1986

Senior Honors Project

Kelly Dyksterhouse
Western Michigan University

Senior Honors Project

Kelly Dyksterhouse

Fall 1986

The Statistical Analysis Project is a computer program that interacts with a person to discover the type of statistical test that person should use. Essentially, this program is intended to assist students, faculty and researchers in choosing the correct statistical procedure to use in a particular situation. For example, there are many different types of tests a person can use to test the mean of data but each test is used in a different situation. This program will tell you which test to use based on the situation you give it. This project was started by Anika Forsblad in 1985. She wrote the program in the standard Pascal programming language on the DEC-10 computer. She used a decision tree structure to make the decisions on which test to use. For example, the first question (How many variables does the problem involve?) has three answers. This produces three branches in the tree, each stemming from the first question and based on a different answer to the first question. This procedure is used on each question. When using the program you just traverse (or follow) the tree based on answers to each question. When you have given the program enough information about the situation, the program prints the test to use if there is one or if there is no test to use the program prints that there does not exist a test to help your situation.

When I received the program it ran on the DEC-10 computer and had a very wide range of statistical procedures on the tree. The help routine was practically unusable. It contained about 30 terms and definitions and would print all of them when asked f

help. It would not allow the user to pick which term that person wanted defined.

The first step to improving the program was to move it to a more useable environment. Dr. Thompson and I established that a better environment would be a microcomputer environment. The most widely used microcomputers are IBM and IBM compatible machines. There also is a type of Pascal called Turbo Pascal available to be used on the IBM machines. This language has a lot of flexibility and many ways to control output. Turbo also has available a tool called Database Toolbox that could solve the problem of the help procedure. After choosing the environment preferred, I needed to download the program to the IBM microcomputer. Another task to be performed on the program was for it to be partitioned into pieces small enough to fit into Turbo Pascal. The program also had to be modified to be screen oriented, and have a database created for the help procedure.

To download a program means to transfer it from a mainframe computer to a microcomputer. There are many programs available to transfer programs to microcomputers. I chose KERMIT. Using Kermit is a relatively easy procedure, although it took a long time to transfer the program since it is extremely large.

Once the program was downloaded to a diskette I needed to edit the program to add things that Turbo expected and delete things it did not expect. I ran into a problem when I tried to use Turbo's Editor. The file was too big for the editor to handle. By consulting manuals and other people more knowledgeable about Turbo, I decided to try to use "includ

files. Include files allow a programmer to split up the original file into pieces and then call those files with one line (an Include command) in the main file. This program was split into seven files.

Once those files were small enough I could edit the files. The original program asked questions and gave answers all on one screen. When the screen was full it scrolled up. Standard Pascal on the DEC-10 did not allow a programmer to control the way output was printed in relation to the screen. For example, standard Pascal did not even recognize that there was a screen to control. It treats screens as if a program was printing output to paper. Turbo could control screen output. Now I could modify the program so each question was on a different screen. Each question could begin in the same spot. Turbo also has a command where it reads one character from a screen which is easier than standard Pascal. When all the output was standardized and modified, it began to be easier to follow the what the program was doing and what input it wanted. All in all the program was easier to understand. All that was left was the hardest part--the help procedure.

To create the database I first had to read all about the Database Toolbox from Turbo. This tool provides the procedures to create a database, add records, delete records, find records and even create an index for the database. The advantage to using this database tool over most other databases is that the programmer has complete control over what is to be done with the database. The disadvantage is that the programmer has to write

procedures to do everything around the database. For example, the programmer writes the procedure to input all the data and then calls a procedure in the toolbox to put it in the database.

I used Turbo to write a short program to create a database for the help procedure. The fields for the database include a code number to categorize the different terms into five categories, the term, a number to tell how many lines the definition has and the definition itself. The categories of the terms are Terms, Tests, Measures of Association, Univariate Measures and Multivariate Techniques. I read all the information from a file that contained the old definitions and added about 30 more that I had found. Once the database was created I wrote more procedures to access it from within the Statistical Analysis Program. One of these procedures shows the user of the program the categories they can choose to receive help on. Another shows them the terms they can choose from once they select a category. Another allows them to select a term and then prints its definition. All of this sounds easy to do, but the programmer has to figure out all the information each routine wants and then get it from somewhere. Once the routines were running they created a nice looking on line help procedure. The one problem is that the routines in the toolbox are not very fast. The procedure that prints the terms available in each category is very slow.

Like any other on-going project there is plenty more that can be done on this program. One of the enhancements would be a report at the end of running of the program that printed the criteria that was just run through the procedure. For example, a

report that gave you all the answers you just gave the program in a one page format. One other idea is to allow the program to go backwards. If the user gives it a statistical test they want to use then the program prints the criteria for that test.

