Chapter 2
The ASSIGN Procedure

Chapter Table of Contents

Chapter 2
The ASSIGN Procedure

Overview

The ASSIGN procedure finds the minimum or maximum cost assignment of *m sink nodes* to *n source nodes*. The procedure can handle problems where $m =, <, > n$,.

When *n* = *m*, the number of source nodes equals the number of sink nodes and the procedure solves

$$\text{min(max)}\Sigma_{j=1}^{m}\Sigma_{i=1}^{n}c_{ij}x_{ij}$$

subject to: $\Sigma_{j=1}^{m}x_{ij} = 1$ for $i = 1, ..., n$

$\Sigma_{i=1}^{n}x_{ij} = 1$ for $j = 1, ..., m$

where $x = 0$ or 1 for $i = 1, ..., n$ and $j = 1, ..., m$

When $n < m$, the number of source nodes is less than the number of sink nodes and the procedure solves

$$\text{min(max)}\Sigma_{j=1}^{m}\Sigma_{i=1}^{n}c_{ij}x_{ij}$$

subject to: $\Sigma_{j=1}^{m}x_{ij} = 1$ for $i = 1, ..., n$

$\Sigma_{i=1}^{n}x_{ij} \leq 1$ for $j = 1, ..., m$

where $x = 0$ or 1 for $i = 1, ..., n$ and $j = 1, ..., m$

When $n > m$, the number of source nodes is greater than the number of sink nodes and the procedure solves

$$\text{min(max)}\Sigma_{j=1}^{m}\Sigma_{i=1}^{n}c_{ij}x_{ij}$$

subject to: $\Sigma_{j=1}^{m}x_{ij} \leq 1$ for $i = 1, ..., n$

$\Sigma_{i=1}^{n}x_{ij} = 1$ for $j = 1, ..., m$

where $x = 0$ or 1 for $i = 1, ..., n$ and $j = 1, ..., m$

Getting Started

The ASSIGN procedure finds the minimum or maximum cost assignment of sink nodes to source nodes. Many practical problems can be formulated in a way that is solvable by PROC ASSIGN.

Introductory Example

Consider assigning five programmers to five programming jobs. Each programmer prefers specific programming jobs over others. You can use PROC ASSIGN to assign jobs to programmers in such a way that the total preferences of the group are maximized. Suppose you ask each programmer to rank the jobs according to preference (using 1 for the most preferred job and 5 for the least preferred job). PROC ASSIGN maximizes the total preference of the group by minimizing the sum of the preferences. In the matrix that follows, each row of the matrix represents a programmer and each column represents a programming job. Each entry in the matrix is a preference ranking each programmer given each programming job.

PRGMER	JOB1	JOB2	JOB3	JOB4	JOB5
PRGMER1	4	1	3	5	2
PRGMER2	2	1	3	4	5
PRGMER3	3	2	4	1	5
PRGMER4	2	3	4	5	1
PRGMER5	4	2	3	1	5

To solve this problem using PROC ASSIGN, the data must be in a SAS data set; the solution is output to a SAS data set and no output is produced. Each observation corresponds to a programmer and contains the programming job assigned to it. In this way, the procedure identifies the assignment of the five jobs to the five programmers. To solve this assignment problem, place the preference data into a SAS data set (PREFER). Then, call PROC ASSIGN, identifying the cost variables in the input data set. The solution is output by PROC ASSIGN to a SAS data set (PREFER1) and displayed with the PRINT procedure. The following statements produce Figure 2.1:

```
title 'Assigning Programming Jobs to Programmers';

data prefer;
   input  prgmer $ job1-job5;
   datalines;
PRGMER1 4 1 3 5 2
PRGMER2 2 1 3 4 5
PRGMER3 3 2 4 1 5
PRGMER4 2 3 4 5 1
PRGMER5 4 2 3 1 5
;
```

```
proc assign data=prefer out=prefer1;
   cost job1-job5;
   id prgmer;
run;

proc print data=prefer1;
   sum _fcost_;
run;
```

The following note is written to the SAS log:

```
NOTE: The minimum cost assignment costs 8.
```

```
                  Assigning Programming Jobs to Programmers

Obs    prgmer     job1    job2    job3    job4    job5    _ASSIGN_    _FCOST_

 1     PRGMER1      4       1       3       5       2       job2          1
 2     PRGMER2      2       1       3       4       5       job1          2
 3     PRGMER3      3       2       4       1       5       job4          1
 4     PRGMER4      2       3       4       5       1       job5          1
 5     PRGMER5      4       2       3       1       5       job3          3
                                                                    =======
                                                                          8
```

Figure 2.1. Assigning Programming Jobs to Programmers

The solution, given in column _ASSIGN_, shows how each programming job should be assigned to each worker in order to minimize the assignment cost, which is equivalent to maximizing the worker preferences. The _FCOST_ column expresses in units of preference the cost of the assignment. The SUM statement in the PRINT procedure is used to total the assignment cost.

