

Developing Your Own SAS Studio Custom Tasks for Advanced Analytics

Elliot Inman and Olivia Wright, SAS Institute Inc., Cary, NC

ABSTRACT

Standard SAS® Studio tasks already include many advanced analytic procedures for data mining and other high-performance models. These existing tasks enable the point-and-click generation and execution of SAS code. However, you can extend the power of standard tasks by creating tasks of your own. You can create tasks for the very latest SAS statistical procedures, your own default model definitions, or your previously developed SAS/STAT® software or SAS macro code. Best of all, these point-and-click tasks can be developed directly in SAS Studio without the need to compile binaries using third-party software.

This paper demonstrates three approaches to developing custom tasks. First, we build a custom task to provide point-and-click access to the new Item Response Theory procedure commonly used to analyze educational test and opinion survey data. Second, we build a custom task that calls a macro for previously developed SAS code, and we show how point-and-click options can be set up to allow users to guide the execution of complex macro code. Third, we demonstrate just enough of the underlying Apache Velocity Template Language to enable you to take advantage of the benefits of that language to support your SAS process. Finally, we show how these custom tasks can easily be shared with a user community, and thereby increase the efficiency of analytic modeling across the organization.

INTRODUCTION

The line dividing Statistics and Computer Science is becoming increasingly blurred. It would be impossible to be productive as a statistician or data scientist without some ability to code. At the same time, an understanding of the essentials of statistics – probability, reliability, and generalizability – is critical to programming machine learning and cognitive computing platforms. Someday, every statistician might also be a web programmer, and every Python or Java programmer will understand the difference between random and fixed statistical effects. But that day is not here yet. In the meantime, for statisticians and data scientists, SAS Studio offers a flexible, easy way to create custom tasks for running complex statistical models.

This paper assumes that the reader understands analytic modeling and the fundamentals of the SAS language: Base SAS, STAT, and the SAS macro language. This paper shows the Apache Velocity Template Language (VTL) code required to start using the SAS Studio Common Task Model (CTM). To that end, in this introduction, we define what a task is and its benefits to statisticians and data scientists. We also briefly describe the Item Response Theory (IRT) models used in the examples. Following this introduction are multiple examples, beginning with the most simple and building to more complex use cases.

SAS STUDIO TASKS AND CUSTOM TASKS

SAS Studio tasks are a visual interface to SAS data and statistical procedures. These tasks are similar to the visual interface available through SAS Enterprise Guide and node-based SAS Enterprise Miner diagrams. However, these tasks are based on Apache VTL, which provides a number of advantages. Tasks can be written and deployed without needing to compile a binary of the task. Code can be written within SAS Studio or any text editor. And the Apache VTL code can be shared just like SAS code.

All of the code for standard SAS Studio tasks can be inspected, modified, and saved as a custom task of your own. Custom tasks can be developed by editing existing tasks, by using the task templates shipped with SAS Studio, or by coding a task starting with a blank file. Finally, modified or custom tasks are easily portable. The CTM file is very small in size and can be copied like any file (without the need for a library of supporting code) and transferred via a shared file repository or, if appropriate, even by email.

Figure 1 shows some of the tasks available in SAS Studio.

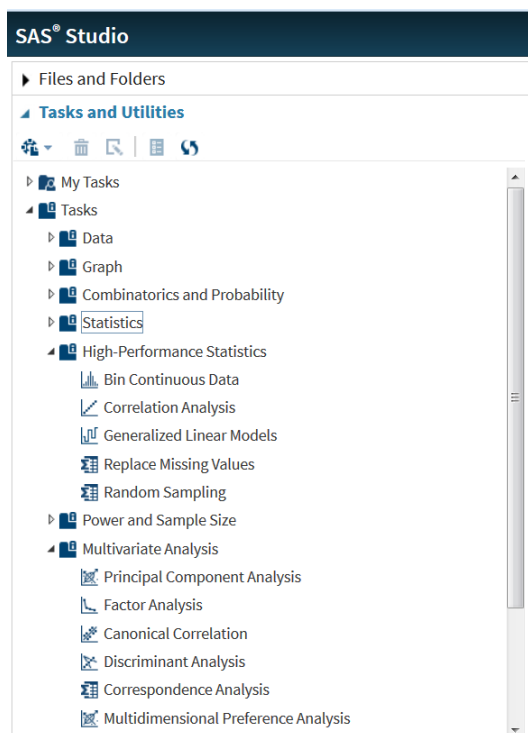


Figure 1. Some Advanced Analytic Procedures Available as SAS Studio Tasks

For the following examples, the Bin Continuous Data task was copied from the standard **Tasks** folder to the **My Tasks** folder. By copying the task to the **My Tasks** folder, you can modify the task. The original task (that shipped with SAS Studio) is still available from the **Tasks** folder. Figure 2 shows the visual user interface for the Bin Continuous Data Task. Figure 3 shows the VTL code that generates the **Method** drop-down list, so the user can select a binning method.

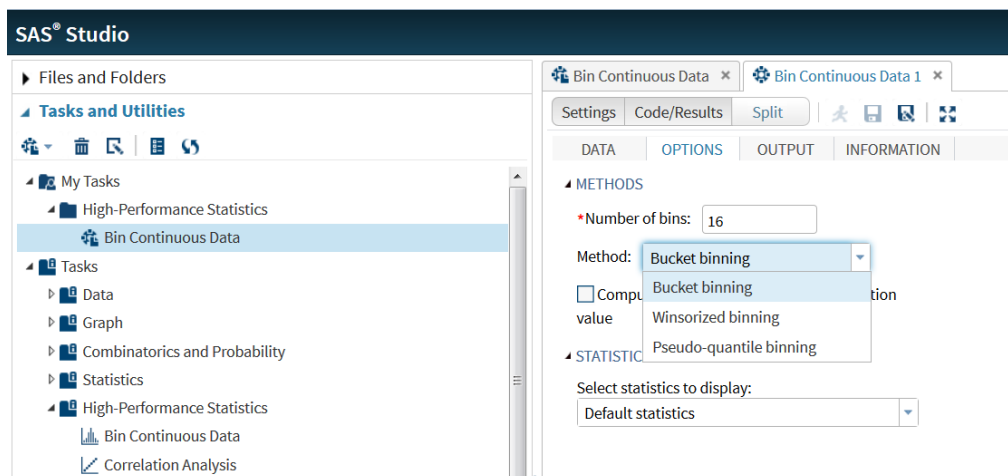


Figure 2. User Interface for the Bin Continuous Data Task

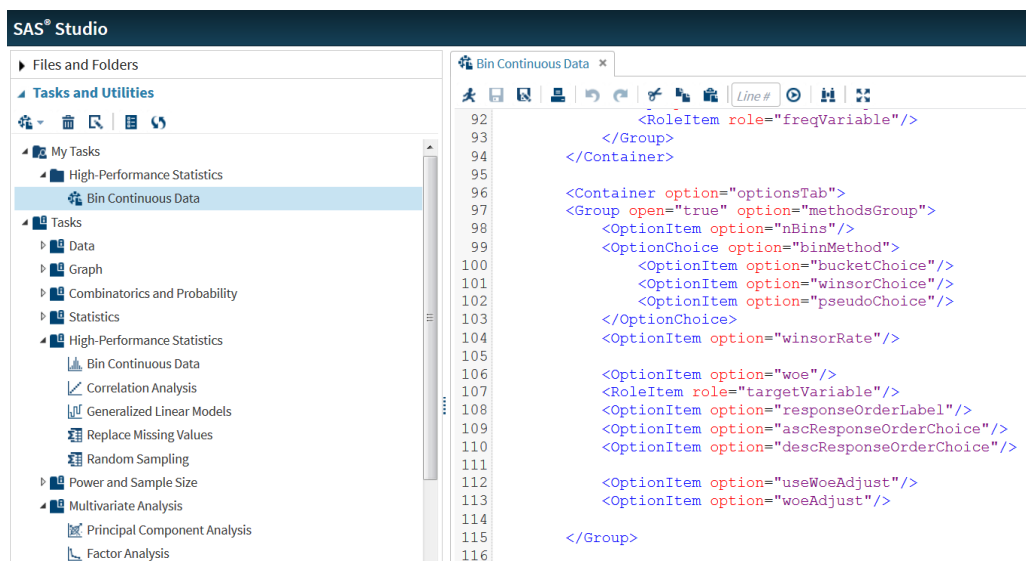


Figure 3. Apache Velocity Code for the Method Drop-Down List

SAS STUDIO TASKS FOR STATISTICIANS

A statistician or data scientist who has already mastered the fundamentals of SAS code already knows how to manage data using Base SAS, run statistical analyses using SAS/STAT, automate those processes using SAS macro language, and structure tabular and graphic output using ODS. Suggesting that an expert SAS coder learn Apache VTL for custom tasks raises the question: Why? There are many reasons.

