NISTIR 5723

Airborne Asbestos Method: Bootstrap Method for Determining the Uncertainty of Asbestos Concentration - Version 1.0

Shirley Turner
Robert L. Myklebust
Barbara B. Thorne
Stefan D. Leigh*
Eric B. Steel

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Surface and Microanalysis Science Division
Chemical Science and Technology Laboratory
Gaithersburg, MD 20899

*Statistical Engineering Division
Computing and Applied Mathematics Laboratory

NIST

NISTIR 5723

Airborne Asbestos Method: Bootstrap Method for Determining the Uncertainty of Asbestos Concentration - Version 1.0

Shirley Turner
Robert L. Myklebust
Barbara B. Thorne
Stefan D. Leigh*
Eric B. Steel

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Surface and Microanalysis Science Division
Chemical Science and Technology Laboratory
Gaithersburg, MD 20899

*Statistical Engineering Division
Computing and Applied Mathematics Laboratory

August 1996

U.S. DEPARTMENT OF COMMERCE
Michael Kantor, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology
NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Arati Prabhakar, Director

Acknowledgements

We thank Kelly Collins for assistance in testing the bootstrapping programs.

Preface

This Interagency Report (IR) is one of a series of IRs that will form the basis of a method for analysis of airborne asbestos by transmission electron microscopy. The form and style of the American Society for Testing and Materials (ASTM) was adopted as a standard format for this series of reports.

Shirley Turner or Eric Steel can be contacted about obtaining the programs on disk or from a planned world wide web site.

1. Scope

1.1 This test method describes a procedure for determining the component of uncertainty in the concentration of asbestos that is due to variation in sampling of a population of values (corresponding to the Type A evaluation of uncertainty described in TN 1297[1]). The component of uncertainty arising from systematic effects (Type B evaluation of uncertainty) is not addressed in this method.

1.2 The test method describes use of a bootstrapping procedure for determination of a 95% confidence interval for the concentration of asbestos deposited onto filters.

1.3 The test method is applied where the concentration of asbestos is determined by transmission electron microscopy.

1.4 This method for determination of a confidence interval is nonparametric, i.e., no probability distribution is assumed for the concentration of asbestos in air or on filters.

1.5 The method can be used to derive 95% confidence intervals for analyses of both indoor and outdoor sets of filters as described in the AHERA method.

1.6 The method can be used to derive confidence intervals for the number of structures per grid square, the number of structures per mm^2, the number of structures per cm^3 of air or for any other relevant unit.

1.7 The method cannot be applied to data sets in which all values are zero.

1.8 The method requires analysis of a minimum of four grid squares on each of five filters and analysis of an approximately equal number of grid squares per filter.

1.9 This method requires use of an Apple, PC (IBM or IBM clone) or a computer that can compile and execute the code in Appendix X4 and X5.

1.10 This method, though written specifically for asbestos, can be applied to other data sets containing at least 20 values.

2. Terminology

2.1 *Definitions:*

2.1.1 *bootstrapping*--a statistical procedure whereby a set of data is uniformly and randomly sampled with replacement. Statistical estimates are then determined from the randomly resampled data sets.

2.1.2 *grid square, grid opening*--an area on a grid used for analysis of asbestos by transmission electron microscopy.

2.1.3 *95% confidence interval*--a statistical interval derived from a random sample of a population that may enclose a value of interest, e.g., the mean of a population. If 95% confidence intervals are repeatedly calculated from many independent random samples of a population, the true value of interest will be contained within the confidence intervals 95% of the time.

2.2 *Description of Terms Specific to This Standard:*

2.2.1 *AHERA method*--method for analysis of asbestos by transmission electron microscopy developed by the Environmental Protection Agency[2] with subsequent modifications by the National Institute of Standards and Technology. This method was developed in response to the Asbestos Hazard Emergency Response Act (AHERA) of 1986.

---

[1]Taylor, B.N. and C.E. Kuyatt, 1994, NIST Technical Note 1297, 1994.

[2]Code of Federal Regulations, 1987, 52, No. 210, 41826-41905.

2.2.2 *bootstrap program*--the computer program (for either the Apple or PC) associated with this test method.

2.2.3 *input data set*--a data set containing the information to be input into the bootstrap program. The data set consists of one column containing an identification number or text for each grid opening analyzed and a second column containing the number (or concentration) of structures found on each grid opening.

2.2.4 *bootstrap data set*--a basic set of values chosen by uniform random resampling in the bootstrap program. In the bootstrap program, the size of the bootstrap data set is equal to the number of grid squares analyzed.

2.2.4.1 *Discussion*--The bootstrap program determines the average number of structures per grid square for each bootstrap data set and stores this value in an internal data file.

2.2.5 *number of iterations*--the number of bootstrap data sets generated by the program.

2.2.5.1 *Discussion*--The number of iterations is set at 6000 for this program.

2.2.6 *grand mean*--the average value of the mean values generated for the bootstrap data sets from a run.

2.2.7 *lower 95% confidence limit*--the lower value of the 95% confidence interval for the average number of structures generated for the bootstrap data sets from a run.

2.2.7.1 *Discussion*--In this program, the mean values generated for 6000 bootstrap data sets are sorted. The lower confidence limit is the mean value that is 150 values from the lowest value in the list.

2.2.8 *upper 95% confidence limit*--the upper value of the 95% confidence interval for the average number of structures generated for the bootstrap data sets from a run.

2.2.8.1 *Discussion*--In this program, the mean values generated for 6000 bootstrap data sets are sorted. The upper confidence limit is the mean value that is 150 values from the highest value in the list.

2.2.9 *run*--an execution of the program.

2.2.9.1 *Discussion*--The output from each run consists of the average number of structures in the input data set, and the grand mean, and lower and upper values of the average number of structures derived from the bootstrap data sets.

3. Significance and Use

3.1 The uncertainty derived by this method is useful for a general comparison of analyses obtained by different operators or laboratories on the same sampling area.

