Appendix B: Computer Programs

This appendix contains the computer programs, some written in Arc/Info’s macro language (aml), and others in C, FORTRAN, or AWK, that organize the data used in the study and calculate the reported statistics. The following list groups the programs according to their function and gives their names. The actual code follows, in the order listed.

WELL AND MEASUREMENT RECORD SELECTION
- test_quad.aml
- test_quad.c
- include.aml

PRECIPITATION CALCULATIONS
- years.aml
- firstyear
- lastyear
- gap
- precmean.aml

SOIL PARAMETER CALCULATIONS
- org_int.aml
- unit_avg.aml

QUADRANGLE GENERATION
- build_quads7.aml
- tx_7m.c

DISCRETE PROBABILITY CALCULATIONS
- count_quad.aml
- count_aq.aml
- count_aqquad.aml
- count_wud.aml
- bino2.f
- bino_quad.aml

LOGNORMAL PARAMETER CALCULATIONS

- logfit.c
- fit_quad.aml

MAP CREATION

- gt1_plot.aml

QUADRANGLE PARAMETER AVERAGES

- aw_avg.aml

```
/*  testquad.aml -- determines whether the number of a well listed
/*  in the TWDB database is consistent with the well's location
/*  data.

tables

  /*  define an info table to hold the results of the test
  define qtest.tab
    wellno 7 7 i
    quad_ok 1 1 c
    quad_err 1 1 c
    ~

  /*  create a text file with the necessary data for the test
  select twdb_wells.dat
    unload qtest.in wellno latitude longitude

  /*  execute the C program the performs the test
  SYSTEM test_quad qtest.in qtest.out

  /*  transfer the results to the new info table
  select qtest.tab
    add from qtest.out

q stop

/*  Expand the well data table with the results of the test.
JOINITEM twdb_wells.dat qtest.tab twdb_wells.dat wellno primeuse

/*  delete the text files and the temporary info table
&sv delstat := [DELETE -FILE qtest.in]
&sv delstat := [DELETE -FILE qtest.out]
&sv delstat := [DELETE -INFO qtest.tab]

&return
```

```
/*  test_quad.c -- tests whether a location, given by latitude and
*   longitude, is consistent with the TWDB number assigned to a
*   well */

/*  USAGE:  test_quad infile outfile */

#include <stdio.h>

/* define an array holding numbers and coordinates for the SE
*  corners of the 89 1-degree quads enclosing Texas, numbered
*  in accordance with TWDB wells */

int secor[89][3] = { {1, 103, 36},{2, 102, 36},{3, 101, 36},
{4, 100, 36},{5, 100, 35},{6, 101, 35},{7, 102, 35},
{8, 103, 35},{9, 103, 34},{10, 102, 34},{11, 101, 34},
{12, 100, 34},{13, 99, 34},{14, 98, 34},{15, 97, 34},
{16, 94, 33},{17, 95, 33},{18, 96, 33},{19, 97, 33},
{20, 98, 33},{21, 99, 33},{22, 100, 33},{23, 101, 33},
{24, 102, 33},{25, 103, 33},{26, 103, 32},{27, 102, 32},
{28, 101, 32},{29, 100, 32},{30, 99, 32},{31, 98, 32},
{32, 97, 32},{33, 96, 32},{34, 95, 32},{35, 94, 32},
{36, 93, 31},{37, 94, 31},{38, 95, 31},{39, 96, 31},{40, 97, 31},
{41, 98, 31},{42, 99, 31},{43, 100, 31},{44, 101, 31},
{45, 102, 31},{46, 103, 31},{47, 104, 31},{48, 105, 31},
{49, 106, 31},{50, 105, 30},{51, 104, 30},{52, 103, 30},
{53, 102, 30},{54, 101, 30},{55, 100, 30},{56, 99, 30},
{57, 98, 30},{58, 97, 30},{59, 96, 30},{60, 95, 30},{61, 94, 30},
{62, 93, 30},{63, 93, 29},{64, 94, 29},{65, 95, 29},
{66, 96, 29},{67, 97, 29},{68, 98, 29},{69, 99, 29},
{70, 100, 29},{71, 101, 29},{72, 102, 29},{73, 103, 29},
{74, 104, 29},{75, 103, 28},{76, 100, 28},{77, 99, 28},
{78, 98, 28},{79, 97, 28},{80, 96, 28},{81, 95, 28},
{82, 96, 27},{83, 97, 27},{84, 98, 27},{85, 99, 27},
{86, 99, 26},{87, 98, 26},{88, 97, 26},{89, 97, 25}};

main(argc, argv)
  int  argc;
  char  *argv[];
{
  int wellno, latitude, longitude, scancnt;
  int qdeg, q7, q2, remain;
  int latd, latm, lats, latmss, latmssmin, latmssmax;
  int lngd, lngm, lngs, lngmss, lngmssmin, lngmssmax;
  char loc_ok = 'y', errtype = 'n', inst[30];
  FILE *datafp, *outfp;

  /*  usage message for careless users */
  if(argc != 3){
    fprintf(stderr, "\nUSAGE:  %s infile outfile\n", argv[0]);
    exit(1);}

  /*  open the data file READ ONLY */
  if((datafp = fopen(argv[1], "r")) == NULL){
    fprintf(stderr, "\nunable to open data file:  %s.\n",
      argv[1]);
    exit(1);}
```

```
/*  open the output file */
if((outfp = fopen(argv[2], "w")) == NULL){
  fprintf(stderr, "\nunable to open output file:  %s.\n",
    argv[2]);
  fclose(datafp);
  exit(1);}

/*  read a line from the input file*/
while((scancnt = fscanf(datafp, "%s", inst)) != EOF){

  /*  if the line is properly formatted, extract and test
      the data  */
  if(sscanf(inst,"%d,%d,%d",
     &wellno, &latitude, &longitude) == 3){

    /*  parse the well number into degree quad, 7.5 min quad,
        and 2.5 min quad numbers */
    qdeg = wellno / 100000;
    remain = wellno % 100000;
    q7 = remain / 1000;
    remain = remain % 1000;
    q2 = remain / 100;
    remain = remain % 100;

    /*  parse latitude into degrees minutes seconds */
    latd = latitude / 10000;
    remain = latitude % 10000;
    latm = remain / 100;
    lats = remain % 100;

    /*  parse longitude into degrees minutes seconds  */
    lngd = longitude / 10000;
    remain = longitude % 10000;
    lngm = remain / 100;
    lngs = remain % 100;

    /*  add minutes and seconds as seconds */
    latmss = latm * 60 + lats;
    lngmss = lngm * 60 + lngs;

    /*  identify inconsistent 1-degree quad numbers */
    if(latd != secor[qdeg-1][2] || lngd != secor[qdeg-1][1]){
      loc_ok = 'n';
      errtype = 'd';}

    /*  define range of lat and long for 7.5 minute quad
         nb:  7.5 min = 450 sec */
    latmssmin = ((64 - q7) / 8) * 450;
    latmssmax = latmssmin + 450;
    lngmssmin = ((64 - q7) % 8) * 450;
    lngmssmax = lngmssmin + 450;

    /*  identify inconsistent 7.5 minute quad numbers */
    if(loc_ok == 'y'){
    if(latmss < latmssmin || latmss > latmssmax ||
       lngmss < lngmssmin || lngmss > lngmssmax){
      loc_ok = 'n';
```

```
        errtype = '7';}}

      /*  define range of lat and long for 2.5 minute quad
           nb:  2.5 min = 150 sec */
      latmssmin = latmssmin + ((9 - q2) / 3) * 150;
      latmssmax = latmssmin + 150;
      lngmssmin = lngmssmin + ((9 - q2) % 3) * 150;
      lngmssmax = lngmssmin + 150;

      /*  identify inconsistent 2.5 minute quad numbers */
      if(loc_ok == 'y'){
      if(latmss < latmssmin || latmss > latmssmax ||
         lngmss < lngmssmin || lngmss > lngmssmax){
        loc_ok = 'n';
        errtype = '2';}}

      /* print results to output file */
      fprintf(outfp, "%d,%c,%c\n", wellno, loc_ok, errtype);

      /*  reinitialize output variables */
      loc_ok = 'y';
      errtype = 'n';
    }  /* end of test loop */
  }  /* end of file read loop */

  fclose(datafp);
  fclose(outfp);

}  /*  end of program test_quad.c */
```

```
/*  include.aml -- selects those records to be included in TWDB
/*  nitrate study.  Should be run after testquad.aml

/*  set up relate to gain access to well records from nitrate
/* measurement table.
