User Guide:
KTAB Sensitivity Analysis for SMP
v 0.5

Introduction

This is a bespoke application to facilitate analyzing sensitivity of the SMP model results with respect to both model parameters and actor attributes. Fundamentally, it works by compiling enumerated lists of possible values, such as *VotingRule = (Binary, Prop, Cubic)*, or *SActor-03.Influence = (50, 55, 60, 65, 70)* – these are examples of what we call a "sensitivity analysis specification" (SAS). These can then be combined through filters and cross products to create a list of SAS's. Once the SAS's are complete, clicking a run button will:

1. Generate xml files for all specifications
2. Run the SMP model on all of them
3. Save results from all into a single database to facilitate analysis

High-Level Navigation

Navigation throughout the application is achieved by way of three main sets of GUI elements – the top menu, toolbar underneath that, and page type buttons at the bottom.

Menu

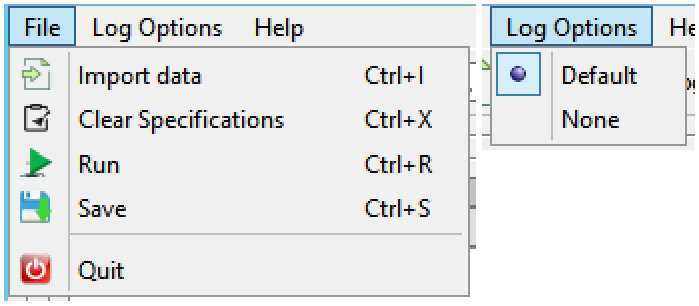


Figure 1 - Menu Items

The *File* menu has three options:

- *Import Data*: Load an existing SMP model input file – both csv and xml-formatted files are accepted *Clear Specifications*: clear all defined specifications. If an xml-formatted input file is loaded, the stored parameters are ignored.
- *Run*: Execute the SMP model on all defined specifications. This performs the following steps:
    - ask the user for the location in which to save output files
    - save a text file named *specList.txt*, which lists all the specifications
    - save xml-formatted SMP model input files for all defined specifications; these are stored in the same location as the original input file; the scenario name & description in these files have "Scen_#" prepended to the original fields, where "#" is a sequential number over all the defined SAS's
    - run the SMP model on all generated SMP input files
- *Quit*: quit the application

There are two options on the *Log Options* menu, which control saving of the SMP model log output:

- *Default*: the log output will be named using the format *ktab-smp-GMT_Timestamp_log.txt*, where "*GMT_Timestamp*" is replaced with the GMT-formatted timestamp of when the model runs begin; the user is asked for the directory in which to save the file
- *None*: don't save the log output (if this option is selected, the user is still asked where to save the specification listing)

Toolbar

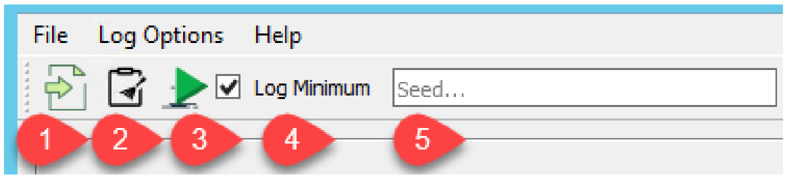


Figure 2 – Toolbar Items

1. Click here to load an existing SMP data file – same as *File->Import Data*.
2. Click here to clear all defined specifications – same as *File->Clear Specifications*.
3. Click here to run the SMP model on all defined specifications – same as *File->Run*.
4. Checking this box enables minimal data logging in the SMP model, which dramatically reduces the size of the database storing model results. It also restricts the detail with which model results can be analyzed.
5. Set the numeric random seed for the SMP model, when it is run on each defined specification. If left blank, a seed is randomly generated.

