Paper 165-2012

Bringing Optimization to the Business: Interfacing SAS/OR® with SAS® Stored Processes, Microsoft Excel, SAS® Enterprise Guide®, SAS® Forecast Studio, and Microsoft Project

Andrew Pease, SAS Institute Inc., Cary, NC

ABSTRACT

SAS/OR® software provides a powerful array of optimization, simulation, and project scheduling techniques to identify the actions that will produce the best results, while operating within resource limitations and tight restrictions. This paper focuses on ways to get this information to business users in an intuitive and interactive way. Via SAS® Stored Processes, users can change parameters and run different configurations of a base SAS/OR program. These stored processes can be surfaced via the SAS® Add-In for Microsoft Office, SAS® Enterprise Guide®, and SAS® Forecast Studio. Also, via Microsoft Project macros, Microsoft Project users can export parameters to SAS/OR, which produces optimized planning, and then push back to the Microsoft Project user. The technical setup of these various channels is explored in demonstrations.

INTRODUCTION

Optimization has often been described as prescriptive analytics and as the top level on the data and analytic continuum. Despite the perceived and demonstrated value that it can bring, it remains an underused analytic capability in the SAS® analytics arsenal. Part of the problem is because of the perceived and real complexity in operations research problem formulation and execution. However, this complexity can be masked to the business user by presenting optimization algorithms via simplified, ready-for-business interfaces.

This paper presents the necessary steps for developing interfaces to SAS/OR programs for use by business users.

The steps include the following:

1. State the business challenge.
2. Formulate the mathematical model.
3. Program the mathematical model.
4. Parameterize the program.
5. Create a SAS stored process.
6. Use it.

For the purposes of this paper, the descriptions of the first three steps are brief. These steps are generic to any operations research application, and they are better outlined elsewhere. That said, because these steps are important to any successful business utilization, the key points are emphasized.

For an introduction to optimization with SAS software, see the “Optimization with SAS® Software” white paper, available at http://www.sas.com/reg/wp/corp/17649.

For more information about SAS stored processes, see http://support.sas.com/documentation/cdl/en/stpug/62758/PDF/default/stpug.pdf.

STATE THE BUSINESS CHALLENGE AND FORMULATE THE MATHEMATICAL MODEL

Optimization is at the top level on the data and analytic continuum.

Figure 1. Data and Analytic Continuum

Nothing can be optimized without first stating the business challenge. In other words, *“What is it that we want to optimize?”* Initially, stating the business challenge has little to do with math and even less to do with programming. It is a simple, plain language description of the business system to be modeled, including the objective, control factors, and constraints. It might at first seem simplistic to mention this as a key step in the optimization process. But, in fact, many optimization efforts fail because of a lack of consensus or completeness in defining the business system and the objective.

The key components to a complete business challenge statement have been described in the "Aligned Resource Optimization" white paper, available at http://www.sas.com/reg/wp/corp/4183. In this paper, the following model is presented, including a stated objective, decision variables, and constraints.

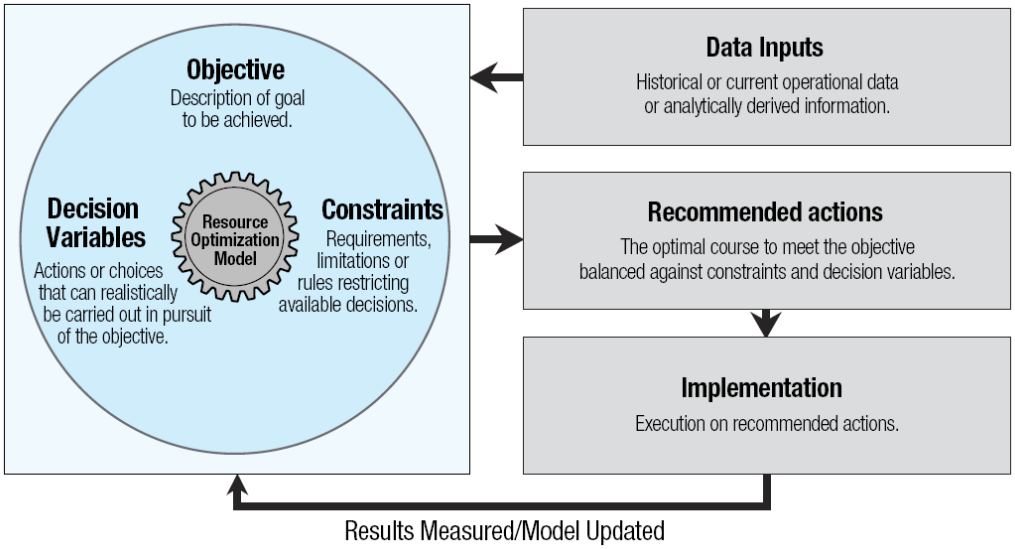


Figure 2. Resource Optimization Model

In the following table, examples are provided. In each of the specific business challenge statements, there is a clearly stated objective highlighted in green, with various decision variables highlighted in blue, and constraints highlighted in red.

Business Domain	High-Level Challenge Statement	Specific Business Challenge Statement
Supply Chain	Reduce the cost of lost or dumped stock.	Minimize waste by setting optimal stock levels to meet stated service levels.
	Increase product availability.	Maximize product availability by setting optimal stock levels while respecting inventory cost and infrastructure limitations.
	Reduce supplier costs.	Minimize supplier costs by using an optimal supplier mix based on cost and ability to deliver while respecting stated service levels.
Marketing	Increase return on marketing investment.	Maximize campaign response rate by setting optimal contact schedule (including channel, message, and timing) while respecting budget constraints and contact policies.
	Decrease marketing costs.	Minimize marketing costs by setting optimal contact schedule (including channel, message, and timing) while respecting established contact policies.
	Re-engineer contact policies to have a more customer-focused approach.	Minimize total number of contacts by setting optimal contact schedule (including channel, message, and timing) while respecting established contact policies and individual campaign objectives.
Production	Increase production line efficiency.	Minimize system down time due to production shifts by setting optimal production times while respecting projected demand amounts and operational constraints.
	Reduce production costs.	Reduce production costs by minimizing waste by defining optimal use of raw materials while respecting projected demand amounts and operational constraints.
Project Management	Reduce project time.	Minimize elapsed project time by allowing for concurrent project tasks in the scheduling while respecting required task completion order.