The latest version of SAS Studio supports the newest SAS procedures for running exceptionally high-speed in-memory analytics. Far from a limited-functionality version of SAS, SAS Studio is the platform through which users can access the most advanced cognitive computing and machine learning statistical procedures. That new functionality is available through SAS Studio tasks. Understanding Apache VTL will allow you to create your own custom versions of those tasks.

Tasks can assist in learning new SAS procedure code. As you interact with the visual user interface of a

task, SAS Studio automatically generates the SAS code based on your selections. This goes beyond code completion; it is code generation on-the-fly. For users wanting to learn the structure and model specifications for new SAS procedures, using a SAS Studio task is an easy way to get started. The basic model code works from the start without the hurdle of typographical and syntactic errors.

Tasks can also assist expert coders in using SAS procedures they know well. Any professional statistician or data scientist with more than a few years of experience has a set of critical, but rarely used procedures. The specifics of the syntax and even decisions about certain variance structures or estimation methods can be easily forgotten from one use to the next. Previous code cannot be automated because each new run of the analysis might require some minor manual adjustment to the model. For such work, tasks are ideal. Instead of having to search through user guides and manuals, you can include your most commonly used model specifications in the task. And the code generated by the task can actually serve as a mnemonic device, assisting in recall of the appropriate syntax and options.

Tasks can also help with documentation and communication of model specifications. SAS data and statistical code (readable using any text browser) has always been a reliable source of documentation for researchers sharing statistical findings. But exceptionally complex multi-stage modeling can generate a tangle of statistical procedures that requires extensive inline commenting and explanation. Data mining experts who have used SAS Enterprise Miner know the value of the Reporter node to represent all nodes and model parameters in a single clear view. That diagram is a useful visual organizer for a complex model. Tasks can serve the same purpose. Figure 4 shows an example of a process flow using standard tasks.

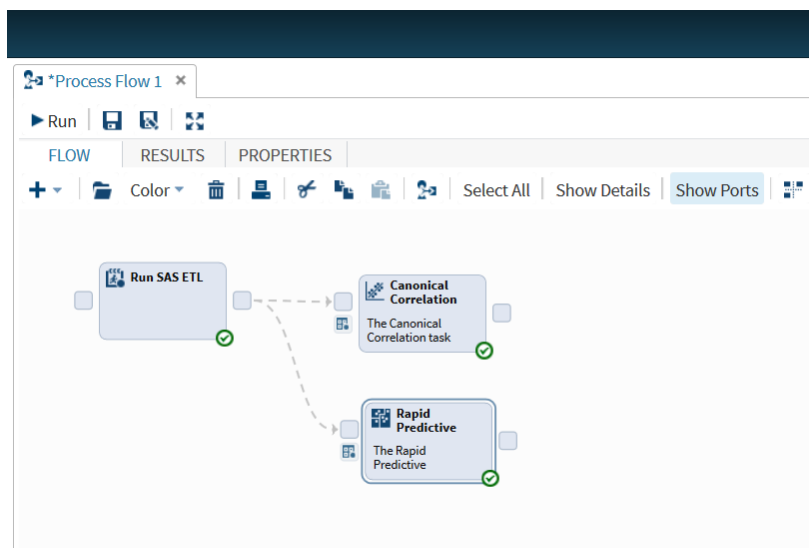


Figure 4. Sample Process Flow

Finally, expert SAS programmers might not know that SAS Studio is now the primary platform for learning for the next generation of SAS users. It is the platform for both of the no-cost educational offerings: SAS® OnDemand (cloud-based) offering and the downloadable virtual machine version. (For links, see the “Recommended Reading and Resources” section at the end of this paper.) Statisticians and data scientists who have already mastered the SAS language using traditional SAS tools will want to take advantage of the growing community of SAS users who know SAS Studio. A new generation of SAS users will share their code as tasks, snippets, and process flows.

ITEM RESPONSE THEORY (IRT)

For this paper, we demonstrate tasks using the Item Response Theory procedure (PROC IRT). (For the full documentation on this procedure, see support.sas.com.) As of this writing, PROC IRT has not been implemented as a standard SAS Studio task. PROC IRT is a relatively new SAS procedure (experimental in SAS/STAT 13.1, production in SAS/STAT 14.1 and later). IRT methods originated in the analysis of educational test data but are now widely used to analyze survey response data in health care, marketing, and other industries.

All IRT models have several characteristics in common. All IRT models assume that performance on a single item (test item or survey question) represents just one measure of an individual on a latent trait. One math test question on the Pythagorean Theorem is merely one measure of a student's understanding of geometry. The student's performance on that item can vary due to many factors such as the student's ability, opportunity to learn, and motivation. In addition, the test question might not be well-written, easily guessed, and so on. The same is true for any survey question. One question might ask about a customer's satisfaction with the staff at a hotel, but the customer's answer to that one question can be due to many factors including the person's overall happiness in general, satisfaction due to factors other than the staff, and so on. IRT models parse the variance due to various factors (the latent trait of interest, general difficulty of the item, guessing or careless responses, and so on) and rescale both the items and people on a common scale to better understand performance on a set of items. A set of item parameters from a previous analysis can be used to score new data or equate new items to a previously developed bank of items.

PROC IRT can be used to implement a wide variety of standard IRT models from Rasch, 1p, 2p, 3p, and 4p to the Graded Response Model and, in SAS/STAT 14.1 and later, the Generalized Partial Credit Model. PROC IRT will automatically generate a factor analysis of the set of items, but there are many options for that model, including specifications for a confirmatory factor analysis to validate new data against previous expectations. For a complete reference to PROC IRT, please see the documentation on support.sas.com.

To get started, we demonstrate a barebones custom task, which includes the minimal VTL code necessary to create the task. Then we explore more complex examples. One example shows how a task can be used to execute a specific version of a PROC IRT model (1p, 2p, 3p, and so on) using check boxes to select one or more models. The next example shows how to use similar task functionality to execute the modeling based on a set of defined projects assuming that different model specifications were already developed for different surveys to be administered. Finally, we present a few of the most useful VTL commands for implementing custom advanced analytic tasks.

In this paper, the example code below is presented as a graphics object to enable users to see exactly how the VTL code will look in SAS Studio. All of the code for each task is included as plain ASCII text as the end of this paper, along with a simple SAS program for quickly generating test and survey data sets for testing this code.

