**Model Builder, Selection and Portability**

Our purpose here is to develop more practice with model builder, and to automate portions of your workflow to ease multiple runs during your runoff mitigation development.

There were three main "branches" in the example flowchart provided earlier in the semester; one that calculated net rainfall (red, below), one that calculated infiltration (green), and a third that calculated watersheds (blue below).

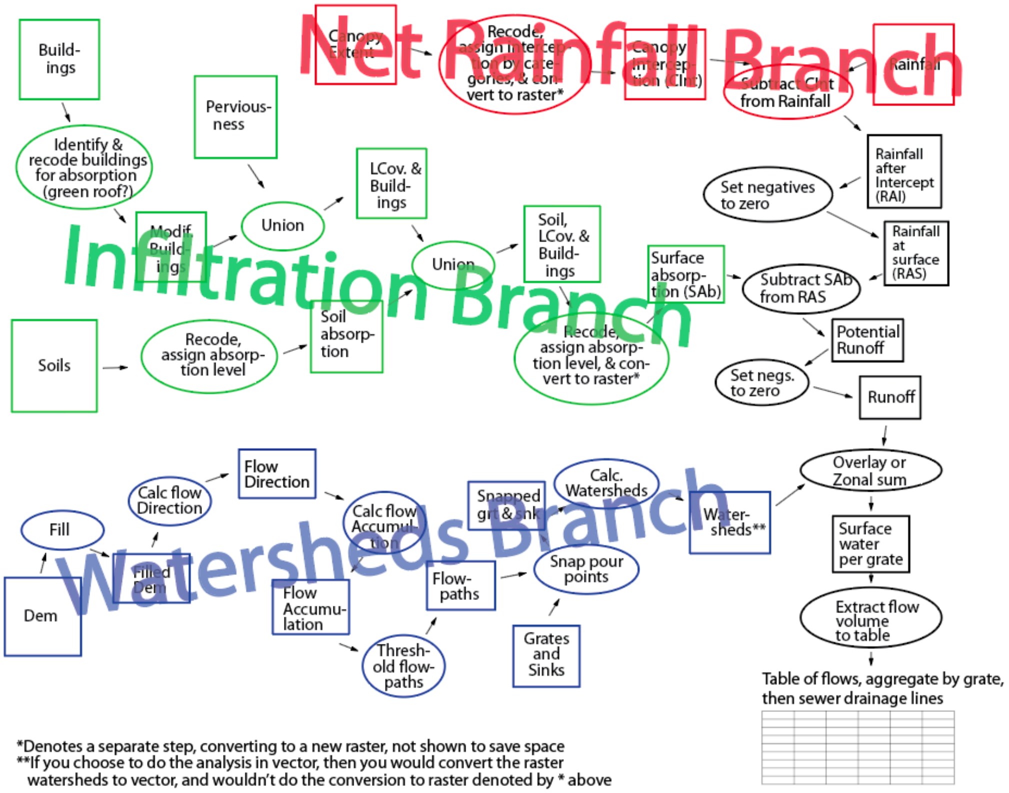
While you may have opted for a different workflow that that embodied in this flowchart, your approach will likely include something similar to the three branches. For example, a vector-only approach could duplicate these three branches, but substitute a set of overlays, calculations, and recodings for the raster conversions and raster algebra on the right side of this flowchart.

Your various model runs involve modifying inputs, e.g., canopy, building roof type or surface permabilities, or watershed pour points, and then re-running the analysis.

This sort of repeat analysis could be greatly eased by models which allow you to change those inputs, but run the same model steps in a more or less automated fashion.

Your Previous Model Builder assignment incorporated the watershed branch into a model.

Your assignment this week is to write models for some version of the net rainfall branch and the infiltration branch.



*Denotes a separate step, converting to a new raster, not shown to save space
**If you choose to do the analysis in vector, then you would convert the raster watersheds to vector, and wouldn't do the conversion to raster denoted by * above

## Infiltration Branch Example

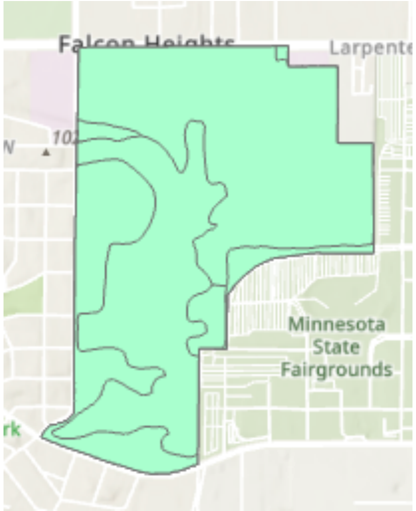
The objective of this model is to create a layer with a column for maximum infiltrations rates, combining buildings, impervious surface, and soils layers.

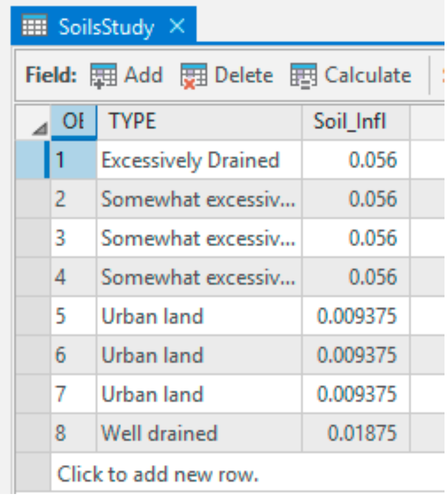
In our problem, it is likely we will modify our pervious surface layer, changing some of the impervious surfaces to pervious surfaces, and modify the original buildings layer, converting some of the flat roofs to green roofs.

Therefore, we want at least the buildings and perviousness layers as a parameter that can be changed with each run.

A few starting assumptions:
First, I won't be modifying the initial soils layer between model runs. I'll make it a parameter, because that provides more flexibility for your main input data, but I don't need to, so could leave it as a fixed data set since I won't be changing any soils polygons or table data between runs.

Also note that the soils data has already been clipped (below left), and a proper table item for the maximum infiltration rates for underlying soils created and with the proper values, named Soil_Infl, as shown in the table at right. There are other columns in the table, hidden in this view.

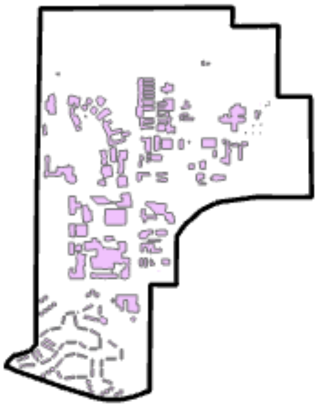


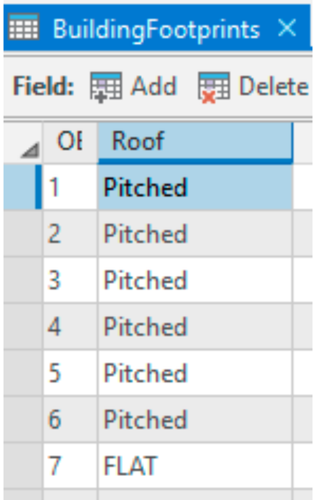


| OI | TYPE | Soil_Infl |
|---|---|---|
| 1 | Excessively Drained | 0.056 |
| 2 | Somewhat excessiv... | 0.056 |
| 3 | Somewhat excessiv... | 0.056 |
| 4 | Somewhat excessiv... | 0.056 |
| 5 | Urban land | 0.009375 |
| 6 | Urban land | 0.009375 |
| 7 | Urban land | 0.009375 |
| 8 | Well drained | 0.01875 |

The Building Footprints layers has a column listing roof type.

