

In Jupyter notebook
Choose how to color vertices in TDA mapper graph

```

nodes = mapper_output.nodes
node_color = None
#node_color = [1, 2, 3, 4, 5, 6, 7, 8] # Coloring the 8 nodes using 1D array
#point_color = None
point_color = f # color nodes using average filter value
#point_color = (data[:,0]**2 + data[:,1]**2)*0.5 # color nodes in mapper output using average length
#point_color = data[:,0] # color nodes in mapper output using average of first coordinates
name = "custom scheme"
node_color = mapper_output.postprocess_node_color(node_color, point_color, point_labels)
    
```

Output the color code for each node in TDA mapper graph

```

node_color
array([-0.03425722, -0.01541899, -0.01547266, -0.000479 , 0.00117447,
        0.01467286, 0.01467839, 0.03424094])
    
```

In Jupyter notebook
Choose how to color vertices in TDA mapper graph

```

nodes = mapper_output.nodes
node_color = None
#node_color = [1, 2, 3, 4, 5, 6, 7, 8] # Coloring the 8 nodes using 1D array
#point_color = None
point_color = f # color nodes using average filter value
#point_color = (data[:,0]**2 + data[:,1]**2)*0.5 # color nodes in mapper output using average length
#point_color = data[:,0] # color nodes in mapper output using average of first coordinates
name = "custom scheme"
node_color = mapper_output.postprocess_node_color(node_color, point_color, point_labels)
    
```

Output the color code for each node in TDA mapper graph

```

node_color
array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.])
    
```

```

In [56]: nodes
Out[56]: [node(0,),array([ 2,  3,  4,  8, 11, 13, 16, 18, 19, 21, 22, 23, 24,
        27, 31, 35, 39, 41, 42, 46, 47, 49, 51, 54, 58, 61,
        64, 69, 70, 71, 74, 75, 77, 80, 81, 82, 84, 86, 88,
        92, 95, 96, 98, 101, 103, 104, 111, 112, 113, 114, 115, 117,
        118, 119, 120, 121, 122, 124, 125, 129, 130, 131, 132, 135, 141,
        145, 148, 152, 153, 154, 155, 156, 157, 161, 162, 163, 166, 167,
        169, 170, 176, 180, 182, 188, 189, 190, 196, 198, 199, 201, 204,
        206, 210, 215, 216, 220, 224, 225, 228, 229, 235, 237, 239, 244,
        245, 250, 251, 252, 256, 259, 262, 264, 271, 275, 277, 280, 290,
        292, 293, 297, 298, 299, 300, 302, 318, 321, 322, 325, 329, 331,
        336, 337, 339, 341, 343, 344, 347, 348, 350, 351, 352, 359, 360,
        361, 362, 367, 369, 375, 376, 377, 378, 379, 380, 383, 384, 385,
        386, 388, 389, 391, 393, 396, 397, 398, 399, 400, 401, 405, 406,
        407, 410, 412, 413, 416, 424, 426, 427, 428, 431, 436, 438, 440,
    
```

```

912, 915, 916, 918, 921, 926, 927, 933, 934, 936, 943, 950, 951,
956, 957, 959, 961, 962, 966, 970, 971, 973, 980, 987, 988, 989, 999]),-0.03658378562302291
2),
node(1,),array([ 1, 16, 24, 49, 55, 68, 72, 114, 128, 132, 133, 135, 137,
138, 162, 180, 185, 187, 189, 192, 193, 201, 206, 213, 215, 222,
235, 244, 247, 256, 263, 290, 294, 307, 323, 350, 357, 369, 374,
378, 380, 383, 395, 400, 401, 405, 412, 420, 431, 434, 435, 451,
454, 455, 465, 469, 486, 487, 500, 521, 534, 535, 539, 556, 560,
574, 582, 584, 591, 595, 596, 597, 599, 604, 612, 613, 640, 642,
643, 645, 646, 660, 661, 672, 674, 676, 677, 708, 710, 713, 735,
744, 757, 762, 780, 783, 788, 792, 798, 803, 819, 820, 821, 823,
875, 880, 887, 888, 889, 896, 901, 912, 958, 971, 983, 989, 991, 994]),-0.01554355948362592
2),
node(1,),array([ 4, 18, 25, 31, 39, 41, 45, 58, 59, 80, 86, 94, 95,
98, 112, 115, 118, 121, 125, 139, 141, 148, 149, 155, 166, 168,
175, 182, 214, 216, 217, 227, 238, 245, 250, 259, 270, 297, 299,
322, 347, 348, 359, 368, 371, 397, 410, 417, 426, 440, 446, 453,
458, 478, 506, 507, 509, 510, 533, 551, 553, 576, 581, 603, 608,
609, 616, 619, 610, 648, 655, 664, 667, 681, 691, 702, 715, 718,
    
```

node_color

```

array([-0.03425722, -0.01541899, -0.01547266, -0.000479 , 0.00117447,
        0.01467286, 0.01467839, 0.03424094])
    
```