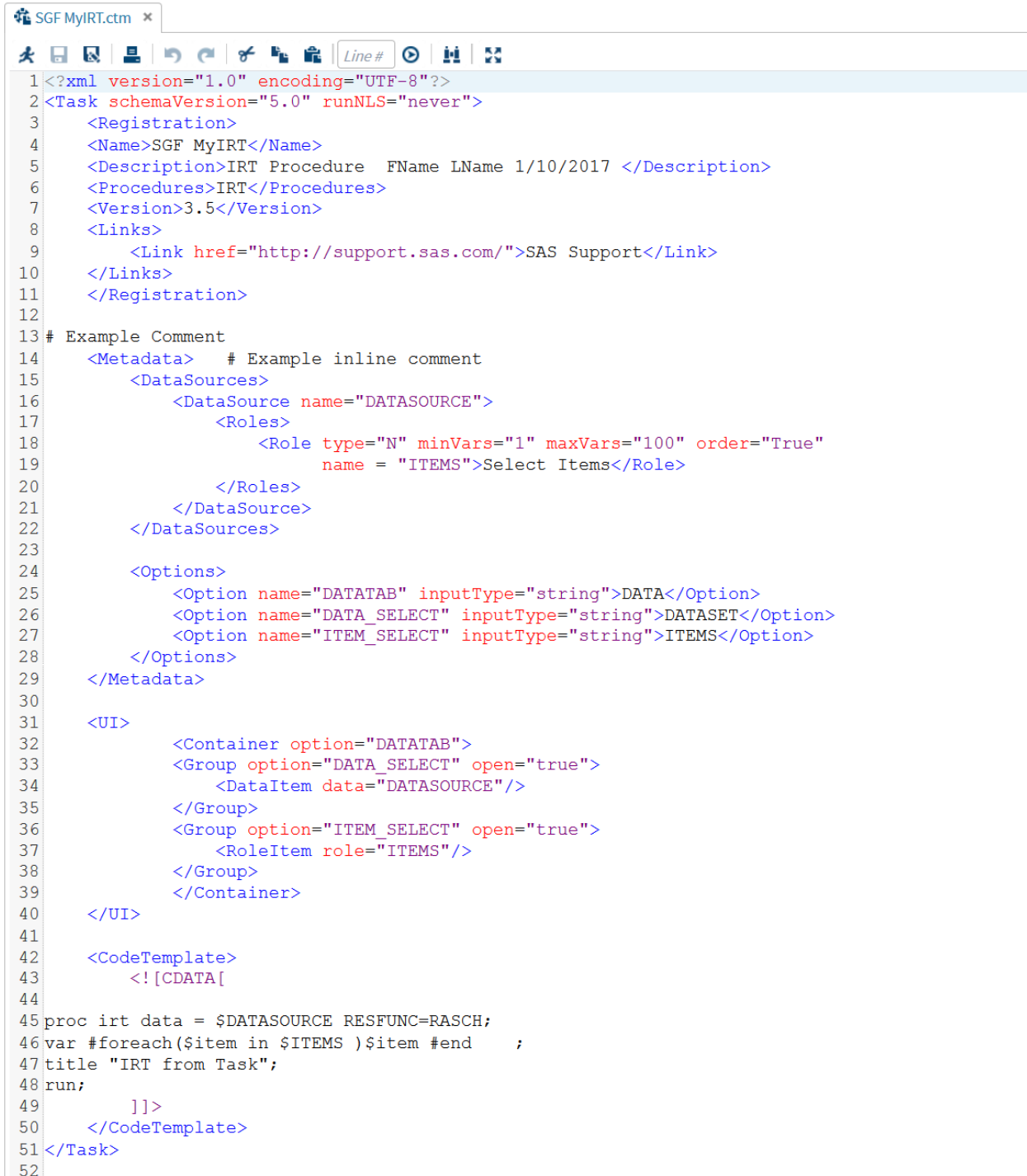
CREATING A BASIC POINT-AND-CLICK TASK

The Common Task Model (CTM) includes several basic elements that must appear in all tasks. Here are the required elements:

- Registration
- Metadata
- User Interface (UI)
- CodeTemplate

Figure 5 shows all of these elements in a CTM program. Figure 6 shows the user interface that is created when you run the CTM code.

Some aspects of the VTL code are self-explanatory. Registration is primarily a header that documents the task name, description, and information about SAS Studio and SAS procedures used. Metadata defines the data source and roles for variables. Metadata also includes an “options” section that defines the elements you plan to use in the task. The UI section builds the interface “containers” for various options. Finally, the CodeTemplate section is for the SAS code to be triggered by the task.



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <Task schemaVersion="5.0" runNLS="never">
3   <Registration>
4     <Name>SGF MyIRT</Name>
5     <Description>IRT Procedure FName LName 1/10/2017 </Description>
6     <Procedures>IRT</Procedures>
7     <Version>3.5</Version>
8     <Links>
9       <Link href="http://support.sas.com/">SAS Support</Link>
10    </Links>
11  </Registration>
12
13  # Example Comment
14    <Metadata> # Example inline comment
15      <DataSources>
16        <DataSource name="DATASOURCE">
17          <Roles>
18            <Role type="N" minVars="1" maxVars="100" order="True"
19              name = "ITEMS">Select Items</Role>
20          </Roles>
21        </DataSource>
22      </DataSources>
23
24      <Options>
25        <Option name="DATATAB" inputType="string">DATA</Option>
26        <Option name="DATA_SELECT" inputType="string">DATASET</Option>
27        <Option name="ITEM_SELECT" inputType="string">ITEMS</Option>
28      </Options>
29    </Metadata>
30
31    <UI>
32      <Container option="DATATAB">
33        <Group option="DATA_SELECT" open="true">
34          <DataItem data="DATASOURCE"/>
35        </Group>
36        <Group option="ITEM_SELECT" open="true">
37          <RoleItem role="ITEMS"/>
38        </Group>
39      </Container>
40    </UI>
41
42    <CodeTemplate>
43      <![CDATA[
44
45proc irt data = $DATASOURCE RESFUNC=RASCH;
46var #foreach($item in $ITEMS )$item #end    ;
47title "IRT from Task";
48run;
49    ]]>
50  </CodeTemplate>
51</Task>
52

```

Figure 5. Barebones VTL Code for IRT Custom Task

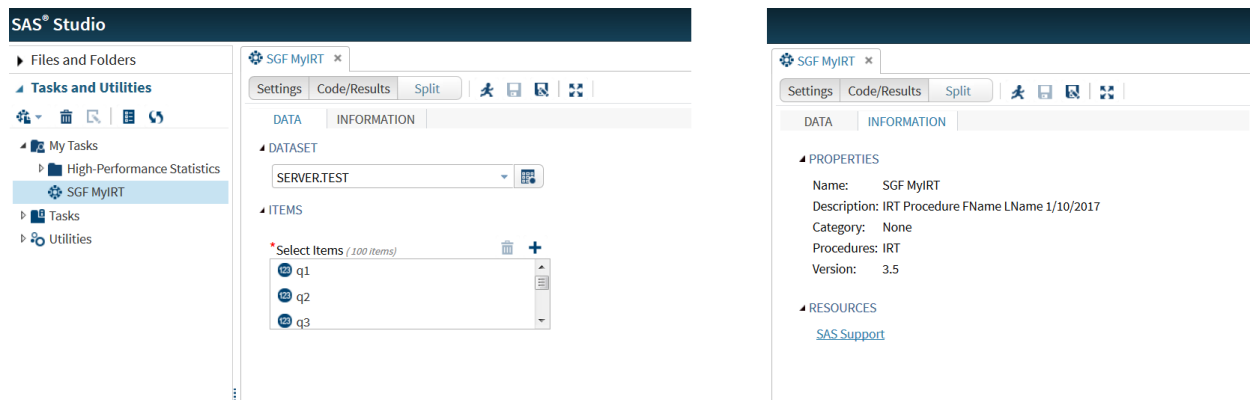


Figure 6. Data and Information Tabs for IRT Custom Task

For expert SAS coders, some of this terminology might be confusing. The easiest way to approach VTL coding is to remember that it is a language of its own, developed outside of SAS. For example, “metadata” is information for the task, not SAS platform metadata. “Options” are really more like tags to identify anything that will be used later in the task, not statistical options for a SAS procedure. “Registration” sounds like it is related to registering a table in SAS metadata; it’s not. Registration is simply a header for the task.

In terms of coding, there are a few things to keep in mind. Comments start with #, not * or /*. Variable names assigned dynamically are prefixed with \$, not &. The color coding (blue, red, and so on) in the editor will assist in identifying typographical errors or syntactic errors. When you save your task, you are notified of any errors with line level references to the problem.

But there are some rather tricky aspects to VTL code. For example, the task in Figure 5 uses `maxVars` and makes that equal to 100. The `maxVars` element identifies the maximum number of variables that can be used from the data source. Values from 1 to any integer are taken as that number, but a value of “0” there actually means unlimited – no maximum at all. Fortunately, all of the VTL code necessary to implement these tasks is fully documented in *SAS Studio: Developer’s Guide to Writing Custom Tasks* available from support.sas.com.

For expert SAS coders, the VTL language might be new, but the structure of the PROC IRT code at the end of Figure 5 should be familiar. Note the way in which VTL and SAS communicate. These are the three most commonly used VTL codes: the identification of a data set using the `$DATASOURCE` name and the enumeration of selected variables using `#foreach` and `$item` to pass those values to the procedure. This is all that is needed to pass information from the user interface to the SAS code, but these are essential for creating any custom task.

Statisticians and data scientists concerned that SAS Studio will not provide the same level of control and feedback that is available through traditional SAS coding tools should know that, even with the task user interface, you can still inspect all code. As Figure 7 shows, the SAS log still provides important feedback on execution of that code. And, as Figure 8 shows, standard SAS output is produced by using output data options in the task. This output includes a standard Rasch model with item slopes fixed at 1 and item difficulties ranging from -1.52 to 1.54. The Eigenvalues table is part of the factor analytic solution and shows that performance on this small set of 10 items is driven largely by a single trait – due to the fact that sample data were created that way. (ASCII code for the task and the code used to create the demo test data are all available at the end of this paper.)

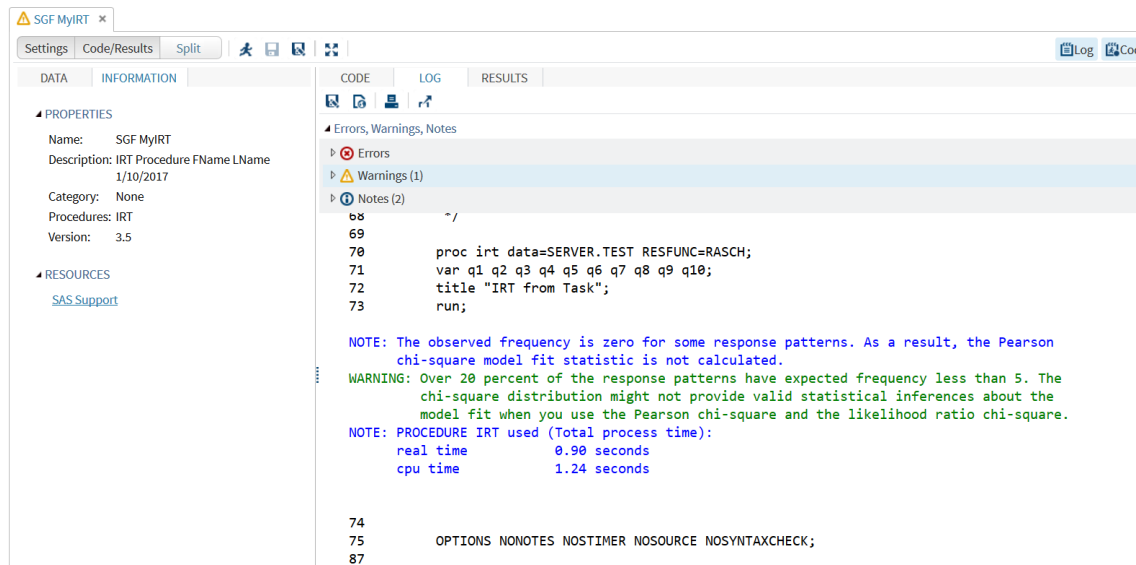


Figure 7. Log for Barebones IRT Custom Task

IRT from Task		IRT from Task				
The IRT Procedure		The IRT Procedure				
Modeling Information		Item Parameter Estimates				
Data Set	SERVER.TEST	Item	Parameter	Estimate	Standard Error	Pr > t
Link Function	Logit	q1	Difficulty	-1.52137	0.03281	<.0001
Response Model	Rasch Model		Slope	1.00000		
Number of Items	10	q2	Difficulty	-1.35062	0.03221	<.0001
Number of Factors	1		Slope	1.00000		
Number of Observations Read	10000	q3	Difficulty	-0.91591	0.03105	<.0001
Number of Observations Used	10000		Slope	1.00000		
Estimation Method	Marginal Maximum Likelihood	q4	Difficulty	-0.30038	0.03021	<.0001
			Slope	1.00000		
		q5	Difficulty	-0.21416	0.03017	<.0001
			Slope	1.00000		
		q6	Difficulty	0.47776	0.03038	<.0001
			Slope	1.00000		
		q7	Difficulty	0.68283	0.03066	<.0001
			Slope	1.00000		
		q8	Difficulty	0.98638	0.03124	<.0001
			Slope	1.00000		
		q9	Difficulty	1.19119	0.03177	<.0001
			Slope	1.00000		
		q10	Difficulty	1.54403	0.03294	<.0001
			Slope	1.00000		

Figure 8. Barebones IRT Custom Task Output

POINT-AND-CLICK SINGLE MACRO EXECUTION

The previous task offers few options. Users can select a set of items and a Rasch model is automatically run. The next task shows a version of the task that allows users to select the IRT model to run. PROC IRT now includes many models: Rasch, 1p, 2p, 3p, and 4p for binary items and the Graded Response and Generalized Partial Credit models for items like constructed response educational test items or Likert

scale survey questions. This task enables users to select one or more of those models. SAS provides an error message if an inappropriate model is attempted (for example, a Rasch model using 7-level Likert scale items).

To build this more flexible task, three changes are required. First, we must add additional options to the Metadata element. Figure 9 shows the addition of the check box definitions for each option.