3.2 The uncertainty derived by this method provides an indication of how an analytical value compares with values set by government regulations.

3.3 The test method should be used for analytical values derived from the AHERA method. Where all values are zero, the confidence interval should be derived assuming a Poisson distribution.

3.4 The uncertainty generated by this method should be included in reports of analysis resulting from use of the AHERA method.

4. Procedure

4.1 Obtain an analysis of air samples using transmission electron microscopy as described in the AHERA method.

4.2 Create a file containing an input data set (see example input data sets for Apple computers in Figure X1.1 of Appendix X1 and for the PC in Figure X2.1 of Appendix X2). Two separate files should be created and run separately if there are both an inside and outside set of filters.

NOTE 1--The input file for the programs is an ASCII file.

4.2.1 The first three lines of the input data set contain header information of the laboratory's choosing.

4.2.2 The fourth line contains the units of the input data. Examples of units include: structures, structures/mm^2, structures/cm^3, etc.

4.2.3 The input for the data starts on the fifth line. There are two columns for the data set. The first column contains identification numbers or text for the grid openings analyzed and the second column contains the number or concentration of asbestos structures identified in each grid opening.

NOTE 2--The entries for the columns cannot exceed 8 characters. The columns can be space or tab delimited.

4.3 Run the bootstrap program. For information purposes, a general outline of the program is given in Appendix X3 and the source codes are given in Appendix X4 and Appendix X5. The Apple version is run by double clicking on the program icon or name and the PC version is run by typing the program name at the DOS prompt.

4.3.1 Input the file name for the input data set (see Figure X1.2 and Figure X2.2).

4.3.2 Wait while the program is performing the bootstrapping procedure until the statistics for the input and bootstrap data sets are displayed. The time required for the calculation varies depending on the type and speed of the computer used.

4.3.3 Respond to questions concerning saving the output data to a file (see Figures X1.3 and X2.3 for example output files).

4.4 Use a statement identical or similar to that given in the output data file for reporting the concentration determined using this method.

NOTE 3--The number of significant figures given by the bootstrapping program is larger than appropriate for most analyses. Laboratory personnel should determine the number of significant figures appropriate to report for the analysis.

5. Precision and Bias

5.1 *Precision*--Data has been collected only on the repeatability of determination of the 95% confidence interval. Data on reproducibilty will be collected later.

5.1.1 *Repeatability*--The repeatability of this test method is dependent upon the nature of the data set. The variation in the confidence limits obtained by repeating the bootstrap procedure 50 times is given for two examples below.

5.1.1.1 Confidence intervals were determined for the data set given in Appendix X1. For the lower 95% confidence limit, a value of 6.7 s/mm^2 was obtained for 31 of 50 bootstrap runs and a value of 10 s/mm^2 was obtained for the other 19 runs. For the upper 95% confidence limit, a value of 46.7 s/mm^2 was obtained 3 times and a value of 50.0 s/mm^2 was obtained 47 times.

5.1.1.2 Confidence intervals were determined for the data set given in Appendix X2. For the lower 95% confidence limit, values of 2.34, 2.36 and 2.38 st./grid square were generated 15, 34 and 1 time, respectively. For the upper 95% confidence limit, values of 3.40, 3.42 and 3.44 st./grid square were generated 21, 28 and 1 time, respectively.

5.2 *Bias*--The bias in the determination of confidence intervals from bootstrapping methods is discussed in Efron and Tibshirani (1993).

6. Keywords

6.1 asbestos; bootstrapping; transmission electron microscopy; confidence interval

Additional References

Efron, B.E. and R.J. Tibshirani (1993) An Introduction to the Bootstrap. Chapman and Hall, NY, NY.

Hahn, G.J. and W.Q. Meeker (1991) Statistical Intervals: A Guide for Practitioners. Wiley-Interscience, NY.

APPENDIX

X1. BOOTSTRAPPING PROGRAM RUN ON AN APPLE COMPUTER

Fig. X1.1 Input file for the bootstrapping program.

```
This is a test file for the bootstrap program.


structures/mm2
1  0
1  0
1  0
1  100
1  0
1  0
2  0
2  0
2  0
2  0
2  0
2  0
3  0
3  0
3  0
3  0
3  0
3  0
4  0
4  0
4  200
4  0
4  100
4  100
5  0
5  0
5  0
5  100
5  0
5  200
```

Fig. X1.2 On screen input and output obtained during a run of the Apple version of the bootstrapping program.

```
NIST data analysis program.
Hit Return and select a file to open.

Calculating bootstrap -


For input data set:
    Mean value                    =   26.6667

For bootstrap data sets:
    Grand mean value              =   26.5894
    Lower 95% confidence limit    =    6.6667
    Upper 95% confidence limit    =   50.0000


The following would be an appropriate way to state the results of
this calculation:

The mean value for this data set is   26.6667 structures/mm2.  The interval
from   6.6667 to   50.0000 structures/mm2 is a 95% confidence interval for the
mean value; i.e., we may assert with 95% confidence that the mean value
lies between   6.6667 and  50.0000 structures/mm2.

Hit Return and select a file for output or cancel.
```

Fig. X1.3 Output file for the Apple version of the bootstrapping program.

```
                example.out   -  6/13/95

Input data set =  example
Number of grid openings =         30

This is a test file for the bootstrap program.


structures/mm2

For input data set:
   Mean value                =     26.6667

For bootstrap data sets:
   Grand mean value          =      26.5894
    Lower 95% confidence limit  =    6.6667
    Upper 95% confidence limit  =   50.0000
```

The following would be an appropriate way to state the results of this calculation:

The mean value for this data set is 26.6667 structures/mm2. The interval from 6.6667 to 50.0000 structures/mm2 is a 95% confidence interval for the mean value; i.e., we may assert with 95% confidence that the mean value lies between 6.6667 and 50.0000 structures/mm2.