Syntax

PROC ASSIGN *options* **;**
  BY *variables* **;**
  COST *variables* **;**
  ID *variable* **;**

The statements and options available on PROC ASSIGN are discussed in the order in which they appear in the preceding list of syntax elements.

Functional Summary

The options available with PROC ASSIGN and its statements are summarized by purpose in the tables that follow.

Table 2.1. Variable Lists

Description	Statement	Option
process data in groups	BY	
cost variables	COST	
source node names	ID	

Table 2.2. Data Set Options

Description	Statement	Option
input data set	ASSIGN	DATA=
output data set containing the solution	ASSIGN	OUT=

Table 2.3. Optimization Control Options

Description	Statement	Option
scaling factor for input cost data	ASSIGN	DEC=
find maximum cost assignment	ASSIGN	MAXIMUM

PROC ASSIGN Statement

PROC ASSIGN *options* **;**

The PROC ASSIGN statement invokes the procedure. The following options can appear in the PROC ASSIGN statement.

Data Set Options

DATA=*SAS-data-set*
names the SAS data set that contains the network specification. If the DATA= option is omitted, the most recently created SAS data set is used.

OUT=*SAS-data-set*

specifies a name for the output data set. If the OUT= option is omitted, the SAS System creates a data set and names it according to the DATA*n* convention. Refer to the “SAS Statements Used in the Data Set” section in base SAS documentation for more information.

Optimization Control Options

DEC=*n*

specifies a scaling factor for the input cost data. The input data are scaled by 10^n. The default value of n is 3. For more information, see the discussion on scaling in the “Details” section.

MAXIMUM

specifies that the objective is to find an assignment that maximizes the sum of the costs. By default, PROC ASSIGN minimizes the sum of the costs.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC ASSIGN to obtain separate solutions on problems in groups defined by the BY variables. When you use a BY statement, the procedure expects the input data to be sorted in ascending order of the BY variables. If your input data set is not sorted, use the SORT procedure with a similar BY statement to sort the data, or, if appropriate, use the BY statement option NOTSORTED or DESCENDING. See Example 2.4 in the “Examples” section. For more information, refer to the discussion of the BY statement in base SAS documentation.

COST Statement

COST *variables* **;**

The COST statement identifies the variables to be interpreted as sink nodes in the input DATA= data set. The values of a COST variable are the costs (or preferences) of assigning each source node (named in the ID variable) to the sink node identified with the COST variable.

If the value of a COST variable is missing, then that particular assignment between source and sink node is infeasible. If you do not use a COST statement, then all numeric variables not specified in the ID or BY statement are assumed to be cost variables.

To find an assignment that maximizes profit instead of minimizing cost, include the MAXIMUM option in the PROC ASSIGN statement and let the COST variables represent profit instead of cost. The COST variables must be numeric. See Example 2.1 for an illustration of the COST statement.

ID Statement

ID *variable* **;**

The ID statement identifies a variable in the input DATA= data set that gives the names of the source nodes. The ID variable can be character or numeric.

Details

Missing Values

Because the value of a cost variable is interpreted as the cost of an assignment, a missing value for a cost variable is assumed to mean that the assignment is not allowed. Refer to Example 2.1 for an illustration of a data set with missing values.

Output Data Set

The output data set contains the *m* cost variables in the input data set, any variables identified in the ID statement, and two new variables named _ASSIGN_ and _FCOST_. The variable named _ASSIGN_ is a character variable containing the names of the sink nodes (names of COST variables) assigned to the source nodes (values of the ID variable). The variable named _FCOST_ is a numeric variable containing the costs of assigning the sink nodes to the source nodes. Note that the values of the *m* cost variables in the output data set reflect any effects of scaling performed by PROC ASSIGN.

The Objective Value

If the problem is infeasible, an error message is written to the SAS log. Otherwise, the value of the objective function

$$\Sigma_{j=1}^{m}\Sigma_{i=1}^{n}c_{ij}x_{ij}$$

under the optimal assignment is reported on the SAS log.

