pysld

Release 0.0.5

Tek Bahadur Kshetri

Jul 11, 2022

TABLE OF CONTENTS

CHAPTER

ONE

ABOUT

The package is useful for the generating the SLD file for vector and raster datasets.

The package can generate the 3 types of styles for vector dataset and 1 type of style for raster,

1. Simple style (for vector dataset)
2. Categorized style (for vector dataset)
3. Classified style (for vector dataset)
4. Raster style (for raster dataset)

CHAPTER

TWO

INSTALLATION

The package can be installed by following command,

```
pip installation pysld
```

2.1 Dependencies

- numpy
- matplotlib
- seaborn
- jenkspy
- psycopg2

CHAPTER

THREE

LICENSE

3.1 MIT License

CHAPTER

FOUR

CONTRIBUTION

`Pysld` is an open source library written in python and contributors are needed to keep this library moving forward. Any kind of contributions are welcome.

CHAPTER

FIVE

ACKNOWLEDGEMENTS

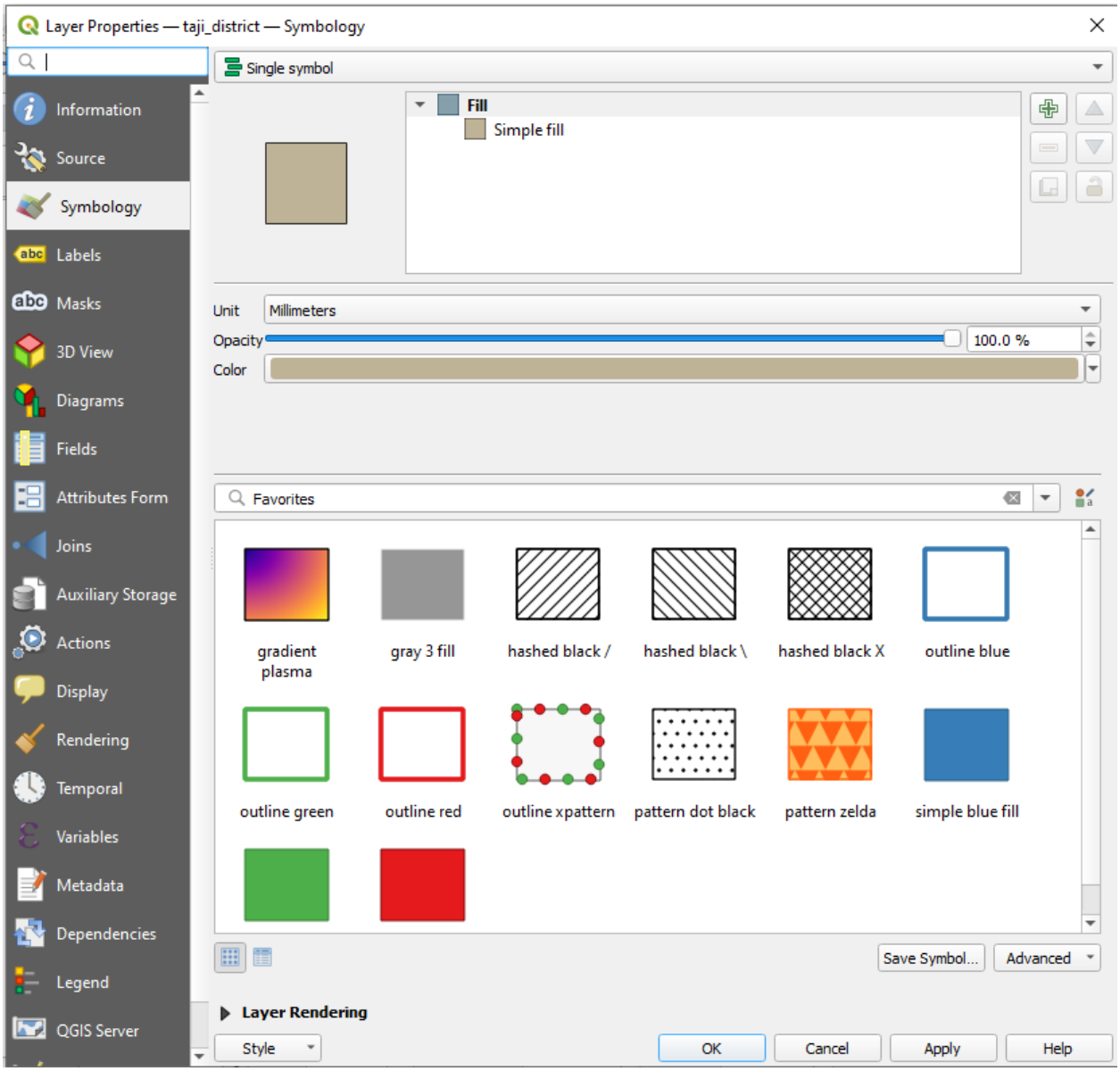
Created and managed by Tek Bahadur Kshetri.

CHAPTER

SIX

SIMPLE STYLE

The simple style is simillar to QGIS single symbol style.