X2. BOOTSTRAPPING PROGRAM RUN ON A PC

Fig. X2.1 Input file for the bootstrapping program.

```
Example data set

st./grid square
1       1
1       0
1       2
1       0
1       2
1       1
1       3
1       0
1       1
1       0
2       4
2       0
2       2
2       6
2       4
2       1
2       3
2       6
2       5
2       4
3       4
3       3
3       3
3       4
3       2
3       2
3       2
3       1
3       3
3       6
4       4
4       6
4       2
4       0
4       2
4       5
4       1
4       6
4       2
4       1
5       4
5       4
5       3
5       4
5       6
5       1
5       5
5       6
5       2
5       5
```

Fig. X2.2 On screen input and output obtained during a run (PC version)

```
        NIST data analysis program

Input the file name: example3.dat

Calculating bootstrap -

For input data set:
    Mean value                  =    2.8800

For bootstrap data sets:
    Grand mean value            =    2.8765
    Lower 95% confidence limit =    2.3600
    Upper 95% confidence limit =    3.4000


The following would be an appropriate way to state the results of
this calculation:

The mean value for this data set is    2.8800 st./grid square.  The interval
from   2.3600 to    3.4000 st./grid square is a 95% confidence interval for the
mean value; i.e., we may assert with 95% confidence that the mean value
lies between    2.3600 and    3.4000 st./grid square.


Save results to a file? Y/N: y
Output to file: BootOut.txt?  Y/N: n
Enter output file name: example3.out
```

Fig. X2.3 Output file for the bootstrapping program (PC version)

```
                EXAMPLE3.OUT  -  6/13/95

Input data set = EXAMPLE3.DAT
Number of grid openings = 50

Example data set


st./grid square

For input data set:
    Mean value                    =    2.8800

For bootstrap data sets:
    Grand mean value              =    2.8765
    Lower 95% confidence limit    =    2.3600
    Upper 95% confidence limit    =    3.4000


The following would be an appropriate way to state the results of
this calculation:

The mean value for this data set is    2.8800 st./grid square.  The interval
from   2.3600 to    3.4000 st./grid square is a 95% confidence interval for t
mean value; i.e., we may assert with 95% confidence that the mean value
lies between    2.3600 and    3.4000 st./grid square.
```

X3. GENERAL OUTLINE OF THE BOOTSTRAPPING PROGRAM

1. Read the input data set
2. Determine the number of values (n) in the input data set
3. Calculate the mean value for the data in the input data set
4. Loop for 6000 iterations
   - Loop for n times
      -- Randomly choose a value from the input data set
      -- Put the value in a temporary bootstrap data set
      -- Replace the value in the input data set
      -- End loop
   - Determine the mean value of the temporary bootstrap data set
   - Put the mean value in a bootstrap array
   - End loop
5. Sort the mean values in the bootstrap array
6. Choose values that are 150 from the top and bottom of the bootstrap array for the lower and upper 95% confidence limits
7. Determine the grand mean of the values in the bootstrap array
8. Report the mean value for the input data set, the grand mean of the bootstrap values and the lower and upper 95% confidence limits.