This project gave me a chance to learn about an environment I have not used much in my academic studies and it let me apply the skills I have learned in my studies. I used microcomputers in only one of my computer classes throughout my whole computer curriculum. I own an IBM PC-jr and used it most to connect with the VAX or DEC mainframes or to do word processing on. Through this project, I now know Turbo Pascal and the Database Toolbox fairly well. I never would have known about Include files or overlays if I had not done this project. I also applied the subjects I did learn in my computer curriculum. The Database Toolbox is based on B+ trees which I learned about in my file structures class. Turbo Pascal is also very similar to the standard Pascal I have learned. I also used the graphics routines I learned about in my computer graphics class. But the most important thing I learned was more about how statistics works. I learned how to classify the terms for the help routines. I learned what questions to ask to decide what statistical procedure to use, and I learned that computers can do more for statistics then just grind out numerical equations. Overall, I would recommend this project highly to any computer science major interested in broadening their horizons.

```
    with help do
      begin
        clrscr;
        gotoXY(5,1);
        writeln(term);
        gotoXY(5,2);
        for i := 1 to num do
          writeln(definition[i]);
      end;
    writeln;
    writeln('Press any key to continue...');
    repeat until keypressed;
    clrscr;
  end;

begin
  ans := 1;
  ch := ' ';
  clrscr;
  while ans < 6 do
    begin
      gotoXY(20,7);
      writeln ('**  MENU **');
      writeln;
      writeln('**  Choose One  **');
      writeln;
      writeln('  1   Terms');
      writeln('  2   Tests');
      writeln('  3   Measures of Association');
      writeln('  4   Univariate Measures');
      writeln('  5   Multivariate Techniques');
      writeln('  6   EXIT To Main Program');
      gotoXY(5,1);
      write ('Please enter the number of your choice:  ');
      read(kbd,ans);write(ans);
      if ans < 6 then
        begin
          nextmenu(ans,term);
          repeat
            findkey(helpindex,recnum,term);
              if ok then
                begin
                  getrec(helpfile, recnum, help);
                  displayrec(help);
                end
              else
                begin
                  gotoxy(5,1);
                  writeln('Term not found'); writeln;
                  writeln('Do you want to continue?y/n');
                  read (kbd,answer); write(answer);
                  if upcase(answer)='Y' then nextmenu(ans,term)
                                       else ok := true;
                end;
          until ok
        end;
    end;
end;

begin
  hsize := sizeof(helprec);
  hkeysize := sizeof(termtype)-1;
  initindex;
  openfile(helpfile, helpname, hsize);
  openindex(helpindex, helpndx, hkeysize, nodupe);
```

```
var
  helpfile : datafile;
  helpindex : indexfile;
  hkeysize : integer;
  nsize : integer;
  answer : char;

procedure helps;

Var
  recnum : integer;
  ans : integer;
  help : helprec;
  ch:char;
  term:termtype;

procedure nextmenu (ans:integer; var term:termtype);
  var
    i,recnum,numrecs:integer;
    help:helprec;
    term2: term2type;

  begin
    numrecs := filelen(helpfile);
    clrscr;
    gotoXY(1,5);
    case ans of
      1: writeln(ch:29, 'TERMS');
      2: writeln(ch:29, 'TESTS');
      3: writeln(ch:20, 'MEASURES OF ASSOCIATION');
      4: writeln(ch:22, 'UNIVARIATE MEASURES');
      5: writeln(ch:20, 'MULTIVARIATE TECHNIQUES');
    end;
    writeln;
    i := 1;
    for recnum := 1 to numrecs-1 do
      begin
        getrec(helpfile, recnum, help);
        if help.category = ans then
          begin
            write(help.term:30, ch:5);
            if i = 1 then i:= 0
            else begin
              i := 1;
              writeln;
            end;
          end;
      end;
    gotoXY(5,1);
    write('which term do you want defined? ');
    readln(term2);
    for i := 1 to 30 do
      if (ord(term2[i]) >= ord('a')) and (ord(term2[i]) <= ord('z')) then
        term2[i] := chr(ord(term2[i]) - 32);
    term := term2;
  end;

procedure displayrec(help: helprec);

  var
    i : integer;
```

```
const
  MaxDataRecSize = 2060;
  MaxKeyLen = 30;
  PageSize = 24;
  Order = 12;
  PageStackSize = 10;
  MaxHeight = 3;
  helpname = 'help.dat';
  helpndx = 'help.ndx';
  nodups = 0;

type
  term2type = array[1..30] of char;
  helprec = record
    helpstatus : integer;
    category : integer;
    term : termtype;
    num : integer;
    definition : definitiontype;
  end;
```

```
    textmode(bw80);
    helps;
    clrscr;
end;
```