Macro Variable _ORASSIG

On termination, the ASSIGN procedure defines a macro variable named _ORASSIG. This variable contains a character string that indicates the status of the procedure on termination and gives the objective value at termination. The form of the _ORASSIG character string is

STATUS=$charstr$ OBJECTIVE=$objective$

where $charstr$ can be any one of the following.

- SUCCESSFUL
- INFEASIBLE
- MEMORY_ERROR
- IO_ERROR
- SYNTAX_ERROR
- SEMANTIC_ERROR
- BADDATA_ERROR
- UNKNOWN_ERROR

This information is useful when PROC ASSIGN comprises one step in a larger program that needs to identify just how the ASSIGN procedure terminated. Because _ORASSIG is a standard SAS macro variable, it can be used in the ways that all macro variables can be used. See the *SAS Guide to Macro Processing* for more information. Example 2.2 illustrates the method to write the _ORASSIG variable to the log.

Scaling

PROC ASSIGN uses a variant of the *out-of-kilter algorithm*. Integral cost data are important for maintaining a rapid rate of convergence with this algorithm. To assure integrality, the cost data are automatically scaled by DEC= decimal places on input to PROC ASSIGN. If this scaling can result in loss of accuracy in the input data, a warning is written to the log indicating a nonzero fractional component in the data after scaling. The output data set produced by PROC ASSIGN contains the scaled input cost data rescaled to its original order of magnitude. Since the input cost data and the output cost data may differ because of scaling, you can use this difference as a measure of how poorly the input cost data were scaled.

Examples

The following examples illustrate some of the capabilities of PROC ASSIGN. These examples, together with the other SAS/OR examples, can be found in the SAS sample library.

Example 2.1. Assigning Subcontractors to Construction Jobs

This example shows how PROC ASSIGN can be used to maximize an objective function. Consider a construction project that consists of nine jobs. Because of the nature of the project, each job must be performed by a different subcontractor. Each job is bid upon by twelve subcontractors. The matrix that follows shows the expected profit to the contractor if each job is given to each subcontractor. Each row in the matrix represents a different job, and each column represents a different subcontractor.

```
SUBCONTRACTOR 1  2  3  4  5  6  7  8  9 10 11 12
________________________________________________
   JOB1    |  79 24 13 53 47 66 85 17 92 47 46 13
   JOB2    |  43 59 33 95 55 97 34 55 84 94 26 56
   JOB3    |  29 52  0 27 13 33  0 11 71 86  6 76
   JOB4    |  88 83 64 72  0 67 27 47 83 62 35 38
   JOB5    |  65 90 56 62 53 91 48 23  6 89 49 33
   JOB6    |  44 79 86 93 71  7 86 59  0 56 45 59
   JOB7    |  35 51 -9 91 39 32  3 12 79 25 79 81
   JOB8    |  50 12 59 32 23 64 20 94 97 14 11 97
   JOB9    |  25 17 39  . 38 63 87 14  4 18 11 45
```

The negative profit in the third column means that if job 7 is awarded to subcontractor 3, the contractor loses money. The missing value in the fourth column means that subcontractor 4 did not bid on job 9. PROC ASSIGN treats a missing value differently from the way it treats a 0. While it is possible that an optimal assignment could include a 0 (or even a negative) contribution to profit, the missing value is never included in an assignment. In this case, subcontractor 4 is never awarded job 9, regardless of the profit structure.

You can use PROC ASSIGN to find how the contractor should award the jobs to the subcontractors to maximize his profit. First, put the data in a SAS data set. Then, call PROC ASSIGN using the MAXIMUM option.

The following statements produce Output 2.1.1:

```
title 'Assigning Subcontractors to Construction Jobs';

data profit;
   input job $ subcon1-subcon12;
   datalines;
JOB1 79 24 13 53 47 66 85 17 92 47 46 13
JOB2 43 59 33 95 55 97 34 55 84 94 26 56
JOB3 29 52  0 27 13 33  0 11 71 86  6 76
JOB4 88 83 64 72  0 67 27 47 83 62 35 38
JOB5 65 90 56 62 53 91 48 23  6 89 49 33
JOB6 44 79 86 93 71  7 86 59  0 56 45 59
JOB7 35 51 -9 91 39 32  3 12 79 25 79 81
JOB8 50 12 59 32 23 64 20 94 97 14 11 97
JOB9 25 17 39  . 38 63 87 14  4 18 11 45
;

proc assign maximum data=profit;
   cost subcon1-subcon12;
   id job;
run;

proc print;
   sum _fcost_;
run;
```

The cost of the optimal assignment written to the SAS log is

```
NOTE: The maximum return assignment yields 814.
```

This means that the contractor can expect a profit of $814 if he follows the optimal assignment.