The simple style is available for all point, line and polygon datasets. Below is the example of generating the simple

style for polygon dataset,

```
# Import and initialized package
from pysld.style import StyleSld
sld = StyleSld(style_name='polygonStyle', geom_type='polygon', fill_color='#ffffff',␣
↪stroke_color='#333333')

# Generate the simple style
sld.generate_simple_style()
```

The above code will generate the following sld file,

```
<StyledLayerDescriptor version="1.0.0" xsi:schemaLocation="http://www.opengis.net/sld␣
↪StyledLayerDescriptor.xsd" xmlns="http://www.opengis.net/sld" xmlns:ogc="http://www.
↪opengis.net/ogc" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.
↪org/2001/XMLSchema-instance">
    <NamedLayer>
        <Name>style</Name>
        <UserStyle>
        <Title>style</Title>
        <FeatureTypeStyle>
    <Rule>
        <PolygonSymbolizer>
            <Fill>
                <CssParameter name="fill">#ffffff</CssParameter>
                <CssParameter name="fill-opacity">1</CssParameter>
            </Fill>
            <Stroke>
                <CssParameter name="stroke">#333333</CssParameter>
                <CssParameter name="stroke-width">1</CssParameter>
            </Stroke>
        </PolygonSymbolizer>
    </Rule>
        </FeatureTypeStyle>
        </UserStyle>
    </NamedLayer>
</StyledLayerDescriptor>
```

6.1 Add feature label

If you want to add the label feature, simple add two more parameters to `StyleSld` class as below,

```
# Import and initialized package
from pysld.style import StyleSld
sld = StyleSld(style_name='polygonStyle', geom_type='polygon', fill_color='#ffffff',␣
↪stroke_color='#333333', feature_label=True, attribute_name_label='name')

# Generate the simple style
sld.generate_simple_style()
```

The above code will add the additional SLD as below,

```
<TextSymbolizer>
    <Label>
        <ogc:PropertyName>name</ogc:PropertyName>
    </Label>
    <Font>
        <CssParameter name="font-family">Arial</CssParameter>
        <CssParameter name="font-size">12</CssParameter>
        <CssParameter name="font-style">normal</CssParameter>
        <CssParameter name="font-weight">bold</CssParameter>
    </Font>
    <Fill>
        <CssParameter name="fill">#990099</CssParameter>
    </Fill>
</TextSymbolizer>
```

Here is the list of available values and their description for simple style,

6.2 Available options for simple style

Table 1: Options for `StyleSld`

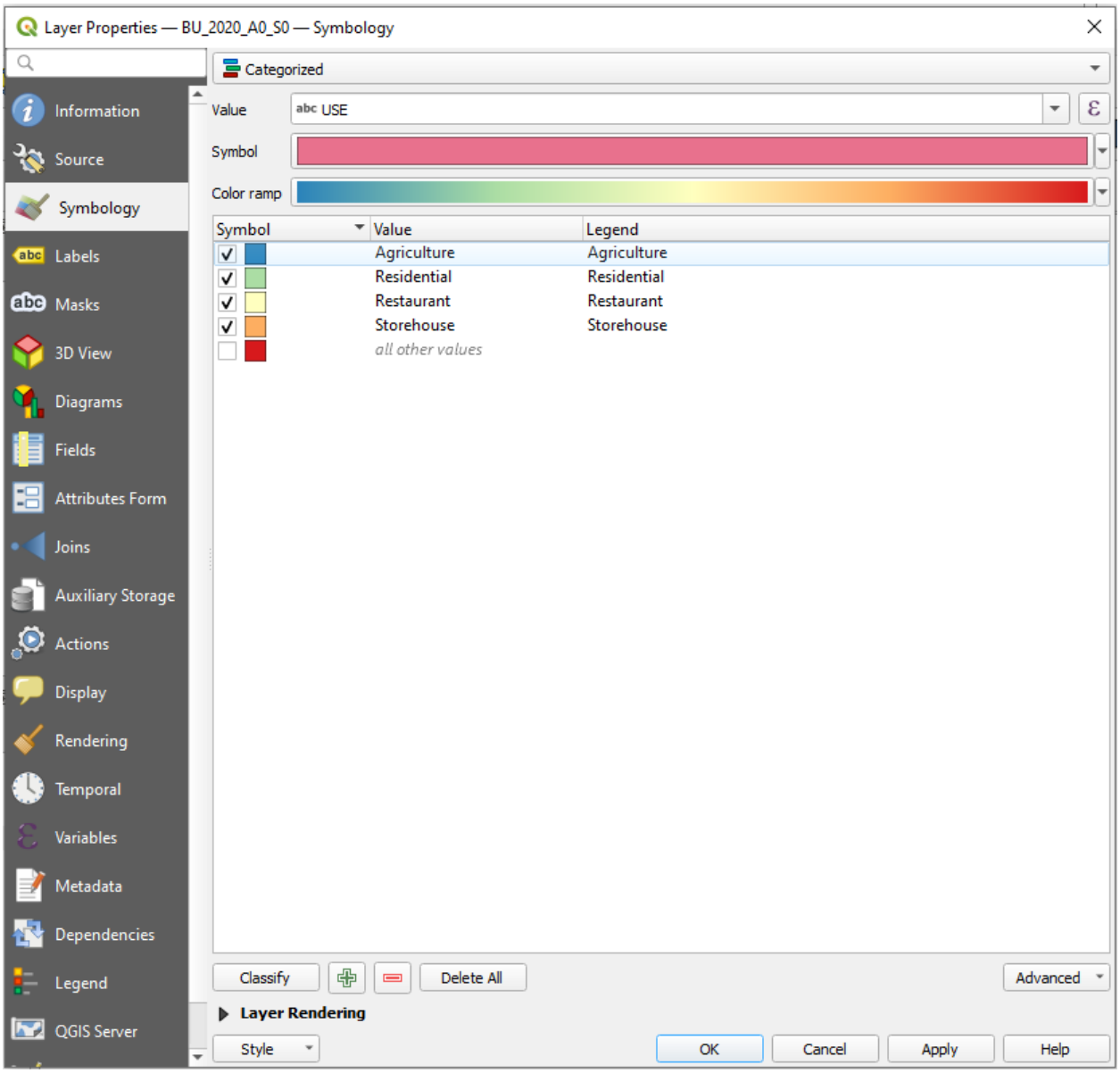
Options	DataType	Default	Description
style_name	string	'style'	The name of the style file.
geom_type	string	'polygon'	The type of geometry. The available values are `point`, `line` and `polygon`.
fill_color	string, color code	'#ffffff'	Fill color for the point or polygon. Not applicable for line.
stroke_color	string, color code	'#333333'	Stroke color (outline color) for the input `geom_type`.
stroke_width	numeric	1	Stroke width (outline width) for the input `geom_type`.
opacity	integer	1	Fill opacity for the point or polygon feature. The value must be between 0 and 1. Not applicable for line.
point_size	numeric	6	The size of the point feature. The parameter will be ignored for polygon and line feature.
well_known_name	string	'circle'	The name of the shape. Available options are, `square`, `circle`, `triangle`, `star`, `cross` and `x` . The parameter will be ignored for polygon and line feature.
point_rotation	integer	0	Rotation of the point in degree. The value must be between 0-360. The parameter will be ignored for polygon and line feature.
stroke_linecap	string	'round'	Determines how lines are rendered at their ends. Possible values are butt (sharp square edge), round (rounded edge), and square (slightly elongated square edge). The parameter will be ignored for point and polygon feature.
stroke_dasharray	string	None	Encodes a dash pattern as a series of numbers separated by spaces. Odd-indexed numbers (first, third, etc) determine the length in pxiels to draw the line, and even-indexed numbers (second, fourth, etc) determine the length in pixels to blank out the line. Default is an unbroken line. The parameter will be ignored for polygon and line feature.
perpendicular_offset	integer	None	Perpendicular offset for the line. The parameter will be ignored for point and polygon feature.
feature_label	boolean	False	It determines whether to add the feature label or not. If feature_label is true, then you need to pass the `attribute_name_label` parameter to label the feature.
attribute_name_label	string	None	The name of the attribute, which you want to label.
font_family	string	'Aerial'	Font family name for the label. eg. 'Aerial', 'Times new roman' etc.
font_color	string, color code	'#333333'	The Font color for the label.
font_size	integer	14	Font size for the label.
font_weight	string	'normal'	Font weight for the label. Available values are, `bold` and `normal`.
font_style	string	'normal'	Font style for the label. Available values are, `normal`, `italic` and `oblique`.
halo_color	string, color code	'#ffffff'	The colored background around the label text, which improves readability in low contrast situations.
halo_radius	numeric	1	The halo radius, in pixels.