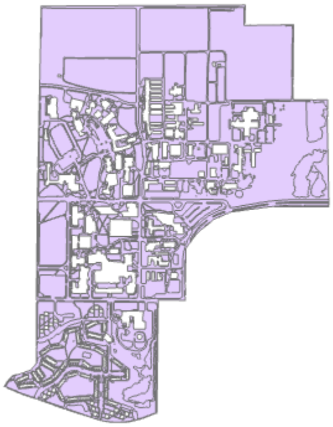
I assume the modified building footprints layer has assigned a Roof value of "Green" for the flat roofs converted to green roofs.

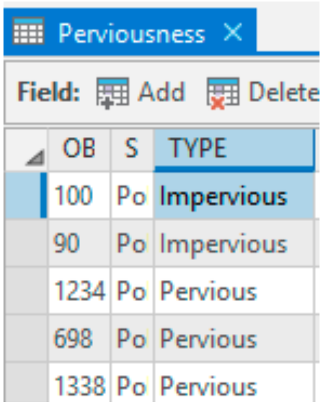




| OI | Roof |
|---|---|
| 1 | Pitched |
| 2 | Pitched |
| 3 | Pitched |
| 4 | Pitched |
| 5 | Pitched |
| 6 | Pitched |
| 7 | FLAT |

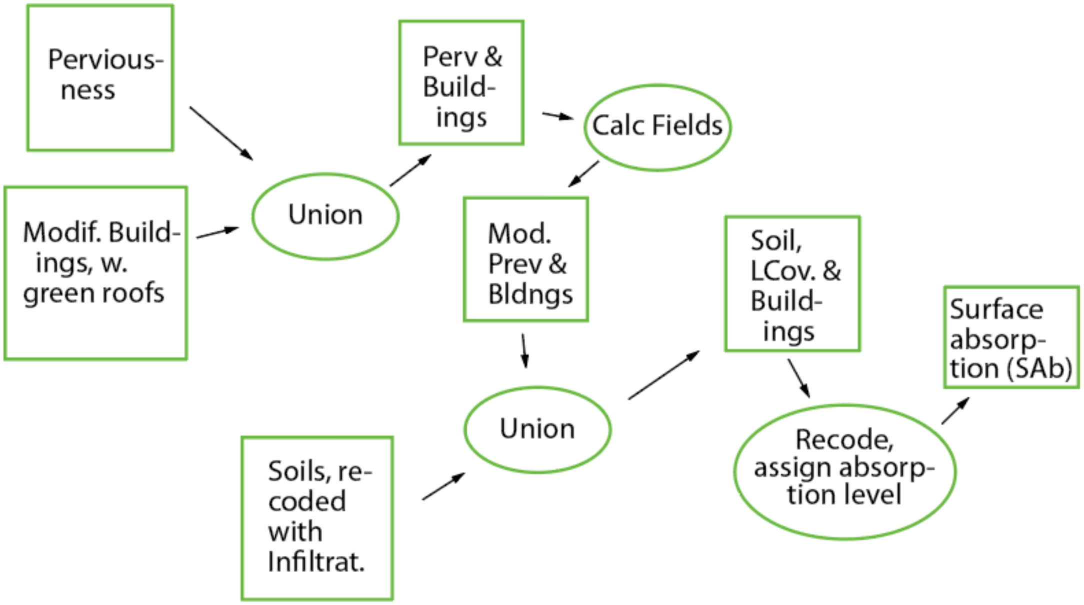
The layer Perviousness contains a field that identifies impervious surfaces:





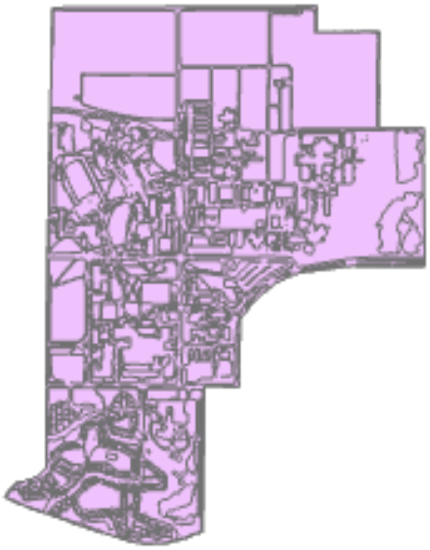
| OB | S | TYPE |
|---|---|---|
| 100 | Po | Impervious |
| 90 | Po | Impervious |
| 1234 | Po | Pervious |
| 698 | Po | Pervious |
| 1338 | Po | Pervious |

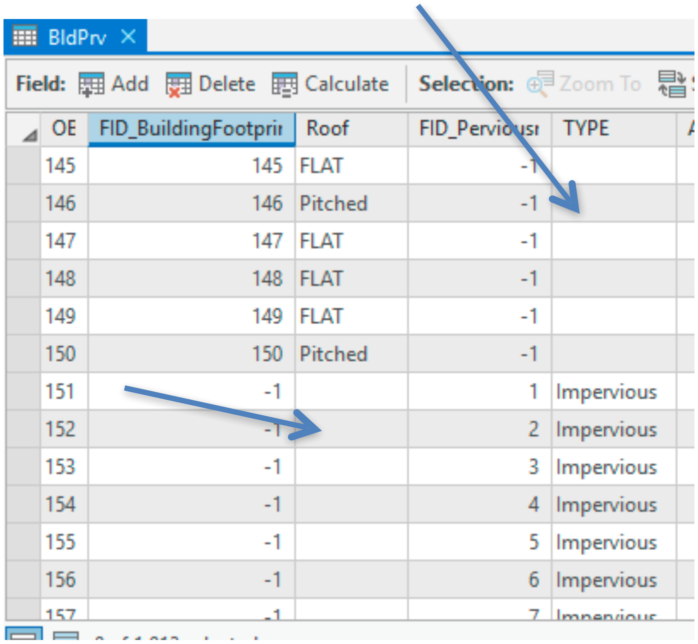
Our original flowchart showed the branch as something like this:



The original flowchart showed us unioning the Pervious and Building Footprints layer, here called BldPrv. However, as you've discovered in your analysis, this union results in a null or unassigned values for the TYPE column, the variable that records pervious status, for building polygons in the union output. It also resulted in null or unassigned output for the Roof value in the polygons from the perviousness layer (see arrows, below right). These

We need to fix this, as we want to record the status of the surface into one column.