relate add
  well
  twdb_wells.dat
  INFO
  wellno
  wellno
  ordered
  rw
~

tables

  /*  add items to well and measurement tables to indicate
  /*  inclusion in study, and wells with included measurements
  additem twdb_wells.dat meas 1 1 c
  additem twdb_wells.dat include 1 1 c
  additem twdb_wells.nit include 1 1 c

  /*  sort the well table by well number to speed up relate
  sel twdb_wells.dat
  sort wellno

  /*  set MEAS field to 'n' (records corresponding to wells with
  /*  included measurements will be changed later in the program.)
  move 'n' to meas

  /*  select the nitrate table and restrict the selection to
  /*  records with remarks indicating poor reliability
  sel twdb_wells.nit
  res reliability_rem = '01' or reliability_rem = '02' ~
      or reliability_rem = '03'

  /*  add to selection records with no corresponding well records
  asel wellno ne well//wellno

  /*  add to selection records for measurements from
  /*  mis-located wells
  asel well//quad_ok = 'n'

  /*  add to selection records with thresholds above
  /*  0.1 mg/l (as N) or 0.45 (as nitrate)
  asel q71850_flag = '<' and q71850_nitrate gt 0.45

  /*  add to selection records with "greater than" flag
  asel q71850_flag = '>'

  /*  add to selection records for samples prior to 1962
  asel yy_date lt 1962

  /*  mark selected records for exclusion
  move 'n' to include
  /*  invert selection and mark those records for inclusion
  nsel
  move 'y' to include

  /*  mark well records corresponding to included measurements
  move 'y' to well//meas

  /*  mark well records to be included in study
  sel twdb_wells.dat
  res quad_ok = 'y' and meas = 'y'
  move 'y' to include
  nsel
  move 'n' to include

  /*  create table include.nit
  copy twdb_wells.nit include.nit
  sel include.nit
  res include = 'n'
  purge
    y

  /*  create table include.wells
  copy twdb_wells.dat include.wells
  sel include.wells
  res include = 'n'
  purge
    y


q stop
relate drop
  wells
  ~

&return
```

```
/*  years.aml -- procedure followed to add items to info table
/*  station.mean indicating first year and last year of data
/*  reported and maximum gap in recording period.

tables

sel prec.dat
  sort station_id station_name year
  unload gap.dat station_id station_name year
  sys awk -f firstyear gap.dat > first.out
  sys awk -f lastyear gap.dat > last.out
  sys awk -f maxgap gap.dat > gap.out

define first.in
  station_id 5 5 i
  station_name 23 23 c
  firstyear 4 4 i
~
redefine
  1
  unique
  28
  28
  c
~
add from first.out

define last.in
  station_id 5 5 i
  station_name 23 23 c
  lastyear 4 4 i
~
redefine
  1
  unique
  28
  28
  c
~
add from last.out

define gap.in
  station_id 5 5 i
  station_name 23 23 c
  gap 2 2 i
~
redefine
  1
  unique
  28
  28
  c
~
add from gap.out

q stop
```

```
joinitem station.mean gap.in station.mean unique op_40
joinitem station.mean last.in station.mean unique op_40
joinitem station.mean first.in station.mean unique op_40

&return
```

```
#  firstyear -- awk script to find first reporting year in annual
#  reports from stations identified by ID number and name.
#  Expects input in form of comma-delimited fields containing
#  station ID, station name, and reporting year.

BEGIN{FS=OFS=",";
  station = "";
  staname = "";}

$1 != station || $2 != staname{print $1,$2,$3;
  station = $1;
  staname = $2;}
```

```
#  lastyear -- awk script to find last reporting year in annual
#  reports from stations identified by ID number and name.
#  Expects input in form of comma-delimited fields containing
#  station ID, station name, and reporting year.

BEGIN{FS=OFS=",";
  lastyear = 0
  station = ""
  staname = ""}

$1 != station || $2 != staname{print station,staname,lastyear;
  station = $1
  staname = $2
  lastyear = $3}

$1 == station && $2 == staname{lastyear = $3;}
END{print station,staname,lastyear}
```

```
#  maxgap -- awk script to find maximum gap in annual reports
#  from stations identified by ID number and name.  Expects
#  input in form of comma-delimited fields containing
#  station ID, station name, and reporting year.

BEGIN{FS=OFS=",";
  gap = 0;
  count = 0;
  lastyear = 0
  station = ""
  staname = ""}

$1 != station || $2 != staname{print station,staname,count;
  count = 0;
  station = $1
  staname = $2
  lastyear = $3}

$1 == station && $2 == staname && $3 == (lastyear + 1)
  {lastyear = $3;}

$1 == station && $2 == staname && $3 != (lastyear + 1)
  {gap = $3 - lastyear - 1;
    if(gap > count) count = gap;
    lastyear = $3;}

END{print station,staname,count}
```

```
/*  precmean.aml -- procedure followed to calculate mean annual
/*  precipitation reported at databsae stations from 1951 to 1980

tables

relate add
  station
  station.mean
  info
  unique
  unique
  ordered
  rw
~

sel prec.dat
  calc station//cnt_40 = station//cnt_40 + 1
  calc station//tot_40 = station//tot_40 + prec
  res year lt 1981
  calc station//cnt_30 = station//cnt_30 + 1
  calc station//tot_30 = station//tot_30 + prec

sel station.mean
  res cnt_40 ne 0
  calc mean_40 = tot_40 / cnt_40
  asel
  res cnt_30 ne 0
  calc mean_30 = tot_30 / cnt_30

q stop
&return
```

```
/*  org_int.aml
/*
/*  Calculates integral of organic material in the soil layer
/*  table for each component.