CHAPTER

SEVEN

CATEGORIZED STYLE

The categorized style is simillar to QGIS categorized style.



The categorized style is available for all point, line and polygon datasets. This class is inherited from simple style.

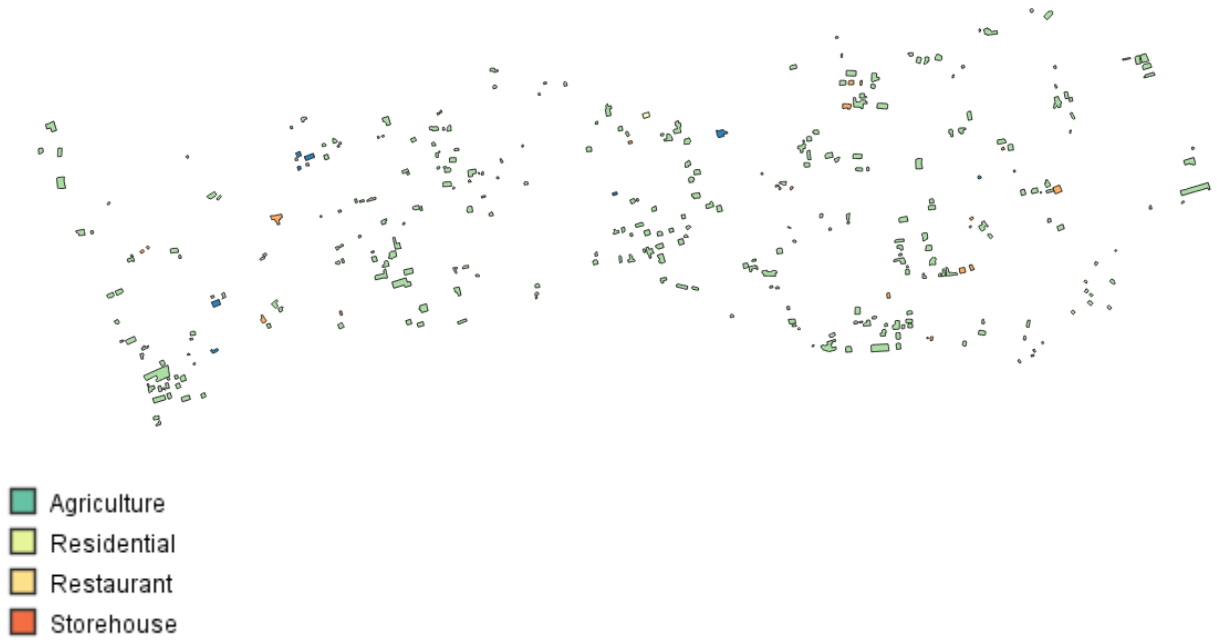
Below is the example of generating the categorized style for polygon dataset,

```
# Import and initialized package
from pysld.style import StyleSld
sld = StyleSld(
            style_name='polygonStyle',
            geom_type='polygon',
            attribute_name='USE',
            values=['Agriculture', 'Residential', 'Restaurant', 'Storehouse'],
            color_palette='Spectral_r',
        )

# Generate the categorized style
style = sld.generate_categorized_style()
print(style)
```

Note: If you want to add the label feature, simple add two more parameters, `feature_label=True` and `attribute_name_label` to the `StyleSld` class.