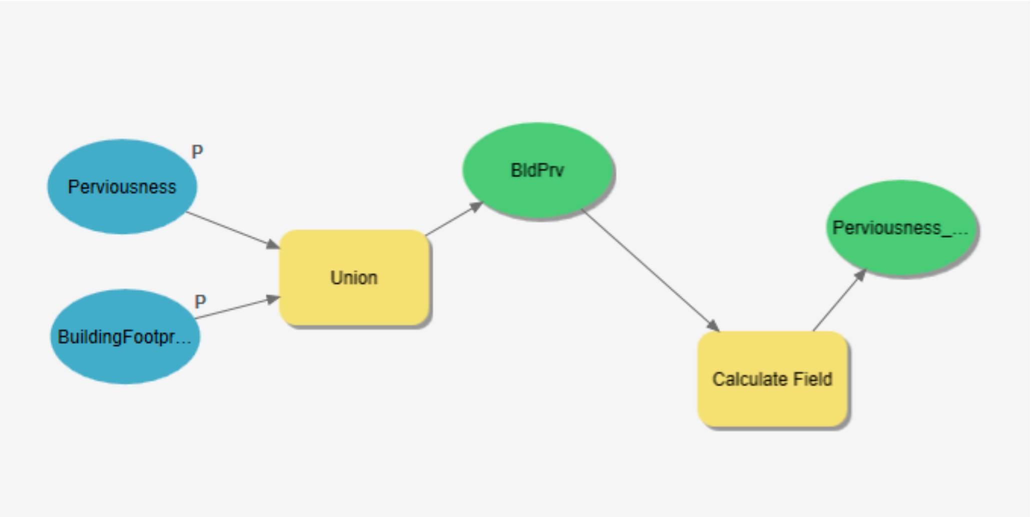
| OE | FID_BuildingFootpri | Roof | FID_Perviousı | TYPE |
|---|---|---|---|---|
| 145 | 145 | FLAT | -1 | |
| 146 | 146 | Pitched | -1 | |
| 147 | 147 | FLAT | -1 | |
| 148 | 148 | FLAT | -1 | |
| 149 | 149 | FLAT | -1 | |
| 150 | 150 | Pitched | -1 | |
| 151 | -1 | | 1 | Impervious |
| 152 | -1 | | 2 | Impervious |
| 153 | -1 | | 3 | Impervious |
| 154 | -1 | | 4 | Impervious |
| 155 | -1 | | 5 | Impervious |
| 156 | -1 | | 6 | Impervious |
| 157 | -1 | | 7 | Impervious |

Here, I'll choose to record the roof's impervious status into the TYPE column, to use in further processing. I need to assign "Impervious" to the Type column for the polygons with pitched or flat roofs, and "Pervious" to the TYPE column for buildings with Green roofs.

When processing manually, you would most likely do this with a Select by Attributes tool, and then a Calculate Field tool. This sequence doesn't work in Model Builder, because the calculate field tool, when done manually, only operates on a selected set, if there is one. Model Builder doesn't do this.

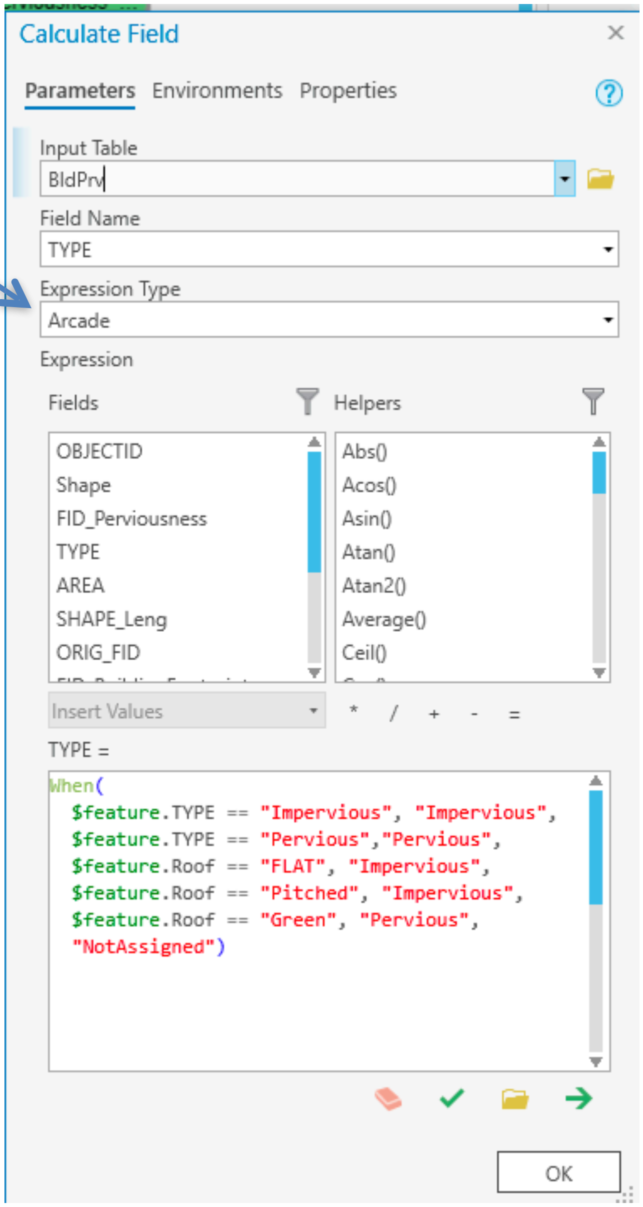
Fortunately, we can use syntax within the Calculate Field tool to select while calculating.

I build my model by dragging and connecting the appropriate inputs and command:



I then open the Calculate Field bubble on my model canvas, and indicate the

Input Table, Field I'll be modifying, entering an expression in the window at the bottom.



Input Table: BldPrv

Field Name: TYPE

Expression Type: Arcade

TYPE =

```
When(
  $feature.TYPE == "Impervious", "Impervious",
  $feature.TYPE == "Pervious","Pervious",
  $feature.Roof == "FLAT", "Impervious",
  $feature.Roof == "Pitched", "Impervious",
  $feature.Roof == "Green", "Pervious",
  "NotAssigned")
```

Take care to note that this is an Arcade Expression Type (see arrow at right)

The window at the bottom shows the expression assigned to TYPE.

This is a selection/assignment, with the ***When (*** part signifying the conditional assignment.

The ***$feature.*** indicates that we want to cycle through all the features in the input table. Then comes the condition; the first line

$feature.TYPE == "Impervious", "Impervious",

is interpreted as: wherever you find a TYPE value of Impervious, assign it a value of Impervious.