Specification Type Buttons

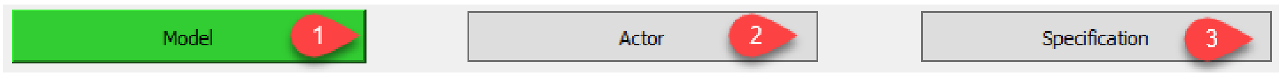


Figure 3 - Specification Type Buttons

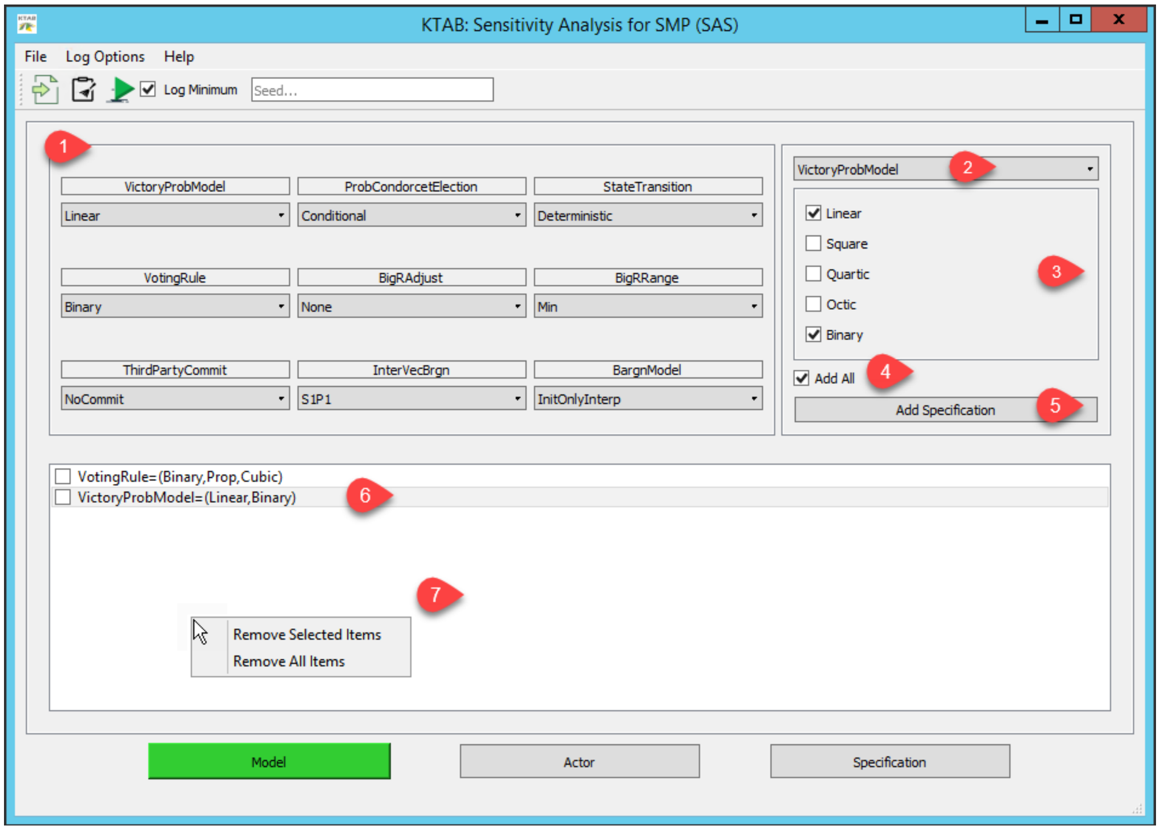
The vast majority of the user interface for defining specifications is spread across three pages:

1. Define simple specifications regarding SMP model parameters.
2. Define simple specifications regarding SMP model input data.
3. Define complex specifications by joining defined specifications.

These three pages will be covered in more detail next.

Model Page

This page is where the user can define simple specifications to test the sensitivity of SMP model results to model parameters. For more information about the model parameters that can be varied, please see the KTAB SMP tutorial. The screenshot in Figure 4 shows two specifications. The first will be used to generate three different SMP model input xml files – the first with nothing changed from the default except for *VotingRule* set to *Binary*, the second with it set to *Prop*, and the third with it set to *Cubic*. The second specification will be used to create two model input files, each varying only *VictoryProbModel* from *Linear* to *Binary*.