The above code will generate the following map with corresponding legend,



Which is simillar to following `xml`,

```
<StyledLayerDescriptor version="1.0.0" xsi:schemaLocation="http://www.opengis.net/sld␣
↪StyledLayerDescriptor.xsd" xmlns="http://www.opengis.net/sld" xmlns:ogc="http://www.
↪opengis.net/ogc" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.
↪org/2001/XMLSchema-instance">
<NamedLayer>
    <Name>polygonStyle</Name>
    <UserStyle>
    <Title>polygonStyle</Title>
    <FeatureTypeStyle>
```

(continues on next page)

(continued from previous page)

```
        <Rule>
            <Name>Agriculture</Name>
            <Title>Agriculture</Title>
            <ogc:Filter>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>USE</ogc:PropertyName>
                    <ogc:Literal>Agriculture</ogc:Literal>
                </ogc:PropertyIsEqualTo>
            </ogc:Filter>
        <PolygonSymbolizer>
            <Fill>
                <CssParameter name="fill">#66c2a5</CssParameter>
                <CssParameter name="fill-opacity">1</CssParameter>
            </Fill>
            <Stroke>
                <CssParameter name="stroke">#333333</CssParameter>
                <CssParameter name="stroke-width">1</CssParameter>
            </Stroke>
        </PolygonSymbolizer>
        </Rule>
        <Rule>
            <Name>Residential</Name>
            <Title>Residential</Title>
            <ogc:Filter>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>USE</ogc:PropertyName>
                    <ogc:Literal>Residential</ogc:Literal>
                </ogc:PropertyIsEqualTo>
            </ogc:Filter>
            <PolygonSymbolizer>
                <Fill>
                    <CssParameter name="fill">#e6f598</CssParameter>
                    <CssParameter name="fill-opacity">1</CssParameter>
                </Fill>
                <Stroke>
                    <CssParameter name="stroke">#333333</CssParameter>
                    <CssParameter name="stroke-width">1</CssParameter>
                </Stroke>
            </PolygonSymbolizer>
        </Rule>
        <Rule>
            <Name>Restaurant</Name>
            <Title>Restaurant</Title>
            <ogc:Filter>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>USE</ogc:PropertyName>
                    <ogc:Literal>Restaurant</ogc:Literal>
                </ogc:PropertyIsEqualTo>
            </ogc:Filter>
            <PolygonSymbolizer>
                <Fill>
                    <CssParameter name="fill">#fee08b</CssParameter>
```

(continues on next page)

(continued from previous page)

```
                    <CssParameter name="fill-opacity">1</CssParameter>
                </Fill>
                <Stroke>
                    <CssParameter name="stroke">#333333</CssParameter>
                    <CssParameter name="stroke-width">1</CssParameter>
                </Stroke>
            </PolygonSymbolizer>
        </Rule>
        <Rule>
            <Name>Storehouse</Name>
            <Title>Storehouse</Title>
            <ogc:Filter>
                <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>USE</ogc:PropertyName>
                    <ogc:Literal>Storehouse</ogc:Literal>
                </ogc:PropertyIsEqualTo>
            </ogc:Filter>
            <PolygonSymbolizer>
                <Fill>
                    <CssParameter name="fill">#f46d43</CssParameter>
                    <CssParameter name="fill-opacity">1</CssParameter>
                </Fill>
                <Stroke>
                    <CssParameter name="stroke">#333333</CssParameter>
                    <CssParameter name="stroke-width">1</CssParameter>
                </Stroke>
            </PolygonSymbolizer>
        </Rule>
    </FeatureTypeStyle>
    </UserStyle>
</NamedLayer>
</StyledLayerDescriptor>
```

7.1 Generate style for PostGIS data

If the shapefile is available in `PostgreSQL` database, then you need to provide the PostgreSQL connection parameters as below,

```
# Import and initialized package
from pysld.style import StyleSld
sld = StyleSld(
            style_name='polygonStyle',
            geom_type='polygon',
            attribute_name='USE',
            color_palette='Spectral_r',

            # Postgres connection parameters
            dbname='postgres',
            user='postgres',
            password='admin',
```

(continues on next page)

(continued from previous page)

```
            host='localhost',
            port='5432',
            schema='public',
            pg_table_name='postgres_table_name'
        )

print(sld.values) # It will print the unique values from postgres_table_name table

style = sld.generate_categorized_style()  # Generate the categorized style
print(style) # print categorized style file
```

The above `StyleSld` will get the values internally and create the categorized style file for `postgres_table_name` table.

7.2 Available options for categorized style

Table 1: Options for `StyleSld`

Options	DataType	Default	Description
attribute_name	string		The attribute name for generating the categorized style. The attribute name either provided directly or can be get using `get_attribute_name()` function. See *Some additional function* for more detail.
values	string		The values based on which the categorized style will be generated. The values either provided directly or can be get using `get_values_from_pg()` function. See *Some additional function* for more detail.
color_palette	string, list of colors, dict	'Spectral_r'	The color palette to represent the layer. Check all the available names of color palette here
dbname	string	None	PostgreSQL database name. This parameter will be used for the PostgreSQL connection.
user	string	'postgres'	PostgreSQL database user. This parameter will be used for the PostgreSQL connection.
password	string	'admin'	PostgreSQL database user password. This parameter will be used for the PostgreSQL connection.
host	string	'localhost'	PostgreSQL database host. This parameter will be used for the PostgreSQL connection.
port	integer	5432	PostgreSQL database host. This parameter will be used for the PostgreSQL connection.
schema	string	'public'	PostgreSQL database data schema. This parameter will be used for get the data from PostgreSQL.
pg_table_name	string	None	PostgreSQL database data table name. This parameter will be used for get the data from PostgreSQL table.

