Building State-of-the-Art Spreadsheet Models

This shares a toolbox for building spreadsheet models using a state-of-the-art approach with rows/columns and scenario model objects. Spreadsheet models are pervasive and powerful, and Excel is great for putting models in the hands of non-coders. Within companies, their use bridges between technical people and business, project management and finance team members. Formatted spreadsheets are also useful as Python model outputs for giving data to non-coders for inspection and study. Excel’s formatting features enable good user interfaces for even large data tables. The best approach shared here evolved over many years of experience building spreadsheet models whose operation is both user facing and VBA macro-driven. It avoids several pitfalls that commonly occur with poorly-designed models.

The following sections define a detailed default configuration for rows/columns tables and “scenario models,” which are an alternate, columns by rows configuration. All configurations can be created manually, but the **ExcelSteps** add-in can be used to refresh these objects automatically. Usage and advantages of the add-in are described in a third section. Finally, VBA projects can customize rows/columns and scenario model objects to create custom-designed user interfaces for models. This is described in a separate document, **Using ExcelSteps as an API for Coding Advanced VBA models**.

## Rows/Columns Model Object

A first building block for spreadsheet models and simulations is rows/columns tables with user-facing wayfinding via range names. The best-approach configuration uses named ranges for table columns, the variable name header row and the overall table range. Named ranges allow lookup formulas to utilize symbolic variable names instead of obscure range references like $C2:$C22. The approach allows column formulas to be built with standard (legacy) formula syntax.

Designing to use legacy formula syntax is a conscious and advantageous choice in most situations. It avoids needing to use Excel’s alternate, “structured reference” syntax that came with addition of the Tables feature in Excel 2007. The Tables syntax necessitates learning a second language for users and makes models less transparent. Named ranges provide the same benefits of being able to reference tables and data as objects and being able to build symbolic formulas that are easy to understand.

Default rows/columns table architecture

* One table object per sheet
* Table’s header row with variable names is homed to cell A1 and data rows begin in cell A2
* Contiguous column and row regions with no blanks or gaps in the header row cells or between data rows
* Table wayfinding by named ranges – optionally these include the Sheet name as a prefix for multi-table models
  + Individual column named ranges based on header row variable names
  + Header row named range for entire Row 1
  + Table region named range for entire columns having populated variable names
* For robustness, avoid placing extraneous data below or to the right of the table rows and columns

A Default Rows/Columns Table

A screenshot of a computer

Description automatically generated

The example table shows preferred formatting and curation:

* Use Excel’s built-in Calculation style for column formulas (e.g. formulas that are the same in every data row)
* Use Excel’s built-in Neutral style to flag cells containing ad hoc formulas such as Cell B4 above
* Use cell comments to share a description and units for variables such as **area** column D example above.

A screenshot of a computer

Description automatically generated

With a few exceptions, a table range can be selected by clicking in any of its cells and first typing Control-A (Windows) or either Control-A or ⌘-A on Mac to select the “CurrentRegion.” [[1]](#footnote-1) This can be followed by typing Control-Space to extend the selection to entire columns.

One notable exception to using CurrentRegion (Control-A) is that the table range should be restricted to columns with a populated variable name in its header cell. In the example below, Control A selects $A$1:$D$4 which would result in named table range $A:$D. The table range should be restricted to $A:$C for this table. The ExcelSteps add-in checks for this and defines it accordingly. This example is easy to detect, but CurrentRegion can also be confused by a seemingly blank cell containing a single space or other non-visible string.

|  |  |
| --- | --- |
| CurrentRegion Erroneously Includes Column D | Table3 Range is $A:$C |
|  |  |

Named column and table ranges are useful for building readable lookup formulas. Making these be the entire row and column ranges allows tables to scale as data are added. Note that, with addition of the XLOOKUP function to replace VLOOKUP, the column named ranges are the most critical, and the header row and overall table named ranges become optional. They are still recommended here because many people still use the older VLOOKUP function.