Output 2.1.1. Assigning Subcontractors to Construction Jobs

```
                 Assigning Subcontractors to Construction Jobs

                                                      s   s   s      _
          s   s    s   s   s   s   s   s   s   u   u   u      A       _
          u   u    u   u   u   u   u   u   u   b   b   b      S       F
          b   b    b   b   b   b   b   b   b   c   c   c      S       C
          c   c    c   c   c   c   c   c   c   o   o   o      I       O
 O   j    o   o    o   o   o   o   o   o   o   n   n   n      G       S
 b   o    n   n    n   n   n   n   n   n   n   1   1   1      N       T
 s   b    1   2    3   4   5   6   7   8   9   0   1   2      _       _

 1  JOB1  79  24   13  53  47  66  85  17  92  47  46  13  subcon9    92
 2  JOB2  43  59   33  95  55  97  34  55  84  94  26  56  subcon6    97
 3  JOB3  29  52    0  27  13  33   0  11  71  86   6  76  subcon10   86
 4  JOB4  88  83   64  72   0  67  27  47  83  62  35  38  subcon1    88
 5  JOB5  65  90   56  62  53  91  48  23   6  89  49  33  subcon2    90
 6  JOB6  44  79   86  93  71   7  86  59   0  56  45  59  subcon3    86
 7  JOB7  35  51   -9  91  39  32   3  12  79  25  79  81  subcon4    91
 8  JOB8  50  12   59  32  23  64  20  94  97  14  11  97  subcon12   97
 9  JOB9  25  17   39   .  38  63  87  14   4  18  11  45  subcon7    87
                                                                     ===
                                                                     814
```

Note that three subcontractors, SUBCON5, SUBCON8, and SUBCON11, are not assigned to any jobs.

Example 2.2. Assigning Construction Jobs to Subcontractors

Suppose that the data from Example 2.1 are transposed so that variables are jobs. Then each observation contains the profit from awarding each job to a single subcontractor. The following program finds the maximum profit assignment and produces Output 2.2.1.

```
title 'Assigning Construction Jobs to Subcontractors';

data profit;
   input subcont $ job1-job9;
   datalines;
SUBCON1   79   43   29   88   65   44   35   50   25
SUBCON2   24   59   52   83   90   79   51   12   17
SUBCON3   13   33    0   64   56   86   -9   59   39
SUBCON4   53   95   27   72   62   93   91   32    .
SUBCON5   47   55   13    0   53   71   39   23   38
SUBCON6   66   97   33   67   91    7   32   64   63
SUBCON7   85   34    0   27   48   86   32    0   87
SUBCON8   17   55   11   47   23   59   12   94   14
SUBCON9   92   84   71   83    6    0   79   97    4
SUBCON10  47   94   86   62   89   56   25   14   18
SUBCON11  46   26    6   35   49   45   79   11   11
SUBCON12  13   56   76   38   33   59   81   97   45
;

proc assign maximum data=profit;
   cost job1-job9;
   id subcont;
run;

proc print;
   sum _fcost_;
run;
```

The cost of the optimal assignment written to the SAS log is

```
   NOTE: The maximum return assignment yields 814.
```

This means that the contractor can expect a profit of $814 if the optimal assignment is followed.

Output 2.2.1. Assigning Subcontractors to Construction Jobs

```
                 Assigning Construction Jobs to Subcontractors

 Obs subcont  job1 job2 job3 job4 job5 job6 job7 job8 job9 _ASSIGN_ _FCOST_

   1 SUBCON1   79   43   29   88   65   44    35  50   25    job4      88
   2 SUBCON2   24   59   52   83   90   79    51  12   17    job5      90
   3 SUBCON3   13   33    0   64   56   86    -9  59   39    job6      86
   4 SUBCON4   53   95   27   72   62   93    91  32    .    job7      91
   5 SUBCON5   47   55   13    0   53   71    39  23   38               0
   6 SUBCON6   66   97   33   67   91    7    32  64   63    job2      97
   7 SUBCON7   85   34    0   27   48   86    32   0   87    job9      87
   8 SUBCON8   17   55   11   47   23   59    12  94   14               0
   9 SUBCON9   92   84   71   83    6    0    79  97    4    job1      92
  10 SUBCON10  47   94   86   62   89   56    25  14   18    job3      86
  11 SUBCON11  46   26    6   35   49   45    79  11   11               0
  12 SUBCON12  13   56   76   38   33   59    81  97   45    job8      97
                                                                    =======
                                                                      814
```

The macro variable _ORASSIG defined by PROC ASSIGN contains information regarding the termination of the procedure.