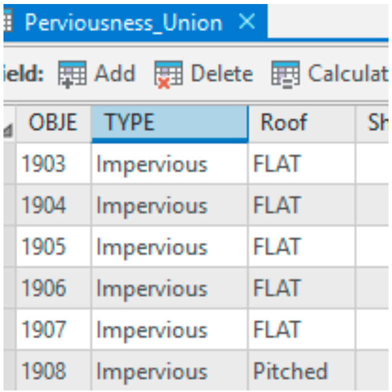
The clever parts start at the third line, with

$feature.Roof == "FLAT", "Impervious",

This allows us to assign a TYPE value of Impervious whenever the Roof is FLAT.

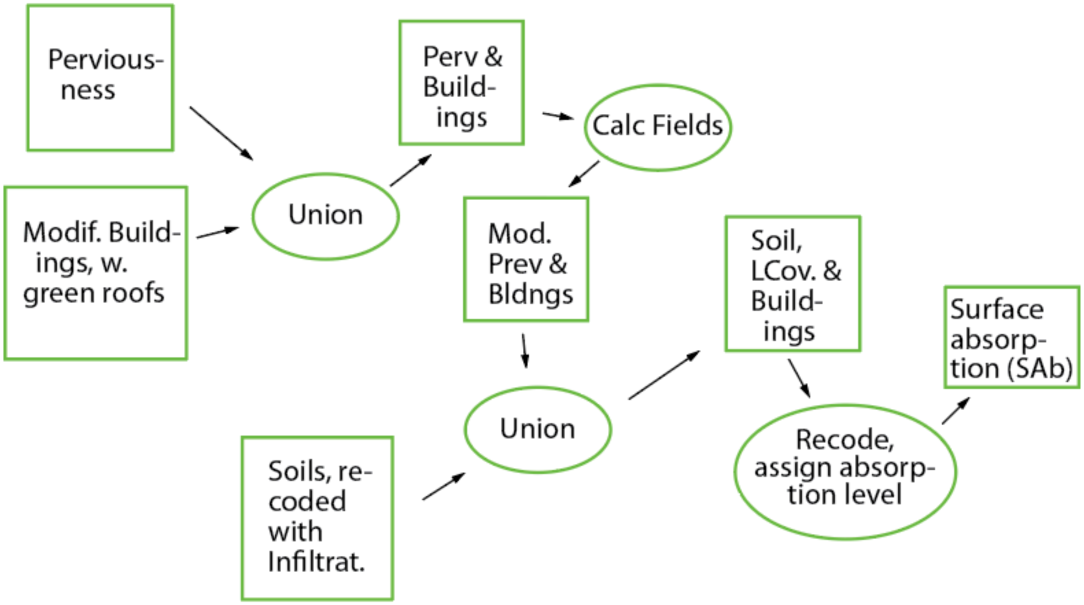
The rest of the expression is similar, assigning values based on a set of conditions. Note that there is a final "NotAssigned" flag, if none of our conditions apply to a feature. Also note the commas following each condition, and the general syntax. Model Builder is pretty unforgiving for errors in format or content.

After running the Calculate Field command, I check and note that the TYPE column is correctly assigned for my polygons that came from the Building Footprints layer, but are now part of my unioned perviousness layer:

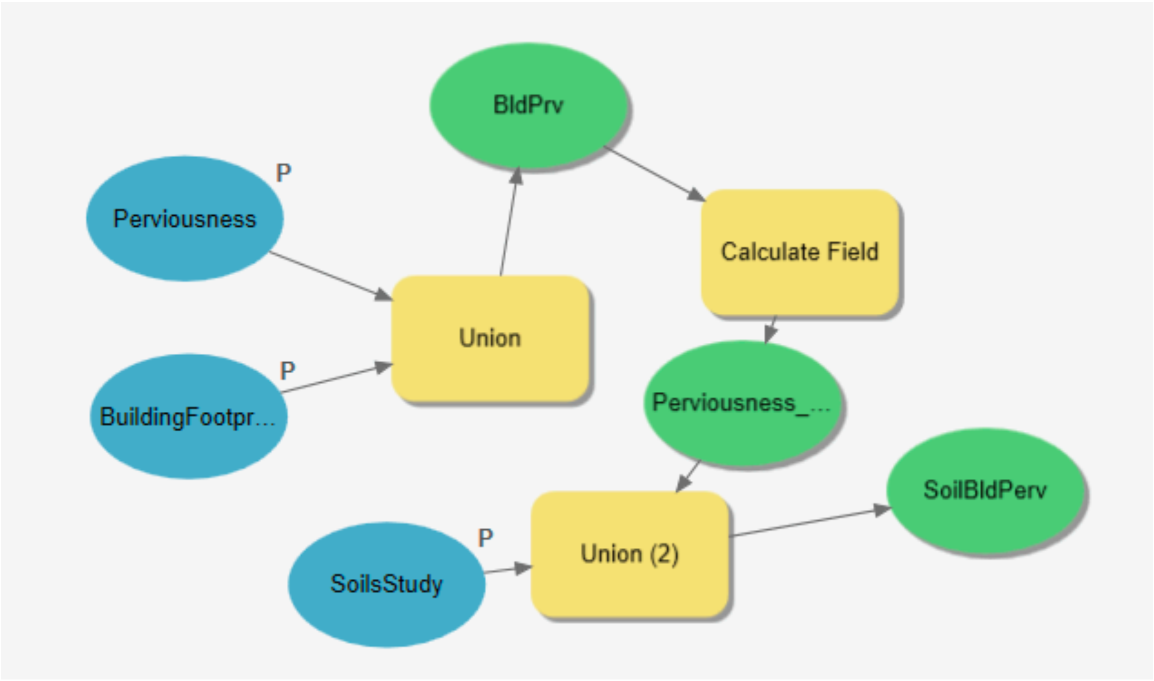


| OBJE | TYPE | Roof | Sh |
|---|---|---|---|
| 1903 | Impervious | FLAT | |
| 1904 | Impervious | FLAT | |
| 1905 | Impervious | FLAT | |
| 1906 | Impervious | FLAT | |
| 1907 | Impervious | FLAT | |
| 1908 | Impervious | Pitched | |