```
<Metadata>
  <DataSources>
    <DataSource name="DATASOURCE">
      <Roles>
        <Role type="N" maxVars="100" minVars="1" order="True"
          name="ITEMS">Select Items</Role>
      </Roles>
    </DataSource>
  </DataSources>

  <Options>
    <Option name="DATATAB"      inputType="string">DATA</Option>
    <Option name="DATA_SELECT"  inputType="string">DATASET</Option>
    <Option name="ITEM_SELECT"  inputType="string">ITEMS</Option>
    <Option name="GROUPCHECK"   inputType="string">MODEL</Option>
    <Option name="chkRASCH"      inputType="checkbox"  defaultValue="0" >Rasch</Option>
    <Option name="chkONE"        inputType="checkbox"  defaultValue="0" >One Parameter</Option>
    <Option name="chkTWO"        inputType="checkbox"  defaultValue="0" >Two Parameter</Option>
    <Option name="chkTHREE"      inputType="checkbox"  defaultValue="0" >Three Parameter</Option>
    <Option name="chkFOUR"       inputType="checkbox"  defaultValue="0" >Four Parameter</Option>
    <Option name="chkGRADED"     inputType="checkbox"  defaultValue="0" >Graded Response</Option>
    <Option name="chkGPC"        inputType="checkbox"  defaultValue="0" >Generalized Partial Credit</Option>
  </Options>
</Metadata>
```

Figure 9. New Check Box Definitions for Each New Option

After those options are defined, we need to surface them in the user interface. Figure 10 shows that VTL code.

```
<UI>
  <Container option="DATATAB">
    <Group open="true" option="DATA_SELECT">
      <DataItem data="DATASOURCE"/>
    </Group>
    <Group open="true" option="ITEM_SELECT">
      <RoleItem role="ITEMS"/>
    </Group>
    <Group open="true" option="GROUPCHECK">
      <OptionItem option="chkRASCH"/>
      <OptionItem option="chkONE"/>
      <OptionItem option="chkTWO"/>
      <OptionItem option="chkTHREE"/>
      <OptionItem option="chkFOUR"/>
      <OptionItem option="chkGRADED"/>
      <OptionItem option="chkGPC"/>
    </Group>
  </Container>
</UI>
```

Figure 10. Defining Model Options in the UI Element

Next, we must tell SAS what to do when one or more of those options are checked in the user interface. Figure 11 shows how we have used a single SAS macro and the SAS RESFUNC specification to supply the model name. The VTL code `#if` passes the model choice selection from the user interface to the SAS procedure.

