THE TRAVELING SALESMAN PROBLEM AND THE SAS SYSTEM

Akos Felsovalyi, J.C. Penney Company, Inc.
Erzsebet Felsovalyi, Bloomfield College

ABSTRACT

The paper describes the SAS tools to solve the Traveling Salesman Problem (TSP) numerically and to present the solution in graphical form. The paper consists of two main sections:

1. Defines the problem and describes the possible methods of the solution. Introduces three SAS macros that yield the numerical solution of TSP. The solution is either exact or approximate depending upon the number of cities. Illustrates an interesting programming technique using the DATA step.

2. Presents some SAS graphic macros that generate the graphical solution of TSP. The graphic macros introduced here are more general. They draw the tour of the specified cities on the US map along with the names of the cities. The tour can be polygon-type or "hub & spoke"-type tour.

INTRODUCTION

The definition of TSP (see (1) or (8)):

There are given n cities and the distances (or costs) between these cities. A traveling salesman has to visit each of the n cities once and only once, starting from any city and returning to the starting city. The problem is to find the tour with the minimum length (or cost).

In other words: The D=(d(i,j)) distance matrix is given, where d(i,j) represents the distance between cities i and j. Find a permutation P=(i1,i2,...,in) of the integers from 1 through n that minimizes the quantity:

\[ d(i_1,i_2) + d(i_2,i_3) + ... + d(i_n,i_1). \]

The programs in this paper intend to solve the symmetric TSP, where d(i,j) = d(j,i).

The solution is very straightforward. Evaluate all the possible tours and select the one with the minimum length. Unfortunately there are (n-1)! different tours of the n cities in general, and this exact algorithm may require inordinate running time. In the symmetric case, the number of tours is cut to half, but it is a worthless relief, for the number of tours still increases factorially.

When the number of cities exceeds a threshold, the full enumeration method has no practical use. It can be seen that the problem is NP-hard ((3) or (6)) meaning that it is unlikely that a polynomially bounded exact algorithm exists. That threshold in the macros of this paper is ten cities. For more than ten cities, some kind of approximation is needed. The literature of the TSP approximations is very broad, (1) provides a good survey of the different methods.

Usually the following fundamental methods are distinguished:

1. Tour building (starts with a city and the other cities are successively included until a tour is reached),
2. Tour-to-tour improvement (starts with an initial tour which is improved by replacing some of its edges),
3. Subtour elimination (an assignment problem is solved and then its subtours are eliminated until a tour is obtained).

There are many realizations of these fundamental methods, even the combinations of them. E.g. the "Composite method" in (2), which produces a starting tour with one of the tour building algorithms and tries to find a better tour with one or more tour improvement algorithms. One of the most remarkable methods is a tour improvement method by Lin and Kernighan in (4). But their algorithm is somewhat difficult to program in our SAS environment.

Our need is an algorithm that is easy to program and performs well. By reconciling the double need, our choice falls on the insertion method. For complete description of the method consult (8):

Step 1: Start with city i.
Step 2: Find city k closest to city i and form subtour i-k.
Step 3: Select city k not in the subtour.
Step 4: Determine two successive cities, i and j such that the quantity d(i,k)+d(k,j)-d(i,j) is minimized. Insert city k into the subtour between cities i and j.
Step 5: Repeat Steps 3 and 4 until a tour is formed.

The SAS macros of this paper take each city as starting city in Step 1, and choose the best tour. In Step 3, we can select the next city to be included into the tour in different ways, defining different insertion methods. The fol-
Following three insertion methods are of interest to us:

Nearest insertion: Let the next city be the city not in the current subtour closest to the subtour.

Farthest insertion: Let the next city be the farthest city not in the subtour.

Arbitrary insertion: Let the next city be an arbitrary city not in the subtour.

The number of computations of the three insertion methods has an order of \( n^2 \).