## Scenario Model Object

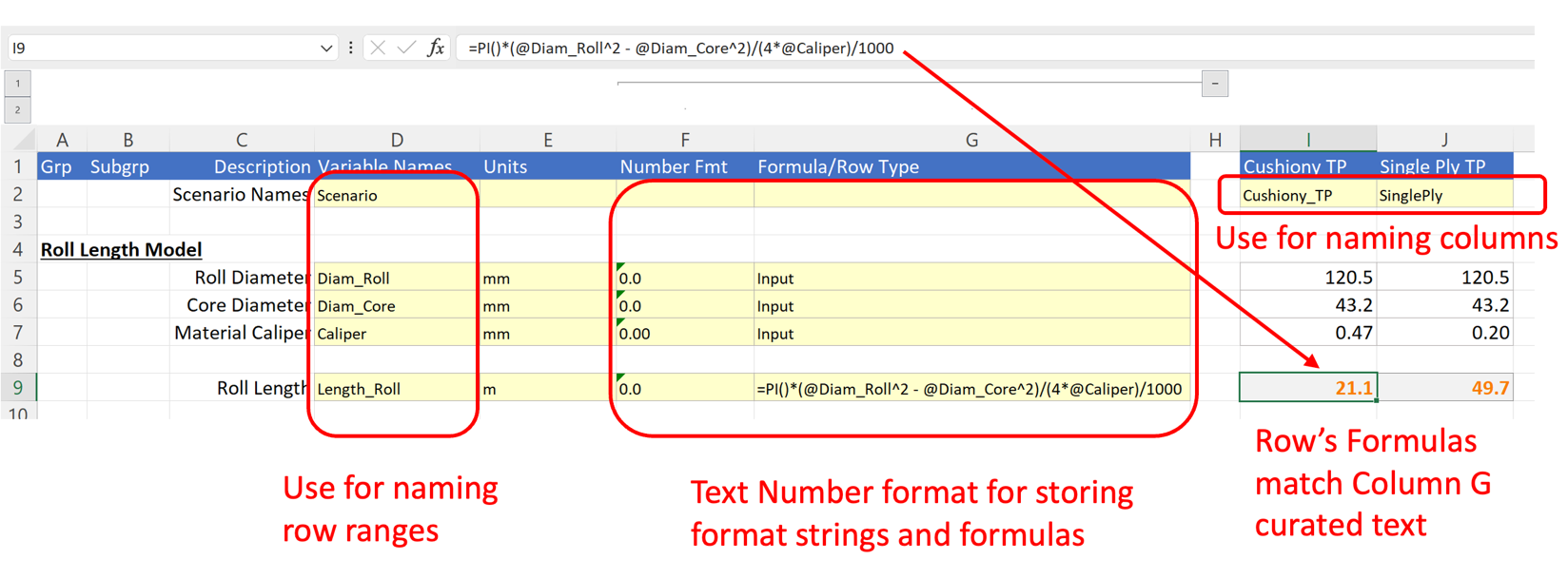
In best-approach spreadsheet models, the second building block is a columns by rows “scenario model” as illustrated below. In multicolumn format, it is simply an alternate aesthetic to a rows/columns table. This is sometimes preferable such as when the outputs in one column feed the inputs in the next such as the example or when its simply nicer to display scenarios side by side. In a single-column “calculator” mode, a scenario model is also a preferred way to present users with input cells to display single-valued, calculated outputs from a model.

The so-called default scenario model has a header region in columns A to G and one or more data columns beginning in Column I (aka scenarios in Columns I, J and beyond. The model’s rows are the variables

Default Scenario Model Architecture

* One scenario model object per sheet
* Model’s header region is Columns A through G, and these header columns are arranged as shown below. The Column D unique variable names are the basis for naming the rows
* The model has unique scenario names in Row 2. Row 2 names are the basis for naming those columns
* Unlike the rows/columns table, scenario models can contain blank rows and columns for spacing –inclusion in the model is determined by presence of a variable name in Column X and a scenario name in row 2
* Columns A, B and C are free text for describing groups and subgroups of variables (Columns A and B) and the individual variable descriptions (Column C)
* Column E is free text for describing variables’ units
* Table wayfinding and formula variables are by named ranges. Like tables, these can optionally include the Sheet name as a prefix to avoid confusion in multi-sheet models
  + Variable-containing rows are named based on Column C cell values
  + Scenario-containing columns are named based on Row 2 values

A Default Scenario Model



There are two, additional, useful feature of scenario models. The first is that they can be easily transposed to rows/columns for graphing or further summarization as shown below. This is done by Paste Special + Values + Transpose followed by deleting extraneous rows and columns and, optionally, adding the variable descriptions and units as header row cell notes as shown. The ExcelSteps addin contains a menu command to do this for default scenario models.

Transposed Scenario Model

A screenshot of a spreadsheet

Description automatically generated

A second useful feature is that the range naming makes it possible to return an individual cell’s value for summarization or other purposes. This can be created as the intersection of a scenario column and variable row using an intersection formula as shown below.

A screenshot of a spreadsheet

Description automatically generated

1. This selection is referred to as the CurrentRegion in VBA. For the example table, its address can be shown by printing Cells(1,1).CurrentRegion.Address in the Visual Basic Editor [↑](#footnote-ref-1)