Orange3-Geo Documentation

-

Feb 12, 2023

Contents

1 Widgets 1

2 Indices and tables 11

CHAPTER 1

Widgets

1.1 Geo Map

Show data points on a map.

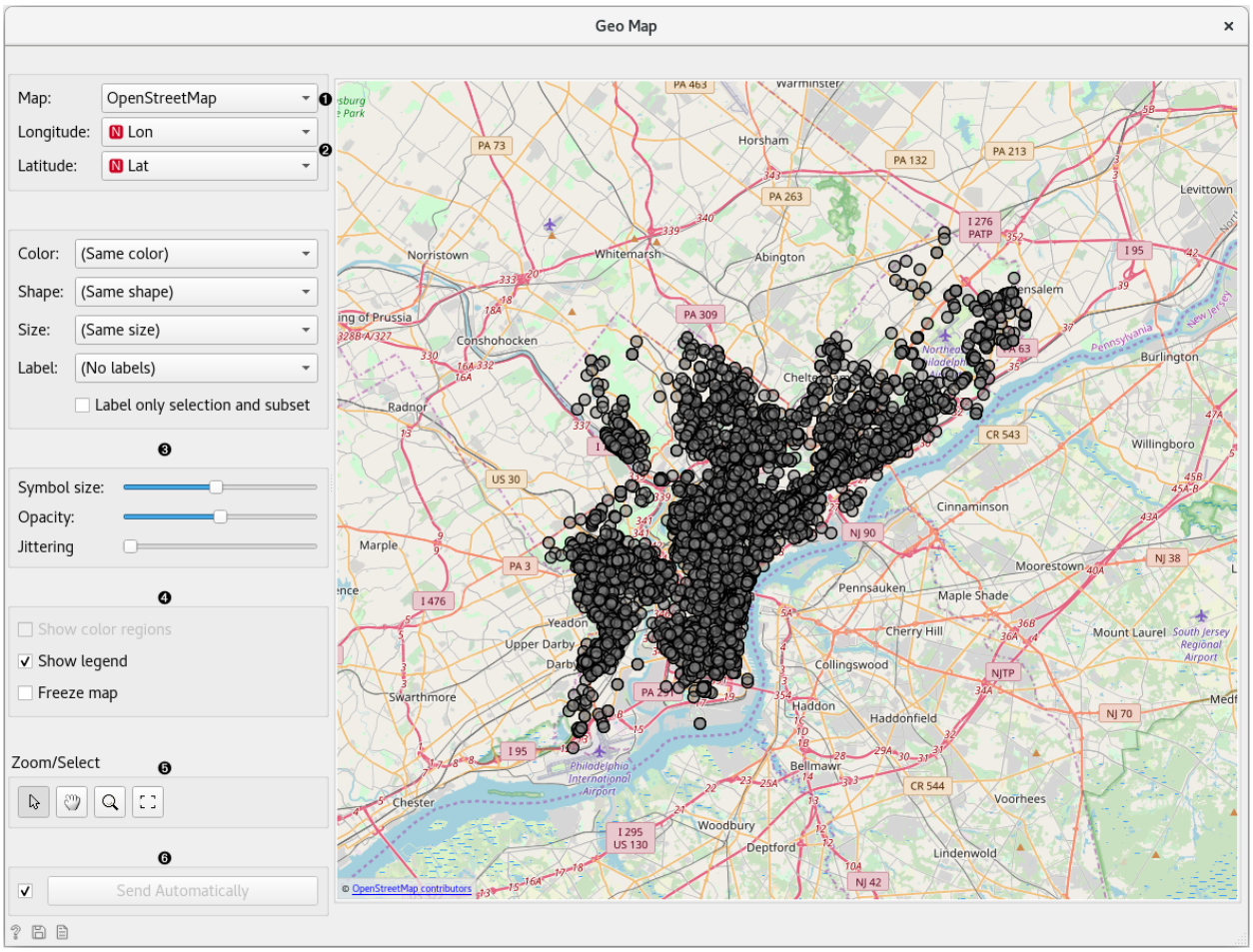
Inputs

- Data: input dataset
- Data Subset: subset of instances

Outputs

- Selected Data: instances selected from the plot
- Data: data with an additional column showing whether a point is selected

Geo Map widget visualizes geo-spatial data on a map. It works on datasets containing latitude and longitude variables in WGS 84 (EPSG:4326) format. We can use it much like we use Scatter Plot widget.