relate add
  comp
  study.comp
  info
  mapseq
  mapseq
  ordered
  rw
~

tables

sel study.comp
  sort mapseq
  calc min-org = 0
  calc max-org = 0
  calc avg-org = 0

sel study.layer

/*  caculate average organic content (pct.)
calc omm = ( omh + oml ) / 2

/*  caculate average bulk density (g/cm^3)
calc bdm = ( bdh + bdl ) / 2

/*  multiply average organic content by average bulk density and
/*  add the result to the average organic field in the component
/*  table.  (0.254 converts inches to cm, g/cm^2 to kg/m^2, and
/*  percents to decimals.)
calc comp//avg-org = comp//avg-org + omm * bdm ~
  * ( laydeph - laydepl ) * 0.254

/*  multiply max organic content by max bulk density and add the
/*  result to the max organic field in the component table.
calc comp//max-org = comp//max-org + omh * bdh ~
  * ( laydeph - laydepl ) * 0.254

/*  multiply min organic content by min bulk density and add the
/*  result to the min organic field in the component table.
calc comp//min-org = comp//min-org + oml * bdl ~
  * ( laydeph - laydepl ) * 0.254

q stop

relate drop
  comp
  ~

&return
```

```
/*  unit_avg.aml
/*
/*  Calculates average of a numerical item in the soil
/*  component table for each map unit.  (executed from
/*  within TABLES)

/*  Read the name of the component item to be averaged, the
/*  destination mapunit item, and, optionally, the definition of
/*  the mapunit item if it does not already exist in the table
&args compitem muitem add itdef:rest

relate add
  mapu
  study.mapu
  info
  muid
  muid
  ordered
  rw
~

/* option to add new item for unit average
&if [TRANSLATE %add%] = 'ADD' &then ~
  additem study.mapu %muitem% [UNQUOTE %itdef%]

sel study.mapu
  sort muid
  calc sum = 0
  /* if new item option not exercised, set item value to zero
  &if [NULL %add%] &then ~
    calc %muitem% = 0

sel study.comp
  calc mapu//%muitem% = mapu//%muitem% + ~
    %compitem% * comppct / 100
  calc mapu//sum = mapu//sum + comppct

sel study.mapu
  res sum ne 100
  list muid %muitem% sum

sel

relate drop
  mapu
  ~

&return
```

```
/*  build_quads7.aml -- builds a polygon coverage in geographic
/*  (DD) co-ordinates of the 7.5 minute quadrangles used for well
/*  numbering by the Texas Water Development Board.

/*  make this a double-precision coverage
precision double

/*  run the C program that generates the quad coordinates
&sys tx7m > tx7m.gen

/*  put the coordinates into a new polygon coverage
generate twdb_7m
  input tx7m.gen
  polys
  quit

/*  tidy up the coverage to get rid of double-listed coordinates
clean twdb_7m

/*  set up the quadrangle ID numbers used for relates to well
data.
tables
  additem twdb_7m.pat quad_7.5m 4 4 i
  sel twdb_7m.pat
  calculate quad_7.5m = twdb_7m-id
  redefine
    25
    quad_1d
    2
    2
    I
  ~
q stop

/* delete the generate data file
&sv delstat := [DELETE tx7m.gen]

&return
```

```
/*  tx7m.c -- generates 7.5 minute cells in decimal degrees (DD)
*   numbered according to the TWDB well-numbering scheme.  Output
*   is an Arc/Info generate file.  */

#include <stdio.h>

/*  set up data matrix for 1-dgree quad locations */
int data[13][4] = {
  {4, 1, -104, 37},  /* One-degree quads in Texas fall in 13    */
  {4, -1, -101, 36}, /* rows. Numbers for quads run west-to-east*/
  {7, 1, -104, 35},  /* and east-to-west in alternating rows.   */
  {10, -1, -95, 34}, /* The matrix "data"   contains 13 four-   */
  {10, 1, -104, 33}, /* column rows.  In each row, the first    */
  {14, -1, -94, 32}, /* column is the number of cells in the    */
  {13, 1, -106, 31}, /* corresponding row of one-degree quads.  */
  {12, -1, -94, 30}, /* The second column is the direction of   */
  {1, 1, -104, 29},  /* the numbering (1 for west-to-east, -1   */
  {6, 1, -101, 29},  /* for east-to-west) in the row.  The third*/
  {4, -1, -97, 28},  /* and fourth columns are the latitude and */
  {3, 1, -100, 27},  /* longitude of the northwest corner of the*/
  {1, 1, -98, 26}};  /* lowest-numbered cell  in the row.  */
                     /* There are 89 one-degree quads. */

main(){
  int dcell = 1;
  int row;
  int cnt, mcell;
  int xcnr, ycnr;
  float xctr, yctr;
  float offset = 0.0625, csize = 0.125;

  /* outer loop for rows of one-degree quads */
  for(row = 0; row < 13; row++){

    /* 1st nested loop for individual one-degree quads */
    for(cnt = 0; cnt < data[row][0]; cnt++){

      /* locate the northwest corner of the one-degree quad */
      xcnr = data[row][2] + data[row][1]*cnt;
      ycnr = data[row][3];

      /* 2nd nested loop for 7.5 minute quads */
      for(mcell = 0; mcell < 64; mcell++){

        /* center of 7.5 minute quad is a half-cell south and east
of
           the quad's NW corner.  NW corner located by integer
division
           of 7.5-minute quad number. Integer quotient is number
of rows
           down from top; remainder is number of columns over from
           edge.  */
        xctr = xcnr + (mcell%8)*csize + offset;
        yctr = ycnr - (mcell/8)*csize - offset;

        /* print cell number and center co-ordinates, followed by
           cell corner co-ordinates.  */
        printf("%d, %f, %f\n", 100*dcell + mcell + 1, xctr, yctr);
        printf("  %f, %f\n", xcnr + (mcell%8)*csize, ycnr -
               (mcell/8)*csize);
        printf("  %f, %f\n", xcnr + (mcell%8)*csize + csize,
               ycnr - (mcell/8)*csize);
        printf("  %f, %f\n", xcnr + (mcell%8)*csize + csize,
               ycnr - (mcell/8)*csize - csize);
        printf("  %f, %f\n", xcnr + (mcell%8)*csize,
               ycnr - (mcell/8)*csize -csize);

        /* print "end" to close polygon */
        printf("%s\n", "end");
      } /* end 7.5-minute quad loop */

      /* step to next one-degree quad */
      dcell++;
    } /* close one-degree quad loop */
  } /* close row loop */

  /* print "end" to close generate file */
  printf("%s\n", "end");
}
```

```
/******************************************************************
/*
/*  count_quad.aml -- counts number of wells, number of nitrate
/*  measurements, number of nitrate measuements exceeding 0.1, 1,
/*  5, and 10 mg/l (N) thresholds for counties in Texas, based on
/*  TWDB data.
/*
/*    CALLED BY:  user
/*
/*    CALLS:  none
/*
/*    USAGE:  &r count_quad
/*
/******************************************************************
/*
/*  VARIABLE LIST
/*
/*  LOCAL
/*    none
/*
/*  GLOBAL
/*    none
/*
/******************************************************************

tables

  relate add
    quad
    counts.quad
    info
    quad_7.5m
    quad_7.5m
    ordered
    rw
  ~

  /*  initialize the counts and sort the quad data table
    sel counts.quad
    calc well_cnt = 0
    calc meas_cnt = 0
    calc dtct_cnt = 0
    calc gt1_cnt = 0
    calc gt5_cnt = 0
    calc gt10_cnt = 0
    calc dtct_prob = 0
    calc gt1_prob = 0
    calc gt5_prob = 0
    calc gt10_prob = 0
    sort quad_7.5m

  sel include.wells
  calc quad//well_cnt = quad//well_cnt + 1

  sel include.nit
  calc quad//meas_cnt = quad//meas_cnt + 1
  res nit_adj gt 0.1
  calc quad//dtct_cnt = quad//dtct_cnt + 1

  res nit_adj gt 1.0
  calc quad//gt1_cnt = quad//gt1_cnt + 1

  res nit_adj gt 5.0
  calc quad//gt5_cnt = quad//gt5_cnt + 1

  res nit_adj gt 10.0
  calc quad//gt10_cnt = quad//gt10_cnt + 1

  sel counts.quad
  res meas_cnt ne 0
  calc dtct_prob = dtct_cnt / meas_cnt
  calc gt1_prob = gt1_cnt / meas_cnt
  calc gt5_prob = gt5_cnt / meas_cnt
  calc gt10_prob = gt10_cnt / meas_cnt

  relate drop
    quad
    ~

q stop
&return
```

```
/*****************************************************************
/*
/*  count_aq.aml -- counts number of wells, number of nitrate
/*  measurements, number of nitrate  measuements exceeding 0.1, 1,
/*  5, and 10 mg/l (N) thresholds for study aquifers, based on
/*  TWDB data.