VictoryProbModel: Linear
ProbCondorcetElection: Conditional
StateTransition: Deterministic
VotingRule: Binary
BigRAdjust: None
BigRRange: Min
ThirdPartyCommit: NoCommit
InterVecBrgn: S1P1
BargnModel: InitOnlyInterp

VictoryProbModel
[✓] Linear
[ ] Square
[ ] Quartic
[ ] Octic
[✓] Binary

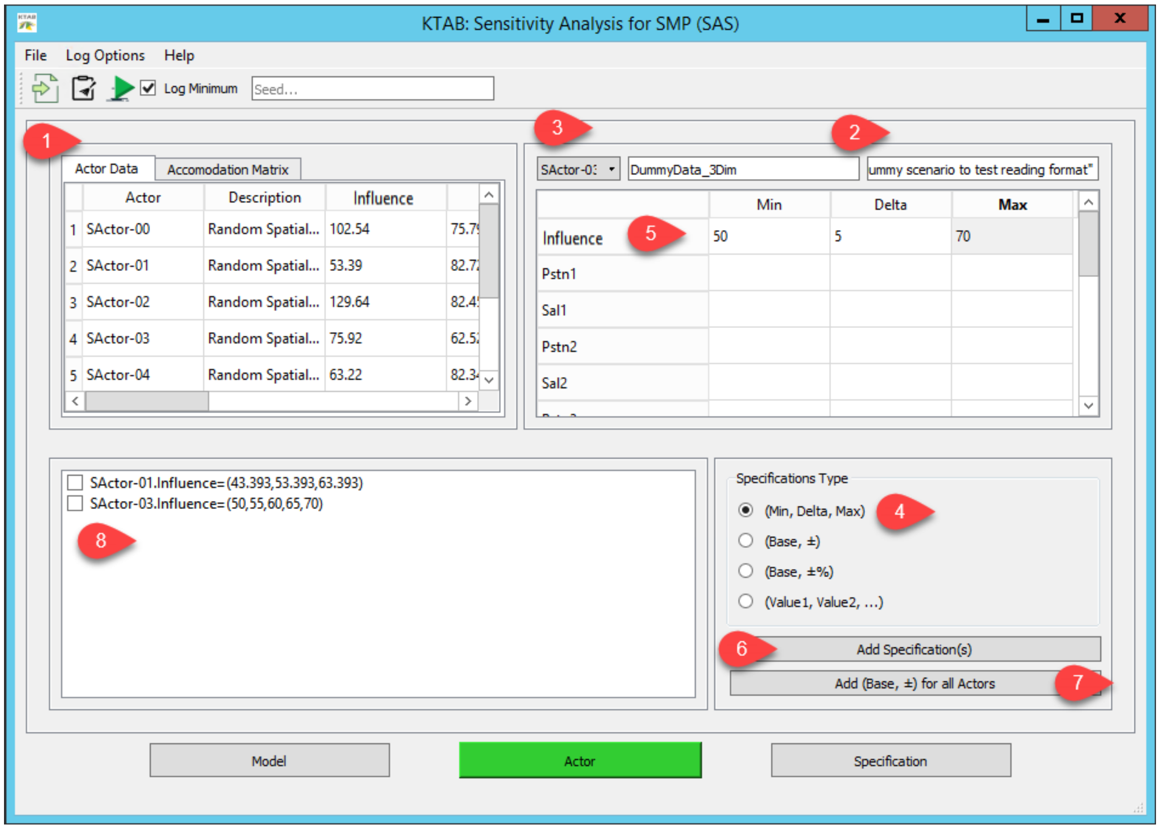
[ ] VotingRule=(Binary,Prop,Cubic)
[ ] VictoryProbModel=(Linear,Binary)

Figure 4 - Model Specifications Page

1. This grid lists the nine model parameters which can be varied, and each has a dropdown box which enumerates the possible values. Selecting a parameter value in this grid has no affect – the grid is merely informative. Hovering the mouse pointer over a parameter or specific value will cause a descriptive tooltip to appear.
2. This is a dropdown box which lists the model parameters. Select an item from this dropdown to vary it in a SAS.
3. When a parameter is selected in **2**, this grid automatically populates with the possible values. Toggle the checkboxes on to indicate which of the values should be included in a specification.
4. Checking here will toggle on/off all the checkboxes in **3**.
5. Click here to generate the sensitivity analysis specification for the selected model parameter. It will then be listed in **6**.
6. All the defined model parameter specifications are listed here; each is of the form *parameter=(value1, value2, …)*. Specifications added here are also added to the main listing on the Specifications page.
7. Right-clicking anywhere in **6** will bring up a context menu from which the user may either remove all the defined specifications, or only checked specifications.