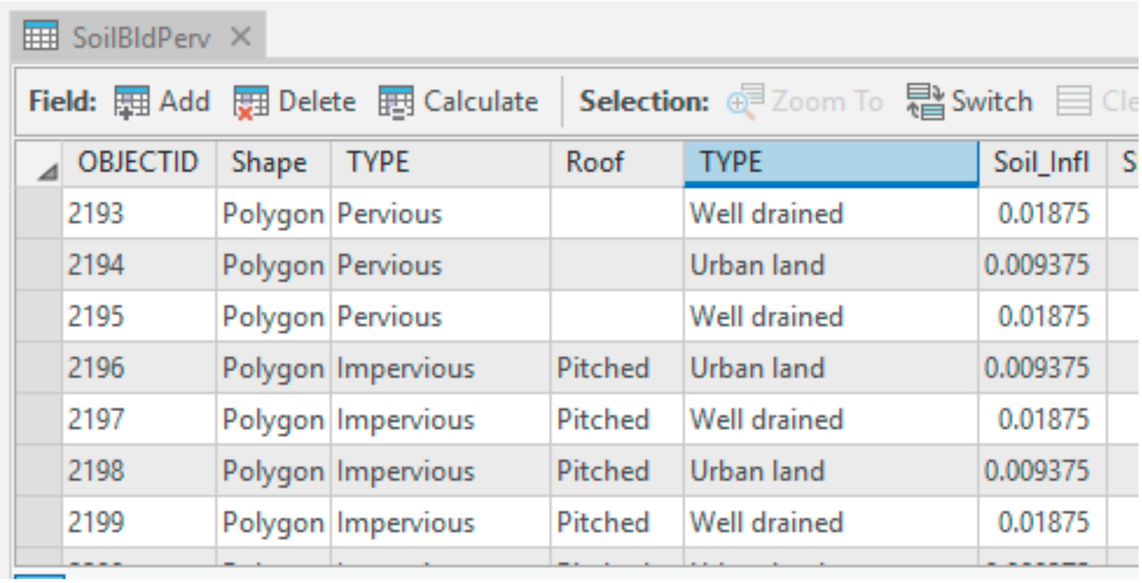
Our flowchart shows the next step as a union of soils with our buildings and previous layer:



When implemented in Model Builder, it looks something like this:



After running the above model, the table for the output combined soils/buildings/perviousness layer looks something like this, below:



| OBJECTID | Shape | TYPE | Roof | TYPE | Soil_Infl | S |
|---|---|---|---|---|---|---|
| 2193 | Polygon | Pervious | | Well drained | 0.01875 | |
| 2194 | Polygon | Pervious | | Urban land | 0.009375 | |
| 2195 | Polygon | Pervious | | Well drained | 0.01875 | |
| 2196 | Polygon | Impervious | Pitched | Urban land | 0.009375 | |
| 2197 | Polygon | Impervious | Pitched | Well drained | 0.01875 | |
| 2198 | Polygon | Impervious | Pitched | Urban land | 0.009375 | |
| 2199 | Polygon | Impervious | Pitched | Well drained | 0.01875 | |

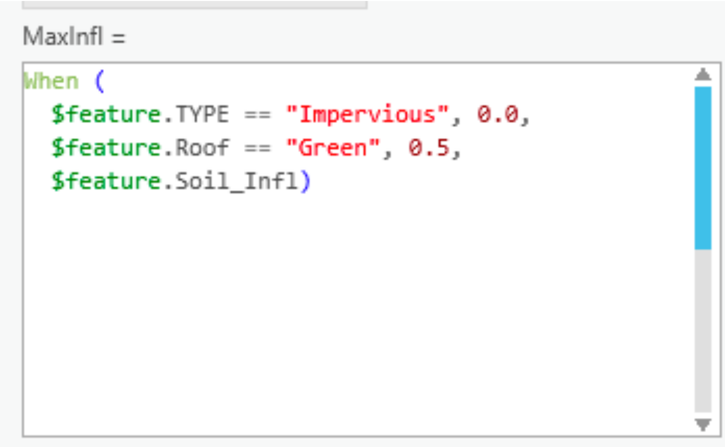
It is close to what we want, in that we have our various types combined. However, the infiltration of impervious surfaces should be set to zero, as they are impervious.

In addition, the infiltration for green roofs should be set to their maximum value.

It is perhaps safest to create a new column, and calculate the maximum infiltration into that column. I can drag the Add Field operation onto the Model Builder Canvas, and specify the input, a new field and type (I named it **MaxInfl**, type double, precision 12 and scale 5).

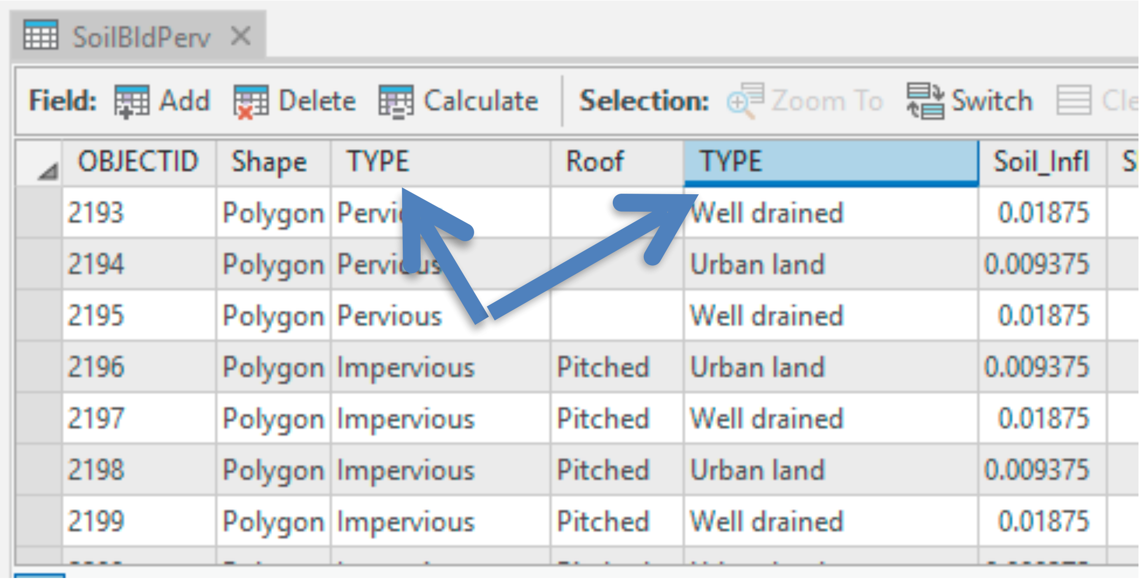
I can then use Calculate Field and a When conditional assignment to assign the proper values to my MaxInfl variable. I might create an Arcade assignment equation something like this:

Remember in this syntax, it checks each row, and if it is impervious, it assigns and infiltration of zero. My previous processing ensures this converts both the impervious roofs and land surface to zero infiltration. But since I haven't set an infiltration rate for green roofs, I need to do that now (Note that the figure should read 0.05 rather than 0.5 for Green roofs). The rest of the polygons are previous, and get the underlying soil infiltration rates.



```
MaxInfl =
When (
  $feature.TYPE == "Impervious", 0.0,
  $feature.Roof == "Green", 0.5,
  $feature.Soil_Infl)
```