This information can be useful when you use PROC ASSIGN as part of a larger SAS program. For example, the following information is written to on the log using the macro language with the statement:

```
%put  &_orassig;
```

On the log the following appears

Output 2.2.2. _ORASSIG macro variable

```
STATUS=SUCCESSFUL OBJECTIVE=814.
```

Example 2.3. Minimizing Swim Times

A swimming coach needs to assign male and female swimmers to each stroke of a medley relay team. The swimmers' best times for each stroke are stored in a SAS data set. The ASSIGN procedure is used to evaluate the times and to match strokes and swimmers to minimize the total relay swim time. The following statements produce Output 2.3.1:

```
title 'Assigning Strokes Using the BY Statement';

data relay;
   input name $ sex $ back breast fly free;
   datalines;
SUE     F 35.1 36.7 28.3 36.1
KAREN   F 34.6 32.6 26.9 26.2
JAN     F 31.3 33.9 27.1 31.2
ANDREA  F 28.6 34.1 29.1 30.3
CAROL   F 32.9 32.2 26.6 24.0
ELLEN   F 27.8 32.5 27.8 27.0
JIM     M 26.3 27.6 23.5 22.4
MIKE    M 29.0 24.0 27.9 25.4
SAM     M 27.2 33.8 25.2 24.1
CLAYTON M 27.0 29.2 23.0 21.9
;

proc assign out=fast;
   cost back--free;
   id name;
   by sex;

proc print;
   by sex;
   sum _fcost_;
run;
```

Output 2.3.1. Assigning Strokes Using the BY Statement

```
                    Assigning Strokes Using the BY Statement

------------------------------------ sex=F ------------------------------------

     Obs    name      back    breast     fly    free    _ASSIGN_    _FCOST_

       1    SUE       35.1     36.7     28.3    36.1                    0.0
       2    KAREN     34.6     32.6     26.9    26.2     breast        32.6
       3    JAN       31.3     33.9     27.1    31.2     fly           27.1
       4    ANDREA    28.6     34.1     29.1    30.3                    0.0
       5    CAROL     32.9     32.2     26.6    24.0     free          24.0
       6    ELLEN     27.8     32.5     27.8    27.0     back          27.8
     ---                                                            -------
     sex                                                              111.5


------------------------------------ sex=M ------------------------------------

    Obs    name       back    breast     fly    free    _ASSIGN_    _FCOST_

      7    JIM        26.3     27.6     23.5    22.4     free          22.4
      8    MIKE       29.0     24.0     27.9    25.4     breast        24.0
      9    SAM        27.2     33.8     25.2    24.1     back          27.2
     10    CLAYTON    27.0     29.2     23.0    21.9     fly           23.0
    ---                                                             -------
    sex                                                                96.6
                                                                    =======
                                                                      208.1
```

On the basis of this solution, Jim will swim freestyle, Mike will swim breast stroke, Sam will swim back stroke, and Clayton will swim butterfly. For the women's team, Karen will swim breast stroke, Jan will swim butterfly, Carol will swim freestyle, and Ellen will swim back stroke.

Example 2.4. Using PROC ASSIGN with a BY Statement

A major beverage company wants to assign TV commercials to television commercial time slot openings in a way that maximizes the overall effectiveness of its television advertising. The time slots in this example begin at 7:00 on a Saturday morning and run hourly through 3:00 p.m. A combination of Nielsen TV ratings and market research testing produces an effectiveness rating for each time slot and commercial combination. The commercials are of three types: children, lifestyle, and sports. The company is willing to show up to three commercials in each time slot as long as the commercials are of different types. Which commercials should be assigned to which time slots in order to maximize the total effectiveness of its television advertising campaign? Data are missing for those time slots where certain programs are not available; for instance, no sports shows are presented during the 7:00 a.m. time slot.