/*
/*    CALLED BY:  user
/*
/*    CALLS:  none
/*
/*    USAGE:  &r count_aq
/*
/*****************************************************************
/*
/*  VARIABLE LIST
/*
/*  LOCAL
/*    none
/*
/*  GLOBAL
/*    none
/*
/*****************************************************************

tables

  relate add
    aq
    counts.aq5
    info
    aqf
    aqf
    ordered
    rw
  ~

  /*  initialize the counts and sort the aquifer data table
    sel counts.aq5
    calc well_cnt = 0
    calc meas_cnt = 0
    calc dtct_cnt = 0
    calc gt1_cnt = 0
    calc gt5_cnt = 0
    calc gt10_cnt = 0
    calc dtct_prob = 0
    calc gt1_prob = 0
    calc gt5_prob = 0
    calc gt10_prob = 0
    sort aqf

  sel aq5.wells
  calc aq//well_cnt = aq//well_cnt + 1

  sel aq5.nit
  calc aq//meas_cnt = aq//meas_cnt + 1
  res nit_adj gt 0.1
  calc aq//dtct_cnt = aq//dtct_cnt + 1

  res nit_adj gt 1.0
  calc aq//gt1_cnt = aq//gt1_cnt + 1

  res nit_adj gt 5.0
  calc aq//gt5_cnt = aq//gt5_cnt + 1

  res nit_adj gt 10.0
  calc aq//gt10_cnt = aq//gt10_cnt + 1

  sel counts.aq5
  res meas_cnt ne 0
  calc dtct_prob = dtct_cnt / meas_cnt
  calc gt1_prob = gt1_cnt / meas_cnt
  calc gt5_prob = gt5_cnt / meas_cnt
  calc gt10_prob = gt10_cnt / meas_cnt

q stop
&return
```

```
/*****************************************************************
/*
/*  count_aqquad.aml -- counts number of wells, number of nitrate
/*  measurements, number of nitrate  measuements exceeding the
/*  0.1, 1, 5, and 10 mg/l (N) thresholds for 7.5-minute quads in
/*  five study aquifers in Texas, based on TWDB data.
/*
/*    CALLED BY:  user
/*
/*    CALLS:  none
/*
/*    USAGE:  &r count_aqquad
/*
/*****************************************************************
/*
/*  VARIABLE LIST
/*
/*  LOCAL
/*    none
/*
/*  GLOBAL
/*    none
/*
/*****************************************************************

tables

  relate add
    aq_q
      aq_quad.dat
      info
      quad_7.5m
      quad_7.5m
      ordered
      rw
    ed_q
      ed_quad.dat
      info
      quad_7.5m
      quad_7.5m
      ordered
      rw
    cw_q
      cw_quad.dat
      info
      quad_7.5m
      quad_7.5m
      ordered
      rw
    og_q
      og_quad.dat
      info
      quad_7.5m
      quad_7.5m
      ordered
      rw
    hm_q
      hm_quad.dat
      info
      quad_7.5m
      quad_7.5m
      ordered
      rw
    sr_q
      sr_quad.dat
      info
      quad_7.5m
      quad_7.5m
      ordered
      rw
  ~

  sel aq_quad.dat
    sort quad_7.5m
    calc ebfz = 0
    calc czwx = 0
    calc ogll = 0
    calc symr = 0
    calc symr = 0
    calc aq_cnt = 0

  sel ed_quad.dat
    sort quad_7.5m
    calc well_cnt = 0
    calc meas_cnt = 0
    calc dtct_cnt = 0
    calc gt1_cnt = 0
    calc gt5_cnt = 0
    calc gt10_cnt = 0
    calc dtct_prob = 0
    calc gt1_prob = 0
    calc gt5_prob = 0
    calc gt10_prob = 0

  sel cw_quad.dat
    sort quad_7.5m
    calc well_cnt = 0
    calc meas_cnt = 0
    calc dtct_cnt = 0
    calc gt1_cnt = 0
    calc gt5_cnt = 0
    calc gt10_cnt = 0
    calc dtct_prob = 0
    calc gt1_prob = 0
    calc gt5_prob = 0
    calc gt10_prob = 0

  sel og_quad.dat
    sort quad_7.5m
    calc well_cnt = 0
    calc meas_cnt = 0
    calc dtct_cnt = 0
    calc gt1_cnt = 0
    calc gt5_cnt = 0
    calc gt10_cnt = 0
    calc dtct_prob = 0
    calc gt1_prob = 0
    calc gt5_prob = 0
    calc gt10_prob = 0

  sel hm_quad.dat
    sort quad_7.5m
    calc well_cnt = 0
    calc meas_cnt = 0
    calc dtct_cnt = 0
    calc gt1_cnt = 0
    calc gt5_cnt = 0
    calc gt10_cnt = 0
    calc dtct_prob = 0
    calc gt1_prob = 0
    calc gt5_prob = 0
    calc gt10_prob = 0

  sel sr_quad.dat
    sort quad_7.5m
    calc well_cnt = 0
    calc meas_cnt = 0
    calc dtct_cnt = 0
    calc gt1_cnt = 0
    calc gt5_cnt = 0
    calc gt10_cnt = 0
    calc dtct_prob = 0
    calc gt1_prob = 0
    calc gt5_prob = 0
    calc gt10_prob = 0

  /*  set aquifer flags on quads where wells located, and count
  /*  the number of wells in each quad in the aquifer tables.
  sel aq5.wells
    res aqf = 'EBFZ'
      calc ed_q//well_cnt = ed_q//well_cnt + 1
      calc aq_q//ebfz = 1
    nsel
    res aqf = 'CZWX'
      calc cw_q//well_cnt = cw_q//well_cnt + 1
      calc aq_q//czwx = 1
    nsel
    res aqf = 'OGLL'
      calc og_q//well_cnt = og_q//well_cnt + 1
      calc aq_q//ogll = 1
    nsel
    res aqf = 'SYMR'
      calc sr_q//well_cnt = sr_q//well_cnt + 1
      calc aq_q//symr = 1
    nsel
    res aqf = 'HMBL'
      calc hm_q//well_cnt = hm_q//well_cnt + 1
      calc aq_q//hmbl = 1

  /*  Count number of measurements and exceedences of detection
  /*  limit,1, 5, and 10 mg/l (N) thresholds.