Table 1. Example Challenge Statements

Reflect on the examples in this table as you develop a business challenge statement and problem formulation for a single, real-life case.

Optimizing a Credit Marketing Campaign

The marketing department of a major bank is looking to expand its credit business.

This might not seem like an appropriate objective for our troubled economic climate. However, credit remains a very profitable business. Financial organizations need to continually secure safe and profitable credit business, without over-contacting high-potential customers. Getting the contact mix just right can make the difference between acquiring good customers and simply picking up the more risky scraps.

So, expand on this simple objective with a more specific objective.

BUSINESS CHALLENGE: Maximize the return on credit business by making a constrained number of offers to existing customers, optimized on default and incurred loss probabilities and propensity to respond.

That's a concrete business challenge with some key elements for the optimization. The objective is to maximize the return on new credit business. The control variables are the offer and customer pairs: the decision of which customer receives which offer. The constraints are the number of offers to make and the acceptable loss levels.

PROBLEM FORMULATION

Maximize *Expected_Profit = Exp_Profit[customer,offer] * chosen_combinations[customer,offer]*, which is subject to maximum and minimum number of contacts, and limit the maximum amount of risk exposure.

The expected profit for each customer and each offer is derived from a series of predictive models.

This simple credit marketing campaign business example is used for the remainder of this paper.

PROGRAM THE MATHEMATICAL MODEL

Now that there is a clear problem formulation, use PROC OPTMODEL to drive the optimization.

First, read in the relevant data. The data should contain the problem parameters.

```
proc optmodel;
   /* declare sets and parameters */
   set Customers;
   set Offers = 1..3;
   num Credit_Loss_Limit init 100000 ;
   num Minimum_Offer_Level = 0;
   num Exp_Profit{Customers,Offers}, Credit_Loss{Customers,Offers};

   /* read data from SAS data sets */
   read data Opt.Bank_Offers into Customers=[ID]
     {j in Offers} <Exp_Profit[ID,j]=col('exprofit'||j)
       Credit_Loss[ID,j]=col('creditloss'||j)>;
```

bank_offers

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	id	creditloss1	creditloss2	creditloss3	exprofit1	exprofit2	exprofit3	decision
1	10001	125.84015439	62.920077194	106.96413123	-1.321316619	-1.035642274	0.1896791579	3
2	10003	133.13669262	66.568346308	113.16618872	0.0243399037	-2.837544966	-0.136592398	1
3	10005	121.52743561	60.763717804	103.29832027	-0.11892676	-1.619434772	0.1264097883	3
4	10006	138.3747094	69.187354701	117.61850299	-0.442538855	-0.850556301	-0.071936622	0
5	10008	148.23540138	74.11770069	126.00009117	-0.51855832	-2.862581071	-0.079484402	0
6	10009	127.02802833	63.514014167	107.97382408	-1.397183613	-1.848022056	0.1323977009	3
7	10011	143.80010524	71.900052622	122.23008946	0.1987336163	-1.79391068	-0.371282671	1
8	10014	140.60470369	70.302351847	119.51399814	-0.331607504	-0.448644439	0.0913721682	3
9	10019	123.76535827	61.882679137	105.20055453	0.1831902447	-1.037221612	-0.041343616	1
10	10020	144.46968288	72.234841442	122.79923045	-1.53616152	-1.328196204	-0.030192304	0
11	10024	138.19941056	69.099705282	117.46949898	-1.208189544	0.4325687091	-0.354208986	2
12	10028	149.70925433	74.854627164	127.25286618	0.3239471761	-1.444832742	-0.494162423	1
13	10029	151.42617437	75.713087187	128.71224822	0.2400992435	-1.741863044	-0.461551107	1
14	10031	125.12382509	62.561912545	106.35525133	-1.206858013	-1.447645702	0.2776241932	3
15	10032	155.79950167	77.899750837	132.42957642	-0.091966956	-1.489458516	-0.464689215	0
16	10033	144.60348733	72.301743664	122.91296423	-1.282231073	-1.688703064	-0.050889686	0
17	10040	124.45123895	62.225619475	105.78355311	-0.30659943	-1.734772534	-0.057120755	0
18	10041	138.86907739	69.434538694	118.03871578	-1.330455058	-0.801388672	0.1497782995	3
19	10043	126.71846608	63.359233038	107.71069616	0.61412182	-0.474446431	-0.388632109	1
20	10044	150.14535846	75.072679232	127.6235547	0.3624314827	-1.695022047	-0.35098978	1

Figure 3. Table View of Opt.Bank_Offers

The input table contains calculated loss given default (LGD) values and expected profits for each customer. These figures have been calculated using predictive models.

The control variables are next. In this case, the control variables are the decision variables for each customer and offer combination.

```
/* declare variables */
var x{Customers,Offers} binary;
```

It is useful to program the three constraints—the minimum and maximum number of offers for each customer to define the contact policies, and the total accepted project credit loss.

```
/* declare constraints */
con At Most One Offer{i in Customers}:
  sum{j in Offers} x[i,j] = 1;

con Minimum_Offers{j in Offers}:
  sum{i in Customers} x[i,j] >= Minimum_Offer_Level;

con Maximum_Credit_Loss:
  sum{i in Customers, j in Offers} Credit_Loss[i,j] * x[i,j] <= Credit_Loss_Limit;
```

The numeric variables have already been created with the NUM statements. This makes the ensuing parameterization of these values much easier.

Next comes the mathematical expression of the objective. This contains both the mathematical relationship between the objective parameter and the indicative direction (MAX or MIN).

```
/* declare objective */
max Expected_Profit = sum{i in Customers, j in Offers}
  Exp_Profit[i,j] * x[i,j];
```

All that remains is to solve.

```
solve;
```

You can specify which solver PROC OPTMODEL should use. But, for now, the SOLVE statement is enough. The results of the optimization can be exported.

```
print Minimum_Offers.body;
print Maximum_Credit_Loss.body;

create data offers from [Customers Offers] x;

quit;
```

There is now a complete list of which customers receive which offers. This information can be used by marketing to execute the campaign.

PARAMETERIZE THE PROGRAM

It is essential to have the entire OPTMODEL program running and generating relevant results before you start to parameterize the program for users. Decide which parameters you want to give the user control over. In the current business example, perhaps the user wants to play with the number of offers or the acceptable level of credit risk.

