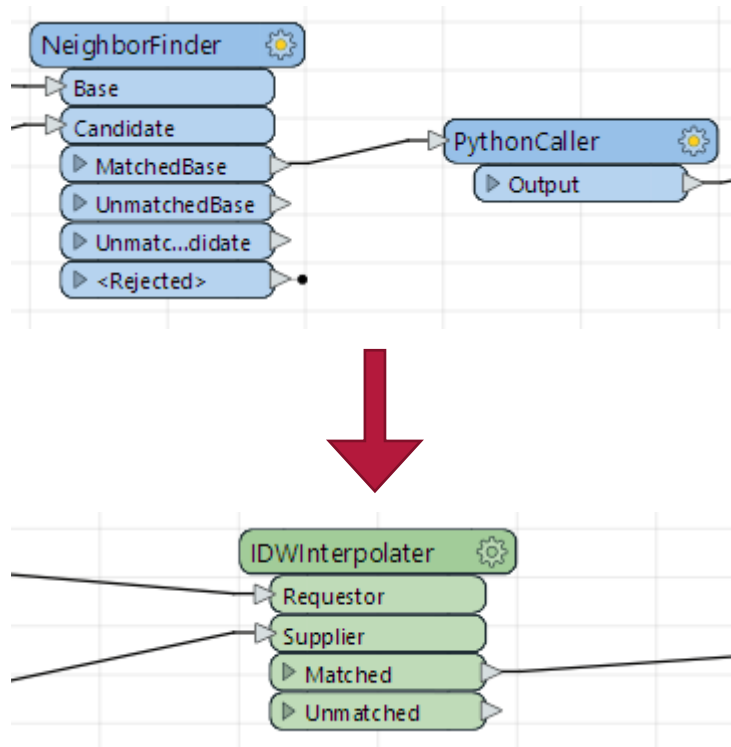


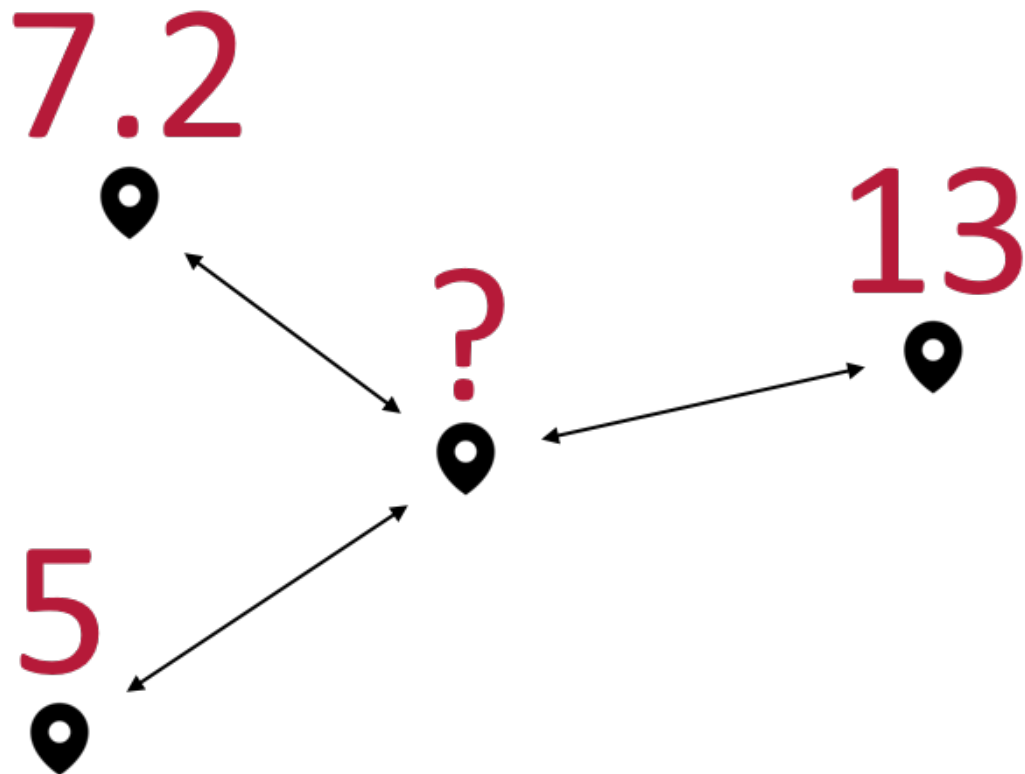
# Quick and easy custom transformers

# Steps Overview

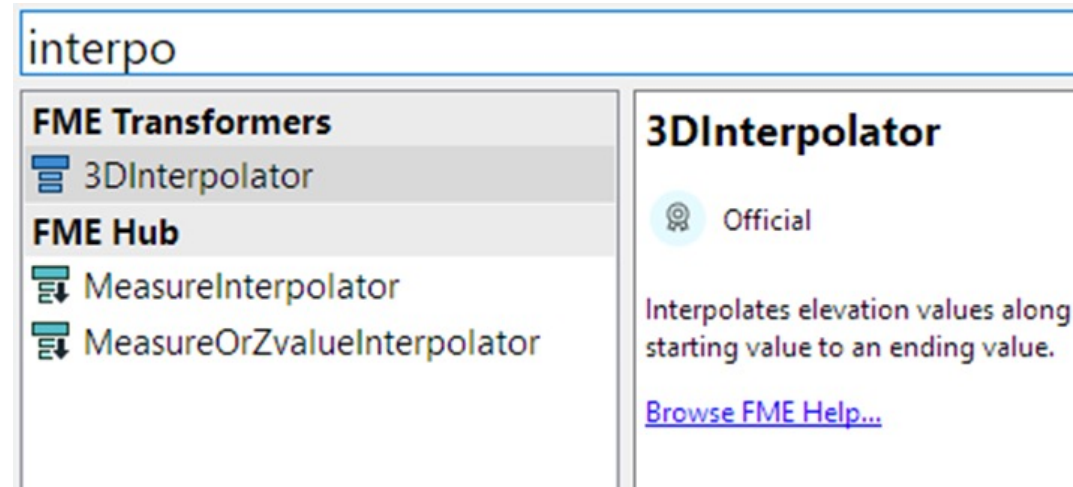


- Idea, check it needs to be made.
- Research, test to understand inputs.
- Generate script for IDW process in python.
- Plug it together, test it works.
- Package into custom transformer.
- Name ports, set up input parameters, attribute handling to be dynamic. Unmatched Inputs. Filter Suppliers. Version compatibility?
- Retest, tweak input parameters, retest.
- Name, title, description, export.

# Interpolation



- Want to estimate the value at a point based on its relative distance to points with known values
- Inverse Distance Weighting



- Doesn't appear to exist yet?

# Test Scenario

- Set up four points with values, and a fifth to be determined.
- Format to be passed to python?
- Target point has a list containing the values, and the distances to each value.
- Values array
- Distances array

The screenshot displays a software interface with a data flow diagram at the top and a visual preview window at the bottom. The data flow diagram shows a 'Creator' block connected to four 'VertexCreator' blocks (VertexCreator\_1 to VertexCreator\_4), which then feed into an 'AttributeCreator\_3' block. This block is connected to a 'NeighborFinder' block. The 'NeighborFinder' block has several output ports: 'Base', 'Candidate', 'MatchedBase', 'UnmatchedBase', and '<Rejected>'. The 