X4. PROGRAM FOR BOOTSTRAPPING ON AN APPLE COMPUTER

```
PROGRAM BOOT_STRAP;

    USES    Types, QuickDraw, Packages, CursorCtl, ToolUtils,
                SANE, PasLibIntf, StandardFile, PackIntf, OSUtils;

CONST
  np = 6000;

TYPE
      RealArrayNP = ARRAY [1..np] OF real;
      IntArray50 = array [1..50] of real;
      fnamestr = STRING[63];

VAR
   out_data        : RealArrayNP;
   Ran2Iy          : longint;
   Ran2Ir          : ARRAY [1..97] OF longint;
   in_data         : IntArray50:    {structure values}
   temp_data       : IntArray50;
   n_dat, ii, jj : integer;
   i_ran           : longint;
   r_ran           : real;
      head_x           : ARRAY [1..3] OF str255;     {3 lines of header information}
      str1          : ARRAY [1..50] OF STRING[10];
      min_val, max_val       : IntArray50;
      mean_val  : IntArray50;
      mean_2000       : IntArray50;
      median_val      : IntArray50;
      ji, n_p         : integer;
      fndrInfo    : FInfo;
      errCode         : integer;
      runs, ir    : integer;
      tt              :     real;
      ave_tt          : real;
      units       : string[20];
      f_name          : fnamestr;
      dateInfo    : DateTimeRec;
      isec        : longint;

LABEL
      100;
  FUNCTION ItsNaN(theString: str255): boolean;
{function is a test to determine if ASCII characters are numbers}

  BEGIN
    ItsNaN := false;
     IF (ord(ClassExtended(Str2Num(theString)))) < 3 THEN ItsNaN := true;
  END;

FUNCTION read_file:   integer;
{function reads the input data file}

VAR
     myReply         : SFReply;
     myTypes         : SFTypeList;
     myErr           : integer;
```

```
    str             : Str255:
    theFile         : text:
    temp            : extended:
    ij,nt       : integer:
{   f_name          : fnamestr:    }
    p       : Point:
    ch              :CHAR:
    strx        : String[1];
    er_NaN          : boolean:

    BEGIN
        myTypes[0] := 'TEXT';
        ch := Chr(ord(9));
        strx := ch:
        (*myForm.Style := FixedDecimal:
        myForm.Digits := 4;*)
        p.h := 100;   p.v := 80;
        SFGetFile(p, 'Select File ', NIL, 1, myTypes, NIL, myReply);
        if myReply.good then
        BEGIN
            myerr := 0;
            f_name := myReply.fname;
            reset(theFile,f_name);
            readln(theFile,head_x[1]);    {read three header lines}
            readln(theFile,head_x[2]);
            readln(theFile,head_x[3]);
            readln(theFile,units);        {read the units for the values}
            (*readln(theFile.str);
            temp := Str2Num(str);
            n_dat := Num2Integer(temp);
            FOR ij := 1 to n_dat do BEGIN*)
            ij := 0;
            tt := 0;
            REPEAT

                readln(theFile,str);
                {read in the identification for the grid openings analyzed
                {maximum eight charactgers} and the value for the number or
                concentration of asbestos structures}
                ij := ij + 1;
            nt := Pos(strx, str) -1;
            str1[ij] := Copy(str, 1, nt);
            nt := nt + 1;
            Delete (str, 1, nt);
            er_NaN := ItsNaN(str);
            if er_NaN = false then begin
                temp := Str2Num(str);
                in_data[ij] := temp;
                    tt    := tt + in_data[ij];       {sum of data values}
                    (*Num2Dec(myForm, temp, in_data[ij]);*)
            end
            else myErr := 2;
            UNTIL (EOF(theFile) = true) OR (myerr = 2);
                n_dat := ij;        {number of values in the input data set}
                ave_tt := tt/ij;    {calculate the mean value for data in the
                                     input data set}
```

```
                    close(theFile);
                    {myErr := 0;}
            END
            ELSE
                  myErr := 1;
            read_file := myErr;
      END;

PROCEDURE write_xfile;      {procedure for writing the output file}
VAR
      my_Reply  : SFReply;
      myErr           : integer;
      f_nam           : fnamestr;
      the_File    : text;
      temp            : extended;
      ij, vol_num     : integer;
      myForm          : DecForm;
      str, str2   : DecStr;
      str3, str4  : DecStr;
      str5        : DecStr;
      str6        : Str255;
      p       : Point;

      BEGIN
            my_Reply.fType := 'TEXT';
            p.h := 100;   p.v := 80;
            getdatetime(isec);

            secs2Date(isec, dateInfo);

            f_nam := 'Bootruns.txt';
            SFPutFile( p,'Save an Output File', f_nam, NIL, my_Reply);
            if my_Reply.Good then BEGIN
                  f_nam := my_Reply.fname;
                  vol_num := my_Reply.VRefNum;
                  rewrite(the_File,f_nam);
{                 fndrInfo.fdcreator := 'XCEL';   }
                  fndrInfo.fdcreator := 'MSWD';
                  fndrInfo.fdtype := 'TEXT';
                  fndrInfo.fdFlags := 0{$100};
                  myErr := setfinfo(f_nam, vol_num, fndrInfo);

                  writeln(the_File,'                  ',f_nam,'    - ',dateInfo.month:2,'/',dateInfo.day:2,'/',dateInfo.year-
1900:2 );
                  writeln(the_File);
                  writeln(the_File,'Input data set =' ',chr(9),f_name);
                  writeln(the_File,'Number of grid openings = ',n_dat);
                  writeln(the_File);
                  writeln(the_File,head_x[1]);
                  writeln(the_File,head_x[2]);
                  writeln(the_File,head_x[3]);
                  writeln(the_File,units);
                  str6 := concat('Mean value', chr(9), 'Grand Mean', chr(9), 'Median value', chr(9), 'Lower
limit', chr(9), 'Upper limit');
            {     writeln(the_File,str6); }
                  myForm.Style := FixedDecimal;
```

```
                myForm.Digits := 3;
                FOR ij := 1 to runs DO begin
                      Num2str(myForm.  mean_val[ij],  str);
                      Num2str(myForm.  mean_2000[ij],  str2);
                      Num2str(myForm.  median_val[ij],  str3);
                      Num2str(myForm.  min_val[ij],  str4);
                      Num2str(myForm.  max_val[ij],  str5);

                      str6 := concat(str. chr(9), str2. chr(9), str3. chr(9), str4. chr(9), str5);
          {           writeln(the_File,str6);   }

            writeln(the_File);

      writeln (the_File,' For input data set: ');
         writeln (the_File,'     Mean value                  = ',chr(9),mean_val[ij]:9:4);