The first step is making macro variables from the hardcoded values in the program.

For example, replace the numeric value assigned to Credit_Loss_Limit with the macro variable &Credit_Loss_Limit.

```
num Credit_Loss_Limit init &Credit_Loss_Limit ;
```

Replace the numeric value assigned to Minimum_Offer_Level with the macro variable &Minimum_Offer_Level.

```
num Minimum_Offer_Level = &Minimum_Offer_Level ;
```

Test the program again to make sure there are no mistakes with the replacement macro variables. For now, use %LET statements to assign values to the newly created macro variables.

```
%let Minimum_Offer_Level=0 ;
```

You now have a program from which you can create a stored process.

CREATE A SAS STORED PROCESS

If you have not written your program in SAS Enterprise Guide, you need to copy or open your debugged program in SAS Enterprise Guide. Then, right-click on the program in a SAS Enterprise Guide diagram, and select **Create Stored Process**.

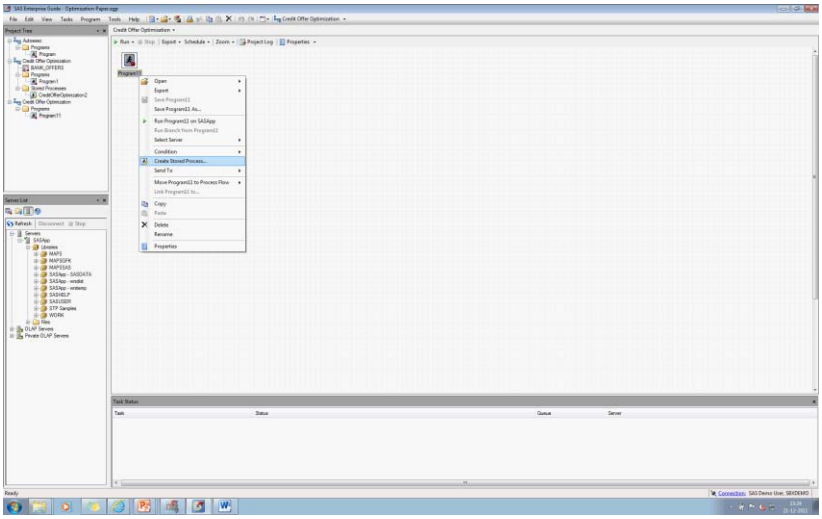


Figure 4. Start the Stored Process Creation Task

On the first page, specify a name and location for your stored process. For more information about configuring your environment for stored processes, see http://support.sas.com/documentation/cdl/en/stpug/62758/PDF/default/stpug.pdf.

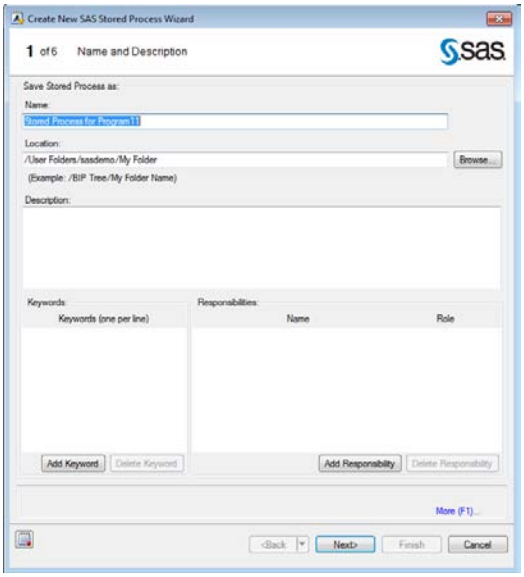


Figure 5. First Page—Name and Location

The next page provides a view of the code that you want to use for the stored process creation. It is important to drop the LIBNAME and macro variable assignments. These assignments will be handled by the parameters that you will create subsequently.

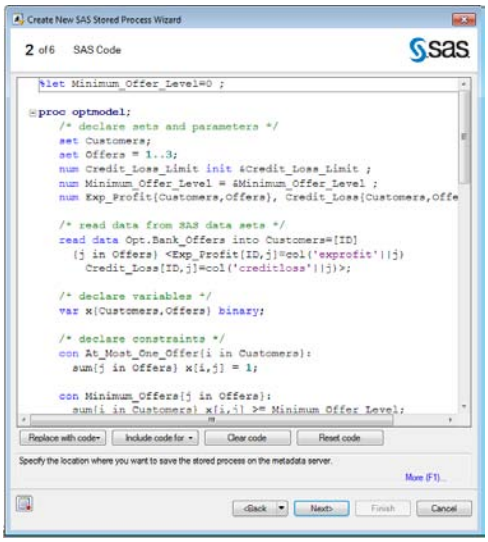


Figure 6. Second Page—Code View

The wizard asks you to set the run-time parameters. At this point, the defaults are acceptable.

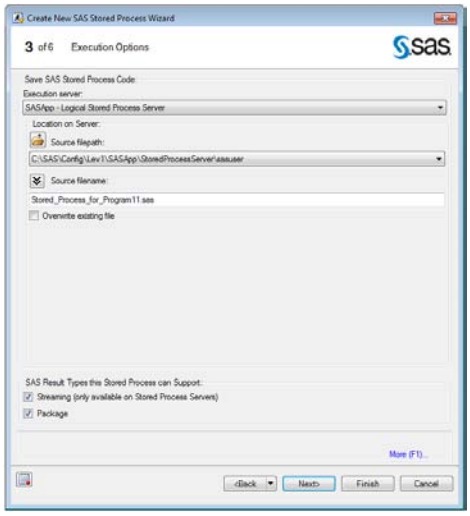


Figure 7. Third Page—Run-Time Parameters

The wizard asks whether you want to make parameters from the scanned program. Basically, with this option, the program is scanned for macro variables, and you can create parameters from these for use in your stored process. Here is an example of making a parameter for the macro variable value Credit_Loss_Limit:

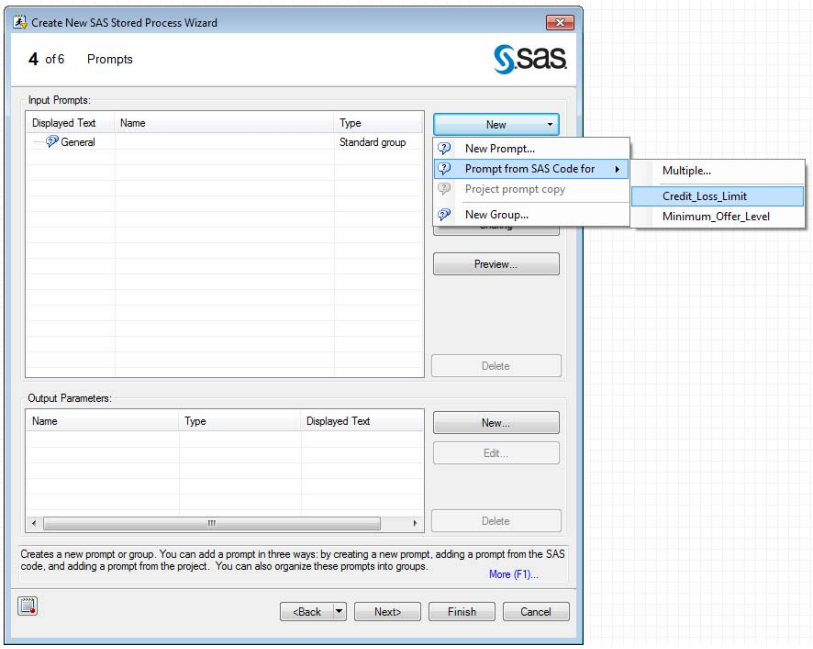


Figure 8. Fourth Page—Prompt for Parameters

After selecting the parameter that you want to use, the wizard enables you to set restrictions and types for the parameter values that will be used during the stored process execution. You should focus on the second tab, where you can change a few characteristics of the parameter vales.

- Make the **Prompt type Numeric**.
- Select **User enters values** for **Method for populating prompt**.
- Select values for **Maximum number of decimal places allowed**, **Minimum value allowed**, **Maximum value allowed**, and **Default Value**.

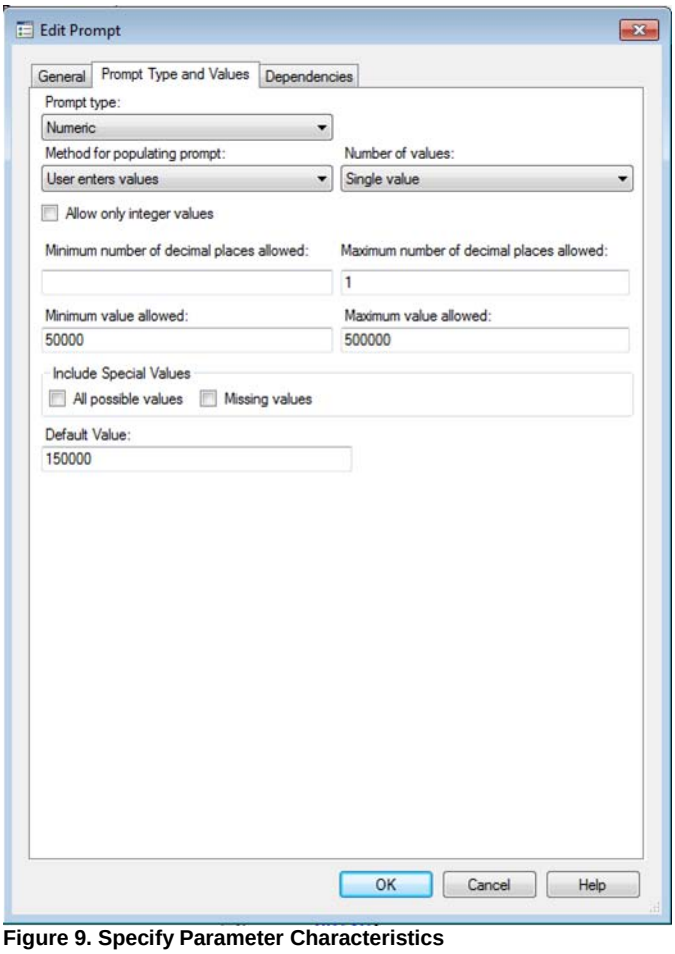
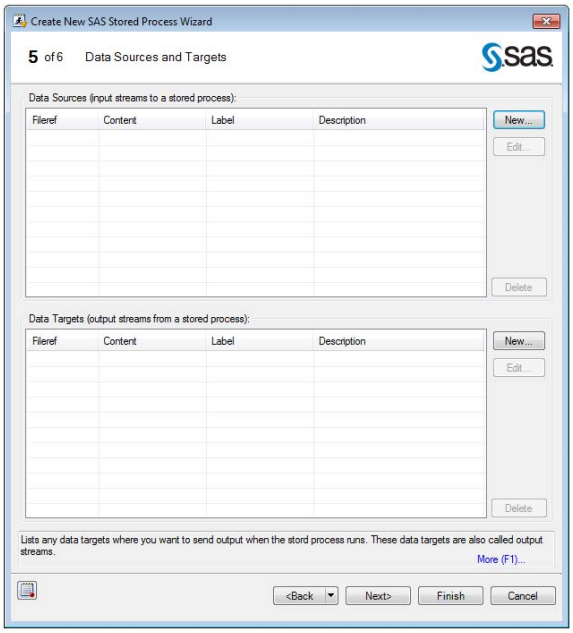


Figure 9. Specify Parameter Characteristics

Repeat this process for the second parameter, Minimum_Offer_Level.

For the last two pages, select the defaults, and click **Finish**.