'MatchedBase' output is connected to a 'NeighborFinder' block. The 'Visual Preview' window shows a 2D coordinate system with a red dot at the center (X: 29.8549, Y: 12.7217) and four black dots representing the vertices. The 'Feature Information' window shows the properties of the selected feature, 'NeighborFinder\_MATCHED'. The properties include a 'Value' of 7.8, a list of 'X()' values, and various distance and angle measurements.

Property	Value
_closest_base_y	7.8
X() (4)	
X(0)	
VWWW	15
distance	5.189412298131649
angle	154.9164059938091
candidate_angle	0
candidate_label_angle	0
closest_candidate_x	0
closest_candidate_y	10
closest_base_x	4.7
closest_base_y	7.8
X(1)	
VWWW	12
distance	5.738466694161429
angle	22.543061000057882

Transformer Name: Python

Group Processing

Python Script

Class to Process Features: Features

FME Feature Attributes  
No Attributes Available

User Parameters  
< Create User Parameter

FME Parameters

Expand categories  
Drag or double-click  
to add the correct

Options AI Assist

Output Attributes  
Attributes to Expose:

Attributes To Hide: No

Lists to Hide: No

Help Presets

AI Assist (Tech Preview)

Request

Action: Generate Code

Python Description

In python I have an array of values, and then a corresponding array of distances which describes how far away each of those values are from my location. The arrays could be of any length. What is an inverse distance weighting formula to estimate the value at my current location?