Actor Page

On this page, the user can define sensitivity analysis specifications for actor attributes as defined in the SMP model input data. Since actor attributes are continuous numeric data, there are more possibilities for how to build the specifications. No matter the logic used, however, the end result of a SAS is still an enumeration of values. In Figure 5, two defined specifications can be seen. The first is a *(Base, ±)* type, which was designed for the influence of actor SActor-01 with the ± delta set to 10. The second was defined as a (Min, Delta, Max) specification for actor SActor-03's influence, with values (50, 5, 70).



	Actor	Description	Influence	
1	SActor-00	Random Spatial...	102.54	75.7
2	SActor-01	Random Spatial...	53.39	82.7
3	SActor-02	Random Spatial...	129.64	82.4
4	SActor-03	Random Spatial...	75.92	62.5
5	SActor-04	Random Spatial...	63.22	82.3

SActor-03 | DummyData_3Dim | ummy scenario to test reading format"

	Min	Delta	Max
Influence	50	5	70
Pstn1			
Sal1			
Pstn2			
Sal2			

SActor-01.Influence=(43.393,53.393,63.393)
SActor-03.Influence=(50,55,60,65,70)

[✓ (Min, Delta, Max)] (Base, ±) (Base, ±%) (Value1, Value2, ...)

Figure 5 - Actor Specifications Page

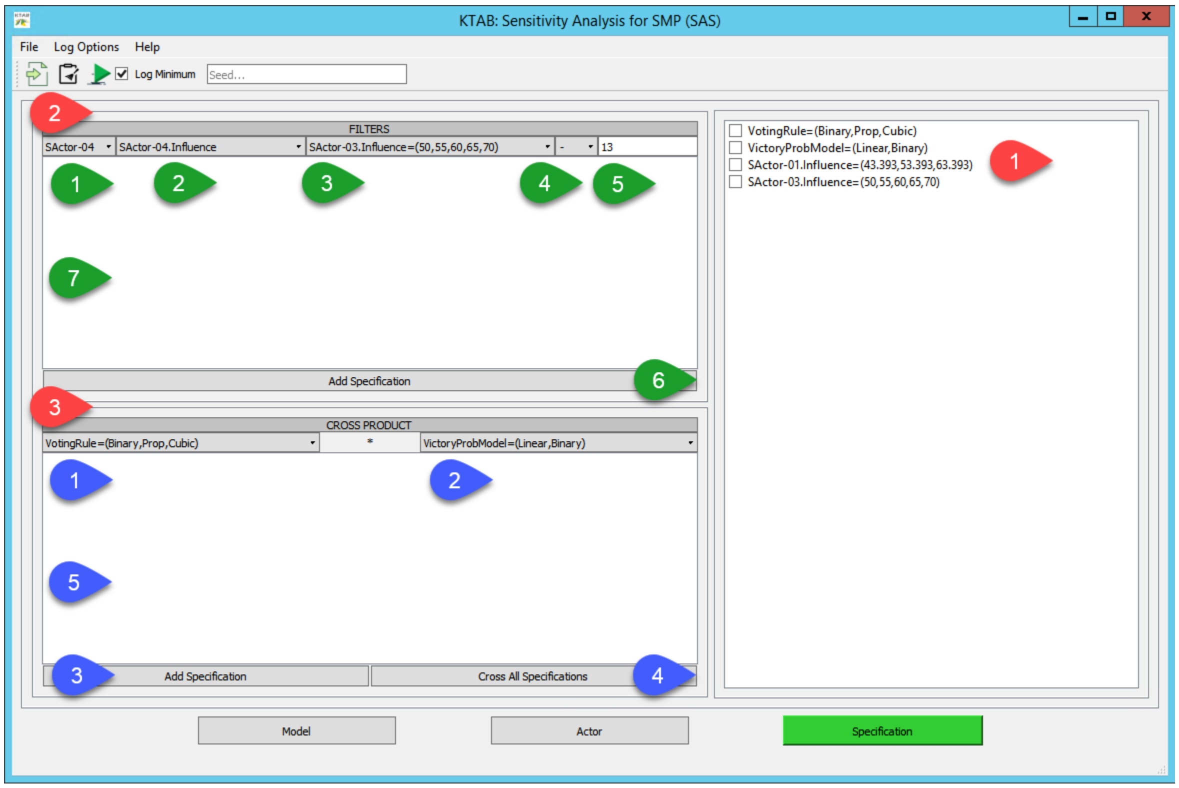
1. This grid lists the actor attributes, as defined in the input data file, in two tabs. The *Actor Data* input tab list the influence, position(s), and salience(s) for each actor. The *Accommodation Matrix* tab is a placeholder for future SMP model development. While specifications can be defined for values in the accommodation matrix, there's no point in doing so, as the current SMP model does not use them. This data is shown here just for reference.
2. Shows the scenario name and description from the input data file.
3. This is a dropdown box which lists all the actors. Selecting an actor here will populate and prepare the grid of attributes in **5**.
4. There are four ways actor attribute specifications can be defined, corresponding to the four radio buttons here:
    - *(Min, Delta, Max)*: create a SAS by specifying the lowest value, highest value, and the increment size; if this is selected, the grid in **5** will have columns headed *Min*, *Delta*, *Max*
    - *(Base, ±)*: create a SAS by specifying an amount to be added to / subtracted from the input attribute; if this is selected, the grid in **5** will have columns headed *Base* and *±*, and the base column will populate with the actor's attributes
    - *(Base, ±%)*: this is identical to *(Base, ±)*, except that the delta is in percentage terms; the header of the second column in **5** is changed accordingly

