mguchiQ

Model Building Tutorial 3

Conditions, Scenarios, and Others

Contents

Overview

The objective of this document is to build on the model we created in the tutorial described in the document **022.mguchiQ – Model Building Tutorial 2** by introducing some more features of mguchiQ.

Specifically we will be introducing **Conditions, Scenarios, Function Properties,** and **Restrictions**.

Conditions

When initiating a run of our model against a set of data, all the product instances from our set of data are included in the result set.

Conditions allow us to create additional result sets that will only include portions of our data set based on **conditions** that we specify in our model.

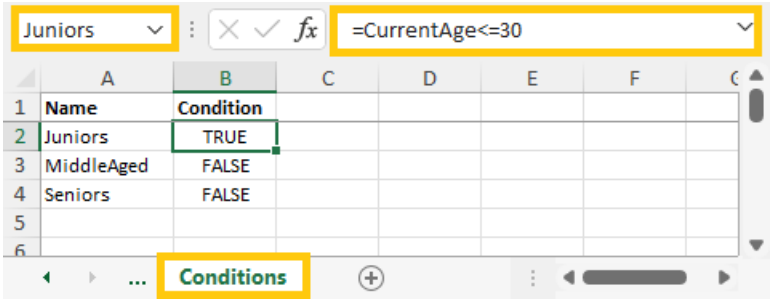
Let's assume that we want to see separate results sets based on the **current age** of our policy holders, possibly based on the following:

Condition	Description
Juniors	30 years old or younger
Middle Aged	Between 31 and 59
Seniors	60 years old or older

We would then specify our **conditions** as follows:

Condition	Formula
Juniors	CurrentAge <= 30
MiddleAged	AND(CurrentAge > 30,CurrentAge < 60)
Seniors	CurrentAge >= 60

Conditions are specified in the **Conditions** sheet as follows:



Scenarios

Scenarios allow us to change **Non-Instance Variables** in order to arrive at a new result set. So, for example, we can say:

- How will our results change if **expenses increase**, or
- How will the results change if **expenses increase** and **inflation increases**, or
- How will the results change if **mortality decreases** (i.e., policy holders live longer)

A scenario is thus defined as a change to the input variables to a run. Note that only **Single Variables, Series Variables** and **Table Variables** can be changed in a scenario – it does not make sense to be able to change **Instance Variables** in a scenario.

The simplest scenario is where there is only one change to a **Single Variable**. We only have one **Single Variable** in our model so far, **Expense**. We could create a scenario, called **Shock Expense**, where we increased the **Expense** by, say 10%. We can then look at the results in one of two ways:

- What is the **absolute** result for **Shock Expense.**
- What is the **change** as a result of **Shock Expense**. This will look at the effect that **Shock Expense** has compared to our **base** scenario (where no changes have been made to the input variables).

We would describe our **Shock Expense** scenario as follows:

Scenario	Variable	Formula
ShockExpense	Expense	Expense * 1.1

What this describes is that our **Shock Expense** scenario:

- Affects a **Single Variable, Expense**
- **Expense** is **multiplied** by **1.1** (i.e., expenses are increased by 10%)

Given there is a general increase in expenses we may wish to add another adjustment to our **Shock Expense** scenario, and that is to adjust **Inflation** upwards as well.

For our example we will assume we want to add 2% to inflation. We would describe this scenario adjustment as follows:

Scenario	Variable	Formula
ShockExpense	Inflation	Inflation[i] + 0.02

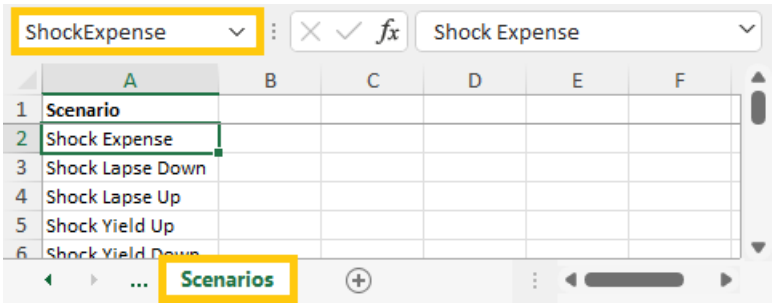
Thus, our new **Shock Expense** scenario would look as follows:

Scenario	Variable	Formula
ShockExpense	Expense	Expense * 1.1
	Inflation	Inflation[i] + 0.02

Some important notes, with examples, about specifying scenario adjustments:

Variable Type	Can Reference	Restrictions and Notes	Examples
Single variable	• Single variables • Constants	**Single Variables** can only reference other **Single variables** or **Constants**	• Expense * 1.15 • Expense * ExpenseIncrease *(where **ExpenseIncrease** is a **single variable** or a **constant**)*
Series variable	• Single variables • Series variables • Constants	**Series variables** can reference other **Single variables**, **Series variables** or **Constants**. Evaluated for a range of values **i**, from **1** to **the size of the series**	• Inflation[i] + 0.02 • Inflation[i] + InflationAdjustment *(where **InflationAdjustment** is a **single variable** or a **constant**)* • Yield[i] * YieldAdjustment[i] *(where **YieldAdjustment** is a **series variable**)* • IF(i < 360, LapseRate[i] *0.5, IF(i < 720, LapseRate[i] *0.6, LapseRate[i] *0.7))
Table variable	• Single variables • Series variables • Table variables • Constants	**Table variables** can reference other **Single variables**, **Series variables, Table variables,** or **Constants**. Evaluated for a range of values **i** and **j**. **i** ranges from 1 to the number of **rows** in the table. **j** ranges from **1** to the number of **columns** in the table	• IF(i < 30, Qx[i,j] * 0.9, Qx[i,j] * 0.8)

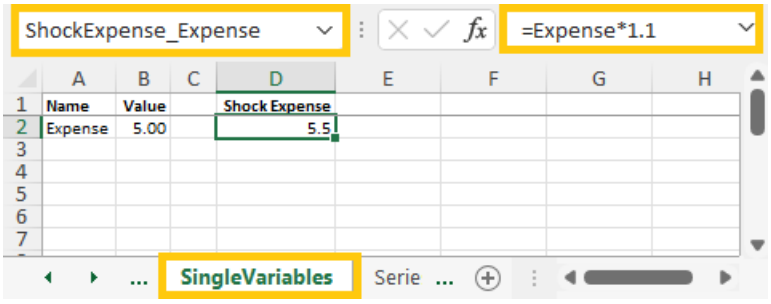
Scenarios names are specified in the **Scenarios** sheet as below:



Scenario Adjustments

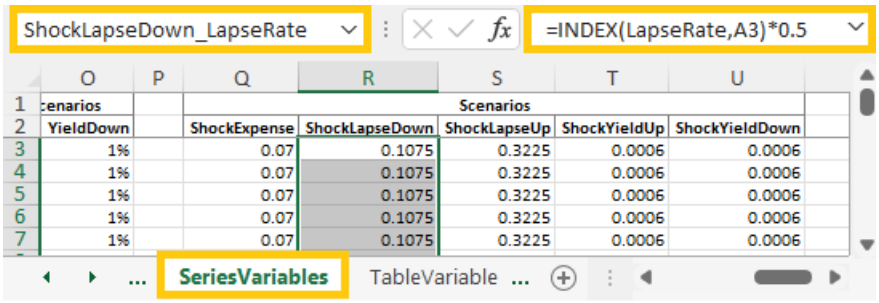
Scenarios are made up of a set of **Scenario Adjustments**. **Scenario Adjustments** are specified in the sheet in which the variable they affect are located.

A **Scenario Adjustment** to a **Single Variable** will be specified in the **SingleVariables** sheet as follows:



Note that the name of the **Scenario Adjustment** is a combination of the scenario name (**ShockExpense**) and the variable it affects (**Expense**), giving the name **ShockExpense_Expense**.

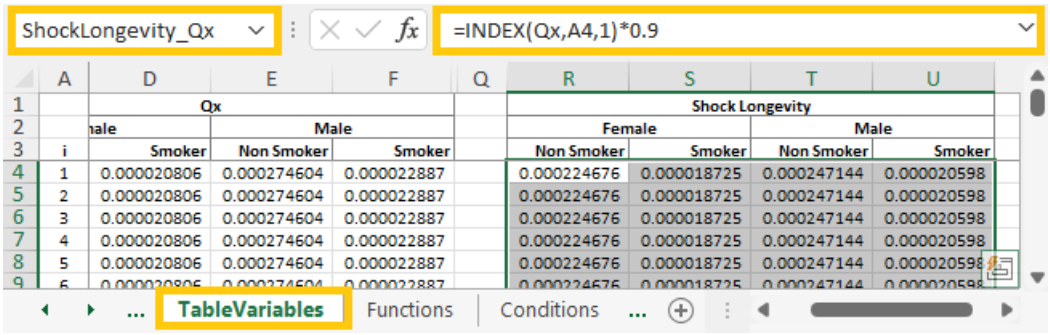
A **Scenario Adjustment** to a **Series Variable** will be specified in the **SeriesVariables** sheet as follows:



Note that the name of the **Scenario Adjustment** is a combination of the scenario name (**ShockLapseDown**) and the variable it affects (**LapseRate**), giving the name **ShockExpense_LapseDown**.

Note also that the entire range of cells that makes up the **Scenario Adjustment** must be named.

A **Scenario Adjustment** to a **Table Variable** will be specified in the **TableVariables** sheet as follows:



Note that the name of the **Scenario Adjustment** is a combination of the scenario name (**ShockLongevity**) and the variable it affects (**Qx**), giving the name **ShockLongevity_Qx**.

Note also that the entire range of cells that makes up the **Scenario Adjustment** must be named.