```

```
sel aq5.nit
  res aqf = 'EBFZ'
    calc ed_quad//meas_cnt = ed_quad//meas_cnt + 1
    res nit_adj gt 0.1
      calc ed_quad//dtct_cnt = ed_quad//dtct_cnt + 1
    res nit_adj gt 1.0
      calc ed_quad//gt1_cnt = ed_quad//gt1_cnt + 1
    res nit_adj gt 5.0
      calc ed_quad//gt5_cnt = ed_quad//gt5_cnt + 1
    res nit_adj gt 10.0
      calc ed_quad//gt10_cnt = ed_quad//gt10_cnt + 1

  asel
  res aqf = 'CZWX'
    calc cw_quad//meas_cnt = cw_quad//meas_cnt + 1
    res nit_adj gt 0.1
      calc cw_quad//dtct_cnt = cw_quad//dtct_cnt + 1
    res nit_adj gt 1.0
      calc cw_quad//gt1_cnt = cw_quad//gt1_cnt + 1
    res nit_adj gt 5.0
      calc cw_quad//gt5_cnt = cw_quad//gt5_cnt + 1
    res nit_adj gt 10.0
      calc cw_quad//gt10_cnt = cw_quad//gt10_cnt + 1

  asel
  res aqf = 'OGLL'
    calc og_q//meas_cnt = og_q//meas_cnt + 1
    res nit_adj gt 0.1
      calc og_q//dtct_cnt = og_q//dtct_cnt + 1
    res nit_adj gt 1.0
      calc og_q//gt1_cnt = og_q//gt1_cnt + 1
    res nit_adj gt 5.0
      calc og_q//gt5_cnt = og_q//gt5_cnt + 1
    res nit_adj gt 10.0
      calc og_q//gt10_cnt = og_q//gt10_cnt + 1

  asel
  res aqf = 'HMBL'
    calc hm_q//meas_cnt = hm_q//meas_cnt + 1
    res nit_adj gt 0.1
      calc hm_q//dtct_cnt = hm_q//dtct_cnt + 1
    res nit_adj gt 1.0
      calc hm_q//gt1_cnt = hm_q//gt1_cnt + 1
    res nit_adj gt 5.0
      calc hm_q//gt5_cnt = hm_q//gt5_cnt + 1
    res nit_adj gt 10.0
      calc hm_q//gt10_cnt = hm_q//gt10_cnt + 1

  asel
  res aqf = 'SYMR'
    calc sr_q//meas_cnt = sr_q//meas_cnt + 1
    res nit_adj gt 0.1
      calc sr_q//dtct_cnt = sr_q//dtct_cnt + 1
    res nit_adj gt 1.0
      calc sr_q//gt1_cnt = sr_q//gt1_cnt + 1
    res nit_adj gt 5.0
      calc sr_q//gt5_cnt = sr_q//gt5_cnt + 1
      res nit_adj gt 10.0
        calc sr_q//gt10_cnt = sr_q//gt10_cnt + 1

  /*  Count the number of aquifers present in each quad
  sel aq_quad.dat
    calc aq_cnt = ebfz + czwx + ogll + symr + hmbl

  /*  Calculate the estimated probabilities of exceeding 0.1, 1,
  /*  5, and 10 mg/l (N) thresholds.

  sel ed_quad.dat
    res meas_cnt ne 0
      calc dtct_prob = dtct_cnt / meas_cnt
      calc gt1_prob = gt1_cnt / meas_cnt
      calc gt5_prob = gt5_cnt / meas_cnt
      calc gt10_prob = gt10_cnt / meas_cnt

  sel cw_quad.dat
    res meas_cnt ne 0
      calc dtct_prob = dtct_cnt / meas_cnt
      calc gt1_prob = gt1_cnt / meas_cnt
      calc gt5_prob = gt5_cnt / meas_cnt
      calc gt10_prob = gt10_cnt / meas_cnt

  sel og_quad.dat
    res meas_cnt ne 0
      calc dtct_prob = dtct_cnt / meas_cnt
      calc gt1_prob = gt1_cnt / meas_cnt
      calc gt5_prob = gt5_cnt / meas_cnt
      calc gt10_prob = gt10_cnt / meas_cnt

  sel hm_quad.dat
    res meas_cnt ne 0
      calc dtct_prob = dtct_cnt / meas_cnt
      calc gt1_prob = gt1_cnt / meas_cnt
      calc gt5_prob = gt5_cnt / meas_cnt
      calc gt10_prob = gt10_cnt / meas_cnt

  sel sr_quad.dat
    res meas_cnt ne 0
      calc dtct_prob = dtct_cnt / meas_cnt
      calc gt1_prob = gt1_cnt / meas_cnt
      calc gt5_prob = gt5_cnt / meas_cnt
      calc gt10_prob = gt10_cnt / meas_cnt

q stop

relate drop
  quad
  ed_quad
  cw_quad


~&return
```

```
/*****************************************************************
/*
/*  count_quad.aml -- counts number of wells, number of nitrate
/*  measurements, number of nitrate measuements exceeding 0.1, 1,
/*  5, and 10 mg/l (N) thresholds for counties in Texas, based on
/*  WUD data.
/*
/*    CALLED BY:  user
/*
/*    CALLS:  none
/*
/*    USAGE:  &r count_quad
/*
/******************************************************************
/*
/*  VARIABLE LIST
/*
/*  LOCAL
/*    none
/*
/*  GLOBAL
/*    none
/*
/******************************************************************

tables

  relate add
    quad
    counts.quad
    info
    quad_7.5m
    quad_7.5m
    ordered
    rw
  ~

  /*  initialize the counts and sort the quad data table
    sel counts.quad
    calc wud_meas = 0
    calc wud_dtct = 0
    calc wud_gt1 = 0
    calc wud_gt5 = 0
    calc wud_gt10 = 0
    calc wdt_prob = 0
    calc w1_prob = 0
    calc w5_prob = 0
    calc w10_prob = 0
    sort quad_7.5m

  sel nit.wrk
  calc quad//wud_meas = quad//wud_meas + 1

  res no3 gt 0.1
  calc quad//wud_dtct = quad//wud_dtct + 1

  res no3 gt 1.0
  calc quad//wud_gt1 = quad//wud_gt1 + 1

  res no3 gt 5.0
  calc quad//wud_gt5 = quad//wud_gt5 + 1

  res no3 gt 10.0
  calc quad//wud_gt10 = quad//wud_gt10 + 1

  sel counts.quad
  res wud_meas ne 0
  calc wdt_prob = wud_dtct / wud_meas
  calc w1_prob = wud_gt1 / wud_meas
  calc w5_prob = wud_gt5 / wud_meas
  calc w10_prob = wud_gt10 / wud_meas

  relate drop
    quad
    ~

q stop
&return
```

```
      PROGRAM BINO2
C*****************************************************************
C  bino2.f -- Uses a cumulative binomial probability function to
C  find confidence limits on the single-event probability for a
C  series of trials.  Built around a binomial distribution
C  approximator in the SCDFLIB fortran library, developed at
C  M. D. Anderson Cancer Center.
C*****************************************************************
C  USAGE:  bino2 confidence_level
C    Where confidence level is a number between 0 and 1.
C*****************************************************************

      REAL XN, S, GAMMA1, GAMMA2, PP, PUP, PLO, BOUND
      INTEGER SWITCH, STATUS, IDW, XNW, SW
      CHARACTER*10 ARG1
      CHARACTER*16 IDENT

C  Open input and output files.
      OPEN(7, FILE='bino2.fmt', STATUS='OLD')
      OPEN(8, FILE='bino2.in', STATUS='OLD')
      OPEN(9, FILE='bino2.out', STATUS='UNKNOWN')

C  Read the confidence level off the command line, and calculate
C  the one-sided equivalent of the two-sided confidence level.
      CALL GETARG(1, ARG1)
      READ(ARG1, '(F6.4)') GAMMA2
      GAMMA1 = (1 + GAMMA2)/2.

C  Set the switch variable for the SDFBIN function to 4, so it
C  will find the unknown binomial parameter.
      SWITCH = 4

C  Read the field widths from the Arc/Info unload format file.
      READ(7, '(3I2)') IDW, XNW, SW

C  Read an identifier, number of trials, and number of successes
C  from a line in the input file.
10    CONTINUE
      READ(8, '(A<IDW>, F<XNW>, F<SW>)', END=9999) IDENT, XN, S

C  Special case for all successes.
      IF (S .EQ. XN) THEN
        PUP = 1.0
        PLO = (1 - GAMMA2)**(1./XN)
        STATUS = 0
      END IF

C  Special case for no successes.