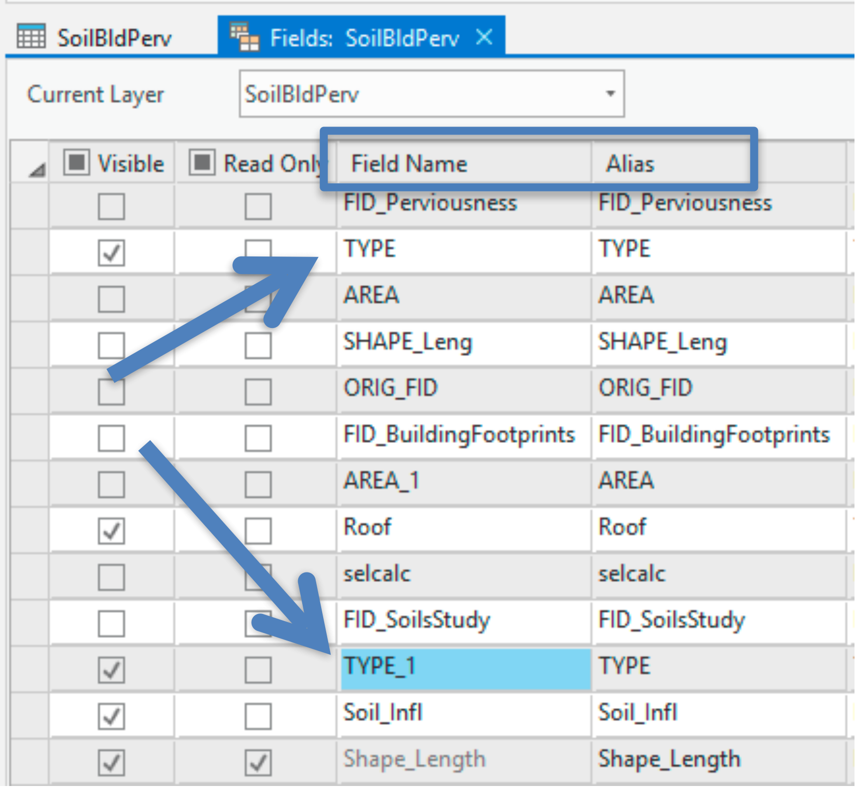
There is one wrinkle that we might find disconcerting. If I look at the input table, there are two columns named type:



| OBJECTID | Shape | TYPE | Roof | TYPE | Soil_Infl | S |
|---|---|---|---|---|---|---|
| 2193 | Polygon | Pervi | | Well drained | 0.01875 | |
| 2194 | Polygon | Pervi us | | Urban land | 0.009375 | |
| 2195 | Polygon | Pervious | | Well drained | 0.01875 | |
| 2196 | Polygon | Impervious | Pitched | Urban land | 0.009375 | |
| 2197 | Polygon | Impervious | Pitched | Well drained | 0.01875 | |
| 2198 | Polygon | Impervious | Pitched | Urban land | 0.009375 | |
| 2199 | Polygon | Impervious | Pitched | Well drained | 0.01875 | |

How can I be sure that my selection will work on the correct column. In this case I want to use the left TYPE column, which contains imperviousness, and not the right, with comes from soil types.

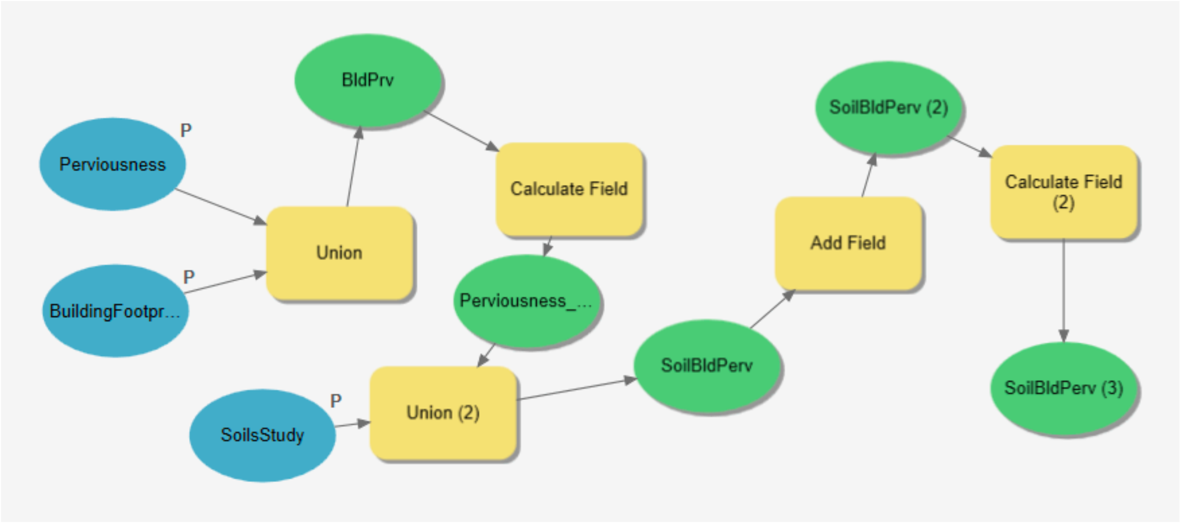
Not to worry, I am only looking at an alias. If I open the Fields view on the table (right click on any column and select Fields, near the bottom of the drop down), I see that the columns have distinct names, TYPE and TYPE_1. The Union command creates these distinct Field Names, and assigns aliases. The Arcade expression references the Field Names, and not the aliases.



Current Layer: SoilBldPerv

| Visible | Read Only | Field Name | Alias |
|---|---|---|---|
| ☐ | ☐ | FID_Perviousness | FID_Perviousness |
| ☑ | ☐ | TYPE | TYPE |
| ☐ | ☐ | AREA | AREA |
| ☐ | ☐ | SHAPE_Leng | SHAPE_Leng |
| ☐ | ☐ | ORIG_FID | ORIG_FID |
| ☐ | ☐ | FID_BuildingFootprints | FID_BuildingFootprints |
| ☐ | ☐ | AREA_1 | AREA |
| ☑ | ☐ | Roof | Roof |
| ☐ | ☐ | selcalc | selcalc |
| ☐ | ☐ | FID_SoilsStudy | FID_SoilsStudy |
| ☑ | ☐ | TYPE_1 | TYPE |
| ☑ | ☐ | Soil_Infl | Soil_Infl |
| ☑ | ☑ | Shape_Length | Shape_Length |

My Model will now look something like this:



Once I validate, run, and verify the model, I can now calculate a new maximum infiltration layer after any modification of the inputs. In this case I could change the perviousness to convert some of my roads or sidewalks to pervious pavement, and change a building to have a green roof, and then re-run my model fairly simply. This would greatly speed iteration through my mitigation development.

If this were a stand-alone model, I would probably want to make the output a parameter. However, the goal is to create a larger model that incorporates my entire workflow.

Once I've completed this "branch" of my overall model, I'll have two of my three main branches codified in Model Builder. I could then make a full model by completing the third branch, and after debugging, combine all three branches with an "overlay" of sorts of the three branch layers, either through conversion/raster calculator or vector union/recode, and then to raster or vector aggregation functions.