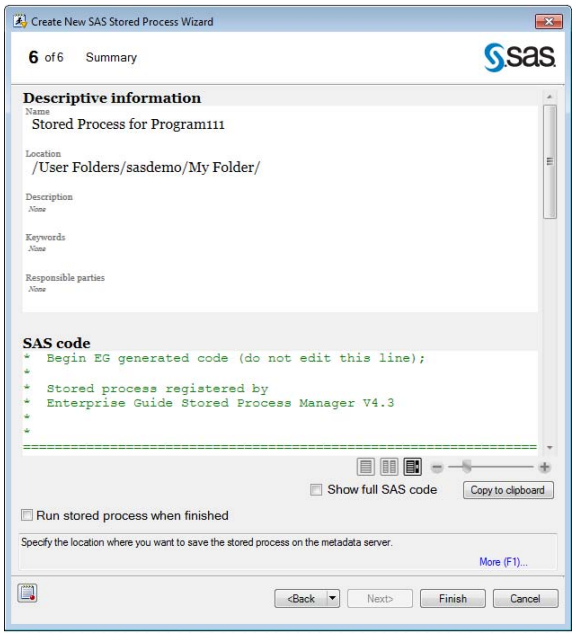


Figure 10. Last Pages of Stored Process Creation

You have created a stored process. You can test its execution by right-clicking on the stored process icon in SAS Enterprise Guide, and selecting **Run Credit Offer Optimization**.

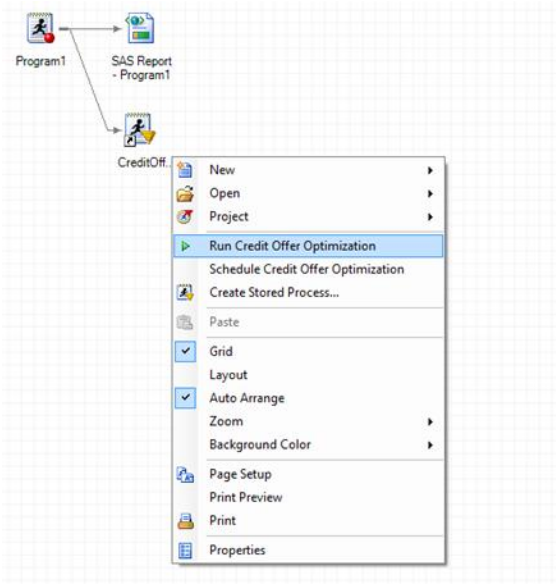


Figure 11. Test the Stored Process

The execution begins with a request for the parameter values that need to be specified.

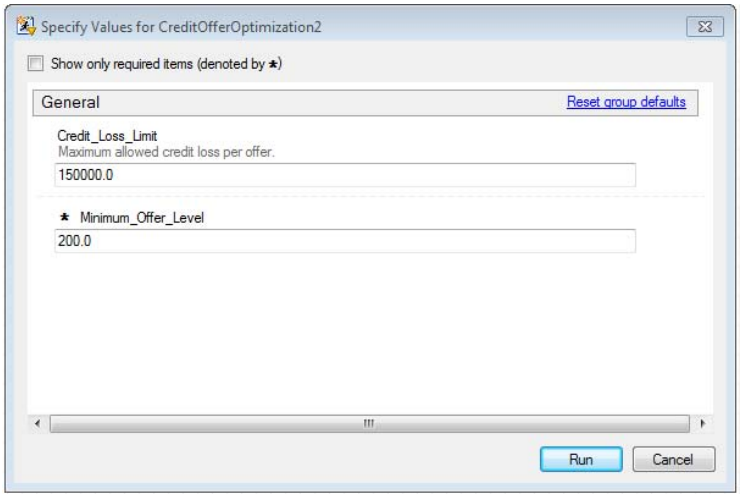


Figure 12. Specify Parameter Values

If you simply click **Run**, then the stored process executes with the default values. In a few seconds, you will see the output of the optimization process and the data set containing individual customer and offer combinations.

The OPTMODEL Procedure

Problem Summary	
Objective Sense	Maximization
Objective Function	Expected_Profit
Objective Type	Linear
Number of Variables	5136
Bounded Above	0
Bounded Below	0
Bounded Below and Above	5136
Free	0
Fixed	0
Binary	5136
Integer	0
Number of Constraints	1716
Linear LE (<=)	1
Linear EQ (=)	1712
Linear GE (>=)	3
Linear Range	0
Constraint Coefficients	15408

CreditOfferOptimization2

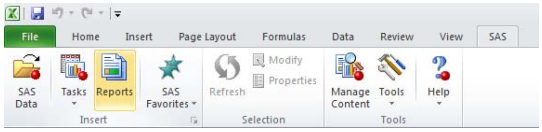
	id	creditloss1	creditloss2	creditloss3	exprofit1	exprofit2	exprofit3	decision	Offers
1	10001	125.84015439	62.920077194	106.96413123	-1.321316619	-1.035642274	0.1896791579	3	2
2	10003	133.13669262	66.568346308	113.16618872	0.0243399037	-2.837544966	-0.136592398	1	3
3	10005	121.52743561	60.763717804	103.29832027	-0.11892676	-1.619434772	0.1264097883	3	3
4	10006	138.3747094	69.187354701	117.61850299	-0.442538855	-0.850556301	-0.071936622	0	2
5	10008	148.23540138	74.11770069	126.00009117	-0.51855832	-2.862681071	-0.079484402	0	3
6	10009	127.02802833	63.514014167	107.97382408	-1.397183613	-1.848022056	0.1323977009	3	3
7	10011	143.80010524	71.900052622	122.23008946	0.1987336163	-1.79391068	-0.371282671	1	2
8	10014	140.60470369	70.302351847	119.51399814	-0.331607504	-0.448644439	0.0913721682	3	2
9	10019	123.76535827	61.882679137	105.20055453	0.1831902447	-1.037221612	-0.041343616	1	2
10	10020	144.46968288	72.234841442	122.79923045	-1.53616152	-1.328196204	-0.030192304	0	2
11	10024	138.19941056	69.099705282	117.46948698	-1.208189544	0.4325687091	-0.354208986	2	2
12	10028	149.70925433	74.854627164	127.25286618	0.3239471761	-1.444832742	-0.494162423	1	2
13	10029	151.42617437	75.713087187	128.71224822	0.2400992435	-1.741863044	-0.461551107	1	2
14	10031	125.12382509	62.561912545	106.36525133	-1.206858013	-1.447645702	0.2776241932	3	3
15	10032	155.79950167	77.899750837	132.42957642	-0.091966956	-1.489458516	-0.464689215	0	2
16	10033	144.60348733	72.301743664	122.91296423	-1.282231073	-1.688703064	-0.050889686	0	2
17	10040	124.45123895	62.225619475	105.78355311	-0.30659943	-1.734772534	-0.057120755	0	3
18	10041	138.86907739	69.434538694	118.03871578	-1.330455058	-0.801388672	0.1497782995	3	2
19	10043	126.71846608	63.359233038	107.71069616	0.61412182	-0.474446431	-0.388632109	1	2
20	10044	150.14535846	75.072679232	127.6235547	0.3624314827	-1.695022047	-0.35098978	1	2
21	10047	143.38807289	71.694036447	121.87986196	-0.254516106	0.3359695248	-0.483433503	2	2

Figure 13. View Optimization Output

MAKING IT ACCESSIBLE WITH SAS ENTERPRISE GUIDE OR MICROSOFT EXCEL

Users can access the optimization stored process with their choice of client. Because you have already tested it using SAS Enterprise Guide, you can see how it looks in Microsoft Excel.