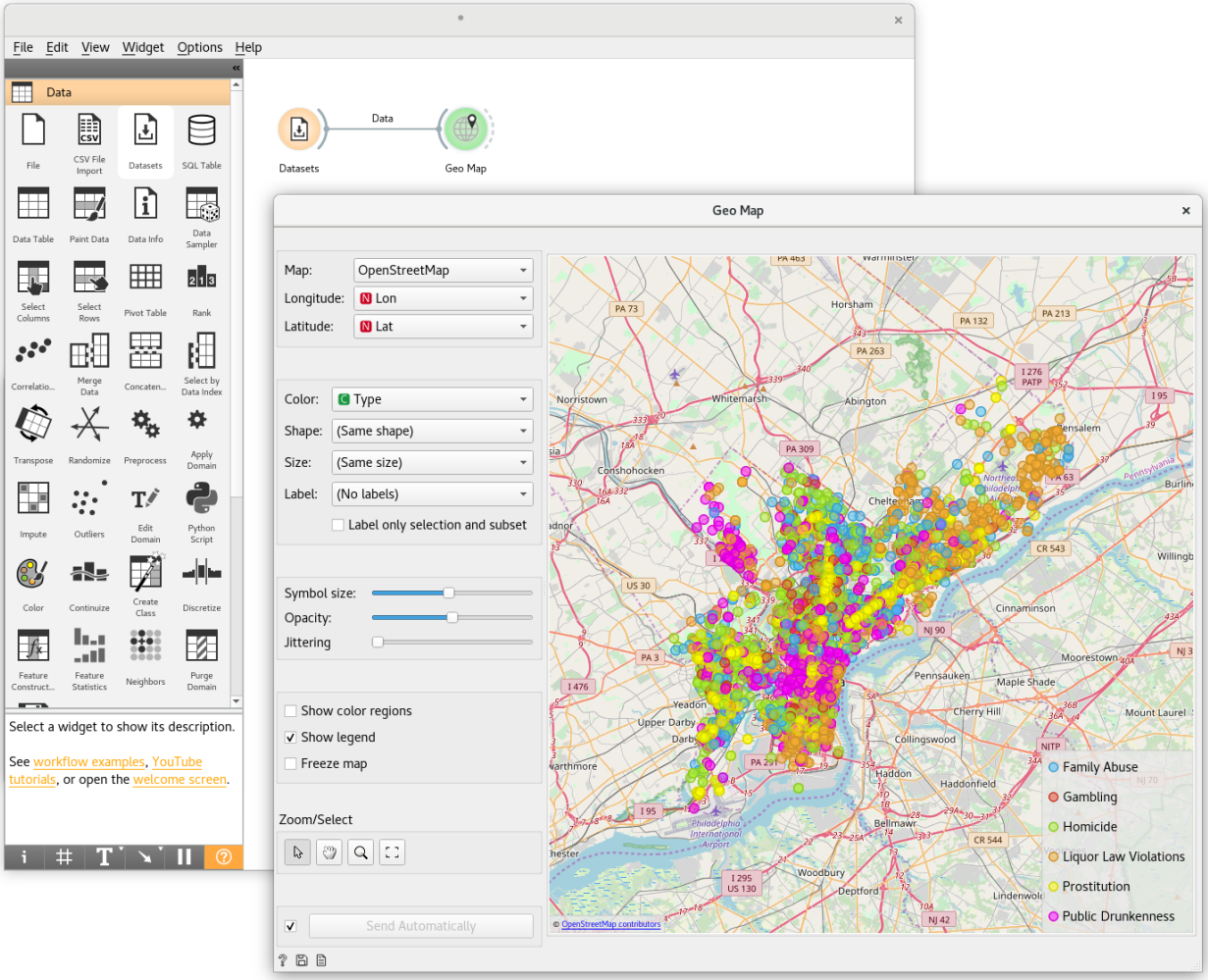


1. Set the type of map: OpenStreetMap, Black and White, Topographic, Satellite, Print, Dark.
2. Set latitude and longitude attributes, if the widget didn't recognize them automatically. Latitude values should be between -85.0511(S) and 85.0511(N) (a limitation of the projections onto flat maps) and longitude values between -180(W) and 180(E).
3. Set color, shape, size and label to differentiate between points. Set symbol size, opacity and jittering for all data points.
4. Adjust *plot properties*:
   - *Show color region* colors the graph by class (color must be selected).
   - *Show legend* displays a legend on the right. Click and drag the legend to move it.
   - *Freeze map* freezes the map so it doesn't update when input data changes.
5. *Select, zoom, pan and zoom to fit* are the options for exploring the graph. The manual selection of data instances works as an angular/square selection tool. Scroll in or out for zoom.
6. If *Send automatically* is ticked, changes are communicated automatically. Alternatively, press *Send*.