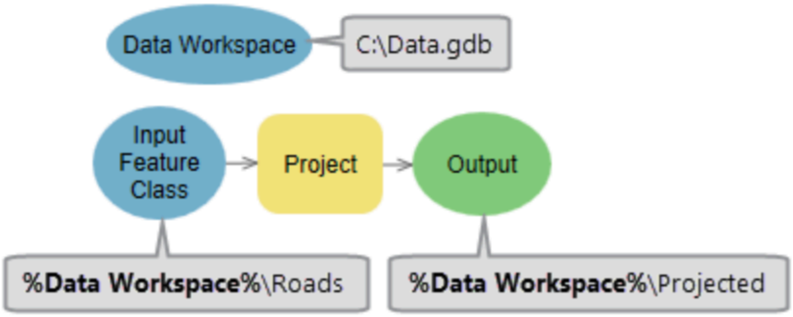
**Making a Model Flexible**

You might notice that when you save a model, the file references are set to the input values you used when debugging, e.g., if an input file was in the C:\users\classdata directory, with the name "elev_rast.tif," then that will be included in the model specification when you save it.

If you want to use the model on different data, in a different directory, you may run in to problems, because the model will look for exactly these data, in the specified directory, when you wish to run it.

One option is to specify all input and output data sets as parameters. This may work in some instances, but in others, you may not wish to have to specify all the inputs. In addition, if there are intermediate data sets, you don't want to have to specify all of them.

A common approach is to take advantage of inline substitution. Any time you enclose text in % symbols, the model will substitute a workspace variable defined for that text. For example, in the figure below, my base data are stored in a geodatabase named Data.gdb. I define a workspace data location with the name Data Workspace, and a value of C:\Data.gdb. If I place the token %Data Workspace% in my Model Builder code, it will substitute the value, C:\Data.gdb when running the model. In this way I can specify a different path and dataset for my input. Note that the data sets themselves must have the same names as specified in the model, e.g., Roads, Projected, in the example below. I could make these variables that one enters when running the model.



Inline substitution is often used to allow for working on different computers, where the paths to a common data set may be different. It is also often used to specify the scratch, or working directory, where intermediate data are stored. Inline substitution can make your models much more flexible.

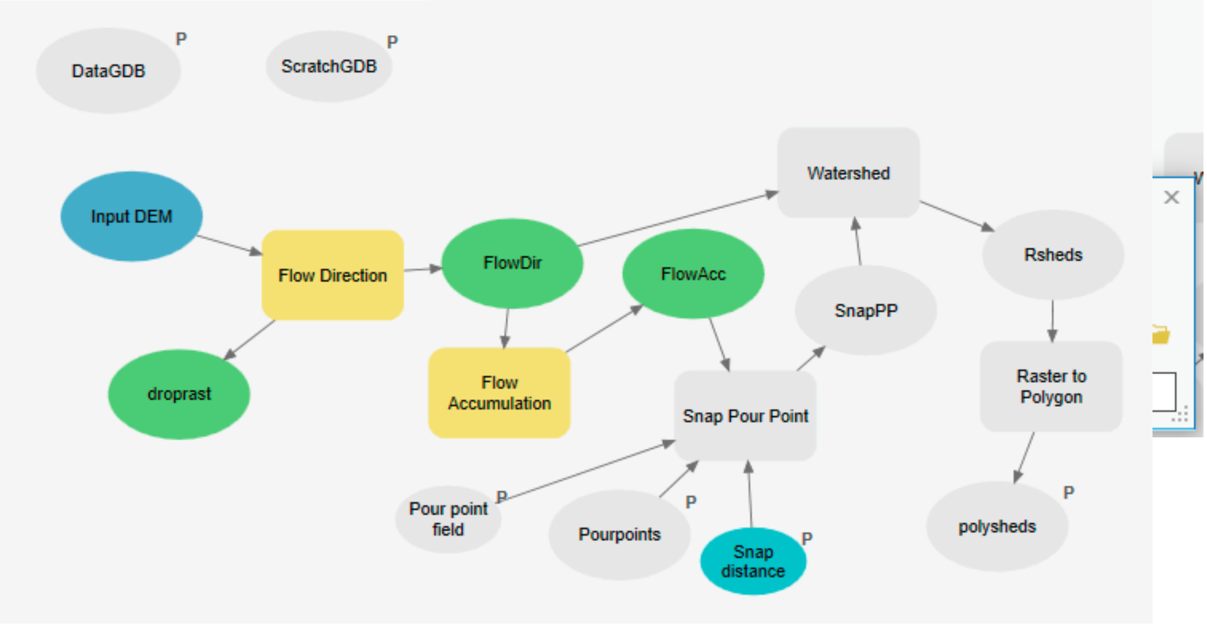
You want to optimize how you divide inputs between parameters and inline substitution. If you are always working in the same project/database, on the same computer, then you probably don't want to make these locations as inline substitutes. You're entering the same things with each run, wasting time.

You likely want to use inline substitution for the working directories/paths, but not make them parameters, if you switch between locations often, but use the same geodatabased and files. For example, if you work is split mainly in the Skok 35 lab and a home or office computer, you'll likely have different paths to your data and scratch workspaces on the two different computers, but use the same geodatabase and data layer names. You don't want to add inline substitution variables or most of the input feature classes as parameters, because you'll have to enter them with each run. Rather, you would use inline substitution, and edit the model to set the data or workspace variables for the model on each computer. As long as the data use the same naming convention, you can move updated data back and forth. Each time you switch between computers, you can just edit the model once, to modify the variable for your data workspace and your scratch workspace, and then do your runs.

If you are changing a data layer, e.g., adding pourpoints, you may wish to make that a parameter, and probably not an inline substitution. Only those values which you which to change regularly between runs should be parameters.
A couple of other notes on inline substitution:

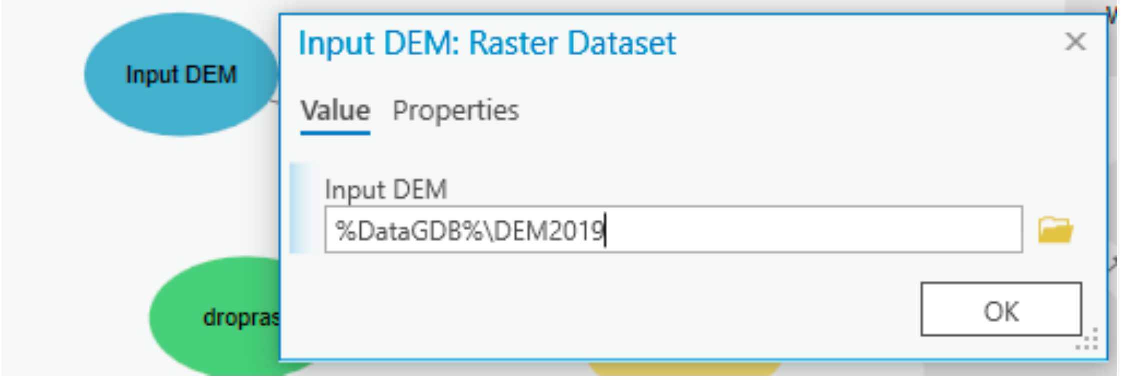
- You should add inline substitution variables after you have written/debugged your model, at least while learning. The Validate/Run sometimes trips up on the inline substitution within the Model Builder Editor, even though the model works in a model run from a toolbox. You get odd error warnings, so while you're learning, you think you've done something wrong, when sometimes you haven't.
- Be prepared for some of your data and operations to be greyed out, seemingly incomplete, when you use inline substitution values as parameters. The program doesn't know what they are until run time, so it will tell you that inputs are undefined or incorrect, for example, saying a data set doesn't exist because a data workspace (geodatabase) is a variable, defined at run time.
- Some tool values need to be defined as parameters when you use inline substitution that don't need to be parameters when you hard-wire the data source. For example, a pour point field has to be specified when snapping pourpoints. If you build your model and indicate a fixed input layer, you have the opportunity to also identify the column used for identification. If you have the pourpoint input layer defined as part of an inline substitution, Model Builder doesn't know what values might be available, so it remains undefined. You have to set it as a parameter so it can be specified at run time, once the input data layer is known.