```

<CodeTemplate>
  <![CDATA[
%macro irt(resfunc);
proc irt data = $DATASOURCE RESFUNC=&resfunc;
var #foreach($item in $ITEMS )$item #end  ;
title "IRT from Task Using &resfunc Model";
run;
%mend irt;

#if ($chkRASCH == 1)
  %irt(RASCH);
#end
#if ($chkKONE == 1)
  %irt(ONEP);
#end
#if ($chkTWO == 1)
  %irt(TWOP);
#end
#if ($chkTHREE == 1)
  %irt(THREEP);
#end
#if ($chkFOUR == 1)
  %irt(FOURP);
#end
#if ($chkGRADED == 1)
  %irt(GRADED);
#end
#if ($chkGPC == 1)
  %irt(GPC);
#end

  ]]>
</CodeTemplate>

```

Figure 11. VTL and SAS Communication

Figure 12 shows the user interface and Figure 13 shows some of the output for the Generalized Partial Credit Model results.

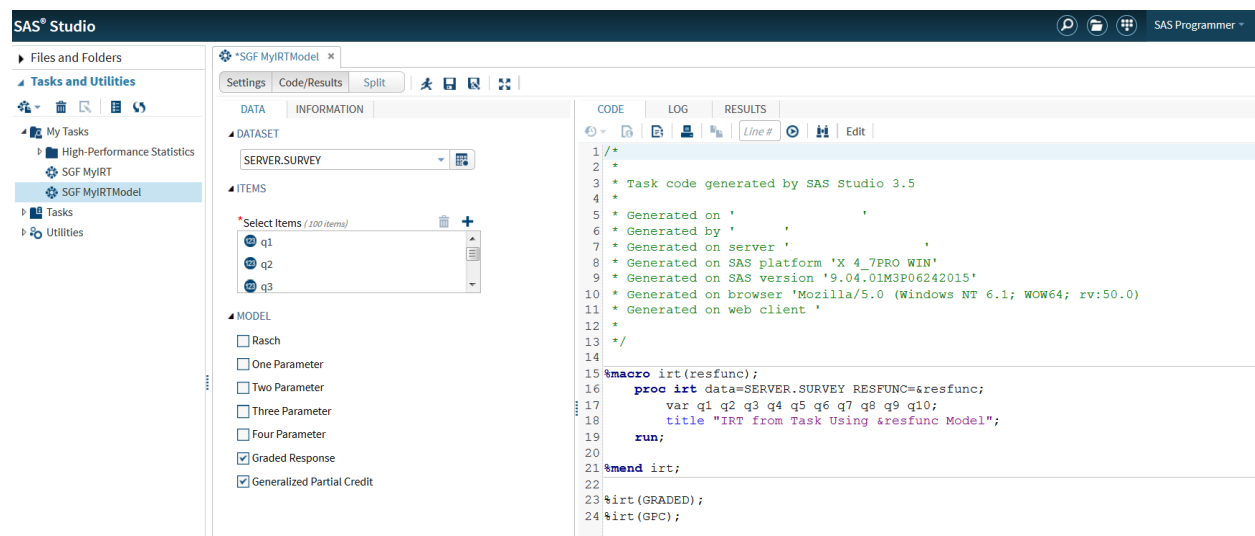


Figure 12. User Interface for SAS Macro Example

IRT from Task Using GPC Model			Item Parameter Estimates				
The IRT Procedure			Item	Parameter	Estimate	Standard Error	Pr > t
Modeling Information							
Data Set	SERVER SURVEY						
Link Function	Logit						
Response Model	Generalized Partial Credit						
Number of Items	10						
Number of Factors	1						
Number of Observations Read	1000						
Number of Observations Used	1000						
Estimation Method	Marginal Maximum Likelihood						

Item Information		
Item	Levels	Values
q1	7	1 2 3 4 5 6 7
q2	7	1 2 3 4 5 6 7
q3	7	1 2 3 4 5 6 7
q4	7	1 2 3 4 5 6 7
q5	7	1 2 3 4 5 6 7
q6	7	1 2 3 4 5 6 7
q7	7	1 2 3 4 5 6 7
q8	7	1 2 3 4 5 6 7
q9	7	1 2 3 4 5 6 7
q10	7	1 2 3 4 5 6 7

q1	Step 1	-2.63196	0.23675	<.0001
	Step 2	-1.29130	0.13348	<.0001
	Step 3	-0.54763	0.11105	<.0001
	Step 4	0.22530	0.10635	0.0171
	Step 5	1.37472	0.13247	<.0001
	Step 6	2.02403	0.18847	<.0001
	Slope	0.94498	0.06345	<.0001
q2	Step 1	-2.77572	0.22780	<.0001
	Step 2	-1.43858	0.10763	<.0001
	Step 3	-0.45833	0.08350	<.0001
	Step 4	0.37029	0.08395	<.0001
	Step 5	1.26070	0.10260	<.0001
	Step 6	2.42011	0.18008	<.0001
	Slope	1.26800	0.08439	<.0001
q3	Step 1	-2.31346	0.21070	<.0001
	Step 2	-1.54654	0.13067	<.0001
	Step 3	-0.70375	0.09751	<.0001
	Step 4	0.33596	0.09311	0.0002
	Step 5	1.21361	0.11385	<.0001
	Step 6	2.07271	0.16912	<.0001
	Slope	1.09290	0.07300	<.0001
q4	Step 1	-2.50091	0.20511	<.0001
	Step 2	-1.19555	0.11499	<.0001
	Step 3	-0.51840	0.09574	<.0001

Figure 13. Output for SAS Macro Example

POINT-AND-CLICK MULTIPLE MACRO EXECUTION

The previous task enables users to select different model specifications by using check boxes for the RESFUNC option, but statisticians often have an entirely different model code process that needs to be run for a particular survey or project. Some projects might require descriptive statistics, graphics output for reports, or other modeling typically run in a batch job. By modifying the previous task, we can create a task that provides users a simple user interface for executing a more complex SAS job, and at the same time, select output options that apply across the various jobs.

The previous tasks included only the **Data** and **Information** tabs. This custom task adds the **Options** tab which is very common in the tasks that ship with SAS Studio. The word “option” can be confusing here. In this case, the **Options** tab is more akin to the traditional idea of SAS “options,” which allow users to select different versions of a model or output specifications. In truth, the IRT model selection on the **Data** tab from our earlier custom task would be better placed on an **Options** tab.

For this task, we are working under the assumption that customer survey title (Loyalty Long Form, Loyalty Short Form, and Satisfaction) provides all we need to know to run the analysis. An IRT model and the supporting code have been already developed. Only one survey can be analyzed at a time. Also note that, unlike the check box in the previous task that allowed users to select one or more models to run at the same time, the survey title selector in this task uses a combobox control so only one survey can run at a time.

Figures 14 and 15 show the necessary code changes in the Metadata and UI elements. Figure 16 shows the SAS code triggered depending on those selections. This code is deliberately redundant to show how a complex set of code could be triggered by a simple selection in the user interface.

```

<Metadata>
  <DataSources>
    <DataSource name="DATASOURCE">
      <Roles>
        <Role type="N" minVars="1" maxVars="100" order="True"
              name = "ITEMS">Select Items</Role>
      </Roles>
    </DataSource>
  </DataSources>
  <Options>
    <Option name="DATATAB"      inputType="string">DATA</Option>
    <Option name="DATAGROUP"    inputType="string">DATA</Option>
    <Option name="ITEM_SELECT"  inputType="string">SURVEY QUESTIONS</Option>
    <Option name="OPTIONSTAB"   inputType="string">OPTIONS</Option>
    <Option name="GROUPLISTING" inputType="string">ITEM OUTPUT FOR LISTING</Option>
    <Option name="GROUPOUTPUT"  inputType="string">OUTPUT DATASETS</Option>
    <Option name="chkITEMS"     inputType="checkbox"   defaultValue="0">Item Statistics</Option>
    <Option name="chkICC"       inputType="checkbox"   defaultValue="0">Item Characteristic Curves</Option>
    <Option name="chkOUTPUT"    inputType="checkbox"   defaultValue="0">Create Output Datasets</Option>
    <Option name="GROUPCOMBO"   inputType="string">CUSTOMER SURVEY</Option>
    <Option name="comboTYPE"    inputType="combobox" defaultValue="LSF">Choose from the list below:</Option>
    <Option name="LSF"          inputType="string">Loyalty (Short Form)</Option>
    <Option name="LLF"          inputType="string">Loyalty (Long Form)</Option>
    <Option name="SAT"          inputType="string">Satisfaction</Option>
    <Option name="labelOUTPUT"  inputType="string">Produces three output datasets:
                                _Scores, _Model, and _ParameterEstimates</Option>
  </Options>
</Metadata>

```

Figure 14. Creating a Combobox for Survey Titles and Options Tab Variables

```

<UI>
  <Container option="DATATAB">
    <Group option="GROUPCOMBO" open="true">
      <OptionChoice option="comboTYPE">
        <OptionItem option="LSF"/>
        <OptionItem option="LLF"/>
        <OptionItem option="SAT"/>
      </OptionChoice>
    </Group>
    <Group option="DATAGROUP" open="true">
      <DataItem data="DATASOURCE"/>
    </Group>
    <Group option="ITEM_SELECT" open="true">
      <RoleItem role="ITEMS"/>
    </Group>
  </Container>
  <Container option="OPTIONSTAB">
    <Group option="GROUPLISTING" open="true">
      <OptionItem option="chkITEMS"/>
      <OptionItem option="chkICC"/>
    </Group>
    <Group option="GROUPOUTPUT" open="true">
      <OptionItem option="chkOUTPUT"/>
      <OptionItem option="labelOUTPUT"/>
    </Group>
  </Container>
</UI>

```

Figure 15. User Interface for Survey Task

Figure 16 shows that this task allows for the creation of three output data sets: Scores, Model, and Parameter Estimates. A look at the SAS code reveals two of these output data structures are available from the PROC IRT statement, but the Parameter Estimates data set is an ODS data set. All such ODS data sets can be written as standard SAS data sets using ODS OUTPUT, but expert IRT modelers should know that many of the data structures they will find useful are available only via this ODS method. (A full listing of all ODS data set names is available in the IRT procedure documentation in the SAS/STAT:

User's Guide.) One of the advantages in this task is that users of the user interface do not need to understand this distinction. Users of this task simply click one box and get all three data output structures automatically.