The worst case behaviour of the methods is:

\[
\text{length of the insertion tour} \leq \frac{1}{\log_2 n} + 1,
\]

\[
\text{length of the optimum tour}
\]

where \( \log \) is logarithm to the base 2 and \( \lfloor x \rfloor \) denotes the smallest integer greater than or equal to \( x \). For the nearest insertion method we have a better upper limit:

\[
\text{length of nearest inser. tour} \leq 2(1 - \frac{1}{n}).
\]

For the comparative computation results, consult (2). Testing the methods on known problems we find (2) that the farthest and the arbitrary insertions produce solutions 3-7% above the optimum tour and the nearest insertion has an error of 10-20%. As the number of the cities increases so do the errors of the insertions. Of course, there is no general rule as far as their performances are concerned and the insertion methods work differently on different problems. Because of this, it is reasonable to try all three insertions on the same problem and take the shortest tour. See macro %TSPBINS.

1. SAS MACROS TO SOLVE TSP

This paragraph presents three SAS macros to provide the numerical solution of TSP. Macro %TSPFULL calculates the exact solution, macro %TSPINS executes one of three insertion methods and macro %TSPBINS determines the best approximation by running all three insertion methods.

1.1 INPUT AND OUTPUT DATA SETS

The input data set contains only the symmetric distance matrix. It has as many variables as the number of observations. No other variables are allowed in the input data set. Follow the practice of naming the variables with the city names. The output data set gives the solution of TSP. It has one observation and \( n+1 \) variables. The first \( n \) variables are ORDER1, ORDER2, ... ORDERn and the last one is LENGTH. The values of ORDER1, ORDER2, ... are the original variable names (the names of the cities) and their sequence determines the traveling salesman tour. The value of LENGTH gives the length of the traveling salesman tour. See 1.5 for an example of the use of the input and the output data sets.

1.2 MACRO %TSPFULL

Macro %TSPFULL determines the exact solution of TSP by evaluating all possible tours and selecting the tour with the minimum length. It works only up to ten cities. Beyond ten cities the running time would not be reasonable. The macro examines \((n-1)!/2\) tours, which are produced by Robinson's algorithm (7). The second \( n!/2 \) permutations produced by that algorithm are the reverse of the first \( n!/2 \) permutations. E.g. the permutations \((1,2,3,...,n)\) and \((n,...,2,1)\) are reverse permutations.

Parameters:

DATA = Input data set containing the symmetric distance matrix. See 1.1. Default is LAST.

OUT = Output data set containing the exact solution of TSP. See 1.1. Default is OUT.

1.3 MACRO %TSPINS

Macro %TSPINS finds an approximate solution of TSP by executing one of the three insertion methods discussed in the introduction. The macro executes the insertion algorithms for each city as starting city. See Step 1.

Parameters:

DATA = Input data set containing the symmetric distance matrix. See 1.1. Default is LAST.

OUT = Output data set containing the solution of TSP. See 1.1. Default is OUT.

METHOD = specifies the method to be used. Its value can be NINS for nearest insertion, FINS for farthest insertion and AINS for arbitrary insertion. Default is FINS.

Limitation:

The number of cities, \( n \), has to satisfy the following restriction: \( n^2 + 74n + 24 \leq \) maximum number of variables allowed by the SAS system.
1.4 Macro %TSPBINS

Macro %TSPBINS finds the best approximate solution of TSP by running all three insertion methods and choosing the best tour. Since the arbitrary insertion method may give a different result each time, the macro may run that algorithm more than once.

Parameters:
DATA = Input data set containing the symmetric distance matrix. See 1.1. Default is _LAST_.
OUT = Output data set containing the solution of TSP. See 1.1. Default is OUT.
NRAINS = number of times the arbitrary insertion algorithm has to be run. Default is 1.

Find the traveling salesman tour of these cities

Find the traveling salesman tour of these cities

1.5 EXAMPLE

There are given ten cities of the United States as shown in Figure 1. Find the shortest tour connecting those cities. Figure 2 is the SAS program calculating the solution. The program runs under CMS and the SAS macro %TSPINS resides in a file called "TSPINS.MCR". The program first sets up the input data set of the distance matrix. The distances between the ten cities are in miles and taken from (11). Then macro %TSPINS is called with METHOD=NINS specified. Finally the output data set, TOUR is printed out. The bottom part of Figure 3 shows the traveling salesman tour. Although this solution is determined by an approximate algorithm (nearest insertion), this time it coincides with the exact solution.