The following statements produce Output 2.4.1:

```
 title 'Assigning Televison Commercials Using the BY Statement';

 data beverage;
    input commercl $ type $ slot1-slot9;
    datalines;
 COMM1 KIDS 27.2 32.8 30.4 31.5 20.9 19.8   .    .    .
 COMM2 KIDS 37.4 33.5 38.4 32.4 25.6 27.2   .    .    .
 COMM3 KIDS 32.5 31.9 34.6 34.5 26.7 28.3   .    .    .
 COMM4 LIFEST .  22.6 25.9 25.3 26.4 28.3 29.1 22.2 20.2
 COMM5 LIFEST .  25.1 36.6 36.8 38.2 33.5 33.2 33.1 30.1
 COMM6 LIFEST .  20.2 31.3 29.3 24.6 25.1 20.0 22.4 23.1
 COMM7 SPORTS .    .  25.1 26.1 28.3 36.1 29.4 31.7 34.5
 COMM8 SPORTS .    .  24.7 27.2 36.4 31.2 28.7 33.2 33.1
 COMM9 SPORTS .  20.2 20.4 20.2 25.6 37.8 35.6 32.4 34.3
 ;

proc assign maximum out=newslots;
   cost slot1-slot9;
   id commercl;
   by type;
run;

proc print;
   by type;
   sum _fcost_;
run;
```

Output 2.4.1. Assigning Television Commercials using the BY Statement

```
             Assigning Televison Commercials Using the BY Statement

---------------------------------- type=KIDS -----------------------------------

            c                                                             _
            o                                                             A      _
            m                                                             S      F
            m      s     s     s     s     s     s   s  s  s   S      C
            e      l     l     l     l     l     l   l  l  l   I      O
       O    r      o     o     o     o     o     o   o  o  o   G      S
       b    c      t     t     t     t     t     t   t  t  t   N      T
       s    l      1     2     3     4     5     6   7  8  9   _      _

       1  COMM1  27.2  32.8  30.4  31.5  20.9  19.8  .  .  .  slot2   32.8
       2  COMM2  37.4  33.5  38.4  32.4  25.6  27.2  .  .  .  slot3   38.4
       3  COMM3  32.5  31.9  34.6  34.5  26.7  28.3  .  .  .  slot4   34.5
     ----                                                            -----
     type                                                            105.7


--------------------------------- type=LIFEST ---------------------------------

         c                                                               _
         o                                                               A      _
         m                                                               S      F
         m    s    s     s     s     s     s     s     s     s     S      C
         e    l    l     l     l     l     l     l     l     l     I      O
    O    r    o    o     o     o     o     o     o     o     o     G      S
    b    c    t    t     t     t     t     t     t     t     t     N      T
    s    l    1    2     3     4     5     6     7     8     9     _      _

    4  COMM4  .  22.6  25.9  25.3  26.4  28.3  29.1  22.2  20.2  slot7  29.1
    5  COMM5  .  25.1  36.6  36.8  38.2  33.5  33.2  33.1  30.1  slot5  38.2
    6  COMM6  .  20.2  31.3  29.3  24.6  25.1  20.0  22.4  23.1  slot3  31.3
  ----                                                                  ----
  type                                                                  98.6


--------------------------------- type=SPORTS ---------------------------------

         c                                                               _
         o                                                               A      _
         m                                                               S      F
         m    s    s     s     s     s     s     s     s     s     S      C
         e    l    l     l     l     l     l     l     l     l     I      O
    O    r    o    o     o     o     o     o     o     o     o     G      S
    b    c    t    t     t     t     t     t     t     t     t     N      T
    s    l    1    2     3     4     5     6     7     8     9     _      _

    7  COMM7  .     .  25.1  26.1  28.3  36.1  29.4  31.7  34.5  slot9   34.5
    8  COMM8  .     .  24.7  27.2  36.4  31.2  28.7  33.2  33.1  slot5   36.4
    9  COMM9  .  20.2  20.4  20.2  25.6  37.8  35.6  32.4  34.3  slot6   37.8
  ----                                                                  -----
  type                                                                  108.7
                                                                        =====
                                                                        313.0
```

On the basis of this survey, this company has decided to drop commercial advertising from the 7:00 a.m. (slot1) and 2:00 p.m. (slot8) time slots.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/OR® User's Guide: Mathematical Programming, Version 8*, Cary, NC: SAS Institute Inc., 1999. 566 pp.

SAS/OR® User's Guide: Mathematical Programming, Version 8

Copyright © 1999 SAS Institute Inc., Cary, NC, USA.

ISBN 1–58025–491–8

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999