1.1.1 Examples

In this simple example we visualize the *Philadelphia Crime* dataset that we can find in the **Datasets** widget. We connect the output of that widget to the **Map** widget. Latitude and longitude variables get automatically detected and

we additionally select the crime type variable for color. We can observe how different crimes are more present in specific areas of the city.



1.2 Geocoding

Encode region names into geographical coordinates, or reverse-geocode latitude and longitude pairs into regions.

Inputs

- Data: An input data set.

Outputs

- Coded Data: Data set with new meta attributes.

Geocoding widget extracts latitude/longitude pairs from region names or synthesizes latitude/longitude to return region name. If the region is large, say a country, encoder with return the latitude and longitude of geometric centre.



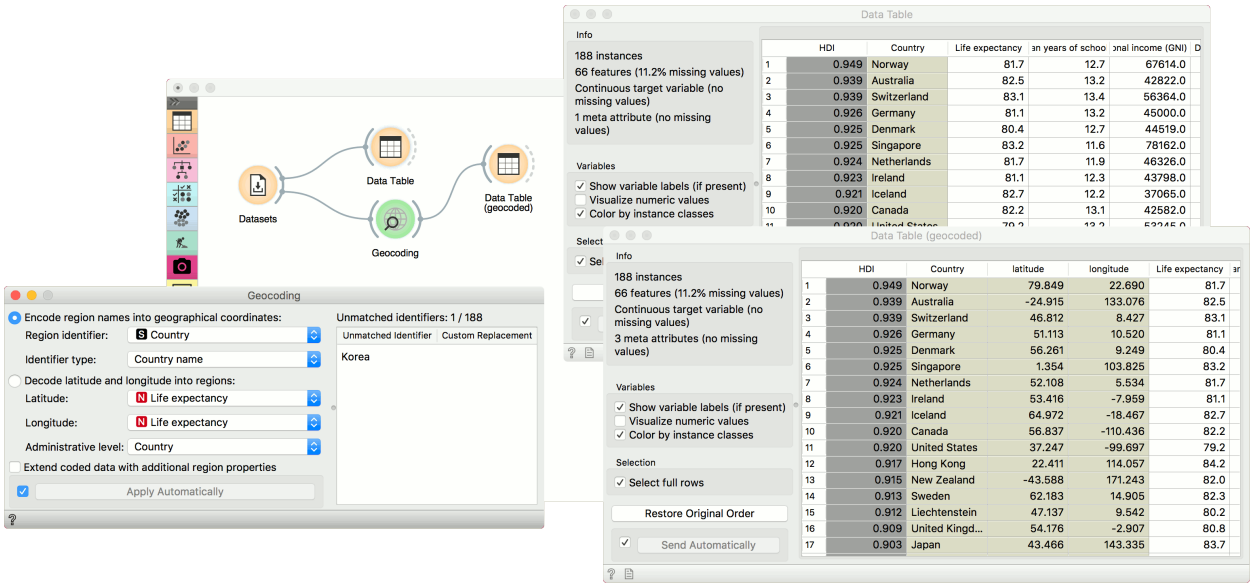
1. Use region names to extract the corresponding latitude/longitude pairs:
   - Region identifier: attribute with the information on region names. Can be discrete or string.
   - Identifier type: define how the data is coded. Supports ISO codes and some major cities and countries.
2. Use latitude and longitude pairs to retrieve region names:
   - Latitude attribute.
   - Longitude attribute.
   - Administrative level of the region you wish to extract.
3. Extend coded data adds additional information on the region of interest. For countries, for example, one would get economy, type, continent, etc.
4. If *Apply Automatically* is ticked, the changes will be communicated automatically. Alternatively, press *Apply*.
5. Unmatched identifiers editor. Match regions names that couldn't be matched automatically with their corresponding name.