1.6 AN INTERESTING PROGRAMMING TECHNIQUE

The modified and renewed ARRAY command in Version 5, namely its explicit subscription feature makes it possible to use the DATA step as an ordinary program. See pages 42-50 of (9). The user now is able to write real program in the frame of a DATA step. In the previous SAS versions, the implicit subscription was awkward; the index was not visible and only one variable (defined in the ARRAY statement) could function as index. In version 5, the explicit sub-
cription is like that of any programming language. The only shortcoming that still exists is that the multidimensional array is not supported under all operating systems. However, this will not last long, based on SAS' promise: "Multidimensional explicitly subscripted arrays will be available in a future release." on page 50 of (9). For the time being, the Institute provides a macro called %MDARRAY (5) that simulates the definition of and the reference to a multidimensional array.

The following example illustrates how the DATA step can function as a real program. Let us have a data set of three variables and three observations, in other words a 3 x 3 matrix. Let us determine the trace of the matrix, which is the sum of its diagonal elements. The sample DATA step defines matrix M, reads the data into M and does the necessary calculation in a real programming fashion. Array ROW is used to read in one observation.

```
DATA A; INPUT C1 C2 C3; CARDS;
  10 38 8
  6 12 24
  15 5 3
DATA NULL; FILE PRINT;
ARRAY ROW(3) C1-C3;
MDARRAY M(3,3);
DO I=1 TO 3;
  SET A; DO J=1 TO 3;
    %M(I,J):ROW(J);
  END;
END;
TRACE:%M(1,1)+%M(2,2)+%M(3,3); *
PUT TRACE:;
```

The statements marked by *; will be replaced in the future with the natural statements M(I,J)=ROW(J); and TRACE=M(1,1)+M(2,2)+M(3,3); respectively. In this example, the trace equals to 10+12+3=25.

After we find the traveling salesman tour, we would like to present it in graphical form. The macros described in this section automatically draw the tour of the specified cities of the United States. The user has to supply only the city names and their sequence in the tour.

The macros themselves do not make the actual drawing, but create special annotate data sets that yield the tour drawn when those data sets are specified as the ANNOTATE option of the SAS/GRAPH procedures. Macro %DTOUR creates a polygon-type tour (traveling salesman tour) and macro %CTNAME places the names of the cities onto the US map. The third macro, %SUBSTOUR is a complement to macro %DTOUR because it produces a "hub & spoke"-type tour.

The macros are more general and can be used independently of TSP.

When a character type information appears in the macro invocation, it should not be surrounded with quotation marks. E.g. say %DTOUR(COLOR='GREEN') instead of %DTOUR(COLOR='GREEN').

### 2.1 Macro %DTOUR

This macro creates the annotate data set of a polygon-type tour. When the annotate option is specified with the name of the data set created by %DTOUR, the tour is drawn. The names of the cities making up the tour reside in the input data set. The coordinates of the cities are taken from the USCITY data set supplied with SAS Version 5. See pp. 445-448 of (10). The variables of the input data set are CITY = city name, CHAR 30. See p. 445 of (10).

STATE = state FIPS code. Use only for city names in more than one state. See p.445 of (10).

ORDER = defines the sequences of the cities in the tour. Its values are 1, 2, ... , n. City with ORDER=1 value is the first city in the tour, ... , city with ORDER=n value is the last city in the tour.