      IF (S .EQ. 0.) THEN
        PLO = 0.0
        PUP = 1 - (1 - GAMMA2)**(1./XN)
        STATUS = 0
      END IF

C  General case.
      IF ((S .NE. 0) .AND. (S .NE. XN)) THEN
        CALL SDFBIN(SWITCH, GAMMA1, (S-1), XN, PLO, STATUS, BOUND)
```

```
          CALL SDFBIN(SWITCH, GAMMA1, (XN-S-1), XN, PP, STATUS,
     &      BOUND)
          PUP = 1 - PP
        END IF

C  Print results.
        IF(STATUS .EQ. 0)
     &  WRITE(9, '(A<IDW>, 2(F6, X), F6.4, 2(X, F6.4))')
     &  IDENT, XN, S, GAMMA2, PLO, PUP
        IF(STATUS .NE. 0)
     &  WRITE(9, '(A<IDW>, 2(F6, X), 3(F6.4, X), I2, X, F3.1)')
     &  IDENT, XN, S, GAMMA2, PLO, PUP, STATUS, BOUND
        GOTO 10


9999  CONTINUE
      CLOSE(7)
      CLOSE(8)
      CLOSE(9)
      STOP
      END
```

```
/******************************************************************
/*  bino_quad.aml -- calculates confidence intervals for
/*  probability of detecting nitrate in concentrations above 0.1,
/*  1, 5, and 10 mg/l.  Calls system program bino2, which
/*  calculates two-sided confidence intervals.
/*
/*  No arguments.  No variables.
/******************************************************************

TABLES  /*  database operations

  /*  define temporary INFO tables to hold program outputs
  DEFINE dtct.temp
    QUAD_7.5M 4 4 I
    DTCT_LO 8 6 F 4
    DTCT_UP 8 6 F 4
    DTCT_GAP 8 6 F 4
  ~
  DEFINE gt1.temp
    QUAD_7.5M 4 4 I
    GT1_LO 8 6 F 4
    GT1_UP 8 6 F 4
    GT1_GAP 8 6 F 4
  ~
  DEFINE gt5.temp
    QUAD_7.5M 4 4 I
    GT5_LO 8 6 F 4
    GT5_UP 8 6 F 4
    GT5_GAP 8 6 F 4
  ~
  DEFINE gt10.temp
    QUAD_7.5M 4 4 I
    GT10_LO 8 6 F 4
    GT10_UP 8 6 F 4
    GT10_GAP 8 6 F 4
  ~

  /* select table of measurement and detection counts
  SELECT counts.quad

  /*  write text file of quad numbers, number of measurements and
  /*  number of nitrate detections (> 0.1 mg/l).
  UNLOAD bino2.in quad_7.5m meas_cnt dtct_cnt ~
    COLUMNAR bino2.fmt INIT

  /*  call FORTRAN program to calculate confidence intervals
  SYSTEM bino2 0.90

  /*  post-process output file to create INFO input file
  /*  containing  quad number, upper & lower confidence limits,
  /*  and difference between upper & lower limits.
  SYSTEM awk '{gap = $6 - $5; print $1","$5","$6","gap}' ~
    bino2.out > dtct.csv

  /*  remove bino2 input & output files
  SYSTEM yes|rm bino2.in bino2.fmt bino2.out
  /*  repeat for 1, 5, and 10 mg/l detection limits.
  UNLOAD bino2.in quad_7.5m meas_cnt gt1_cnt ~
    COLUMNAR bino2.fmt INIT
  SYSTEM bino2 0.90
  SYSTEM awk '{gap = $6 - $5; print $1","$5","$6","gap}' ~
    bino2.out > gt1.csv
  SYSTEM yes|rm bino2.in bino2.fmt bino2.out
  UNLOAD bino2.in quad_7.5m meas_cnt gt5_cnt ~
    COLUMNAR bino2.fmt INIT
  SYSTEM bino2 0.90
  SYSTEM awk '{gap = $6 - $5; print $1","$5","$6","gap}' ~
    bino2.out > gt5.csv
  SYSTEM yes|rm bino2.in bino2.fmt bino2.out
  UNLOAD bino2.in quad_7.5m meas_cnt gt10_cnt ~
    COLUMNAR bino2.fmt INIT
  SYSTEM bino2 0.90
  SYSTEM awk '{gap = $6 - $5; print $1","$5","$6","gap}' ~
    bino2.out > gt10.csv
  SYSTEM yes|rm bino2.in bino2.fmt bino2.out

  /*  write results into new INFO tables
  SELECT DTCT.TEMP
  ADD FROM dtct.csv
  SELECT GT1.TEMP
  ADD FROM gt1.csv
  SELECT GT5.TEMP
  ADD FROM gt5.csv
  SELECT GT10.TEMP
  ADD FROM gt10.csv

  /*  duplicate INFO table for addition of confidence limit data
  COPY counts.quad bino.quad

q stop  /*  exit tables for last step.

/*  expand INFO table with columns containing confidence limits.
JOINITEM bino.quad dtct.temp bino.quad quad_7.5m dtct_prob
JOINITEM bino.quad gt1.temp bino.quad quad_7.5m gt1_prob
JOINITEM bino.quad gt5.temp bino.quad quad_7.5m gt5_prob
JOINITEM bino.quad gt10.temp bino.quad quad_7.5m gt10_prob

&return
```

```
/* logfit.c
*  5/10/94
*  revised 9/14/94, changed from natural to common logs.
*  Tom Evans
*  Civil Engineering Department, University of Texas at Austin
*/
/*****************************************************************
*  logfit.c -- fits data in a two-column comma-delimited input
*  file to a series of lognormal distributions.  The first column
*  of the input file should contain a string to identify a group
*  that the data in the second column belong to.  All records
*  associated with a single group should appear on consecutive
*  lines.
*
*  The output file will contain one record for each group
*  identifier of the input file.  These records consist of the
*  identifier followed by the mean, standard deviation,
*  coefficient of variation, and regression parameters.
*
*  This program uses functions from _Numerical Recipes in C_
*  (Press et al., 1988), identified in comments as NR.
