Let’s explore SAS PROC OPTMODEL

ESUG meeting April 2016

Gunjan Kaur

Goals of the presentation

- Introduction to Mathematical Optimization
- Syntax of PROC OPTMODEL
- Building an optimization problem using PROC OPTMODEL

Optimization can answer all these questions….

What PRICE should I CHARGE for product A ?

How much MARKDOWN should I give so that the customer buys my product ?

How much cash should a particular ATM STORE ?

Which customers deserve a CREDIT LIMIT INCREASE ?

What product needs to be PROMOTED in local area B ?

How much product X INVENTORY should I keep in store Y ?

Which PROMOTION should I run in store Z ?

How does a Mathematical Optimization problem look like ?

Optimization => Maximizing or minimizing an objective function subject to certain constraints

Minimize OR Maximize f(x) → **Objective Function Example: $x_1+2x_2+3x_3$**

Subject to g(x) {≤, =, ≥} b → **Constraint: : $x_1+x_3<5$**

$l \leq x \leq u$ → **Lower and upper bound on different decision variables : $2<x_1<10$; $x_2 >= 0$**

Some types of optimization problems :
Linear programming , Integer Linear Programming, Mixed Integer Linear programming, Non Linear programming

Using Proc OPTMODEL to build, solve and maintain optimization problems

Seven solvers available to solve your problem :

Solver	Problem
LP	Linear programming
MILP	Mixed integer linear programming
NLPU	Unconstrained nonlinear programming
NLPC	General nonlinear programming
IPNLP	Interior point nonlinear programming
QP	Quadratic programming (experimental)
SQP	Sequential quadratic programming

- User friendly

Algebra mimics programming syntax

- Models can read data from SAS data sets as well as output data.
- Models can be easily modified and corrected in the proc itself

SYNTAX – Declaration + Programming Statements

PROC OPTMODEL options ;
CONSTRAINT constraints ;
IMPVAR optimization expression declarations ;
MAX objective ;
MIN objective ;
NUMBER parameter declaration;
PROBLEM problem declaration ;
SET < types > parameter declarations ;
STRING parameter declarations ;
VAR variable declarations ;
Assignment parameter = expression ;
CALL name (expressions) ;
CLOSEFILE files ;
CONTINUE ;
CREATE DATA SAS-data-set FROM columns ;
DO ; statements ; END ;
DO variable = specifications ; statements ; END ;
DO UNTIL (logic) ; statements ; END ;
DO WHILE (logic) ; statements ; END ;
DROP constraint ;
EXPAND name / options ;

```
var x, y; number low;
con a: low <= x+y <= low+10;

impvar total_weight = sum{p in PRODUCTS} Weight[p]*x[p];
con prod1_limit: Weight['Prod1'] * x['Prod1'] <= 0.3 * total_weight;

string dn{1..5} = [Monday Tuesday Wednesday Thursday Friday];

var x init 0.5 >= 0 <= 1;

proc optmodel;
number m = 7, n = 5;
create data example from m n ratio=(m/n);
proc print; run;
```

SYNTAX - Declaration + Programming Statements

FILE file ;
FIX variable = expression ;
FOR { index-set } statement ;
IF logic THEN statement ; ELSE statement ;
LEAVE ;
;
PERFORMANCE options ;
PRINT print items ;
PUT put items ;
QUIT ;
READ DATA SAS-data-set INTO columns ;
RESET OPTIONS options ;
RESTORE constraint ;
SAVE MPS SAS-data-set (OBJECTIVE | OBJ) name ;
SAVE QPS SAS-data-set (OBJECTIVE | OBJ) name ;
SOLVE WITH solver OBJECTIVE name RELAXINT / options ;
STOP ;
SUBMIT arguments / options ;
UNFIX variable = expression ;
USE PROBLEM problem ;

```
proc optmodel;
var x{1..10};
fix x = 0;
fix x[10] = 1;
```

```
data invdata;
input item $ invcount;
datalines; table 100 sofa 250 chair 80 ;
```

```
proc optmodel;
set<string> Items;
number invcount{Items};
read data invdata into Items=[item] invcount;
print invcount;
```

Building an optimization problem : Example

A manufacturer is trying to find out how much of Product A, B, and C he should produce ?

He knows the profit of

he each product…..

Product	Profit/pound
A	$0.70
B	$0.45
C	$0.50

He knows the steps and their costs too….

	Available Time (sec)	Product A (sec)	Product B (sec)	Product C (sec)
Step 1	25,000	10	10	15
Step 2	25,000	20	10	10
Step 3	20,000	20	15	5
Step 4	20,000	15	13	5

```
proc optmodel;
  set <string> Products;
  set <string> Processes;
  num Profit{Products};
  num AvailableTime{Processes};
  num RequiredTime{Products,Processes};

  var Amount{Products} ;

  maximize TotalProfit = sum{p in Products} Profit[p]*Amount[p];
  con Availability{r in Processes}:
    sum{p in Products} RequiredTime[p,r]*Amount[p] <= AvailableTime[r];

  read data Products into Products=[name] Profit;
  read data Processes into Processes=[name] AvailableTime=Available_time
    {p in Products} <RequiredTime[p,name]= col(p)>;

  solve with lp / solver = primal_spx;
  print Amount;
quit;
```

Declaring data

Declaring variable whose value needs to be determined

Objective function

Constraints

Read in data to populate the model

Print the solution

Problem Summary	
Objective Sense	Maximization
Objective Function	TotalProfit
Objective Type	Linear
Number of Variables	3
Bounded Above	0
Bounded Below	0
Bounded Below and Above	0
Free	3
Fixed	0
Number of Constraints	4
Linear LE (<=)	4
Linear EQ (=)	0
Linear GE (>=)	0
Linear Range	0
Constraint Coefficients	12

The SAS System

The OPTMODEL Procedure

Solution Summary	
Solver	Primal Simplex
Objective Function	TotalProfit
Solution Status	Optimal
Objective Value	1066.6666667
Iterations	3
Primal Infeasibility	0
Dual Infeasibility	0
Bound Infeasibility	0

[1]	Amount
A	583.33
B	166.67

Resources :

- SAS/OR manual
- SAS course on optimization
- Whitepapers – Lex Jansen.com

Thank you !