Below is a model that contains inline substitution, and (perhaps too) many parameters for the inputs and outputs. I first created this model as shown earlier, without any inline substitution, verified it worked, and then introduced variables for an input database (DataGDB) and scratch workspace (ScratchGDB). This "broke" some of the pieces, and so I had to modify the model a bit, adding the pourpoint field as a parameters. This model assumes we'll be moving our main data between computers, and mostly modifying our pourpoint layer in this branch, as we add new rain gardens for our project. This means we're always using the same input DEM, so it doesn't need to be a parameter, and that I'll be working with the output vector watershed polygons, so I don't need to specify the output raster watersheds as a



parameter. Different workflows would likely have different mixes of parameters and inline substitution.

If I right click and open the Input DEM bubble, I see the following figure:



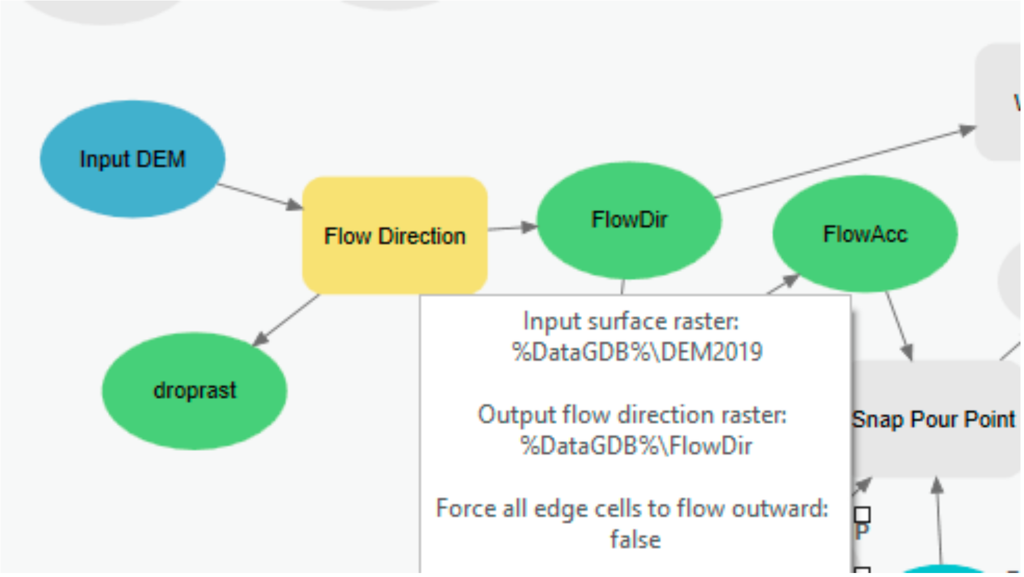
Input DEM: Raster Dataset

Input DEM

%DataGDB%\DEM2019

It shows that we're accessing our study area DEM, with the data path pre-pended. When I set this as a parameter during a model run, it will substitute the path into the %DataGDB% part.

If I hover the cursor over an operation, it will temporarily display the input variables, here for the Flow Direction operation:



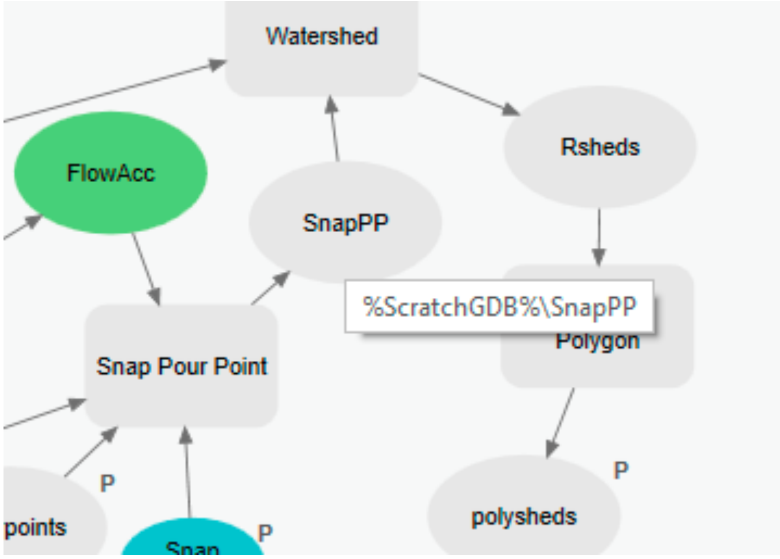
Input surface raster:
%DataGDB%\DEM2019

Output flow direction raster:
%DataGDB%\FlowDir

Force all edge cells to flow outward:
false

You can see that the inline substitution is in place for both my input and output data layers.

In this instance, I decided I don't want to save my snapped pour points to my main data geodatabase, but rather to the scratch geodatabase. I could have done this for any of the intermediate layers, it just depends on what I want to save where. I specified this by Opening and typing the inline variable, %ScratchGDB%\SnapPP when specifying the output, and it recorded that into the model. Hovering over the SnapPP bubble shows this:



%ScratchGDB%\SnapPP

Once I've specified the inline and parameter variables, I save the model, and then run it from the Catalog Toolbox.

**Your assignment:**
Create 2 models (of both branches in the same model), that calculate

1. the pervious/building/soil combination branch described above, appropriate to your workflow, and,
2. the creation of a net rainfall layer. This second model will be a bit more complicated, in that I want you to start with the input POINTS for your new canopy as one of the layers, as a parameter, along with your existing canopy, buildings layer, and gross rainfall layer (2.5 or 5 cm) as inputs. Your model needs to
    - buffer the points (perhaps based on canopy type),
    - then union the new canopy with the existing canopy
    - then erase out the current buildings and current canopy from your buffered new canopy (to create a new layer that allows you to calculate cost of the new canopy area only)
    - assign rainfall interception to the expanded canopy layer,
    - calculate net rainfall from the gross rainfall plus canopy interception.

Turn in pdfs of the model builder graphic, and the models themselves, in to the Canvas site.