          writeln(the_File);
          writeln (the_File,' For bootstrap data sets: ');
         writeln (the_File,'     Grand mean value            = ',chr(9),mean_2000[ij]:9:4);
         writeln (the_File,'     Lower 95% confidence limit  = ',min_val[ij]:9:4);
         writeln (the_File,'     Upper 95% confidence limit  = ',max_val[ij]:9:4);
          writeln(the_File);
          writeln (the_File,' The following would be an appropriate way to state the ',
                 'results of');
         writeln (the_File,' this calculation: ');
         writeln(the_File);
         writeln(the_File,' The mean value for this data set is ',
                  mean_val[ij]:9:4,' ', units,'.  The interval');
         writeln (the_File,' from',min_val[ij]:9:4,
                ' to ',max_val[ij]:9:4,' ',units,' is a 95% confidence'
               ,' interval for the');
         writeln (the_File,' mean value; i.e., we may'
                ,' assert with 95% confidence that the true mean value ');
         writeln (the_File,' lies between ',min_val[ij]:9:4,' and ',max_val[ij]:9:4,' ',
             units,'.');
         writeln(the_File);

                END; {FOR ij := 1 to runs}
                close(the_File);
                myErr := FlushVol(NIL, vol_num);
          END;
    END;

FUNCTION ran2(VAR idum: longint): real;
{function to generate random numbers for the bootstrapping program}

CONST
   m = 714025;
  ia = 1366;
  ic = 150889;
   rm = 1.4005112e-6;
VAR
  j: integer;
BEGIN
  IF idum < 0 THEN BEGIN
     idum := (ic-idum) MOD m;
   FOR j := 1 TO 97 DO BEGIN
```

```
        idum := (ia*idum+ic) MOD m;
        Ran2Ir[j] := idum;
    END;
    idum := (ia*idum+ic) MOD m;
    Ran2Iy := idum;
  END;
  j := 1 + (97*Ran2Iy) DIV m;
 { IF (j > 97) OR (j < 1) THEN BEGIN
      writeln('pause in routine RAN2');
    readln;
  END;}
  Ran2Iy := Ran2Ir[j];
  ran2 := Ran2Iy*rm;
  idum := (ia*idum+ic) MOD m;
  Ran2Ir[j] := idum;
END;


FUNCTION calc_mean(na : integer; z : RealArrayNP): REAL;
{function to calculate the grand mean of the means for all bootstrap data sets}

VAR
      s      : real;
      i      : integer;
BEGIN
      s := 0.0;
      FOR i := 1 to na DO
            s := s + z[i];
      calc_mean := s/na;
END;

FUNCTION calc_mean50(na : integer; z : IntArray50): REAL;
{function to calculate the mean value for a single bootstrap data set}

VAR
      s      : real;
      i      : integer;
BEGIN
      s := 0.0;
      FOR i := 1 to na DO
            s := s + z[i];
      calc_mean50 := s/na;
END;

PROCEDURE sort_data(n: integer;
        VAR ra: RealArrayNP);
{procedure to sort the mean values determined for the 6000 bootstrap data sets}

LABEL 99;
VAR
  l,j,ir,i: integer;
  rra: real;
BEGIN
  l := (n DIV 2)+1;
  ir := n;
  WHILE true DO BEGIN
```

```
    IF l > 1 THEN BEGIN
       l := l-1;
       rra := ra[l]
    END
    ELSE BEGIN
       rra := ra[ir];
       ra[ir] := ra[1];
       ir := ir-1;
       IF ir = 1 THEN BEGIN
          ra[1] := rra;
          GOTO 99;
       END;
    END;
    i := l;
    j := l+l;
    WHILE j <= ir DO BEGIN
       IF j < ir THEN
       IF ra[j] < ra[j+1] THEN
          j := j+1;
       IF rra < ra[j] THEN BEGIN
          ra[i] := ra[j];
          i := j;
          j := j+j;
       END
       ELSE
       j := ir+1;
    END;
     ra[i] := rra;
  END;
99:
END;

BEGIN       {Main program}
            writeln(' NIST data analysis program.');
      {     writeln(' Enter number of iterations (max 6000, min 40).');
            readln(n_p);
            writeln(' Enter number of runs (max 50).');
            readln(runs);     }

            InitCursorCtl(NIL);

            n_p := 6000;
            runs := 1;
            writeln(' Hit Return and select a file to open.');
            readln;
100:
      errCode := read_file;
      if (errCode = 0) then begin
            writeln(' Calculating bootstrap - ');
            SetCursor(GetCursor(WATCHCURSOR)^^);
            i_ran := -1 {TickCount};
            {This line uses the tick count to generate an initial random starting value
            for the random number generator}

            if (n_p < 40) then n_p := 40;
            FOR ir := 1 to runs DO BEGIN
```

```
                FOR ii := 1 TO n_p DO
                BEGIN
                RotateCursor(TickCount);
                    FOR ji := 1 TO n_dat DO
                    BEGIN
                        r_ran := ran2(i_ran);
                        jj := TRUNC(r_ran * n_dat) + 1;
                        if jj > n_dat then jj := n_dat;
                        temp_data[ji] := in_data[jj];
                    END;
                    out_data[ii] := calc_mean50(n_dat,temp_data);
                END; {FOR ii := 1 TO n_p}
                mean_val[ir] := calc_mean50(n_dat,in_data);
                mean_2000[ir] := calc_mean(n_p,out_data);
                sort_data(n_p,out_data);
                ji := TRUNC(n_p * 0.025);
                min_val[ir] := out_data[ji];
                max_val[ir] := out_data[n_p - ji];
                ii := n_p DIV 2;
                median_val[ir] := out_data[ii];
                InitCursor;
                writeln;

    {This information is given on the screen as the program is run}

      writeln;
       writeln (' For input data set: ');
       writeln ('     Mean value                = ',mean_val[ir]:9:4);
       writeln;
       writeln (' For bootstrap data sets: ');
       writeln ('     Grand mean value          = ',mean_2000[ir]:9:4);
       writeln ('     Lower 95% confidence limit = ',min_val[ir]:9:4);
       writeln ('     Upper 95% confidence limit = ',max_val[ir]:9:4);
       writeln;
       writeln;
        writeln (' The following would be an appropriate way to state the ',
               'results of');
       writeln (' this calculation: ');
       writeln;
        writeln(' The mean value for this data set is ',
                 mean_val[ir]:9:4,' ', units,'.  The interval');
        writeln (' from',min_val[ir]:9:4,
               ' to ',max_val[ir]:9:4,' ',units,' is a 95% confidence'
              ,' interval for the');
        writeln (' mean value; i.e., we may'
               ,' assert with 95% confidence that the true mean value ');
         writeln (' lies between',min_val[ir]:9:4,' and',max_val[ir]:9:4,' ',
             units,'.');

          END;
          writeln;
          writeln(' Hit Return and select a file for output or cancel.');
          readln;
          write_xfile:      {write the output file}
    end
else if (errCode = 2) then goto 100;
```

```
  EXITtoshell;
END.
```