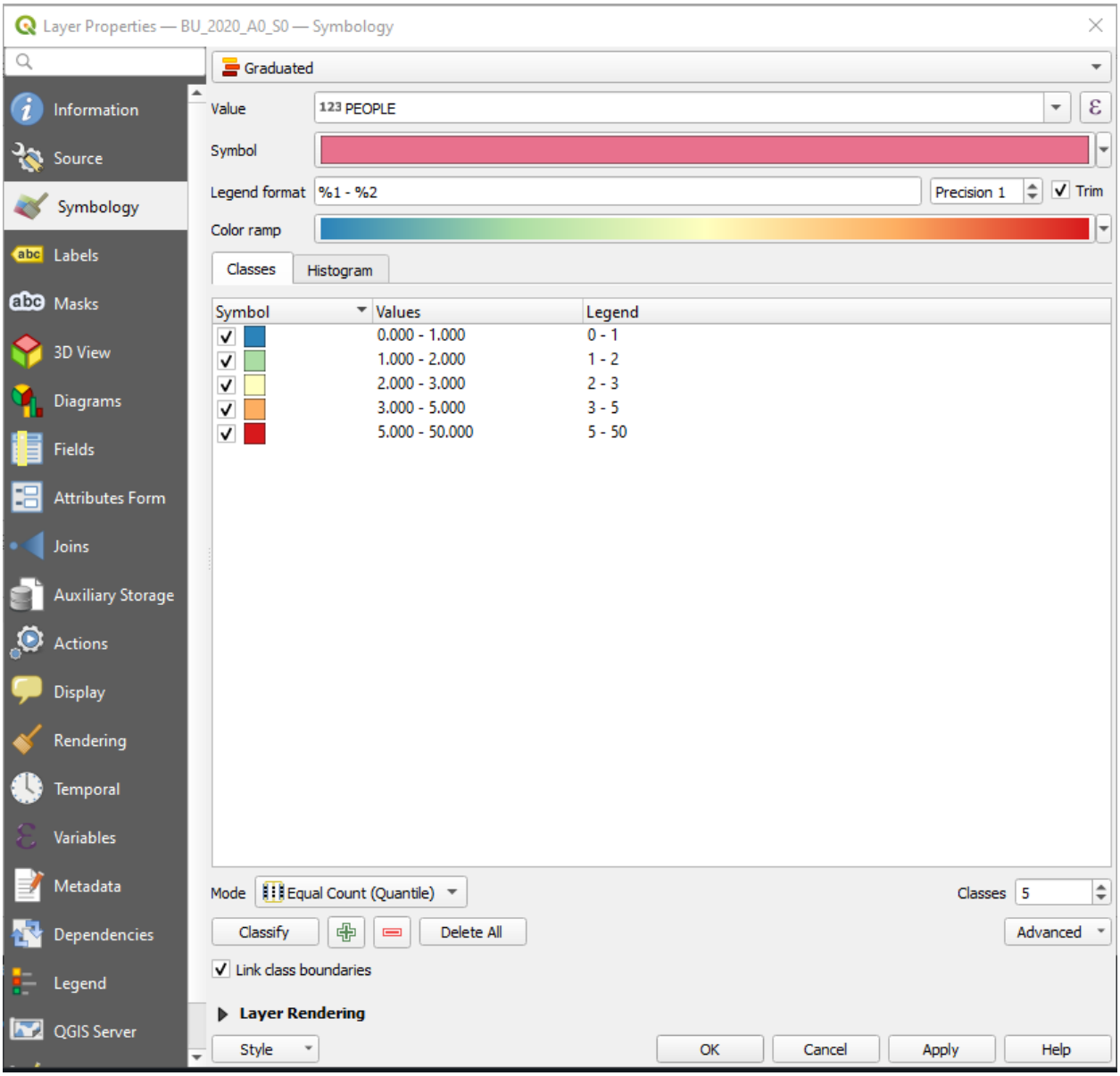
Since the `categorized style` is inherited from `simple style`, it supports all the parameters from `simple style` as well, see *Available options for simple style*.

CHAPTER

EIGHT

CLASSIFIED STYLE

The Classified style is simillar to QGIS graduated style.



The Classified style is available for all point, line and polygon datasets. This class is inherited from categorized style