Function Properties

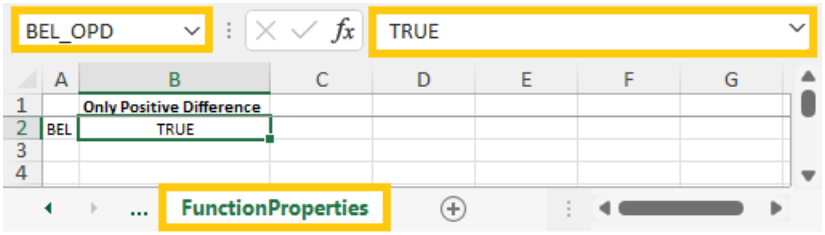
Function properties allow us to control how aggregation works when calculating scenario results.

When a run is initiated the first result set calculated is called the **Base** Scenario and it has **Absolute** values. The **Base** Scenario uses the input variables as they are initially defined, i.e. without any adjustments.

Adding scenarios to a run necessitates a recalculation as input variables to functions may have changed.

When an additional scenario is specified in a run it instructs mguchiQ to recalculate the chosen **Functions** again with an adjusted set of input values, the adjustments being the specification of the scenario, e.g. adjusting mortality rates upwards. This new result set is called an **Absolute** result set for the chosen **Scenario** as it provides absolute values. mguchiQ will also, if instructed, create another result set called a **Difference** result set, which is, on the face of it, the result of subtracting the **Absolute** result set for the scenario from the **Base** result set. In practice the **Difference** result set is not always equal to (**Absolute – Base**), the **Difference** result set may not equal (**Absolute – Base**) if any of the **Functions** in the model are specified with the attribute **Only Positive Difference**. This attribute instructs mguchiQ to only add this **Functions** value to the **Difference** result set if the **Difference** is **positive**.

Specifying the **Only Positive Difference** attribute for a function is done in the **FunctionProperties** sheet as follows:

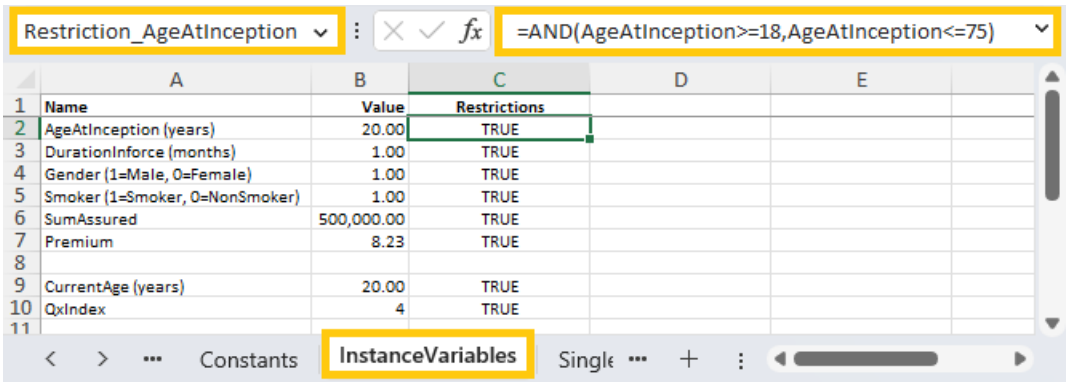


You name a cell with the combination of the function name (**BEL**) and the abbreviation for **Only Positive Difference** (**OPD**), resulting in the name **BEL_OPD**. The contents of the cell must indicate whether the function must only aggregate positive differences (TRUE / FALSE). Note that it is only necessary to specify those functions that only aggregate positive differences, as the default is FALSE.

Restrictions

Restrictions allow us to specify the acceptable range of values for **Instance Variables** of a Product. So, for example, we may want all **Life** products to have an **AgeAtInception** between 18 and 75. When we initiate a **Run** against a **Model** we can instruct the **Run** to check whether the data conforms to the restrictions of the **Model**. Data that does not conform will be excluded from the calculations and the extent of excluded data will be noted in the result details.

Specifying a **Restriction** is done in the **InstanceVariables** sheet as follows:



	A	B	C	D	E
1	**Name**	**Value**	**Restrictions**		
2	AgeAtInception (years)	20.00	TRUE		
3	DurationInforce (months)	1.00	TRUE		
4	Gender (1=Male, 0=Female)	1.00	TRUE		
5	Smoker (1=Smoker, 0=NonSmoker)	1.00	TRUE		
6	SumAssured	500,000.00	TRUE		
7	Premium	8.23	TRUE		
8					
9	CurrentAge (years)	20.00	TRUE		
10	QxIndex	4	TRUE		
11					

Your supply an appropriate name to a cell, starting with **Restriction_**, such as **Restriction_AgeAtInception**. You then supply the restrictive condition as a formula, as in the example above.

Note that a **Restriction** can reference more than one **Instance Variable**s in its formula, for example, we could have a restriction on the current age of a policy holder as follows:

Restriction_CurrentAge: = (AgeAtIncetption *12) + DurationInForce <= 1200

In reality we would rather specify this restriction on our **Derived Instance Variable** as follows:

Restriction_CurrentAge: = CurrentAge <= 100