```
<CodeTemplate>

<![CDATA[

%macro LSF;
  ods graphics on;
  proc freq data=$DATASOURCE; tables #foreach($item in $ITEMS )$item #end ; run;
  #if ($chkOUTPUT ==1) ods output ParameterEstimates=LSF_ParameterEstimates ; #end
  proc irt data=$DATASOURCE resfunc=GRADED
    #if ($chkITEMS == 1) itemstat #end
    #if ($chkICC == 1) plots=icc #end
    #if ($chkOUTPUT == 1) out=LSF_Scores outmodel=LSF_Model #end ;
    var #foreach($item in $ITEMS )$item #end ;
  run;
  ods graphics off;
%mend LSF;

%macro LLF;
  ods graphics on;
  proc means data=$DATASOURCE; var #foreach($item in $ITEMS )$item #end ; run;
  #if ($chkOUTPUT ==1) ods output ParameterEstimates=LLF_ParameterEstimates ; #end
  proc irt data=$DATASOURCE resfunc=GRADED
    #if ($chkITEMS == 1) itemstat #end
    #if ($chkICC == 1) plots=icc #end
    #if ($chkOUTPUT == 1) out=LLF_Scores outmodel=LLF_Model #end ;
    var #foreach($item in $ITEMS )$item #end ;
  run;
  ods graphics off;
%mend LLF;

%macro SAT;
  ods graphics on;
  proc univariate data=$DATASOURCE; var #foreach($item in $ITEMS )$item #end ; run;
  #if ($chkOUTPUT ==1) ods output ParameterEstimates=SAT_ParameterEstimates ; #end
  proc irt data=$DATASOURCE resfunc=GPC
    #if ($chkITEMS == 1) itemstat #end
    #if ($chkICC == 1) plots=icc #end
    #if ($chkOUTPUT == 1) out=SAT_Scores outmodel=SAT_Model #end ;
    var #foreach($item in $ITEMS )$item #end ;
  run;

#if ($comboTYPE == "LSF") %LSF; #end
#if ($comboTYPE == "LLF") %LLF; #end
#if ($comboTYPE == "SAT") %SAT; #end

]]>
</CodeTemplate>
```

Figure 16. SAS Code for Survey Task

One aspect of this task cannot be demonstrated in a written paper. When any customer survey is selected, the macro code for all three tasks is automatically generated. Only the selected survey code is run, but all three macros are written to the SAS code window. This is easily overcome by moving the macro code to a macro library and adding a call to that library in the task code. Securitized macros can be called in the same way to protect coded production models.

Figure 17 shows the **Data** and **Options** tab generated by our VTL code. In the user interface, the **Item Characteristic Curve** option is checked. Figure 18 shows the output that results from checking that box.

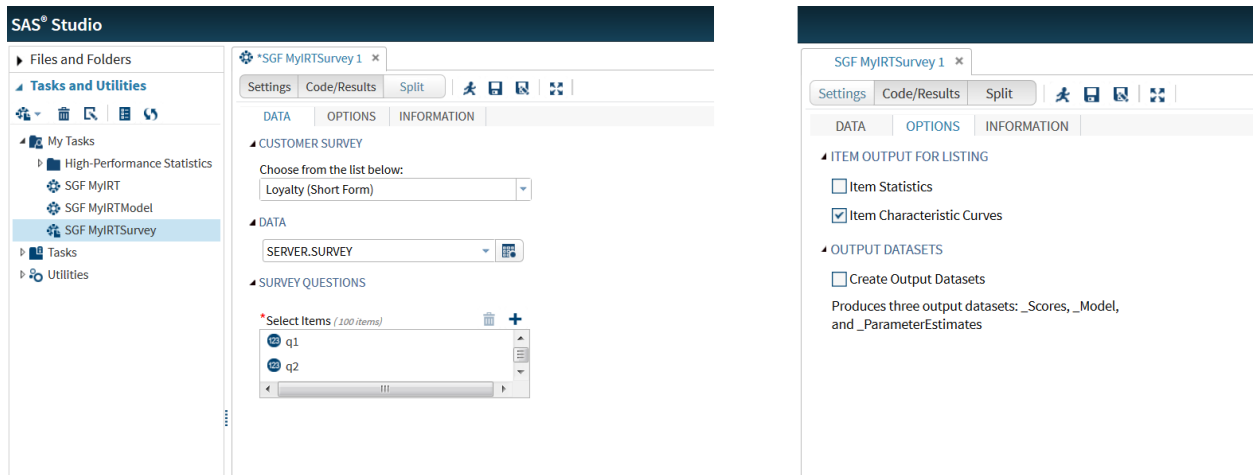


Figure 17. User Interface for MyIRTSurvey Task

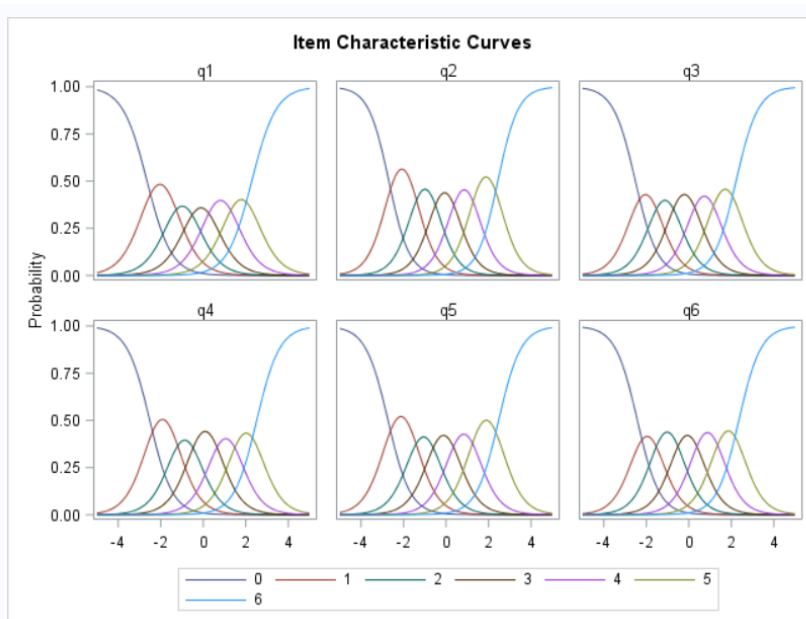


Figure 18. Item Characteristic Curve Output

The VTL code for this task is included at the end of this paper. To use the code, simply paste the ASCII text into any standard text editor (Notepad, Notepad++, and similar tools) and save the code as a CTM file.

MORE ABOUT VELOCITY TEMPLATE LANGUAGE (VTL)

SAS Studio tasks are built using the open-source Apache Velocity Template Language and the SAS Studio Common Task Model. The example tasks in this paper use only a minimal amount of VTL code to simplify the process of learning to use custom tasks. Tasks show the use of the `$` to call variables and the `$DATASOURCE` variable. Tasks demonstrate the `#foreach` to enumerate a list of variables and `#if` to indicate to SAS which check boxes in the user interface were selected.

But the VTL language, and its implementation within SAS Studio, goes well beyond the examples provided here. You can manage conditional code execution, perform quality checks, and manage data. Some of this VTL code is in Table 1. A complete reference can be found in the *SAS Studio: Developer's Guide to Writing Custom Tasks* at support.sas.com.

Code	Purpose	Example
<code>size()</code>	Checks that a list is not empty before executing code assuming a list.	<pre>#if(\$BYVAR.size() > 0) #foreach(\$item in \$BYVAR) proc sort data=\$textDATASET; by \$item; #end run; #end</pre>
<code>length()</code>	Checks to see if a string is empty before executing code assuming a string.	<pre>#if(\$textTITLE.length() > 0) title "\$textTITLE"; #end</pre>
<code>getLibrary()</code>	Automatically specifies a library using the library used for the data source.	<pre>%let SASLIBRARY=\$DATASOURCE.getLibrary(); data &SASLIBRARY.new; run;</pre>

Table 1. Some Additional VTL Codes

CONCLUSION

Statisticians and data scientists can increase their efficiency and productivity by taking advantage of the benefits of SAS Studio tasks – in particular, the custom tasks they develop themselves. SAS Studio already supports the latest version of the newest SAS procedures for running exceptionally high-speed in-memory analytics. All of the VTL code for those tasks is readily available to view, modify, and save as a custom task to meet your unique needs. And other procedures, like PROC IRT, can be called through a custom task, built from scratch, to execute a simple IRT model or a much more complex IRT-based project that includes multiple SAS procedures and various output and reporting options. SAS Studio includes everything necessary to create and deploy tasks. One person's efforts to create a custom task can easily be shared with the team, other researchers, and other users just as easily as any SAS code is transferred from one person to another.

It is not yet true that every statistician or data scientist is an equally capable web programmer. Not every programmer is a skilled analytical consultant. However, the Common Task Model using VTL greatly

simplifies the process of creating tasks for executing advanced analytic models. We encourage expert SAS coders who know Base SAS, SAS/STAT, and the SAS macro language to join the growing community of SAS Studio users taking advantage of tasks.

RECOMMENDED READING AND RESOURCES

Here are some useful guides for the technologies used in this paper. The latest version of these resources will be available through support.sas.com and communities.sas.com and previous SAS Global Forum papers.

- SAS Studio: <http://www.sas.com/studio>
- SAS Studio editions for learning: http://www.sas.com/en_us/industry/higher-education/on-demand-for-academics.html
- Olivia Wright's Task Tuesday series on SAS Communities: <https://communities.sas.com/t5/forums/searchpage/tab/message?q=Task+tuesday>
- Apache Velocity Template Language: <http://velocity.apache.org/>

CONTACT INFORMATION

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

Elliot Inman, Ph.D.
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Elliot.Inman@sas.com
<http://www.sas.com>

Olivia Wright
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Olivia.Wright@sas.com
<http://www.sas.com>

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

APPENDIX: CODE FOR ALL TASKS

The code in this appendix includes all code necessary to create the custom tasks in this paper. To use the code, simply paste the ASCII text into any standard text editor (Notepad, Notepad++, and similar tools) and save the code as a CTM type file.

SGF_MyIRT.CTM