In an Excel spreadsheet, select the **SAS** tab, and then select **Reports**.



In the menu, select **CreditOfferOptimization2**.

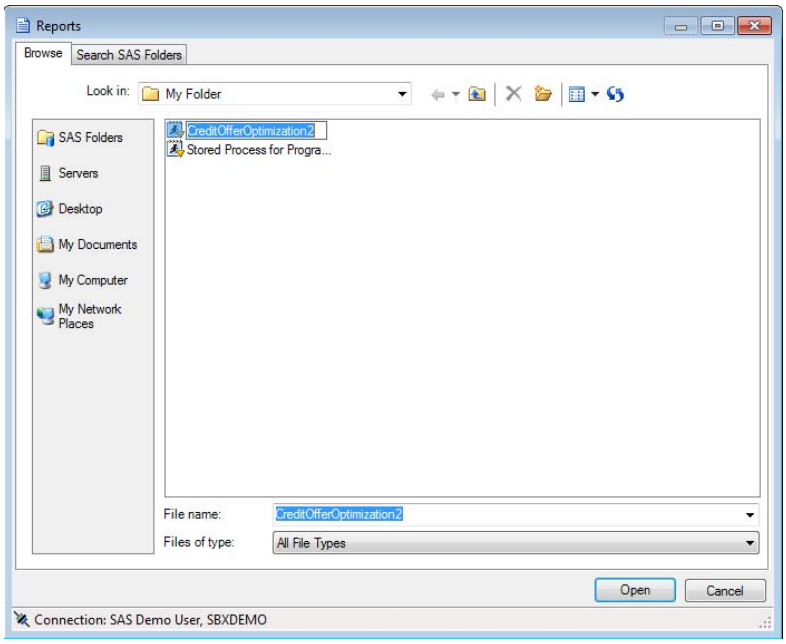


Figure 14. Select Stored Process

When you run the stored process, it executes just as it did in SAS Enterprise Guide.

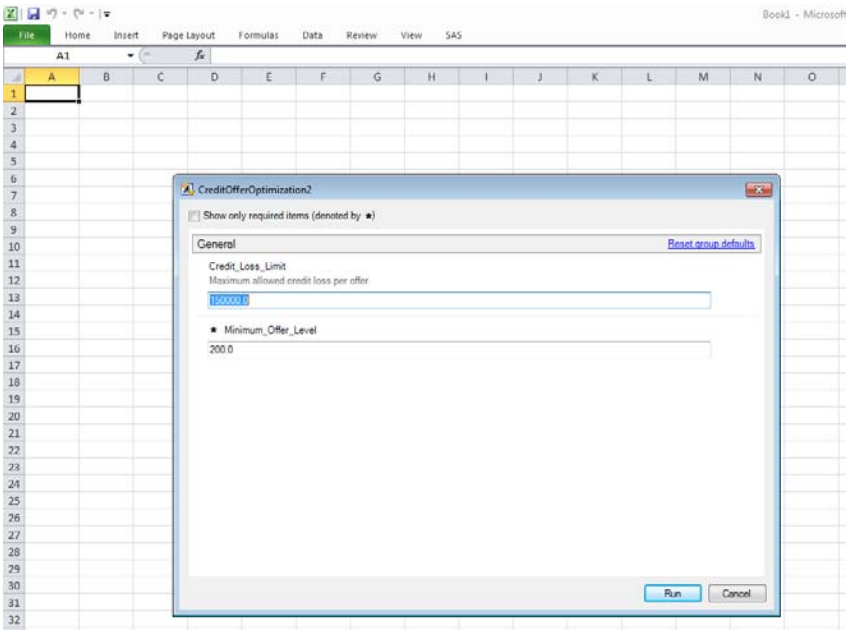
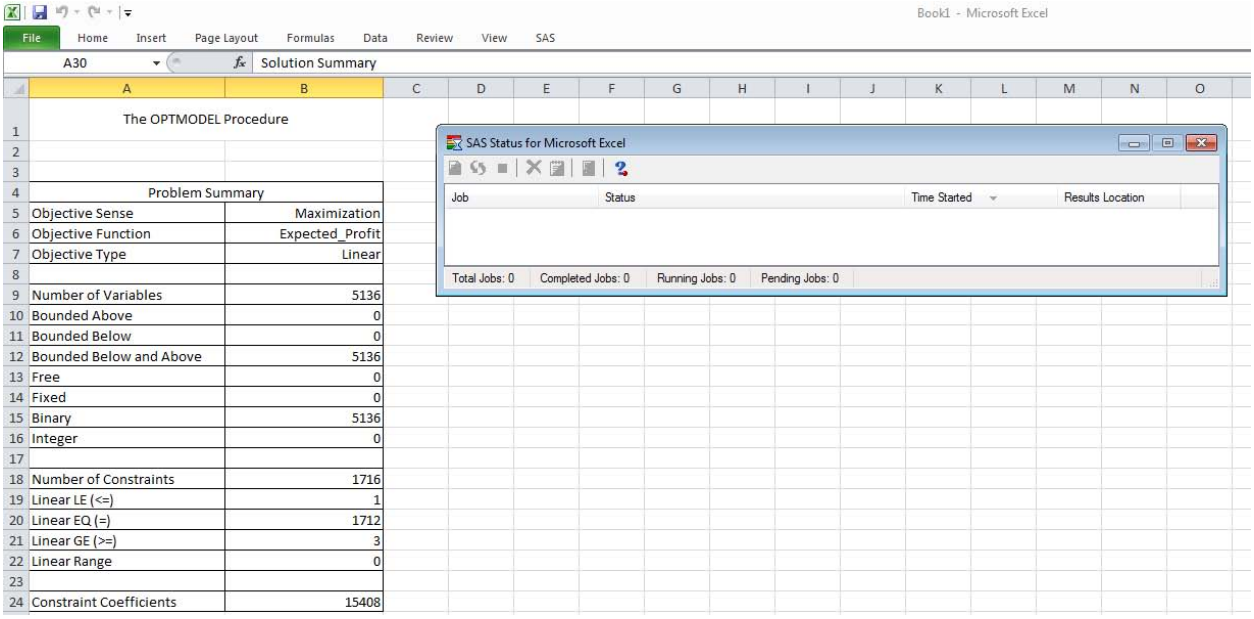


Figure 15. Run the Stored Process

The same results are displayed.