   - *(Value 1, Value 2, …)*: create a SAS by specifying a variable-length list of values; if this is selected, **5** will have columns headed *Val 1* and *Val 2*; right-clicking in the grid will bring up a context menu with an option to add an additional column
5. When an actor is selected in **3**, this grid is automatically populated with all that actor's attributes, and possibly the current values for those attributes. To create a specification, simply type in numerical values in the correct row & column of this grid. If desired, it's possible to create the same type of specification for all attributes of the given actor. However, if different types of specifications are desired, they must be generated sequentially.
6. After the grid in **5** is completed with the generating values for the desired specification(s), click here to create the specification(s), which will then be saved in **8**.
7. This button is a shortcut to create *(Base, ±)* or *(Base, ±%)* specifications for all actors, where the same ± delta will be used for all actors. To use this functionality, select any actor, select an appropriate specification type, then fill in the delta value(s) for the desired attributes in **5**. Click this button to create the same specifications for all actors.
8. All defined specifications go here, as well as on the main list on the Specifications page. As with the Model page, right-clicking will bring up a context menu from where defined specifications can be removed. All specifications are of the form *actor.attribute=(value1, value2, …)*

Specification Page

The last page consists of three major sections, shown in Figure 6 - Specifications Page. This is where all defined specifications ultimately reside, and where more complex – and interesting – sensitivity analysis specifications can be defined. All complex specifications are of the form *(item1, item2, …) = ((item1_value: item2_value, …),(item1_value: item2_value, …),…)*, where an item can be a *parameter* or *actor.attribute*.

1. This is where all defined specifications ultimately reside, and is the list that will be used to generate SMP model input xml files for modeling. As with the other pages, there is a right-click context menu to remove any specifications.
2. In this section, complex specifications for actor attributes can be defined. A Filter specification defines an attribute for an actor as varying in parallel with a previously-defined specification.
3. This is where complex specifications can be defined, in which all possible pairwise combinations of the values in a pair of specifications are generated, in the spirit of a factorial experimental design.



KTAB: Sensitivity Analysis for SMP (SAS)

File Log Options Help

[✓ Log Minimum] Seed...

FILTERS

SActor-04 | SActor-04.Influence | SActor-03.Influence=(50,55,60,65,70) | - | 13

Add Specification

CROSS PRODUCT

VotingRule=(Binary,Prop,Cubic) * VictoryProbModel=(Linear,Binary)

Add Specification | Cross All Specifications

- [ ] VotingRule=(Binary,Prop,Cubic)
- [ ] VictoryProbModel=(Linear,Binary)
- [ ] SActor-01.Influence=(43.393,53.393,63.393)
- [ ] SActor-03.Influence=(50,55,60,65,70)

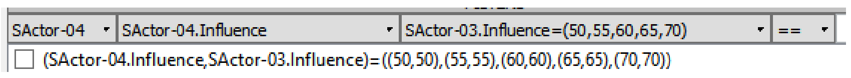
Model | Actor | Specification

Figure 6 - Specifications Page

Filters

A logical and important use for sensitivity analysis is testing the sensitivity to jointly-varying actor attributes. For example, perhaps SActor-03 is a pivotal actor in bargaining in a model run. It's natural to test the sensitivity of the results to his attributes – perhaps we want to see how the results change as his influence is allowed to decrease from ~75 to 70, then 65 and down to 50. This is what the 4th defined spec does (see Figure 6). Note in Figure 5 that SActor-04's influence is approximately 13 less than SActor-03, and suppose that there is some underlying correlation between their influences, such that if one loses influence, so would the other. A Filter specification can be defined to model precisely this kind of jointly-varying behavior. The numbers below refer to the green markers in Figure 6.