X5. PROGRAM FOR BOOTSTRAPPING ON A PC

```
uses   Dos, Crt, windos, strings, printer;
const
  np = 6000;

TYPE   RealArrayNP = ARRAY [1..np] OF real;
    IntArray50  = ARRAY [1..50] of real;
    fnameStr    = ARRAY [0..62] of char;

VAR
    opti        : boolean;
    optr        : boolean;
    def_prt     : boolean;
    out_data    : RealArrayNP;
    Ran2Iy      : longint;
    Ran2Ir      : ARRAY [1..97] OF longint;
    in_data     : IntArray50;  {structure values}
    temp_data   : IntArray50;
    n_dat, ii, jj : integer;
    i_ran       : longint;
    r_ran       : real;
    head_x      : ARRAY [1..3] OF STRING[255];  {3 lines of header information}
    str1        : ARRAY [1..50] OF STRING[10];
    min_val, max_val       : IntArray50;
    mean_val    : IntArray50;
    mean_2000   : IntArray50;
    median_val  : IntArray50;
    ji, n_p     : integer;

    errCode       : integer;
    code          : integer;
    runs, ir      : integer;
    f_name        : fnameStr;
    stra          : string[15];
    strb          : string[15];
    strc          : string[15];
    strd          : string[15];
    stre          : string[15];
    strf          : string[60];
    ave_tt        : real;
    the_File      : text;
    f_nam         : fnameStr;
    lst           : text ;
    prt           : Boolean;
    units         : string[20];

FUNCTION read_file:    integer;
{this function reads the input data file}

VAR
    myErr         : integer;
    strn          : STRING[8];
```

```
    theFile      : text;
    temp         : real;
    ij,nt        : integer;
    ch           : CHAR;
    tt           : real;
    t1           : string[12];  {sum of data}
    l1           : integer;

    S: array[0..fsPathName] of Char;  {used for fileSearch}

  label 10;
  label 20;

   BEGIN

    ch := Chr(ord(9));
10:   FileSearch(S, f_name,");{check to see if file exists}
    if S[0] = #0 then
       Begin   {file does not exist}
       Writeln(f_name,' not found   Please input file name: ');
       readln(f_name);
       goto 10;
    end;

    assign (theFile,f_name);
    reset(theFile);  {open the file for reading}

20:   readln(theFile,head_x[1]);  {read the three header lines}
    readln(theFile,head_x[2]);
    readln(theFile,head_x[3]);
    readln(theFile,units);

    tt := 0;
    ij := 0;
    REPEAT
     ij := ij + 1;
     tt := tt + in_data[ij];   {sum of data}
        readln(theFile,strn,in_data[ij]);
        {read in the identification for the grid openings analyzed
        [maximum eight characters] and the value for the number
        or concentration of asbestos structures}

        tt := tt + in_data[ij];  {sum of data values}

    UNTIL EOF(theFile) = true;
         ave_tt  := tt/ij;   {calculate the mean value for data
                                 in the input data set}
         n_dat := ij;        {number of values in the input data set}

    close(theFile);
    myErr := 0;
    read_file := myErr;

    END;
```

```
PROCEDURE write_xfile;  {procedure for writing the output file}
VAR

    myErr        : integer;
    f_nam        : fnameStr;
    temp         : real;
    ij, vol_num  : integer;
    str3         : String[255];
    S            : ARRAY[0..fsPathName] of Char;
    yr, mon, day, dow : word;
const
    def_file     : ARRAY[0..12] of char = 'BootOut.txt';

label  10;
label  999;
    BEGIN

   ij := 1;
    if (def_prt) then
     begin

  {The information below is written to the screen as the program is run}

    writeln;
    writeln (' For input data set: ');
    writeln ('    Mean value              = ',mean_val[ij]:9:4);
    writeln;
    writeln (' For bootstrap data sets: ');
    writeln ('    Grand mean value        = ',mean_2000[ij]:9:4);
    writeln ('    Lower 95% confidence limit  = ',min_val[ij]:9:4);
    writeln ('    Upper 95% confidence limit  = ',max_val[ij]:9:4);
    writeln;
    writeln;
    writeln (' The following would be an appropriate way to state the ',
          'results of');
    writeln (' this calculation: ');
    writeln;
    writeln(' The mean value for this data set is ',
         mean_val[ij]:9:4,' ', units,'.  The interval');
    writeln (' from',min_val[ij]:9:4,
        ' to ',max_val[ij]:9:4,' ',units,' is a 95% confidence'
        ,' interval for the');
    writeln (' mean value; i.e., we may'
        ,' assert with 95% confidence that the true mean value ');
    writeln (' lies between',min_val[ij]:9:4,' and',max_val[ij]:9:4,' ',
        units,'.');
       end;
    writeln;
    writeln;
    write(' Save results to a file?  Y/N: ');

   readln(str3[1]);
if (prt) then
begin
   writeln(lst);
   writeln(lst);
```

```
   writeln (lst,' The following would be an appropriate way to state the ',
         'results of');
   writeln (lst,' this calculation: ');
   writeln(lst);
   writeln(lst,' The mean value for this data set is ',
         mean_val[ij]:9:4,' ', units,'.  The interval');
   writeln (lst,' from',min_val[ij]:9:4,
       ' to ',max_val[ij]:9:4,' ',units,' is a 95% confidence'
       ,' interval for the');
   writeln (lst,' mean value; i.e., we may'
       ,' assert with 95% confidence that the true mean value ');
   writeln (lst,' lies between ',min_val[ij]:9:4,' and ',max_val[ij]:9:4,' ',
       units,'.');
   writeln(lst);
   writeln(lst);
   writeln(lst,' Save results to a file? Y/N: ',str3[1]);