1.2.1 Example

We will use *HDI* data from the **Datasets** widget. Open the widget, find *HDI* data, select it and press *Send*. First, let us observe the data in a **Data Table**. We have a meta attribute names *Country*, which contains country names. Now we would like to plot this on a map, but **Geo Map** widget requires latitude and longitude pairs. **Geocoding** will help us extract this information from country names.

Connect **Geocoding** to **Datasets**. Region identifier in our case is the attribute *Country* and the identifier type is *Country name*. If our data contained major European cities, we would have to select this from the dropdown. On the right there is the *Unmatched identifier* editor, which shows those instances, for which **Geocoding** couldn't find corresponding latitude/longitude pairs. We can help the widget by providing a custom replacement. Click on the field and start typing *Korea*. The widget will suggest two countries, Democratic Republic of Korea and South Korea. Select the one you wish to use here.

Finally, we can observe the data in the second **Data Table**. We can see our data now has two additional attributes, one for the latitude and one for the longitude of the region of interest. Now, you can plot the data on the map!



1.3 Choropleth Map

A thematic map in which areas are shaded in proportion to the measurement of the statistical variable being displayed.

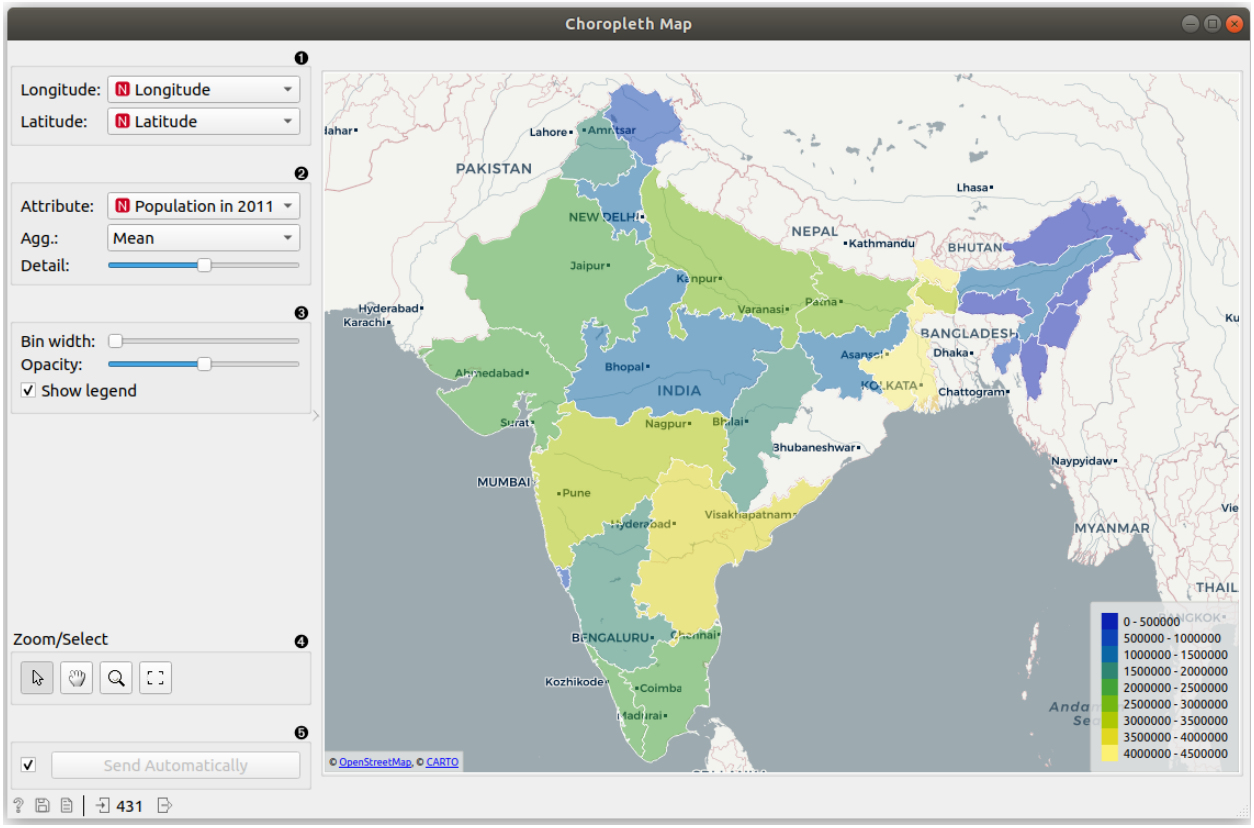
Inputs

- Data: input dataset

Outputs

- Selected Data: instances selected from the map.
- Data: data with an additional column showing whether a point is selected

Choropleth provides an easy way to visualize how a measurement varies across a geographic area or show the level of variability within a region. There are several levels of granularity available, from countries to states, counties, or municipalities.