70/126737 Tokens Generate

Status

AI Results

```

1 def inverse_distance_weighting(values, distances):
2     if not values or not distances or len(values) != len(distances):
3         return None
4     try:
5         weighted_sum = sum(value / distance for value, distance in zip(values, distances) if distance != 0)
6         sum_of_weights = sum(1 / distance for distance in distances if distance != 0)
7         return weighted_sum / sum_of_weights
8     except ZeroDivisionError:
9         return None

```

Explanation

- The function `inverse\_distance\_weighting` takes two lists as input: `values` and `distances`.
- It first checks if either list is empty, or if they are not of equal length, and returns `None` if any of these conditions are true.
- The function calculates the weighted sum of the values by dividing each value by its corresponding distance, ensuring that the distance is not zero.
- It then calculates the sum of weights by taking the inverse of each distance, again ensuring that the distance is not zero.
- The estimated value at the current location is then calculated by dividing the weighted sum of values by the sum of weights.
- If a division by zero occurs during the calculation, the function catches the `ZeroDivisionError` and returns `None`.

# Generate Code

- Introduced FME 2023
- Request -> Response, generate code
- Shop around other services and could get different (better) responses.
- Either way, still requires a little work to use the code in FME – this just gives you the key part.

# Testing

- Test value interpolates as 13.67
- Seems correct. Try with more points!

The screenshot displays a GIS workflow in ArcGIS Pro. The workflow starts with a 'Creator' tool, followed by five 'VertexCreator' tools (VertexCreator\_1 to VertexCreator\_5). The outputs of these tools are combined in 'AttributeCreator\_3', which has a note 'source values to interp'. The output of 'AttributeCreator\_3' is then processed by 'NeighborFinder', which has several input parameters. The output of 'NeighborFinder' is processed by 'PythonCaller'. The output of 'PythonCaller' is then processed by 'Aggregator' and finally 'Geometry extractor'.

Below the workflow, the 'Preview' window shows a table of data for 'PythonCaller\_Output':

WVVVV	_distance	_angle	_cand
1	13.67443030259...	5.189412298131...	154.9164059938...

The 'Graphics' window shows a map with a single point selected. The 'Feature Information' window shows the properties of the selected feature:

Property	Value
Exposed At...	1
_distance	5.189412298131649
_angle	154.9164059938091
_candidate_angle	0
_candidate_labe...	0
_closest_candid...	0
_closest_candid...	10
_closest_base_x	4.7
_closest_base_y	7.8
WVVVV	13.67443030259415
X() (4)	
Unexposed...	1
FME Attribut...	
Geometry	1

# Testing

- Grid of points, view the interpolated value in 3D.
- Something weird is happening at the corners where the nearest point is zero distance.
- Code was excepting out distances that were zero, when instead we want to pick that value.
- Risk of Generated Code!

The screenshot displays a data science workflow and its visual preview. The workflow consists of the following components:

- 2DGridCreator**: A component that outputs a grid of points (121).
- AttributeCreator\_3**: A component that takes 'source values to interp' as input and outputs a grid of points (4).
- NeighborFinder**: A component that takes the grid of points from 2DGridCreator and AttributeCreator\_3 as input and outputs a grid of points (121).
- PythonCaller**: A component that takes the grid of points from NeighborFinder as input and outputs a grid of points (121).
- 3DForcer**: A component that takes the grid of points from PythonCaller as input and outputs a grid of points (121).