Problem Summary	
Objective Sense	Maximization
Objective Function	Expected_Profit
Objective Type	Linear
Number of Variables	5136
Bounded Above	0
Bounded Below	0
Bounded Below and Above	5136
Free	0
Fixed	0
Binary	5136
Integer	0
Number of Constraints	1716
Linear LE (<=)	1
Linear EQ (=)	1712
Linear GE (>=)	3
Linear Range	0
Constraint Coefficients	15408

Figure 16. View the Results

PROJECT PLANNING WITH PROC OPTMODEL AND MICROSOFT PROJECT

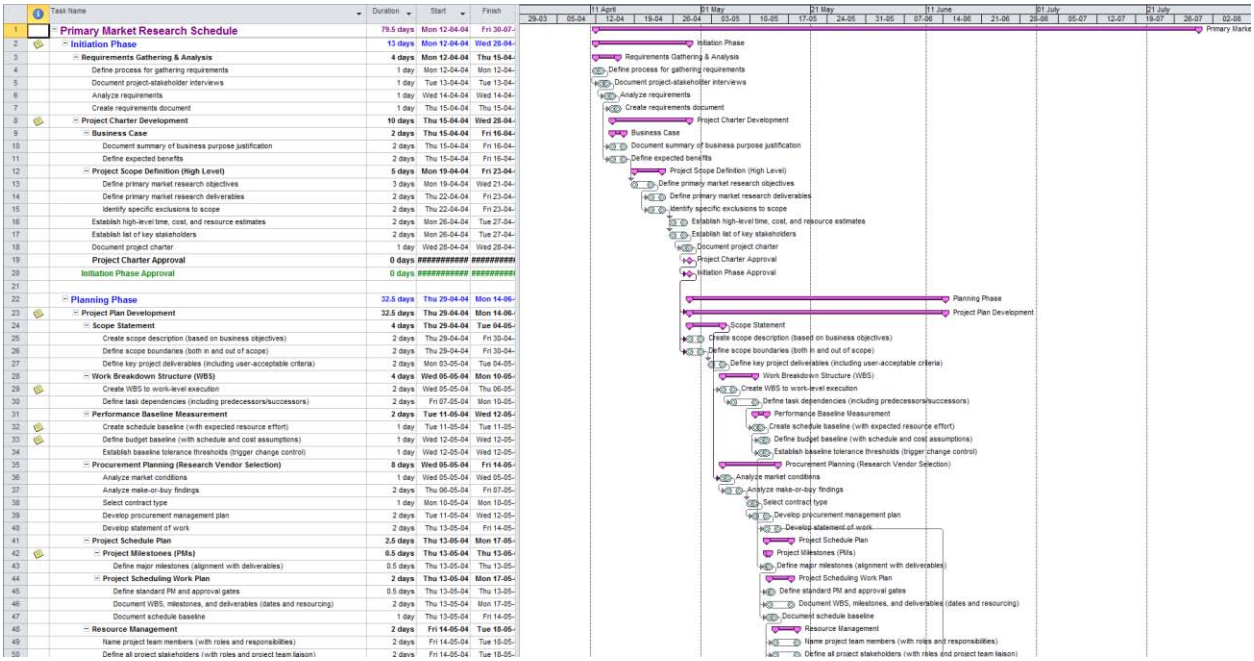
Until now, this paper has focused exclusively on PROC OPTMODEL as an optimization engine under the hood of user-friendly stored processes. In addition, SAS/OR offers a robust suite of project planning capabilities.

- PROC CPM can be used for planning, controlling, and monitoring a project.
- PROC PM is another interactive procedure that can be used for planning, controlling, and monitoring a project.
- PROC DTREE is an interactive procedure for decision analysis.
- PROC GANNT is a graphical scheduling tool for planning and controlling a project.
- PROC NETDRAW is a procedure that draws a network diagram of the activities in a project.

- Earned Value Management (EVM) macros compare an earned value to the work that was planned to measure project performance and to estimate future costs and project completion.

These capabilities become ready for business when project data is read directly from Microsoft Project files using the %MSPTOSAS macro and processed in SAS/OR. Then, the results are converted into Microsoft Project files using the %SASTOMSP macro.

Here is a standard Microsoft Project plan for a market research project:



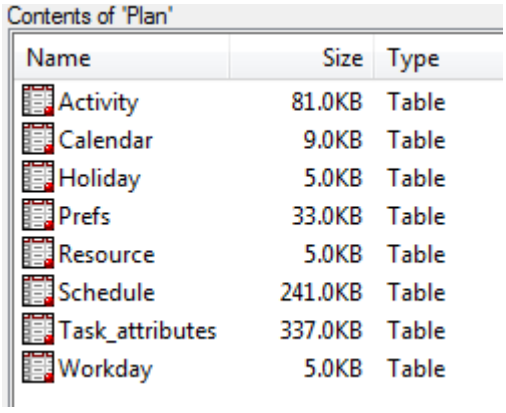
An XML file of this project plan can be saved and then later identified in SAS with the following FILENAME statement:

```
filename check 'C:\Documents and Settings\sbxdemo1\My Documents\Primary market research schedule 2007.xml' ;
```