Variable CITY must be present in the data set, ORDER and STATE may be left out. If ORDER does not appear in the data set, the sequence of the cities in the tour is their sequence in the data set.
CLOSE
COLOR
Parameters:

```
Parameters:
DATA = input data set. See above. Default value is LAST.
OUT = annotate data set produced by the macro. Default is OUT.
CLOSE = YES/NO. Determines whether or not the tour is closed. If it YES, the first and the last cities are connected. Default is NO.
PROJECT = YES/NO. Determines which coordinates the macro has to take: the projected (YES) or the unprojected (NO). Default value is NO.

PROJECT = YES/NO. Determines which coordinates the macro has to take: the projected (YES) or the unprojected (NO). Default value is NO.

COLOR = specify the color, position, size, angle, rotation and style of a dot. See color and style values for all dots or the values can be specified differently for each city and dot. The global values are supplied in the macro call, the individual values are defined in the input data set. The individual values always overwrite the global ones.
The input data set has variables CITY and STATE (see macro %DTour) and the following variables:
COLOR = specify the color, position, size, angle, rotation and style of a dot. See color and style values for all dots or the values can be specified differently for each city and dot. The global values are supplied in the macro call, the individual values are defined in the input data set. The individual values always overwrite the global ones.
COLOR = specify the color, position, size, angle, rotation and style of a dot. See color and style values for all dots or the values can be specified differently for each city and dot. The global values are supplied in the macro call, the individual values are defined in the input data set. The individual values always overwrite the global ones.

COLOR = specify the color, position, size, angle, rotation and style of a dot. See color and style values for all dots or the values can be specified differently for each city and dot. The global values are supplied in the macro call, the individual values are defined in the input data set. The individual values always overwrite the global ones.

COLOR = specify the color, position, size, angle, rotation and style of a dot. See color and style values for all dots or the values can be specified differently for each city and dot. The global values are supplied in the macro call, the individual values are defined in the input data set. The individual values always overwrite the global ones.

COLOR = specify the color, position, size, angle, rotation and style of a dot. See color and style values for all dots or the values can be specified differently for each city and dot. The global values are supplied in the macro call, the individual values are defined in the input data set. The individual values always overwrite the global ones.
```

```
Figure 4 is an example for macro %DTour. Figure 6 lists the program statements producing the traveling salesman tour of Figure 4. In that program macro %DTour is used in conjunction with macro %CTNAME in order to obtain the tour with the names of the cities. The variable POSITION in the input data set is used only by macro %CTNAME.

2.2 MACRO %CTNAME

This macro creates the annotate data set of the specified US cities. When the data set is specified as the annotate option of a GMAP procedure, the city names are placed onto the US map along with dots at the exact locations of the cities.
The macro can apply global color, position, size, angle, rotation and style values for all cities and global color, size and style values for all dots or the values can be specified differently for each city and dot. The global values are supplied in the macro call, the individual values are defined in the input data set. The individual values always overwrite the global ones.
The input data set has variables CITY and STATE (see macro %DTour) and the following variables:
COLOR = specify the color, position, size, angle, rotation and style of a dot. See color and style values for all dots or the values can be specified differently for each city and dot. The global values are supplied in the macro call, the individual values are defined in the input data set. The individual values always overwrite the global ones.
COLOR = specify the color, position, size, angle, rotation and style of a dot. See color and style values for all dots or the values can be specified differently for each city and dot. The global values are supplied in the macro call, the individual values are defined in the input data set. The individual values always overwrite the global ones.
COLOR = specify the color, position, size, angle, rotation and style of a dot. See color and style values for all dots or the values can be specified differently for each city and dot. The global values are supplied in the macro call, the individual values are defined in the input data set. The individual values always overwrite the global ones.
COLOR = specify the color, position, size, angle, rotation and style of a dot. See color and style values for all dots or the values can be specified differently for each city and dot. The global values are supplied in the macro call, the individual values are defined in the input data set. The individual values always overwrite the global ones.
COLOR = specify the color, position, size, angle, rotation and style of a dot. See color and style values for all dots or the values can be specified differently for each city and dot. The global values are supplied in the macro call, the individual values are defined in the input data set. The individual values always overwrite the global ones.
```

```
SAS program to produce Figure 1

```
```
SAS program to produce Figure 8

DATA UT ORDER 21 STATE 31-32; CARDS;
UIAMI CHICAGO DALLAS KANSAS CITY
LOS ANGELES NEW YORK WASHINGTON
COUNTRY SOURCE 2;
DENSITY LT 2 THEN DELETE;
PROC GREUOYE DATA-US OUT-US;
BY COUNTRY;
DATA US; SET US.
IF DENSITY LT 2 THEN OUTPUT;
OAT A RESP. COUNTRY .. I: OUTPUT;
PROC DEFINE DATA-OUT OUT-LABEL,ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,
PROC DEFINE DATA-OUT OUT-LABEL, ABBR .. O, DOT"YES, OCOLOR .. RED,