The visual preview shows a table of 3DForcer\_Output and a 3D scatter plot of red points. The table has the following data:

_column	_row	_distance	
1	0	0	0
2	0	1	1
3	0	2	2
4	0	3	3
5	0	4	4
6	0	5	5
7	0	6	4
8	0	7	3
9	0	8	2

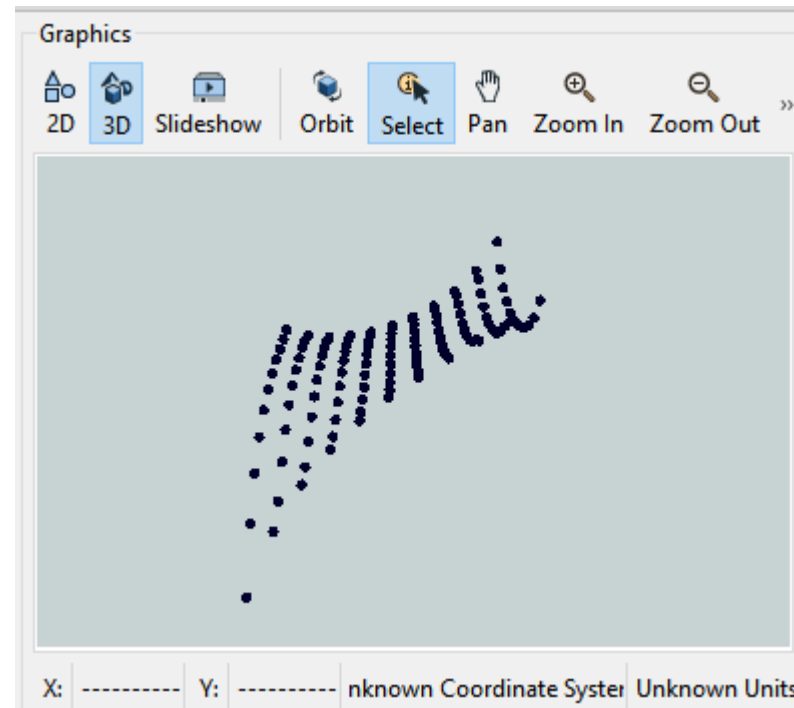
The 3D scatter plot shows a grid of red points in a 3D space, with a cluster of points in the center and a few points at the corners.

# Testing

- Grid of points, view the interpolated value in 3D.
- Something weird is happening at the corners where the nearest point is zero distance.
- Code was excepting out distances that were zero, when instead we want to pick that value.
- Risk of Generated Code!

```
def inverse_distance_weighting(values, distances):
    if distances[0] == 0:
        return values[0]
    weighted_sum = sum(value / distance for value, distance in zip(values, distances))
    sum_of_weights = sum(1 / distance for distance in distances)
    return weighted_sum / sum_of_weights
```

