  for i:=2 to num_reg do
  for j:=num_reg+1 to num_files do hessia[i,j]:=difsumcr[i-1,j-num_reg];

  for i:=2 to num_files do
  for j:=1 to i-1 do
  hessia[i,j]:=hessia[j,i];
```

```
  tihessia[1]:=totsuma;
  for i:=2 to num_reg do tihessia[i]:=sumareg[i-1]-sumareg[num_reg];
  for i:=num_reg+1 to num_files do
         tihessia[i]:=sumavar[i-num_reg]-sumavar[num_var];

  sol_sim_pos;

  nu:=sol[1];
  alfa[num_reg]:=0;
  for i:=1 to num_reg-1 do
  begin
    alfa[i]:=sol[i+1];
    alfa[num_reg]:=alfa[num_reg]-alfa[i];
  end;
  beta[num_var]:=0;
  for j:=1 to num_var-1 do
  begin
    beta[j]:=sol[num_reg+j];
    beta[num_var]:=beta[num_var]-beta[j];
  end;

  writeln(nu:10:4);
  for i:=1 to num_reg do write(alfa[i]:10:4);
  writeln;
  for j:=1 to num_var do write(beta[j]:10:4);
  writeln;
  writeln(' ================================================ ');
  writeln;

  for i:=1 to num_reg do
  for j:=1 to num_var do
  dadesimp[i,j]:=dad[i,j]*inddad[i,j]+(nu+alfa[i]+beta[j])*indmis[i,j];

  for i:=1 to num_reg do
  begin
    for j:=1 to num_var do write(dadesimp[i,j]:10:4);
    writeln;
  end;
  writeln;
  writeln(' ================================================ ');
  writeln;
end;
```

```
{*********************************************************************}
{****************** PROCEDIMENT "GENERA_REPLICA" *********************}
{*********************************************************************}

procedure genera_replica;

var
  i,j, posprevia, posicio:  integer;
  numalea :  real;

begin
  for i:=1 to num_reg do
  begin
    numalea:=num_reg*random(num_reg);
    posprevia:=round(numalea);
    if numalea < posprevia then
         posicio:=posprevia-1 else posicio:=posprevia;
    for j:=1 to num_var do
    begin
         dad[i,j]:=dades[posicio,j];
         mortdad[i]:=mort[posicio];
    end;
  end;
end;

{*********************************************************************}
{****************** PROCEDIMENT "LLEGEIX_NOMS" ***********************}
{*********************************************************************}

procedure llegeix_noms;

var
  f:text;
  i:integer;

begin
  reset(f,'nomsnum.txt');
  for i:=1 to num_boots do readln(f,nom[i]);
  close(f);
end;
```

```
{*********************************************************************}
{****************** PROGRAMA PRINCIPAL ***************************}
{*********************************************************************}

begin
  num_files:=num_reg+num_var-1;
  num_colum:=num_files;
  llegir_dades('svcdppmo.dat');
  mostram(dades);
  writeln;
  llegir_par;

  for i:=1 to num_reg do for j:=1 to num_var do
    dades[i,j]:=(dades[i,j]-mit[j])/sd[j];

  write('Nombre de mostres Bootstrap --> ';
  readln(num_boots);
  llegeix_noms;
  for cont:=1 to num_boots do
  begin
    writeln('Generacio num -----> ',cont:4);
    genera_replica;
    indicadors;
    totals;
    imputacio;

    for i:=1 to num_reg do
    begin
         dadesimp[i,1]:=mitj[1]+dadesimp[i,1]*sd[1];
         dadesimp[i,2]:=mitj[2]+dadesimp[i,2]*sd[2];
         if dadesimp[i,2]<0 then dadesimp[i,2]:=1;
         if dadesimp[i,3]<0 then dadesimp[i,3]:=0 else
         if dadesimp[i,3]>0 then dadesimp[i,3]:=1 else
         if 245*round(random(i))>149 then dadesimp[i,3]:=1
              else dadesimp[i,3]:=0;
    end;

    escriu_imputacio(nom[cont]);
    writeln('OK');
  end;
end.
```