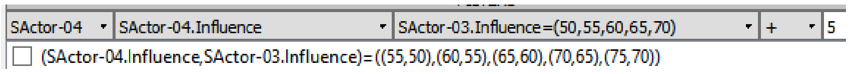
1. This is a dropdown box which lists all the actors. This is used to select the actor whose attribute will vary jointly with a defined specification.
2. Once an actor is selected in **1**, this dropdown box will be populated with that actor's attributes. Select from here the attribute to vary.
3. This dropdown box lists all the simple actor specifications which have been defined. From here, select the specification for varying the attribute in **2**.
4. The nature of the correlative relationship is defined by one of the four mathematical operations listed in this dropdown box:
    - "==": This will build a SAS in which the attribute in **2** takes on values identical to those in the SAS in **4**.



SActor-04 | SActor-04.Influence | SActor-03.Influence=(50,55,60,65,70) | ==

- [ ] (SActor-04.Influence,SActor-03.Influence)=((50,50),(55,55),(60,60),(65,65),(70,70))

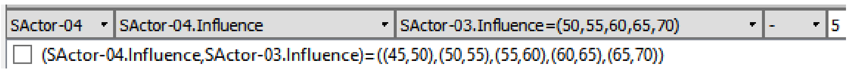
- "+": This will build a SAS in which the attribute in **2** takes on the values in the SAS in **4**, but increased by the numerical value in **5**.



SActor-04 | SActor-04.Influence | SActor-03.Influence=(50,55,60,65,70) | + | 5

(SActor-04.Influence,SActor-03.Influence)=((55,50),(60,55),(65,60),(70,65),(75,70))

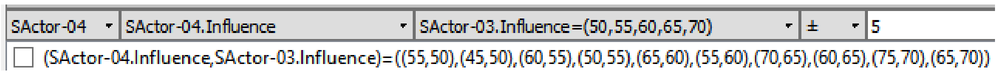
- "-": This will build a SAS in which the attribute in **2** takes on the values in the SAS in **4**, but decreased by the numerical value in **5**.



SActor-04 | SActor-04.Influence | SActor-03.Influence=(50,55,60,65,70) | - | 5

(SActor-04.Influence,SActor-03.Influence)=((45,50),(50,55),(55,60),(60,65),(65,70))

- "±": This will build a SAS that merges the previous two.

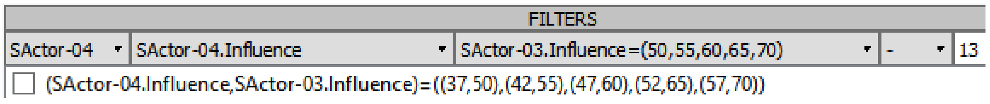


SActor-04 | SActor-04.Influence | SActor-03.Influence=(50,55,60,65,70) | ± | 5

(SActor-04.Influence,SActor-03.Influence)=((55,50),(45,50),(60,55),(50,55),(65,60),(55,60),(70,65),(60,65),(75,70),(65,70))

5. For all filter types listed in **4** except for equality, enter the numeric value of the desired shift in this box.
6. Click here to generate the Filter specification as defined in **1** through **5**. This will add it to the listing in **7** and the main listing on this page.
7. This is where all defined Filter specifications are listed. As with the other listings, there is a right-click context menu to remove defined specifications.

After a Filter specification is defined, it will generally make sense to remove the original simple specification (selected in **3**) from the main specifications list, since the filter supersedes it. This isn't required, though, as it may be desirable to separately test the SMP model sensitivity to just the original specification – as well as the filter. The example above about correlating the influence of SActor-04 with the specification for SActor-03's influence would look like Figure 7.