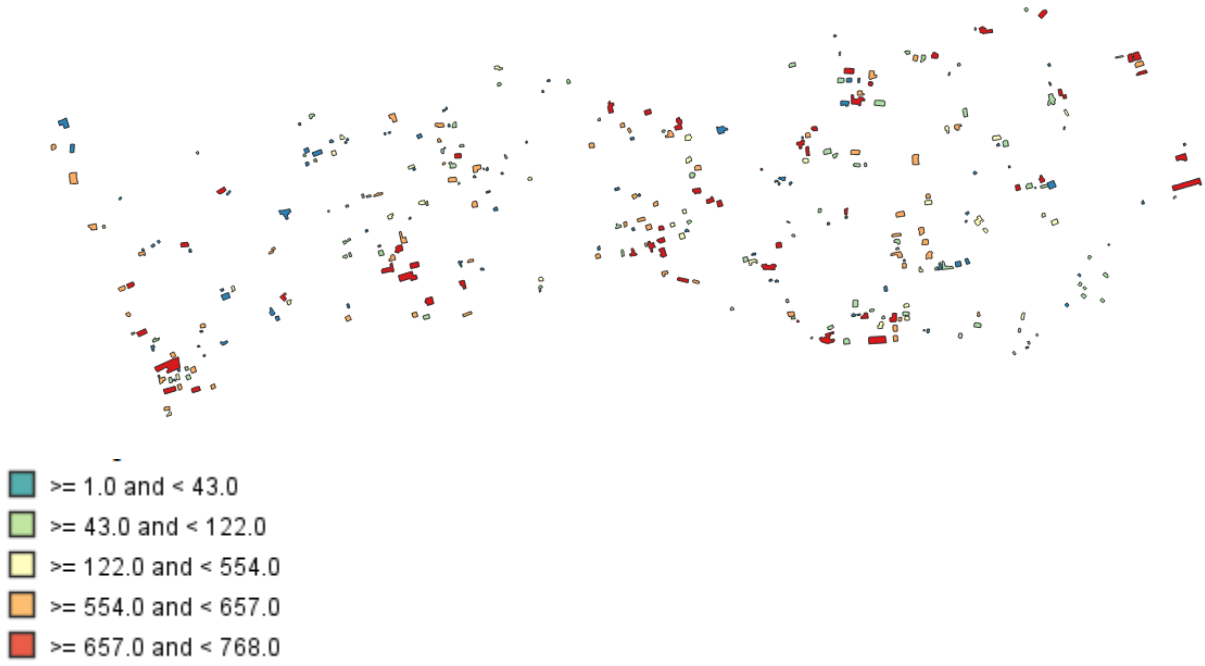
and simple style. Below is the example of generating the Classified style for polygon dataset,

```
# Import and initialized package
from pysld.style import StyleSld
sld = StyleSld(
            style_name='polygonStyle',
            geom_type='polygon',
            attribute_name='PEOPLE',
            values=[1,2,3,34,23,122,12,2,3,21,23,32,1,23,42,1,23,1,1,23,4,3,54,6,768,8,
↪554,3,43,543,6,657,7,75,4,4],
            number_of_class=5,
            classification_method='natural_break',
            color_palette='Spectral_r',
        )

# Generate the Classified style
style = sld.generate_classified_style()
print(style)
```

Note: If you want to add the label feature, simple add two more parameters, `feature_label=True` and `attribute_name_label` to the `StyleSld` class.

The above code will generate the following map with corresponding legend,



Which is simillar to following `xml`,

```
<StyledLayerDescriptor version="1.0.0" xsi:schemaLocation="http://www.opengis.net/sld␣
↪StyledLayerDescriptor.xsd" xmlns="http://www.opengis.net/sld" xmlns:ogc="http://www.
↪opengis.net/ogc" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.
↪org/2001/XMLSchema-instance">
    <NamedLayer>
        <Name>polygonStyle</Name>
```

(continues on next page)

(continued from previous page)

```
        <UserStyle>
        <Title>polygonStyle</Title>
        <FeatureTypeStyle>
            <Rule>
                <Name>&gt;= 1.0 and &lt; 43.0</Name>
                <Title>&gt;= 1.0 and &lt; 43.0</Title>
                <ogc:Filter>
                    <ogc:And>
                    <ogc:PropertyIsGreaterThanOrEqualTo>
                        <ogc:PropertyName>USE</ogc:PropertyName>
                        <ogc:Literal>1.0</ogc:Literal>
                    </ogc:PropertyIsGreaterThanOrEqualTo>
                    <ogc:PropertyIsLessThan>
                        <ogc:PropertyName>USE</ogc:PropertyName>
                        <ogc:Literal>43.0</ogc:Literal>
                    </ogc:PropertyIsLessThan>
                    </ogc:And>
                </ogc:Filter>
                <PolygonSymbolizer>
                    <Fill>
                        <CssParameter name="fill">S</CssParameter>
                        <CssParameter name="fill-opacity">1</CssParameter>
                    </Fill>
                    <Stroke>
                        <CssParameter name="stroke">#333333</CssParameter>
                        <CssParameter name="stroke-width">1</CssParameter>
                    </Stroke>
                </PolygonSymbolizer>
            </Rule>
            <Rule>
                <Name>&gt;= 43.0 and &lt; 122.0</Name>
                <Title>&gt;= 43.0 and &lt; 122.0</Title>
                <ogc:Filter>
                    <ogc:And>
                    <ogc:PropertyIsGreaterThanOrEqualTo>
                        <ogc:PropertyName>USE</ogc:PropertyName>
                        <ogc:Literal>43.0</ogc:Literal>
                    </ogc:PropertyIsGreaterThanOrEqualTo>
                    <ogc:PropertyIsLessThan>
                        <ogc:PropertyName>USE</ogc:PropertyName>
                        <ogc:Literal>122.0</ogc:Literal>
                    </ogc:PropertyIsLessThan>
                    </ogc:And>
                </ogc:Filter>
                <PolygonSymbolizer>
                    <Fill>
                        <CssParameter name="fill">p</CssParameter>
                        <CssParameter name="fill-opacity">1</CssParameter>
                    </Fill>
                    <Stroke>
                        <CssParameter name="stroke">#333333</CssParameter>
                        <CssParameter name="stroke-width">1</CssParameter>
```

(continues on next page)