```
<?xml version="1.0" encoding="UTF-8"?>
<Task schemaVersion="5.0" runNLS="never">
  <Registration>
    <Name>SGF MyIRT</Name>
    <Description>IRT Procedure FName LName 1/10/2017 </Description>
    <Procedures>IRT</Procedures>
    <Version>3.5</Version>
    <Links>
      <Link href="http://support.sas.com/">SAS Support</Link>
    </Links>
  </Registration>

  # Example Comment
  <Metadata> # Example inline comment
    <DataSources>
      <DataSource name="DATASOURCE">
        <Roles>
          <Role type="N" minVars="1" maxVars="100" order="True"
            name = "ITEMS">Select Items</Role>
        </Roles>
      </DataSource>
    </DataSources>

    <Options>
      <Option name="DATATAB" inputType="string">DATA</Option>
      <Option name="DATA_SELECT" inputType="string">DATASET</Option>
      <Option name="ITEM_SELECT" inputType="string">ITEMS</Option>
    </Options>
  </Metadata>

  <UI>
    <Container option="DATATAB">
      <Group option="DATA_SELECT" open="true">
        <DataItem data="DATASOURCE"/>
      </Group>
      <Group option="ITEM_SELECT" open="true">
        <RoleItem role="ITEMS"/>
      </Group>
    </Container>
  </UI>

  <CodeTemplate>
    <![CDATA[
proc irt data = $DATASOURCE RESFUNC=RASCH;
var #foreach($item in $ITEMS )$item #end ;
title "IRT from Task";
run;
      ]]>
    </CodeTemplate>
  </Task>
```

SGF_MyIRTModel.CTM

```
<?xml version="1.0" encoding="UTF-8"?><Task runNLS="never" schemaVersion="5.0">
  <Registration>
    <Name>SGF MyIRTModel</Name>
    <Description>IRT Procedure with Model Select LName FName 1/10/2017 </Description>
    <Procedures>IRT</Procedures>
    <Version>3.5</Version>
    <Links>
      <Link href="http://support.sas.com/">SAS Support</Link>
    </Links>
  </Registration>

  <Metadata>
    <DataSources>
      <DataSource name="DATASOURCE">
        <Roles>
          <Role type="N" maxVars="100" minVars="1" order="True"
            name="ITEMS">Select Items</Role>
        </Roles>
      </DataSource>
    </DataSources>

    <Options>
      <Option name="DATATAB" inputType="string">DATA</Option>
      <Option name="DATA_SELECT" inputType="string">DATASET</Option>
      <Option name="ITEM_SELECT" inputType="string">ITEMS</Option>
      <Option name="GROUPCHECK" inputType="string">MODEL</Option>
      <Option name="chkRASCH" inputType="checkbox" defaultValue="0" >Rasch</Option>
      <Option name="chkONE" inputType="checkbox" defaultValue="0" >One Parameter</Option>
      <Option name="chkTWO" inputType="checkbox" defaultValue="0" >Two Parameter</Option>
      <Option name="chkTHREE" inputType="checkbox" defaultValue="0" >Three Parameter</Option>
      <Option name="chkFOUR" inputType="checkbox" defaultValue="0" >Four Parameter</Option>
      <Option name="chkGRADED" inputType="checkbox" defaultValue="0" >Graded Response</Option>
      <Option name="chkGPC" inputType="checkbox" defaultValue="0" >Generalized Partial Credit</Option>
    </Options>
  </Metadata>

  <UI>
    <Container option="DATATAB">
      <Group open="true" option="DATA_SELECT">
        <DataItem data="DATASOURCE"/>
      </Group>
      <Group open="true" option="ITEM_SELECT">
        <RoleItem role="ITEMS"/>
      </Group>
      <Group open="true" option="GROUPCHECK">
        <OptionItem option="chkRASCH"/>
        <OptionItem option="chkONE"/>
        <OptionItem option="chkTWO"/>
        <OptionItem option="chkTHREE"/>
        <OptionItem option="chkFOUR"/>
        <OptionItem option="chkGRADED"/>
        <OptionItem option="chkGPC"/>
      </Group>
    </Container>
  </UI>

  <CodeTemplate>
    <![CDATA[
%macro irt(resfunc);
proc irt data = $DATASOURCE RESFUNC=&resfunc;
var #foreach($item in $ITEMS )$item #end ;
title "IRT from Task Using &resfunc Model";
run;
%mend irt;

#if ($chkRASCH == 1)
  %irt(RASCH);
#end
#if ($chkONE == 1)
  %irt(ONEP);
#end
#if ($chkTWO == 1)
  %irt(TWOP);
#end
#if ($chkTHREE == 1)
  %irt(THREEP);
#end
#if ($chkFOUR == 1)
  %irt(FOURP);
#end
#if ($chkGRADED == 1)
  %irt(GRADED);
#end
#if ($chkGPC == 1)
  %irt(GPC);
#end

    ]]>
  </CodeTemplate>
</Task>
```

SGF_MyIRTSurvey.CTM

```
<?xml version="1.0" encoding="utf-8"?>
<Task schemaVersion="5.0" runNLS="never">
  <Registration>
    <Name>IRT Survey Analysis</Name>
    <Description>IRT Analysis for Survey LName FName 1/10/2017</Description>
    <Procedures>IRT</Procedures>
    <Version>3.5</Version>
    <Links>
      <Link href="http://support.sas.com">SAS Support</Link>
    </Links>
  </Registration>