1. Set latitude and longitude attributes, if the widget didn't recognize them automatically.
2. Set *Attribute* to color the region by. Set *Agg.* which by default counts the number of occurrences of the region in the data. *Count defined* shows which regions appear in the data. *Sum*, *Mean*, *Median*, *Maximal*, *Minimal* and *Std.* (standard deviation) work for numeric data, while *Mode* works for categorical. Set *Detail* level to countries, states (US)/counties/Bundesländer/provinces or counties (US)/municipalities.
3. Adjust plot properties:
   - *Bin width* for discretize displayed color.
   - *Opacity* sets transparency of regions.
   - *Show legend* displays a legend on the right. Click and drag the legend to move it.
4. *Select, zoom, pan and zoom to fit* are the options for exploring the map. The manual selection of data instances works as an angular/square selection tool. Scroll in or out for zoom.
5. If *Send automatically* is ticked, changes are communicated automatically. Alternatively, press *Send*.

1.3.1 Example

We will use *HDI* data from the **Datasets** widget. Open the widget, find *HDI* data and double click. **Choropleth** widget requires latitude and longitude pairs, so we will use **Geocoding** to extract this information. We used the attribute *Country* and found lat/lon pairs that **Choropleth** can use.

Choropleth will automatically look for attributes named *latitude*, *longitude*, *lat*, *lon* or similar. It will use them for plotting. Alternatively, set the attributes manually.

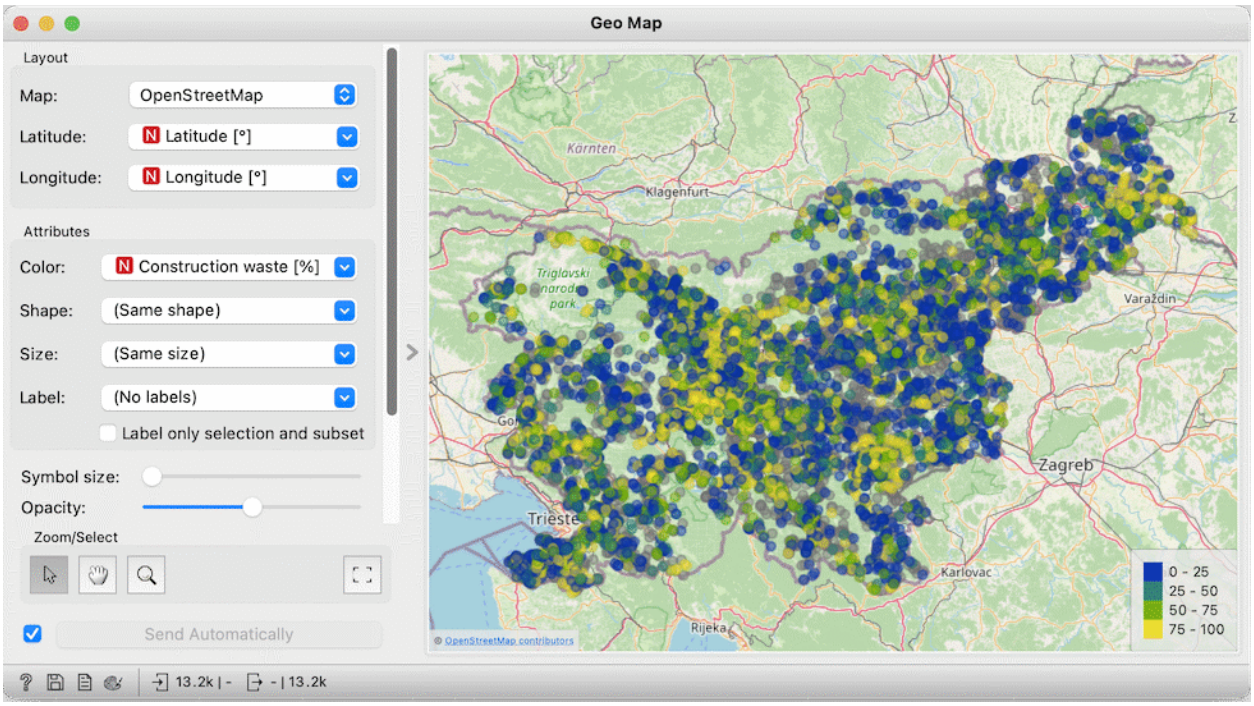
Since *HDI* attribute is our target variable, it will automatically be used for coloring. We change it in the *Attribute* dropdown to *Life expectancy*. We have set the level of aggregation to *Mean*, but since we have only one value per

country, we could use *Sum* or *Median* just as well.