(continued from previous page)

```
                </Stroke>
            </PolygonSymbolizer>
        </Rule>
        <Rule>
            <Name>&gt;= 122.0 and &lt; 554.0</Name>
            <Title>&gt;= 122.0 and &lt; 554.0</Title>
            <ogc:Filter>
                <ogc:And>
                <ogc:PropertyIsGreaterThanOrEqualTo>
                    <ogc:PropertyName>USE</ogc:PropertyName>
                    <ogc:Literal>122.0</ogc:Literal>
                </ogc:PropertyIsGreaterThanOrEqualTo>
                <ogc:PropertyIsLessThan>
                    <ogc:PropertyName>USE</ogc:PropertyName>
                    <ogc:Literal>554.0</ogc:Literal>
                </ogc:PropertyIsLessThan>
                </ogc:And>
            </ogc:Filter>
            <PolygonSymbolizer>
                <Fill>
                    <CssParameter name="fill">e</CssParameter>
                    <CssParameter name="fill-opacity">1</CssParameter>
                </Fill>
                <Stroke>
                    <CssParameter name="stroke">#333333</CssParameter>
                    <CssParameter name="stroke-width">1</CssParameter>
                </Stroke>
            </PolygonSymbolizer>
        </Rule>
        <Rule>
            <Name>&gt;= 554.0 and &lt; 657.0</Name>
            <Title>&gt;= 554.0 and &lt; 657.0</Title>
            <ogc:Filter>
                <ogc:And>
                <ogc:PropertyIsGreaterThanOrEqualTo>
                    <ogc:PropertyName>USE</ogc:PropertyName>
                    <ogc:Literal>554.0</ogc:Literal>
                </ogc:PropertyIsGreaterThanOrEqualTo>
                <ogc:PropertyIsLessThan>
                    <ogc:PropertyName>USE</ogc:PropertyName>
                    <ogc:Literal>657.0</ogc:Literal>
                </ogc:PropertyIsLessThan>
                </ogc:And>
            </ogc:Filter>
            <PolygonSymbolizer>
                <Fill>
                    <CssParameter name="fill">c</CssParameter>
                    <CssParameter name="fill-opacity">1</CssParameter>
                </Fill>
                <Stroke>
                    <CssParameter name="stroke">#333333</CssParameter>
                    <CssParameter name="stroke-width">1</CssParameter>
```

(continues on next page)

(continued from previous page)

```
                    </Stroke>
                </PolygonSymbolizer>
            </Rule>
            <Rule>
                <Name>&gt;= 657.0 and &lt; 768.0</Name>
                <Title>&gt;= 657.0 and &lt; 768.0</Title>
                <ogc:Filter>
                    <ogc:And>
                    <ogc:PropertyIsGreaterThanOrEqualTo>
                        <ogc:PropertyName>USE</ogc:PropertyName>
                        <ogc:Literal>657.0</ogc:Literal>
                    </ogc:PropertyIsGreaterThanOrEqualTo>
                    <ogc:PropertyIsLessThan>
                        <ogc:PropertyName>USE</ogc:PropertyName>
                        <ogc:Literal>768.0</ogc:Literal>
                    </ogc:PropertyIsLessThan>
                    </ogc:And>
                </ogc:Filter>
                <PolygonSymbolizer>
                    <Fill>
                        <CssParameter name="fill">t</CssParameter>
                        <CssParameter name="fill-opacity">1</CssParameter>
                    </Fill>
                    <Stroke>
                        <CssParameter name="stroke">#333333</CssParameter>
                        <CssParameter name="stroke-width">1</CssParameter>
                    </Stroke>
                </PolygonSymbolizer>
            </Rule>
        </FeatureTypeStyle>
        </UserStyle>
    </NamedLayer>
</StyledLayerDescriptor>
```

For PostGIS data, check *Generate style for PostGIS data* section from *Categorized Style*.

For the feature label check *Add feature label* section from *Simple style*.

8.1 Available options for classified style

Since the `classifed style` is inherited from *Categorized Style* and *Simple style*, it supports all the parameters and function from `categorized style` and `simple style` as well, see *Available options for categorized style* and *Available options for simple style*.

Table 1: Options for `StyleSld`

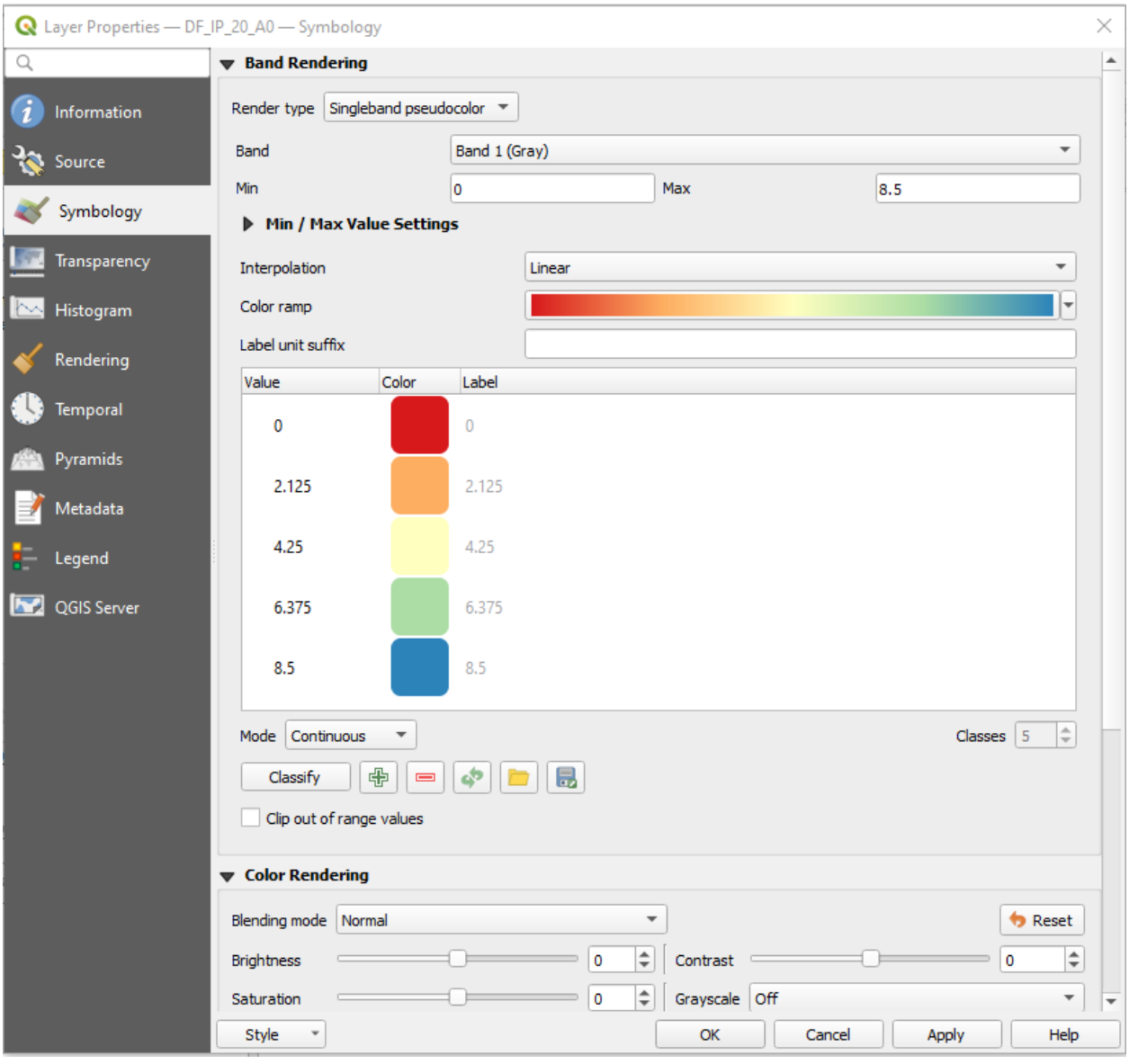
Options	DataType	Default	Description
number_of_class	integer	5	The number of classes for classify the values.
values	list of numeric values		It should be list of numeric values. See *Some additional function* for more detail.
classification_method	string	'natural_break'	The classification method for classify the vlaues. Available options are, `natural_break`, `equal_interval`, `quantile`, `standard_deviation` and `geometrical_interval`.