FILTERS

SActor-04 | SActor-04.Influence | SActor-03.Influence=(50,55,60,65,70) | - | 13

(SActor-04.Influence,SActor-03.Influence)=((37,50),(42,55),(47,60),(52,65),(57,70))

Figure 7 - Example Filter SAS

Cross Product

It might not always – or even usually – make sense for a user to perform sensitivity analysis by varying model parameters from the defaults or actor attributes from the baselines individually in sequence. Consider the specifications defined in Figure 4: *VotingRule=(Binary, Prop, Cubic)* and *VictoryProbModel=(Linear, Binary)*. If we add these only, a total of up to 5 non-default SMP models will be run:

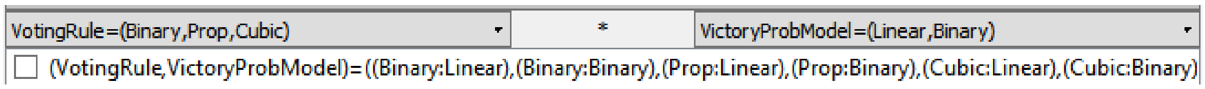
- Voting Rule set to *Binary* and everything else at default
- Voting Rule set to *Prop* and everything else at default
- Voting Rule set to *Cubic* and everything else at default
- Victory Probability Model set to *Linear* and everything else at default
- Victory Probability Model set to *Binary* and everything else at default

Instead, what will often make more sense is to simultaneously vary these two specifications over all the possible combinations of their values. This would be called a "factorial design"; Cross Product specifications do precisely this. The numbers below refer to the blue markers in Figure 6.

1. This dropdown lists all defined specifications in the final specifications listing. Select from here the first specification in the pair which will be crossed to define the new complex specification.

2. This is identical to **1**, and is used to select the second specification in the pair to be crossed.
3. Click here to generate the Cross Product specification.
4. In a full factorial experiment, all combinations of all values from all specifications would be modeled. This could be done manually by defining a sequence of Cross Product SAS's: first crossing pairs of simple (and Filter) specifications, then crossing Cross Product Specifications – removing source SAS's along the way – until there only remained a single large Cross Product SAS. This manual process can be performed automatically by clicking this button.
5. This is where all defined Cross Product specifications are listed. As with the other listings, there is a right-click context menu to remove defined specifications.

After a Cross Product specification is defined, it will generally make sense to remove the two source specifications. From the above example of Voting Rule and Victory Probability Model specifications, the Cross Product SAS would look like Figure 8. This specification would be used to create six SMP model input xml files, varying only those two model parameters, with each file encoding the options in one of the pairs (i.e., the first file would set Voting Rule to *Binary* and Victory Probability Model to *Linear*).


VotingRule=(Binary,Prop,Cubic) * VictoryProbModel=(Linear,Binary)

☐ (VotingRule,VictoryProbModel)=((Binary:Linear),(Binary:Binary),(Prop:Linear),(Prop:Binary),(Cubic:Linear),(Cubic:Binary)

Figure 8 - Example Cross Product SAS

Finishing with the ongoing example of the three simple specifications and single Filter SAS, the full factorial Cross Product SAS would look like Figure 9 (only part can be shown, since it's so long). This Cross Product SAS would be used to generate 90 SMP model input xml files encoding the specified values for the model parameters and actor attributes shown.

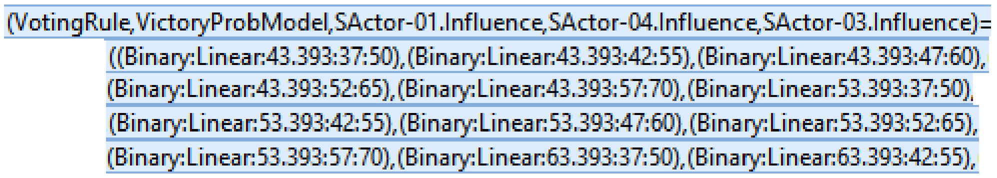

(VotingRule,VictoryProbModel,SActor-01.Influence,SActor-04.Influence,SActor-03.Influence)=
((Binary:Linear:43.393:37:50),(Binary:Linear:43.393:42:55),(Binary:Linear:43.393:47:60),
(Binary:Linear:43.393:52:65),(Binary:Linear:43.393:57:70),(Binary:Linear:53.393:37:50),
(Binary:Linear:53.393:42:55),(Binary:Linear:53.393:47:60),(Binary:Linear:53.393:52:65),
(Binary:Linear:53.393:57:70),(Binary:Linear:63.393:37:50),(Binary:Linear:63.393:42:55),

Figure 9 - Example Full Factorial Cross Product SAS (partial)