end;
   if(UpCase(str3[1]) = 'N') then goto 999;
    write(' Output to file: BootOut.txt?  Y/N ');
    readln(str3[1]);
   if (prt) then writeln(lst,' Output to file: BootOut.txt?  Y/N: ',str3[1]);
10:    if(UpCase(str3[1]) = 'Y') then {use BootOut.txt}
        begin  {check to see if file already exists}
        FileSearch(S,def_file, GetEnvVar('PATH'));
            if S[0] <> #0 then
            Begin
      write(' ',def_file,' already exists.  Do you want to replace it?  Y/N: ');
            readln(str3[1]);
        if (prt) then writeln(lst,' ',def_file,' already exists.',
        '  Do you want to replace it?  Y/N: ',str3[1]);
            if (UpCase(str3[1]) = 'N') then goto 10;
            writeln(' Output sent to ',def_file);
            if(prt)  then writeln(lst,' Output sent to ',def_file);
            assign(the_File,def_file);
            end;

        end
      else
        begin
        write(' Enter output file name: ');
        readln(f_nam);
        FileSearch(S,f_nam, GetEnvVar('PATH'));

   if (prt) then write(lst,' Enter output file name: ',f_nam);
        if S[0] <> #0 then
        Begin
        write(' ',f_nam,' already exists.  Do you want to replace it?  Y/N ');
        readln(str3[1]);
    if (prt)  then write(' ',f_nam,' already exists.  Do you want to replace it?  Y/N: ',str3[1]);
        if (UpCase(str3[1]) = 'N') then goto 10;
        assign(the_File,def_file);
        end;

      assign(the_File,f_nam);
      end;
```

```
   getdate(yr, mon, day, dow);   {get date for output file}
   rewrite(the_File);
   writeln(the_File);
   if (UpCase(str3[1]) = 'Y') then writeln(the_File,'              ',
         StrUpper(def_file),'  -  ',mon,'/', day,'/', yr-1900)
   else
   writeln(the_File,'              ',
         StrUpper(f_nam), '  -  ',mon,'/',day,'/',yr-1900);

 writeln(the_File);
 writeln(the_File,' Input data set = ',StrUpper(f_name));
 writeln(the_File,' Number of grid openings = ',n_dat);

 if( runs > 1 ) then
 begin
 writeln(the_File,' Number of iterations (bootstrap data sets) = ',n_p);
 writeln(the_File,' Number of runs = ',runs);
 end;
 writeln(the_File);
 writeln(the_File,' ',head_x[1]);
 writeln(the_File,' ',head_x[2]);
 writeln(the_File,' ',head_x[3]);
 writeln(the_File,' ',units);
 writeln(the_File);

if ( (runs = 1) and (n_p = 6000) )      then
begin
writeln (the_File,' For input data set: ');
writeln (the_File,'    Mean value              = ',mean_val[ij]:9:4);
writeln(the_File);
writeln (the_File,' For bootstrap data sets: ');
writeln (the_File,'    Grand mean value        = ',mean_2000[ij]:9:4);
writeln (the_File,'    Lower 95% confidence limit  = ',min_val[ij]:9:4);
writeln (the_File,'    Upper 95% confidence limit  = ',max_val[ij]:9:4);

writeln(the_File);
writeln(the_File);
writeln (the_File,' The following would be an appropriate way to state the ',
      'results of');
writeln (the_File,' this calculation: ');
writeln(the_File);
writeln(the_File,' The mean value for this data set is ',
     mean_val[ij]:9:4,' ', units,'.  The interval');
writeln (the_File,' from ',min_val[ij]:9:4,
    ' to ',max_val[ij]:9:4,' ',units,' is a 95% confidence'
    ,' interval for the');
writeln (the_File,' mean value; i.e., we may'
    ,' assert with 95% confidence that the true mean value ');
writeln (the_File,' lies between ',min_val[ij]:9:4,' and ',max_val[ij]:9:4,' ',
    units,'.');
writeln(the_File);
writeln(the_File);
end;

  str3 := concat('Mean value', chr(9), 'Grand Mean', chr(9),
 'Median value', chr(9),       'Lower limit',chr(9), 'Upper limit');
```

```
  writeln;
  if ( opti) then
  begin
   writeln(the_File);
   writeln(the_File,'Run  Mean Value   Grand Mean    '
                  ,'Median        Lower        Upper');

     writeln(the_File,'    (of input   bootstrap                Limit        Limit');
     writeln(the_File,'    counts)     data sets');
     writeln(the_File);

         FOR ij := 1 to runs DO begin

         str(mean_val[ij]:10:4,stra);
         str(mean_2000[ij]:10:4,strb)  ;
         str(median_val[ij]:10:4,strc);
         str(min_val[ij]:10:4,strd);
         str(max_val[ij]:10:4,stre);
   strf := concat(stra,chr(9),strb,chr(9),strc,chr(9),strd,chr(9),stre);
         writeln (the_file,' ',ij,'  ',strf);

         end;

  end
  else
  begin
  end;
    close(the_File);

999:  END;


  FUNCTION ran2(VAR idum: longint): real;
  {function to generate random numbers for the bootstrapping program}

  CONST   m  = 714025;
     ia = 1366;
     ic = 150889;
     rm = 1.4005112e-6;
  VAR
     j: integer;

   BEGIN
   IF idum < 0 THEN
   BEGIN
         idum := (ic-idum) MOD m;
         FOR j := 1 TO 97
         DO BEGIN
                  idum := (ia*idum+ic) MOD m;
                  Ran2Ir[j] := idum;
         END;

   idum := (ia*idum+ic) MOD m;
   Ran2Iy := idum;
   END;
```

```
j := 1 + (97*Ran2Iy) DIV m;
Ran2Iy := Ran2Ir[j];
ran2 := Ran2Iy*rm;
idum := (ia*idum+ic) MOD m;
Ran2Ir[j] := idum;
END;

FUNCTION calc_mean(na : integer; z : RealArrayNP): REAL;
{function to calculate the grand mean for all the bootstrap data sets}

VAR
  s     : real;
  i     : integer;
  BEGIN
    s := 0.0;
    FOR i := 1 to na DO
        s := s + z[i];
        calc_mean := s/na;
    END;

FUNCTION calc_mean50(na : integer; z : IntArray50): REAL;
{function to calculate the mean value for a single bootstrp data set}