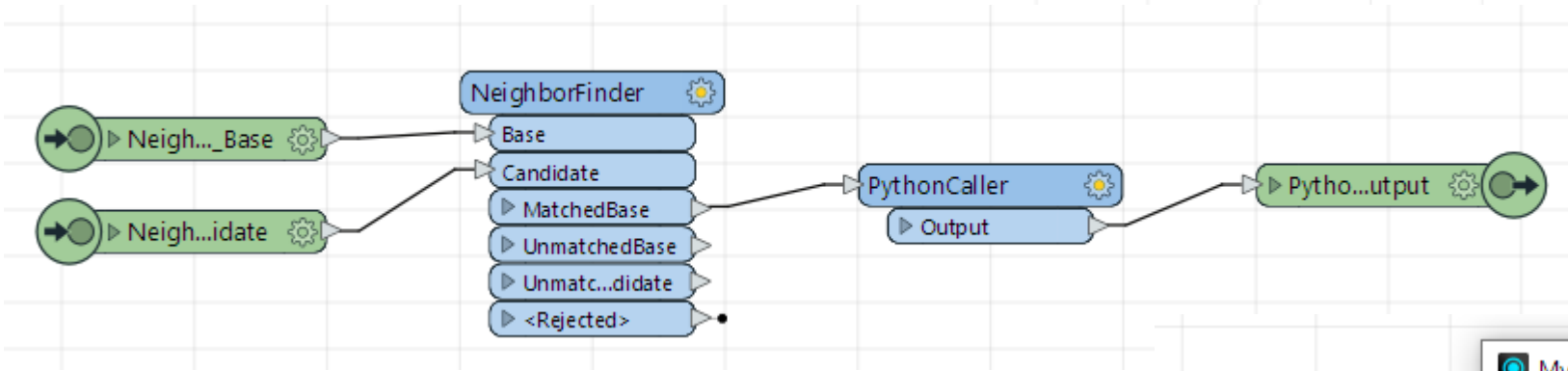
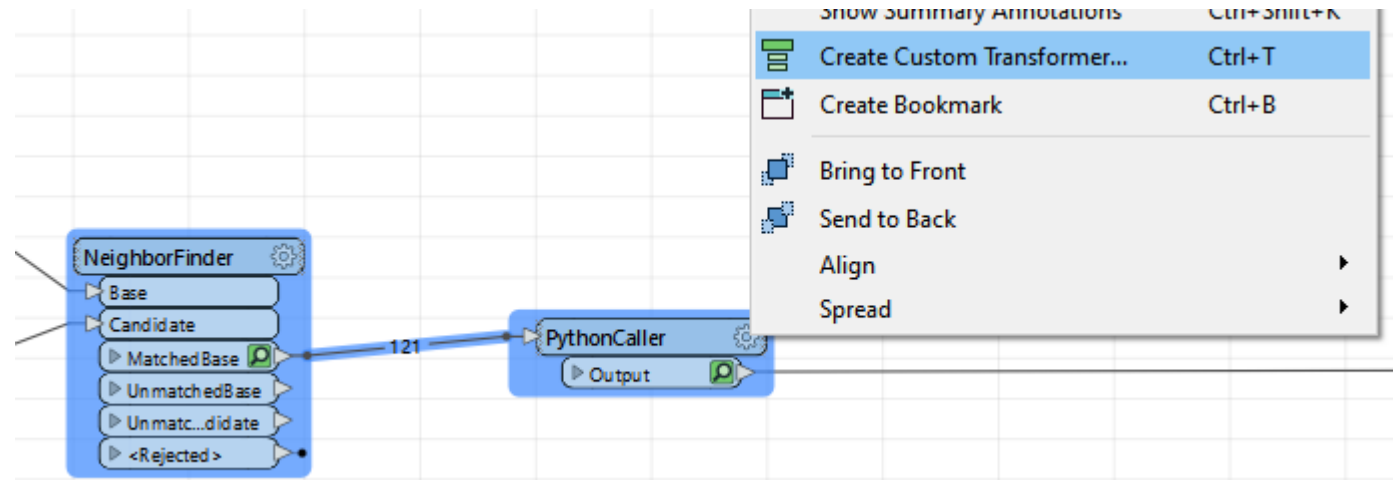
- We know that the NeighbourFinder list will always be sorted by distance ascending, can simplify the function to this to handle these zeros.



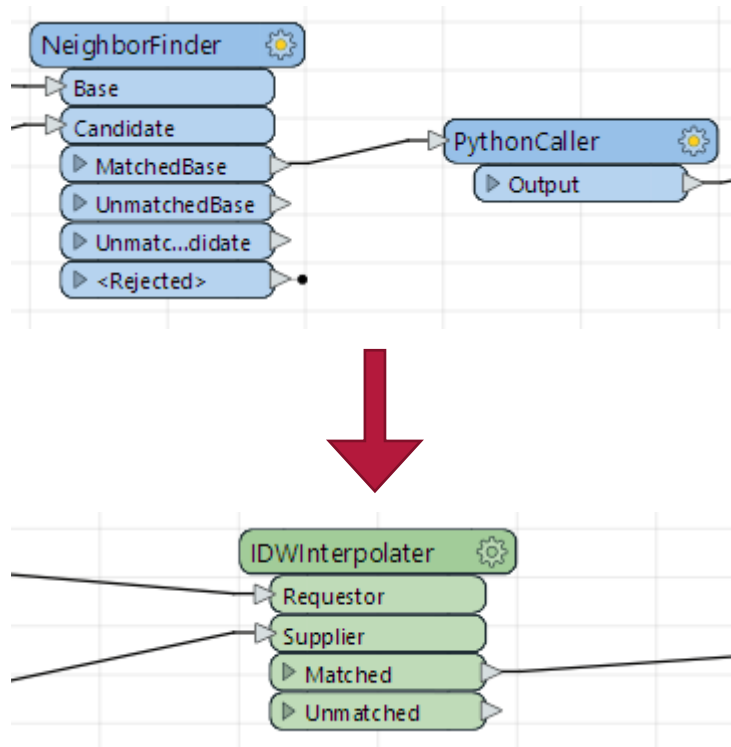


# Create Custom Transformer

- Job done!



