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 \( f(x) \)

Subject to \( g(x) \) \{≤, =, ≥\} \( b \)

\( l \leq x \leq u \)

Objective Function Example: \( x_1 + 2x_2 + 3x_3 \)

Constraint: \( : x_1 + x_3 < 5 \)

Lower and upper bound on different decision variables: \( 2 < x_1 < 10 \); \( x_2 \geq 0 \)

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:

<table>
<thead>
<tr>
<th>Solver</th>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>Linear programming</td>
</tr>
<tr>
<td>MILP</td>
<td>Mixed integer linear programming</td>
</tr>
<tr>
<td>NLPU</td>
<td>Unconstrained nonlinear programming</td>
</tr>
<tr>
<td>NLPC</td>
<td>General nonlinear programming</td>
</tr>
<tr>
<td>IPNLP</td>
<td>Interior point nonlinear programming</td>
</tr>
<tr>
<td>QP</td>
<td>Quadratic programming (experimental)</td>
</tr>
<tr>
<td>SQP</td>
<td>Sequential quadratic programming</td>
</tr>
</tbody>
</table>

- 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
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 each product.....

<table>
<thead>
<tr>
<th>Product</th>
<th>Profit/pound</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$0.70</td>
</tr>
<tr>
<td>B</td>
<td>$0.45</td>
</tr>
<tr>
<td>C</td>
<td>$0.50</td>
</tr>
</tbody>
</table>

He knows the steps and their costs too....

<table>
<thead>
<tr>
<th>Step</th>
<th>Available Time (sec)</th>
<th>Product A (sec)</th>
<th>Product B (sec)</th>
<th>Product C (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>25,000</td>
<td>10</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>Step 2</td>
<td>25,000</td>
<td>20</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Step 3</td>
<td>20,000</td>
<td>20</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>Step 4</td>
<td>20,000</td>
<td>15</td>
<td>13</td>
<td>5</td>
</tr>
</tbody>
</table>
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];  \textbf{Objective function}
    con Availability{r in Processes}:
        sum{p in Products} RequiredTime[p,r]*Amount[p] <= AvailableTime[r];  \textbf{Constraints}

    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;
Resources:

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

Thank you!