  <Metadata>
    <DataSources>
      <DataSource name="DATASOURCE">
        <Roles>
          <Role type="N" minVars="1" maxVars="100" order="True"
            name = "ITEMS">Select Items</Role>
        </Roles>
      </DataSource>
    </DataSources>
    <Options>
      <Option name="DATATAB" inputType="string">DATA</Option>
      <Option name="DATAGROUP" inputType="string">DATA</Option>
      <Option name="ITEM_SELECT" inputType="string">SURVEY QUESTIONS</Option>
      <Option name="OPTIONSTAB" inputType="string">OPTIONS</Option>
      <Option name="GROUPLISTING" inputType="string">ITEM OUTPUT FOR LISTING</Option>
      <Option name="GROUPOUTPUT" inputType="string">OUTPUT DATASETS</Option>
      <Option name="chkITEMS" inputType="checkbox" defaultValue="0">Item Statistics</Option>
      <Option name="chkICC" inputType="checkbox" defaultValue="0">Item Characteristic Curves</Option>
      <Option name="chkOUTPUT" inputType="checkbox" defaultValue="0">Create Output Datasets</Option>
      <Option name="GROUPCOMBO" inputType="string">CUSTOMER SURVEY</Option>
      <Option name="comboTYPE" inputType="combobox" defaultValue="LSF">Choose from the list below:</Option>
      <Option name="LSF" inputType="string">Loyalty (Short Form)</Option>
      <Option name="LLF" inputType="string">Loyalty (Long Form)</Option>
      <Option name="SAT" inputType="string">Satisfaction</Option>
      <Option name="labelOUTPUT" inputType="string">Produces three output datasets:
        _Scores, _Model, and _ParameterEstimates</Option>
    </Options>
  </Metadata>

  <UI>
    <Container option="DATATAB">
      <Group option="GROUPCOMBO" open="true">
        <OptionChoice option="comboTYPE">
          <OptionItem option="LSF"/>
          <OptionItem option="LLF"/>
          <OptionItem option="SAT"/>
        </OptionChoice>
      </Group>
      <Group option="DATAGROUP" open="true">
        <DataItem data="DATASOURCE"/>
      </Group>
      <Group option="ITEM_SELECT" open="true">
        <RoleItem role="ITEMS"/>
      </Group>
    </Container>
    <Container option="OPTIONSTAB">
      <Group option="GROUPLISTING" open="true">
        <OptionItem option="chkITEMS"/>
        <OptionItem option="chkICC"/>
      </Group>
      <Group option="GROUPOUTPUT" open="true">
        <OptionItem option="chkOUTPUT"/>
        <OptionItem option="labelOUTPUT"/>
      </Group>
    </Container>
  </UI>

  <CodeTemplate>
    <![CDATA[
%macro LSF;
  ods graphics on;
  proc freq data=$DATASOURCE; tables #foreach($item in $ITEMS)$item #end ; run;
  #if ($chkOUTPUT ==1) ods output ParameterEstimates=LSF_ParameterEstimates ; #end
  proc irt data=$DATASOURCE resfunc=GRADED
    #if ($chkITEMS == 1) itemstat #end
    #if ($chkICC == 1) plots=icc #end
    #if ($chkOUTPUT == 1) out=LSF_Scores outmodel=LSF_Model #end ;
    var #foreach($item in $ITEMS)$item #end ;
  run;
  ods graphics off;
%mend LSF;

%macro LLF;
  ods graphics on;
  proc means data=$DATASOURCE; var #foreach($item in $ITEMS)$item #end ; run;
  #if ($chkOUTPUT ==1) ods output ParameterEstimates=LLF_ParameterEstimates ; #end
  proc irt data=$DATASOURCE resfunc=GRADED
    #if ($chkITEMS == 1) itemstat #end
```

```

        #if ($chkICC == 1) plots=icc #end
        #if ($chkOUTPUT == 1) out=LLF_Scores outmodel=LLF_Model #end ;
        var #foreach($item in $ITEMS )$item #end ;
    run;
    ods graphics off;
%mend LLF;

%macro SAT;
    ods graphics on;
    proc univariate data=$DATASOURCE; var #foreach($item in $ITEMS )$item #end ; run;
    #if ($chkOUTPUT ==1) ods output ParameterEstimates=SAT_ParameterEstimates ; #end
    proc irt data=$DATASOURCE resfunc=GPC
        #if ($chkITEMS == 1) itemstat #end
        #if ($chkICC == 1) plots=icc #end
        #if ($chkOUTPUT == 1) out=SAT_Scores outmodel=SAT_Model #end ;
        var #foreach($item in $ITEMS )$item #end ;
    run;
    ods graphics off;
%mend SAT;

#if ($comboTYPE == "LSF") %LSF; #end
#if ($comboTYPE == "LLF") %LLF; #end
#if ($comboTYPE == "SAT") %SAT; #end

    >>
</CodeTemplate>
</Task>

```

MakeData.SAS

```

/* Data for IRT Custom Task Development

SURVEY is a dataset with ten customer satisfaction Likert scale
questions (1-7), all of which correlate with the customer's overall
level of satisfaction.

TEST is a dataset with ten educational test items scored as 0 or 1,
all of which were influenced by the trait being tested.

*/

data survey;
do Customer = 1 to 1000;
    Satisfaction = round(7*ranuni(0), 1);
    q1 = round(((rantbl(1, .1, .2, .1, .2, .1, .1, .2) + Satisfaction )/2), 1);
    q2 = round(((rantbl(2, .1, .1, .2, .2, .2, .1, .1) + Satisfaction )/2), 1);
    q3 = round(((rantbl(3, .1, .1, .1, .2, .1, .3, .1) + Satisfaction )/2), 1);
    q4 = round(((rantbl(4, .1, .2, .1, .2, .2, .1, .1) + Satisfaction )/2), 1);
    q5 = round(((rantbl(5, .1, .1, .2, .1, .2, .2, .1) + Satisfaction )/2), 1);
    q6 = round(((rantbl(6, .1, .1, .2, .2, .1, .2, .1) + Satisfaction )/2), 1);
    q7 = round(((rantbl(7, .1, .2, .1, .1, .3, .1, .1) + Satisfaction )/2), 1);
    q8 = round(((rantbl(8, .2, .1, .1, .2, .2, .1, .1) + Satisfaction )/2), 1);
    q9 = round(((rantbl(9, .1, .2, .1, .2, .2, .1, .2) + Satisfaction )/2), 1);
    q10 = round(((rantbl(10, .1, .1, .2, .2, .1, .2, .1) + Satisfaction )/2), 1);
    Overall_Satisfaction = round(mean(of q1-q10), .01);
    output;
end;
drop Satisfaction;
run;

proc means; var q1-q10 Overall_Satisfaction; run;
proc corr; var q1-q10; run;
proc irt data=survey resfunc=GPC itemstat; var q1-q10; run;

data test;
do Student = 1 to 10000;
    Student_Trait = normal(0);
    if Student_Trait > normal(1)-.9 then q1 = 1; else q1 = 0;
    if Student_Trait > normal(2)-.8 then q2 = 1; else q2 = 0;
    if Student_Trait > normal(3)-.5 then q3 = 1; else q3 = 0;
    if Student_Trait > normal(4)-.2 then q4 = 1; else q4 = 0;
    if Student_Trait > normal(5)-.1 then q5 = 1; else q5 = 0;
    if Student_Trait > normal(6)+.3 then q6 = 1; else q6 = 0;
    if Student_Trait > normal(7)+.4 then q7 = 1; else q7 = 0;
    if Student_Trait > normal(8)+.6 then q8 = 1; else q8 = 0;
    if Student_Trait > normal(9)+.7 then q9 = 1; else q9 = 0;
    if Student_Trait > normal(10)+.9 then q10 = 1; else q10 = 0;
    Overall_Performance = sum(of q1-q10);
    output;
end;
drop Student_Trait;
run;

proc means; var q1-q10 Overall_Performance; run;
proc corr; var q1-q10; run;
proc irt data=test resfunc=RASCH itemstat; var q1-q10; run;

```