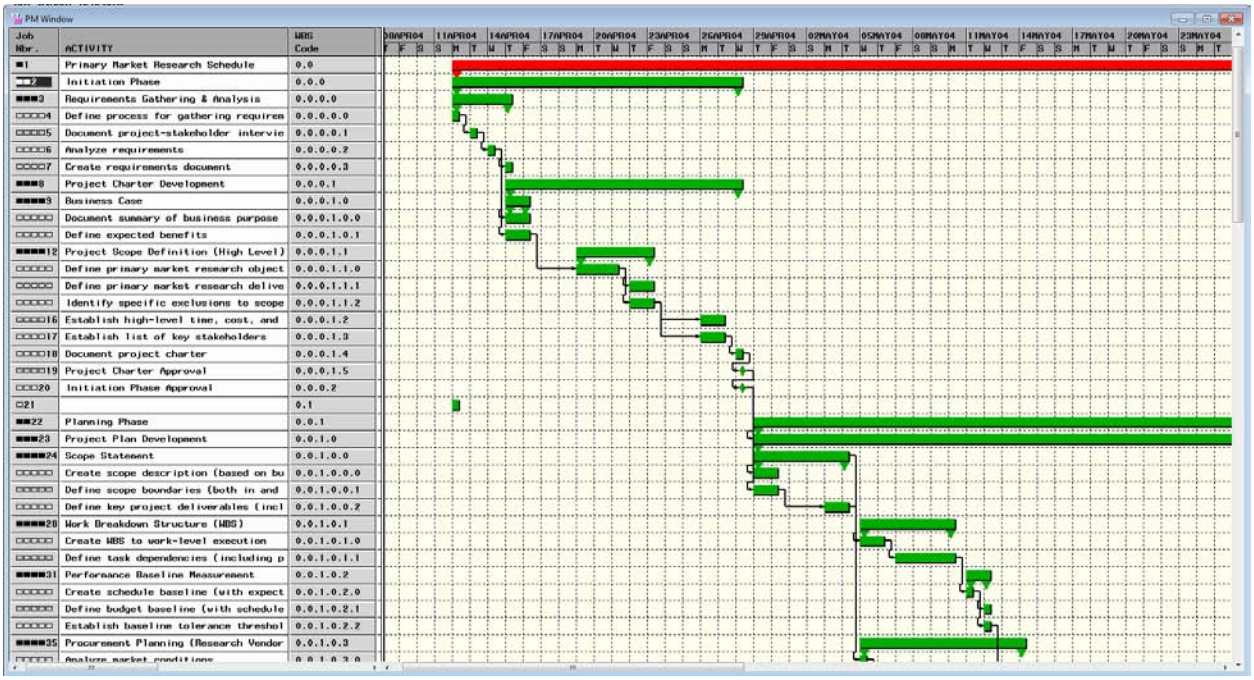
This XML file can be read into SAS using the following simple macro call:

```
libname plan 'D:\Business Development\Events\2012\SAS Global Forum - Orlando\sasdata' ;
%MSPTOSAS (
        LIBRARY=plan,
        VERSION=2007,
        XMLFILE=check    ) ;
```

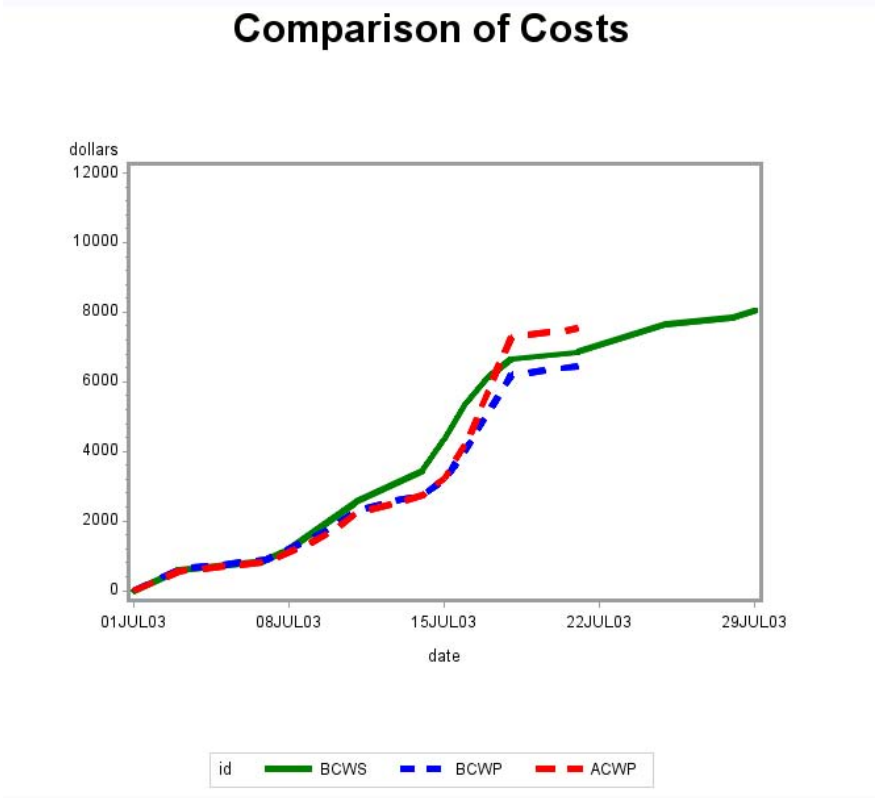
This creates the following data sets, which are already structured for use with the SAS/OR project planning capabilities:



The SAS interactive project management window immediately opens:



You can now run various analyses. For example, here are the results of a cost analyses:



After you have analyzed the project and modified the project flow if needed, you can complete the task by creating an .mdb file for further use in Microsoft Project.

```
filename mdbfile 'C:\Documents and Settings\sbxdemo1\My Documents\revised\Revised market research
schedule 2007.mdb' ;
```

```
%SASTOMSP(
        LIBRARY=plan,
        MDBFILE=mdbfile,
    _DUR=duration,
     actds=activity, calds=calendar, workds=workday,
          scheduleds=schedule,  interval=dtday,
          _date="12APR04:08:00:00"dt,
          _activity=ACTUID, _successor=SUCCUID,
          _lag=LAG,  _project=PNTUID,
          _id=ACTIVITY ACTUID
      ) ;
```

CONCLUSION

For SAS/OR and the optimization possibilities that it offers to reach a business, analysts need to be creative in making SAS/OR more business-ready for business users. This paper discusses a few simple approaches toward this goal.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Andrew Pease
SAS Institute Inc.
E-mail: Andrew.Pease@sas.com