CHAPTER

NINE

RASTER STYLE

The Raster style is simillar to QGIS singleband pseudocolor.



The below is the example code for raster style

```
# Import and initialized package
from pysld.style import StyleSld
sld = StyleSld(
            style_name='polygonStyle',
            color_palette='Spectral_r',
            continuous_legend=True,
        )

# Generate the Raster style
style = sld.generate_raster_style(max_value=100, min_value=0)
print(style)
```

The above code will print the following text,

```
<StyledLayerDescriptor xmlns="http://www.opengis.net/sld" xmlns:gml="http://www.opengis.
↪net/gml" version="1.0.0" xmlns:ogc="http://www.opengis.net/ogc" xmlns:sld="http://www.
↪opengis.net/sld">
    <UserLayer>
        <sld:LayerFeatureConstraints>
        <sld:FeatureTypeConstraint/>
        </sld:LayerFeatureConstraints>
        <sld:UserStyle>
        <sld:Name>polygonStyle</sld:Name>
        <sld:FeatureTypeStyle>
            <sld:Rule>
            <sld:RasterSymbolizer>
                <Opacity>1</Opacity>
                <sld:ChannelSelection>
                <sld:GrayChannel>
                    <sld:SourceChannelName>1</sld:SourceChannelName>
                </sld:GrayChannel>
                </sld:ChannelSelection>
                <sld:ColorMap type="range">
                    <sld:ColorMapEntry color="#54aead" label=" 0.0" quantity="0.0"/>
                    <sld:ColorMapEntry color="#bfe5a0" label=" 25.0" quantity="25.0"/>
                    <sld:ColorMapEntry color="#fffebe" label=" 50.0" quantity="50.0"/>
                    <sld:ColorMapEntry color="#fdbf6f" label=" 75.0" quantity="75.0"/>
                    <sld:ColorMapEntry color="#e95c47" label=" 100.0" quantity="100.0"/>
                </sld:ColorMap>
            </sld:RasterSymbolizer>
            </sld:Rule>
        </sld:FeatureTypeStyle>
        </sld:UserStyle>
    </UserLayer>
</StyledLayerDescriptor>
```

9.1 Get min_value, max_value of raster

If you like to calculate the min_value and max_value of raster dynamically, you can try following line of code,

```
from osgeo import gdal

file = r'path/to/tiff/file.tif'
gtif = gdal.Open(file)

srcband = gtif.GetRasterBand(1)
srcband.ComputeStatistics(0)

min_value = srcband.GetMinimum()
max_value = srcband.GetMaximum()
```

CHAPTER

TEN

SOME ADDITIONAL FUNCTION

```
from pysld.style import StyleSld
sld = StyleSld (
    style_name='polygonStyle',
    geom_type='polygon',
    attribute_name='USE',
    color_palette='Spectral_r',

    # Postgres connection parameters
    dbname='postgres',
    user='postgres',
    password='admin',
    host='localhost',
    port='5432',
    schema='public',
)

sld.get_attribute_name(pg_table_name='postgres_table_name') # get the random attribute_
↪name
print(sld.attribute_name) # Print the random attribute name from ``get_attribute_name()``␣
↪function

sld.get_values_from_pg() # Get and set the values from given ``pg_table_name`` table and␣
↪``attribute_name`` column
print(sld.values)
```

10.1 PostgreSQL functions

Below is the example of some postgres functionalities,

```
from pysld.postgres import Pg

pg = Pg(dbname='dbname', user='postgres', password='admin', host='localhost', port=5432)

pg.connect() # connect the postgresql

pg.set_postgres_schema('public') # set the schema to public

pg.get_column_names(table='table_name') # get all the column names from "table_name"␣
↪table
```

(continues on next page)

(continued from previous page)

```
pg.get_values_from_column(column='column_name', table='table_name', schema='public') #␣
↪get the values from "column_name" column of "table_name" table
```