# Steps Overview



- Idea, check it needs to be made.
- Research, test to understand inputs.
- Generate script for IDW process in python.
- Plug it together, test it works.
- Package into custom transformer.
- Name ports, set up input parameters, attribute handling to be dynamic. Unmatched Inputs. Filter Suppliers. Version compatibility?
- Retest, tweak input parameters, retest.
- Name, title, description, export.

# Tweaks/Finalisation

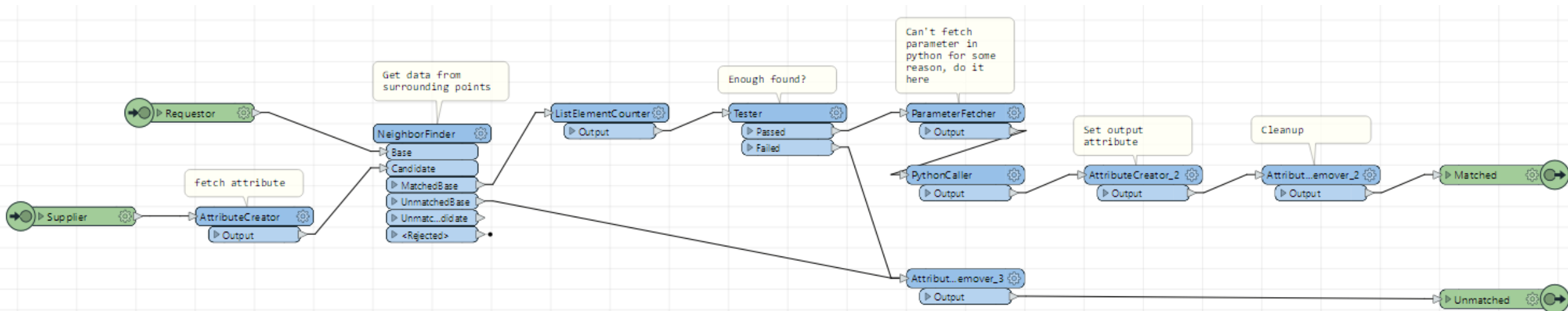
- Attribute Parameters
- Other Parameters
- Ports
- Termination
- Metadata

The top screenshot shows the configuration for the 'AttributeCreator' transformer. It is set to 'Advanced: Attribute Value Handling' and has a 'New Attributes' table with one entry:

Output Attribute	Attribute Value
<code>_interp_attribute</code>	<code>@Value\$(GETATTRIBUTE)</code>