VAR
  s     : real;
  i     : integer;

  BEGIN
    s := 0.0;
    FOR i := 1 to na DO
        s := s + z[i];
        calc_mean50 := s/na;

    END;


PROCEDURE sort_data(n: integer;
        VAR ra: RealArrayNP);
{procedure to sort the mean values determined for the 6000 bootstrap
data sets}

LABEL 99;
VAR
   l,j,ir,i: integer;
   rra: real;
BEGIN
l := (n DIV 2)+1;
ir := n;
WHILE true DO BEGIN
   IF l > 1 THEN BEGIN
    l := l-1;
    rra := ra[l]
   END
   ELSE BEGIN
    rra := ra[ir];
    ra[ir] := ra[1];
```

```
        ir := ir-1;
        IF ir = 1 THEN BEGIN
           ra[1] := rra;
           GOTO 99;
        END;
     END;
     i := l;
     j := l+l;

     WHILE j <= ir DO BEGIN
        IF j < ir THEN
        IF ra[j] < ra[j+1] THEN
         j := j+1;
        IF rra < ra[j] THEN BEGIN
         ra[i] := ra[j];
         i := j;
         j := j+j;
        END
        ELSE
        j := ir+1;
      END;
      ra[i] := rra;
     END;
    99:
  END;


var
    hr, min, sec, hund : word;
    opts : string;

label 10;

    BEGIN {Main program}
    {note: opti, optr, prt refer to additional features not described in
    this NISTIR}

    opti := False;
    optr := False;
    prt := False;
    def_prt := True;
    n_p  := 6000;
    runs := 1;

    clrScr;

    for ii := 1 to ParamCount
    do begin
    opts := concat(opts, ParamStr(ii))  ;
    end;

    if((Pos('R',opts) > 0) or (Pos('r',opts) > 0)) then optr := True;
    if((Pos('I',opts) > 0) or (Pos('i',opts) > 0)) then opti := True;
    if((Pos('P',opts) > 0) or (Pos('p',opts) > 0)) then prt := True;
    if (optr or opti) then def_prt := False;
```

```
    if (prt) then
    begin
    writeln(' Print the screen  inhouse option');
    writeln(' Make sure Printer is online  Hit return when ready');
    readln;
    assign(Lst,'lpt1');
    rewrite(lst);
    end;

    writeln('          NIST data analysis program');
    writeln;
    write(' Input the file name: ');
    readln(f_name);

    if (opti) then
    begin
    write(' Enter number of iterations (max 6000).: ');
    readln(n_p);
    end;

    if (optr) then
     begin
        write(' Enter number of runs (max 50).: ');
        readln(runs);
      end;

    if prt  then
    begin
    {$I-}
10:      writeln(lst,'          NIST data analysis program');
    {$I+}
       if IOResult = 160 then
       begin
            writeln(' Printer is off-line?');
            writeln (' hit Return when ready');
            readln;
            goto 10;
       end;

    writeln(lst);
    write(lst,' Input the file name: ');
    writeln(lst,f_name);
    writeln(lst);

       if (opti) then
       begin
       write(lst,' Enter number of iterations (max 6000).: ');
       writeln(lst,n_p);
       write(lst,' Enter number of runs (max 50).: ');
       writeln(lst,runs);
       end;
    end;

    clrScr;

    writeln(' Calculating bootstrap - ');
```

```
if (prt) then writeln(lst,' Calculating bootstrap - ');
errCode := read_file;

if (errCode = 0) then begin  {file OK}

{The next three lines use the time to generate an initial
random starting value for the random number generator}

 i_ran := -1;
 GetTime(hr,min,sec,hund);
 i_ran := hund * min * sec;

 FOR ir := 1 to runs DO BEGIN
  FOR ii := 1 TO n_p DO BEGIN

    FOR ji := 1 TO n_dat DO BEGIN
      r_ran := ran2(i_ran);
      jj := TRUNC(r_ran * n_dat) + 1;
      if jj > n_dat then jj := n_dat;
      temp_data[ji] := in_data[jj];
    END;
  out_data[ii] := calc_mean50(n_dat,temp_data);

  END;
mean_val[ir] := calc_mean50(n_dat,in_data);
mean_2000[ir] := calc_mean(n_p, out_data);
sort_data(n_p,out_data);
ji := TRUNC(n_p * 0.025);
min_val[ir] := out_data[ji];
max_val[ir] := out_data[n_p - ji];
ii := n_p DIV 2;

median_val[ir] := out_data[ii];
str(mean_val[ir]:10:4,stra);
str(mean_2000[ir]:10:4,strb)  ;
str(min_val[ir]:10:4,strd);
str(max_val[ir]:10:4,stre);
strf := concat(stra,chr(9),strb,chr(9),strc,chr(9),strd,chr(9),stre);

if (optr) then
  Begin
  writeln;
  writeln(' Run number: ',ir);
  writeln(' Mean value is:     ',chr(9),stra);
  writeln(' Mean value of ',n_p,' is:  ',strb);
  writeln(' Lower limit is:    ',chr(9),strd);
  writeln(' Upper limit is:    ',chr(9),stre);
end;

if (prt)  then
Begin
   if (opti) then
   Begin
   writeln(lst);
   writeln(lst,' Run number: ',ir);
   writeln(lst,' Mean value is:        ',chr(9),stra);
```

```
        writeln(lst,' Mean value of ',n_p,' is: ',chr(9),strb);
        writeln(lst,' Median value is:      ',chr(9),strc);
        writeln(lst,' Lower limit is:       ',chr(9),strd);
        writeln(lst,' Upper limit is:       ',chr(9),stre);
        end;

        if (prt and def_prt) then
        begin
        writeln(lst);
        writeln (lst,' For input data set: ');
        writeln (lst,'    Mean value              = ',
              mean_val[ir]:9:4);
        writeln(lst);
        writeln (lst,' For bootstrap data sets: ');
        writeln (lst,'    Grand mean value        = ',
              mean_2000[ir]:9:4);
        writeln (lst,'    Lower 95% confidence limit = ',
              min_val[ir]:9:4);
        writeln (lst,'    Upper 95% confidence limit = ',
              max_val[ir]:9:4);
        end;

     end;

    END;
    write_xfile;      {write the output file}
 end;

  if prt  then writeln(lst,chr(12));
  if prt  then close(lst);
END.
```