*
*  USAGE:  logfit infile outfile
*
*  NOTE ON VARIABLE NAMES:
*     floating point variable names end in 'f'
*     double-precision floating point variable names end in 'lf'
*     integer variable names end in 'i'
*     double-precision integer variable names end in 'li'
*     pointer variable names end in 'p'
*           ???fp is a pointer to a file
*           ???ip is a pointer to an integer
*           ???fpp is a pointer to an float
*
*****************************************************************/
#include <stdio.h>
#include <math.h>

#define MAXCHAR 80         /* Max acceptable input line length */
#define MAX_VECTOR_LENGTH 5000  /* Max length of data vector */

float *vector(); /* NR's variable-offset float vector creator */
void mdian1();   /* NR median-finding function */
int read_line (); /* reads a line of text from a file */
void fit_logn();  /* fits data to a lognormal distribution */
float normz();  /* normal variate for excedance probability */
float normp();  /* exceedance probability for a normal variate */
float betai();  /* NR's incomplete beta function */

/*****************************************************************
*  MAIN PROGRAM
*****************************************************************/

main(argc, argv)
  int  argc;
  char  *argv[];
{
```

```
float dataf, *datavfp; /* datum from file, data vector */
float medf, meanlogf, stdlogf, R2, t, F, Se, sigF;
                       /* calculated by fit_log function */
float Pd, P1, P5, P10;  /* model probabilities */
int ni;  /* number of records in input file */
int cnti = 1; /* number of values in data vector */
char inst[MAXCHAR + 1], idst[10], idnewst[10], tempst[10];
FILE *datafp, *outfp;

/*  usage message for careless users */
if(argc != 3){
  fprintf(stderr, "\nUSAGE:  %s infile outfile\n", argv[0]);
  exit(1);}

/*  open the data file READ ONLY */
if((datafp = fopen(argv[1], "r")) == NULL){
  fprintf(stderr, "\nunable to open data file:  %s.\n",
    argv[1]);
  exit(1);}

/*  open the output file */
if((outfp = fopen(argv[2], "w")) == NULL){
  fprintf(stderr, "\nunable to open output file:  %s.\n",
    argv[2]);
  fclose(datafp);
  exit(1);}

/* create the data vector */
datavfp = vector(1,MAX_VECTOR_LENGTH);

/*  read the first line from the input file */
if(read_line(datafp, inst, MAXCHAR) == EOF){
  printf("\nEmpty input file.  Terminating %s.\n", argv[0]);
  fclose(datafp);
  fclose(outfp);
  exit(1);}

/* extract the id and value from the line, extracting new lines
   from the data file if the line is bad  */
while(sscanf(inst, "%[^,],%f", tempst, &dataf) != 2)
  if(read_line(datafp, inst, MAXCHAR) == EOF){
    printf("\nBad input file.  Terminating %s.\n", argv[0]);
    fclose(datafp);
    fclose(outfp);
    exit(1);}
/*  put the first value in the data vector and set the first
    value of the id string.  */
datavfp[cnti] = dataf;
strcpy(idst, tempst);

while(read_line(datafp, inst, MAXCHAR) != EOF){
  if(sscanf(inst, "%[^,],%f", tempst, &dataf) == 2){

    if(strcmp(idst, tempst) != 0){
      /* when a new id is encountered,
      (1) perform the lognormal fitting for the last set of
          data, */
```

```
        fit_logn(datavfp, cnti, &medf, &meanlogf, &stdlogf,
                 &R2, &t, &F, &Se, &sigF);

        /* (2)print the results to the output file */

fprintf(outfp,"%s,%d,%.2f,%.3f,%.3f,%.4f,%.2f,%.2f,%.3f,%f,",
        idst, cnti, medf, meanlogf, stdlogf, R2, t, F, Se,
        sigF);

        /* (3)calculate probabilities for nitrate at 0.1, 1, 5, &
          10 mg levels */
        Pd = 1. - normp((-1.0 - meanlogf)/stdlogf);
        P1 = 1. - normp((0.0 - meanlogf)/stdlogf);
        P5 = 1. - normp((0.699 - meanlogf)/stdlogf);
        P10 = 1. - normp((1.0 - meanlogf)/stdlogf);

        /* (4)print the results to the output file */
        fprintf(outfp,"%.2f,%.2f,%.2f,%.2f\n", Pd, P1, P5, P10);

        /* (5)reinitialize the data vector. */
        cnti = 0;
        strcpy(idst, tempst);} /* end new id block */

      /* always add new data to the current vector */
      cnti++;
      datavfp[cnti] = dataf;}
    }

  /* do calculations and prints for the last data set */
  fit_logn(datavfp, cnti, &medf, &meanlogf, &stdlogf, &R2, &t, &F,
           &Se, &sigF);
  fprintf(outfp,"%s,%d,%.2f,%.3f,%.3f,%.4f,%.2f,%.2f,%.3f,%f,",
    idst, cnti, medf, meanlogf, stdlogf, R2, t, F, Se, sigF);
  Pd = 1. - normp((-1.0 - meanlogf)/stdlogf);
  P1 = 1. - normp((0.0 - meanlogf)/stdlogf);
  P5 = 1. - normp((0.699 - meanlogf)/stdlogf);
  P10 = 1. - normp((1.0 - meanlogf)/stdlogf);
  fprintf(outfp,"%.2f,%.2f,%.2f,%.2f\n", Pd, P1, P5, P10);

  /* close the files and go home */
  fclose(datafp);
  fclose(outfp);
}  /* end main program */

/*****************************************************************
*  function read_line
*****************************************************************/
/*  Reads a line from the input file pointed to by ifp and places
it in the string str.  Returns 0 if nothing unexpected happens.
Returns 1 if more than maxci charaters appear in the line.
Returns EOF if EOF encountered. */

int read_line(ifp, str, maxci)
  FILE *ifp;
  char *str;
  int  maxci;
{
  char c;
  int  cnti = 0, reti = 0;

  while((c = getc(ifp)) != '\n' && c != EOF){
    if(cnti <= maxci)
      str[cnti] = c;
    cnti++;}
  if(cnti > (maxci+1)){
    reti = 1;
    cnti = maxci + 1;}
  if(c == EOF) reti = EOF;
  str[cnti] = '\0';

  return reti;
}  /* end function count_file_lines */

/*****************************************************************
*  function fit_logn
*****************************************************************/
void fit_logn(vectorf, ni, medfp, meanlogfp, stdlogfp, R2, t, F,
              Se, sigF)
  float  *vectorf, *medfp, *meanlogfp, *stdlogfp, *R2, *t, *F,
         *Se, *sigF;
  int ni;
{
  float *z, *logx, *sig, probx, a, b, siga, sigb, chi2, q, df;
  int ii, ji;  /* counter, number of unique values in vector */

  /* special case for 1 value in data vector */
  if(ni == 1){
    *medfp = vectorf[1];
    *meanlogfp = log10(vectorf[1]);
    *stdlogfp = *t = *F = *Se = *R2 = *sigF = 0;
    return;}

  /* allocate vectors for values and normal variates */
  z = vector(1, ni);
  logx = vector(1,ni);

  /* sort the data vector and find its median value (sorted from
     low to high values). */
  mdian1(vectorf, ni, medfp);

  /* for each unique value in the sorted vector, caculate an
     excedance probability from Blom's formula, the corresponding
     normal variate, and the log of the value */
  ji = 1;
  for(ii = 1; ii < ni; ii++){
    if(vectorf[ii] != vectorf[ii + 1]){
      probx = ((ni-ii+1) - 0.375)/(ni + 0.25);
      z[ji] = normz(probx);
      logx[ji] = log10(vectorf[ii]);
      ji++;}
    }
  /* calculate same values for last value in the data vector */
  probx = ((ni-ii+1) - 0.375)/(ni + 0.25);
  z[ji] = normz(probx);
  logx[ji] = log10(vectorf[ii]);

  /* special case for 1 value */
  if(ji == 1){
    *meanlogfp = log10(vectorf[1]);
    *stdlogfp = *t = *F = *Se = *R2 = *sigF = 0;
    return;}

  fit(logx, z, ji, sig, 0, &a, &b, &siga, &sigb, &chi2, &q);

  *meanlogfp = -a/b;
  *stdlogfp = 1./b;

  df = ji - 2;
  if(df == 0.){
      *t = *F = *Se = *sigF = 0.;
      *R2 = 1.0;
      return;}

  *t = b / sigb;
  *F = *t * *t;
  *Se = sqrt(chi2/(ji-2));
  *R2 = 0;
  for(ii = 1; ii <= ji; ii++)*R2 += z[ii]*z[ii];
  *R2 = 1 - chi2 / *R2;
  *sigF = betai( df/2., 0.5, df/(df+*F) );

  return;
}

/*****************************************************************
*  function normz
*****************************************************************/
float normz(p)
  float p;
{
  float z, w;

  if(p >= 1.0 || p <= 0.0){
    fprintf(stderr, "range error on function normz.\n");
    z = -9999;
    return z;}

  if(p == 0.5) z = 0;

  else{
  if(p < 0.5) w = sqrt(-log(p*p));
  if(p > 0.5) w = sqrt(-log((1-p)*(1-p)));
  z = w - (2.515517+0.802853*w+0.010328*w*w)/
          (1.+1.432788*w+0.189269*w*w+0.001308*w*w*w);}

  if(p <= 0.5) return z;
  else return -z;
}

/*****************************************************************
*  function normp
```

```
*****************************************************************/
float normp(z)
  float z;
{
  float abz, prob;

  abz = fabs(z);
  prob = 1.+0.196854*abz+0.115194*abz*abz
          +0.000344*abz*abz*abz+0.019527*abz*abz*abz*abz;
  prob = 1./(2.*prob*prob*prob*prob);

  if(z<0)return prob;
  else return 1-prob;
}
```

```
/*****************************************************************
/*
/*  fit_quad.aml -- fits nitrate detections for 7.5' quadrangles
/*  to lognormal distributions.