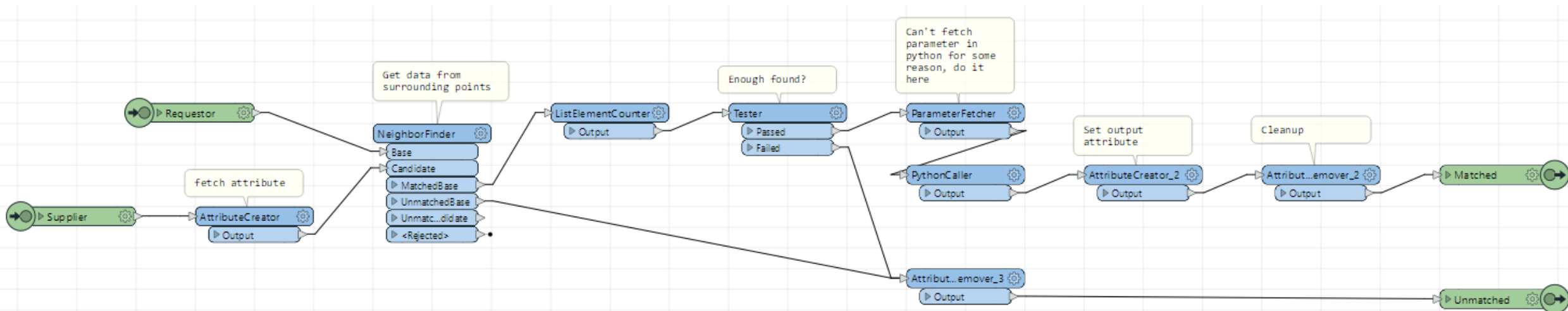
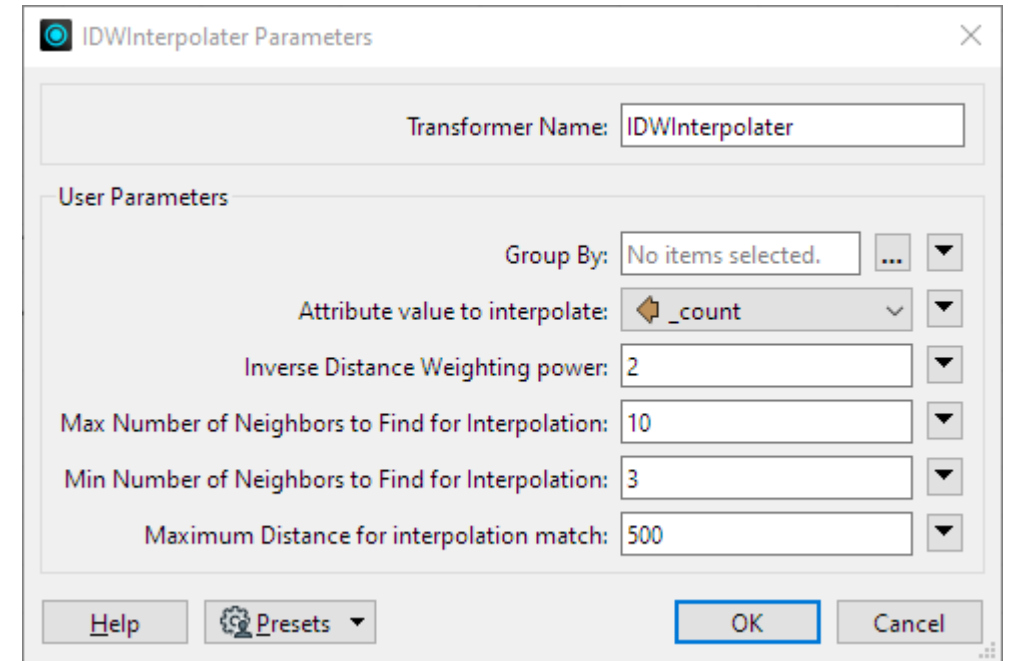
The bottom screenshot shows the configuration for the 'AttributeCreator\_2' transformer. It is also set to 'Advanced: Attribute Value Handling' and has a 'New Attributes' table with two entries:

Output Attribute	Attribute Value
<code>\$(GETATTRIBUTE)</code>	<code>__set_interp_attribute</code>



# Tweaks/Finalisation

- Attribute Parameters
- Other Parameters
- Ports
- Termination
- Metadata



# Generate Code II

- Input is one feature with list attribute from NeighbourFinder
- Copy in generated code snippet, checking handling for zero
- Added power parameter
- Get Attribute, Set Attribute.
- Output is one feature with an interpolated value

```

23 def input(self, feature: fmeobjects.FMEFeature):
24
25     #This script from Claude3
26     def idw_estimate(values, distances, power):
27         weighted_sum = 0
28         weight_sum = 0
29         for value, distance in zip(values, distances):
30             if distance == 0: # Handle the case when distance is 0
31                 return value
32             else:
33                 weight = 1 / (distance ** power)
34                 weighted_sum += value * weight
35                 weight_sum += weight
36
37         return weighted_sum / weight_sum
38
39     given_values = feature.getAttribute('__za{__}.__interp_attribute', list)
40     given_distances = feature.getAttribute('__za{__}.distance', list)
41     power_weighting = float(feature.getAttribute('__idw_weighting'))
42
43     estimated_value = idw_estimate(given_values, given_distances, power_weighting)
44
45     feature.setAttribute('__set_interp_attribute', estimated_value)
46
47     self.pyoutput(feature)
48
49 def close(self):
50     pass
51

```

# Tweaks/Finalisation

- Attribute Parameters
- Other Parameters
- Ports
- Termination
- Metadata

**Transformer Parameters** ✕

Transformer Name:

Category:

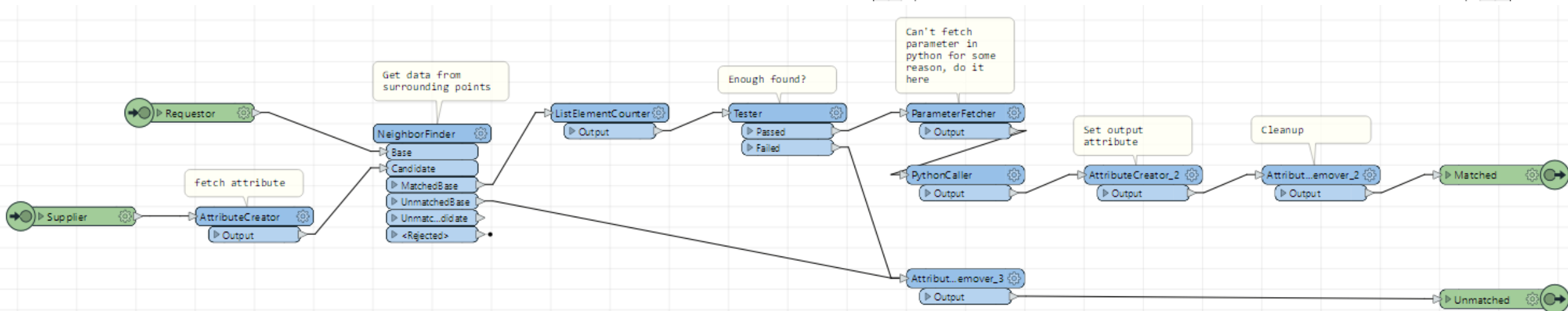
Transformer Description

Use Markdown (Recommended for FME Hub)

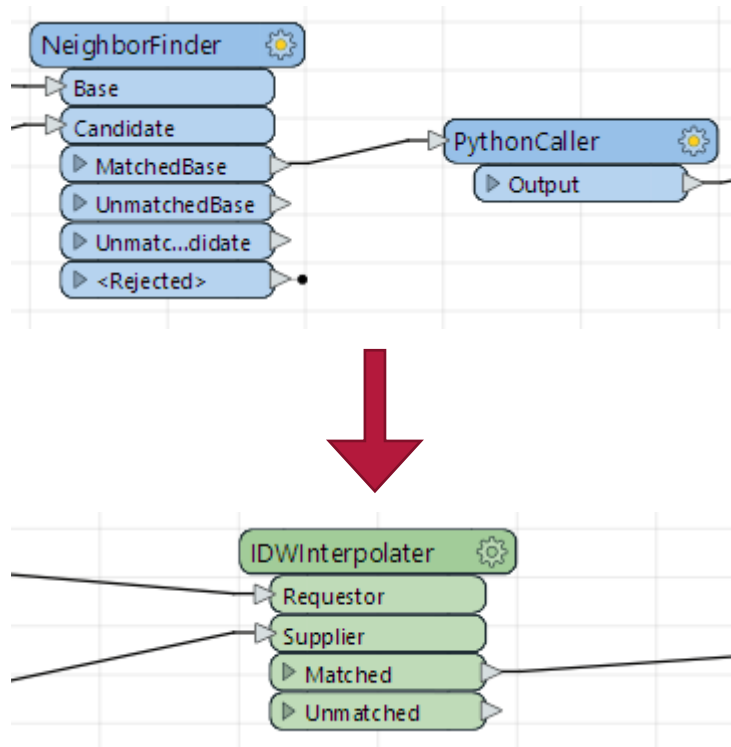
Overview Help History

A transformer to do a quick inverse distance weighting interpolation. Derive a value from multiple input points, weighting more those which are closer. It is simply a NeighbourFinder, followed by a PythonCaller to interpolate the selected value according to the relative distance to all the supplier locations. Supplier features must have a geometry, and attribute for interpolation must be numeric.

Chris Tredinnick, Abley, 2024



# Steps Overview



- Idea, check it needs to be made.
- Research, test to understand inputs.
- Generate script for IDW process in python.
- Plug it together, test it works.
- Package into custom transformer.
- Name ports, set up input parameters, attribute handling to be dynamic. Unmatched Inputs. Filter Suppliers. Version compatibility?
- Retest, tweak input parameters, retest.
- Name, title, description, export.

Questions?



**Auckland**

Level 1/70 Shortland Street  
Auckland 1010  
Aotearoa New Zealand

**Wellington**

Level 1/119-123 Featherston Street  
Wellington 6011  
Aotearoa New Zealand

**Christchurch**

Level 1/137 Victoria Street  
PO Box 36446, Merivale  
Christchurch 8146  
Aotearoa New Zealand

**hello@abley.com**

**+64 3 377 4703**

**abley.com**