/*
/*    USAGE:  &r fit_quad
/*
/*****************************************************************
/*
/*  VARIABLE LIST
/*
/*  LOCAL
/*    delstat -- status of delete operation
/*
/*  GLOBAL
/*    none
/*
/*****************************************************************

tables

  relate add
    quad
    counts.quad
    info
    quad_7.5m
    quad_7.5m
    ordered
    rw
  ~

  sel counts.quad
  sort quad_7.5m

  sel include.nit
  sort quad_7.5m
  /* don't fit quads with single measurements or no detects
  res quad//meas_cnt gt 1 and quad//dtct_cnt gt 0
  unload fit.in quad_7.5m nit_adj

  system logfit fit.in fit.out

  sel logfit.quad
  add from fit.out

  &sv delstat := [DELETE fit.in]
  &sv delstat := [DELETE fit.out]

q stop

&return
/*****************************************************************
/*
/*  gt1_plot.aml -- plots 7.5' quds shaded according to
/*  probability of detecting nitrate gt 1 mg/l.
/*
/*    CALLED BY:  user
```

```
/*
/*    CALLS:  none
/*
/*    USAGE:  &r gt1_plot <screen|ill> {illustrator file name}
/*
/******************************************************************
/*
/*  VARIABLE LIST
/*
/*  LOCAL
/*    output -- (argument, string) "screen" directs output to
/*          display;
/*           "ill" directs output to illustrator file.
/*    filename -- (argument, string) name of illustrator file
/*
/*  GLOBAL
/*    none
/*
/******************************************************************
&args output filename
/*  set up program environment
lineset carto.lin
markerset plotter.mrk
shadeset colornames.shd
&if %output% = 'screen' &then ~
  display 9999 size canvas 500 650
&else &if %output% = 'ill' &then &do /* illustrator block
  &if [NULL %filename%] &then &do
    &ty \using default output filename\
    &sv filename := bw.out
  &end
  &if [exists %filename%] &then &system rm %filename%
  display 1040 3
  %filename%
  &ty \sending output to ILLUSTRATOR file: %filename%\
&end /* end illustrator block
&else &do /* error block
  &type Invalid output option
  &return
&end /* end error block

/*  Begin map composition.
/*  Page size set for Apple LaserWriter page writable page area
/*  loses 0.35" left 0.47" right 0.42" top 0.42" bottom from 8.5
/*  by 11 detrmined by experiment 7/22/93 TAE
/******************************************************************
/*
/*  set appropriate map limits for composition
/*
/******************************************************************
maplimits 1.32 3.01 6.73 8.42 /* shifted up 1.0" from original
units page
pagesize 7.68 10.16
/* draw boundaries of writable area of dissertation page on screen
/*&if %output% = 'screen' &then &do
  linesym 101
  box 0.05 0.05 7.58 10.06 /* page boundaries
  box 1.20 0.89 6.85 8.98  /* dissertation page area, slopped
  line  1.2 1.89 6.85 1.89 /*
/*&end
/*****************************************************************
/*
/*  LL and UR corners of writable zone for dissertation pages are:
/*  (1.15, 0.84) (6.90, 9.08)
/*
/*  Allowing one inch for a title and figure number and about 0.05
/*  inches (based on experiment) all around for slop, the limits
/*  of the mapable region are:
/*    1.20 1.89 6.85 8.98
/*
/*  map contents follow.
/*
/*****************************************************************
mapextent quads_7.5
shadetype color
linesymbol 101

RELATE add
  quad
  bino.quad
  INFO
  quad_7.5m
  quad_7.5m
  ordered
  ro
~

asel quads_7.5 poly
res quads_7.5 poly quad//meas_cnt lt 12
asel quads_7.5 poly quad//quad_7.5m ne quad//quad_7.5m
shadecolor cmy 0 0 0
polygonshades quads_7.5 1000

nsel quads_7.5 poly
res quads_7.5 poly quad//meas_cnt ge 12
res quads_7.5 poly quad//gt1_prob ge 0.0 and quad//gt1_prob lt 0.2
shadecolor cmy 100 0 100
polygonshades quads_7.5 1000

nsel quads_7.5 poly
res quads_7.5 poly quad//meas_cnt ge 12
res quads_7.5 poly quad//gt1_prob ge 0.2 and quad//gt1_prob lt 0.4
shadecolor cmy 39.6 19.6 80.4
polygonshades quads_7.5 1000

nsel quads_7.5 poly
res quads_7.5 poly quad//meas_cnt ge 12
res quads_7.5 poly quad//gt1_prob ge 0.4 and quad//gt1_prob lt 0.6
shadecolor cmy 0 0 100
polygonshades quads_7.5 1000

nsel quads_7.5 poly
res quads_7.5 poly quad//meas_cnt ge 12
res quads_7.5 poly quad//gt1_prob ge 0.6 and quad//gt1_prob lt 0.8
```

```
shadecolor cmy 0 35.3 100
polygonshades quads_7.5 1000

nsel quads_7.5 poly
res quads_7.5 poly quad//meas_cnt ge 12
res quads_7.5 poly quad//gt1_prob ge 0.8
shadecolor cmy 0 100 100
polygonshades quads_7.5 1000

asel quads_7.5 poly
polys quads_7.5

relate drop
  quad
~
/*****************************************************************
/*
/*  end map contents
/*
/*****************************************************************
&if %output% = 'ill' &then ~
  display 9999
&return


relate drop
  quad
~
&return
```

```
/*  aw_avg.aml -- calculates area weighted average values of
/*  soil parameters from STATSGO coverage onto 7.5' quadrangles.

/*  Program runs in TABLES subsystem.

/*  Establish relate environment to access mapunit and
/*  quadrangle tables.

relate add
  quad
  params.quad
  INFO
  quad_7.5m
  quad_7.5m
  ordered
  rw
  mapu
  study.mapu
  INFO
  muid
  muid
  ordered
  ro
~
/*  In quadrangle data file, clear old soil parameter values
sel params.quad
  calculate soilarea = 0
  calculate ommar = 0
  calculate thkar = 0

/*  SOILCELL is polygon intersection of quads and STATSGO mapunits
/*  select polygon table and remove non-soil polygons from
/*  calculations.
sel soilcell.pat
  reselect muid ne 'TXW'
  reselect muid cn 'TX'

  /*  sum soil area, parameter-area products
  /*  in related quadrangle data table.
  calculate quad//soilarea = quad//soilarea + area
  calculate quad//ommar = quad//ommar + area * mapu//av-av-org
  calculate quad//thkar = quad//thkar + area * mapu//avthk

/*  In quadrangle table, calculate area-weighted parameter
/*  averages by dividing summed area-parameter products by
/*  soil areas.
sel params.quad
  reselect soilarea gt 0
  calculate avsoilomm = ommar / soilarea
  calculate avsoilthk = thkar / soilarea

/*  clean up and quit
relate drop
  quad
  mapu
~
&return
```