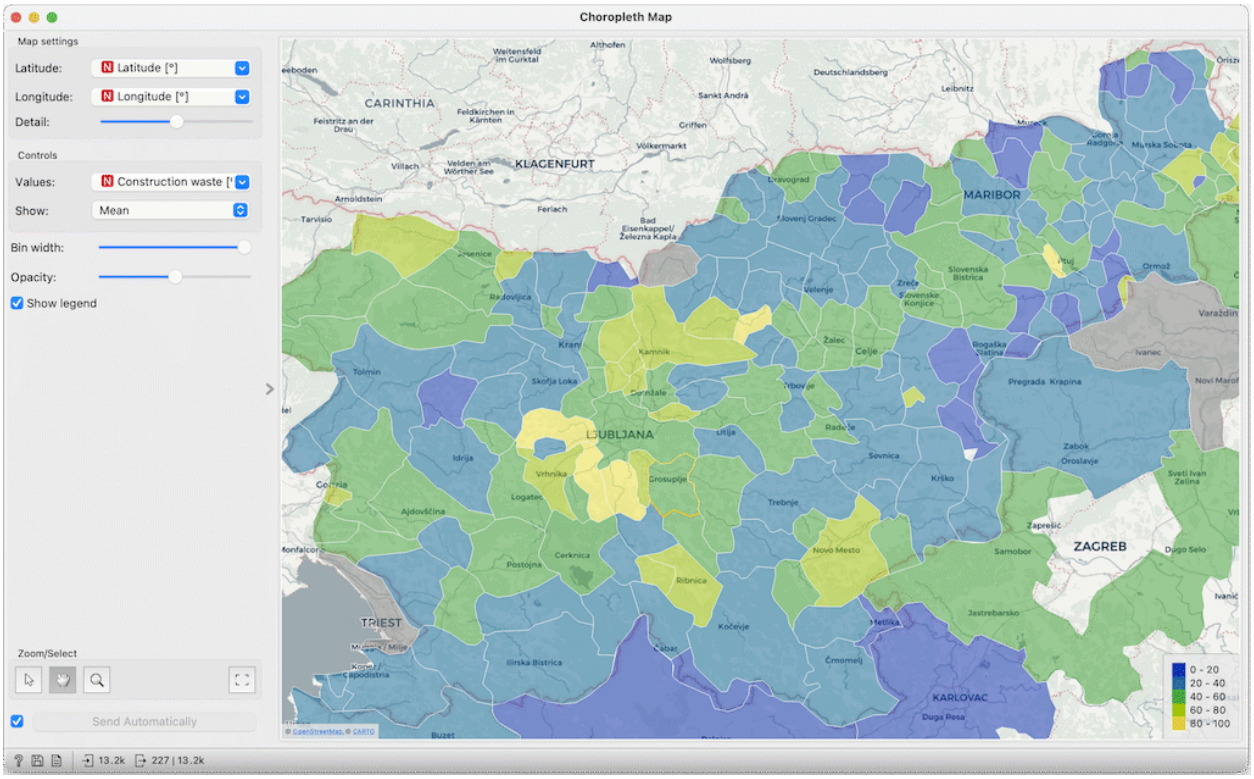
The widget shows life expectancy as reported by the United Nations per country. Yellow countries are those with a high Life expectancy and blue ones are the ones with a low life expectancy.



Choropleth can also aggregate data for points belonging to the same region. The data on *Illegal waste dumps in Slovenia* (available through the Datasets widget) contains coordinates of dumps sites and the composition of the waste. Suppose that we are interested in the proportion of construction waste. Shown in Geo Map and coloring points by that feature, the map looks like this:



Choropleth can provide a much better picture: we set the Detail to maximum, choose *Construction waste* and show its *mean*.



Dumps with the largest proportion of construction waste (or the lowest proportion of other types?) can be found in central Slovenia.

1.4 Geo Transform

Transform geographic coordinates from one system to another.

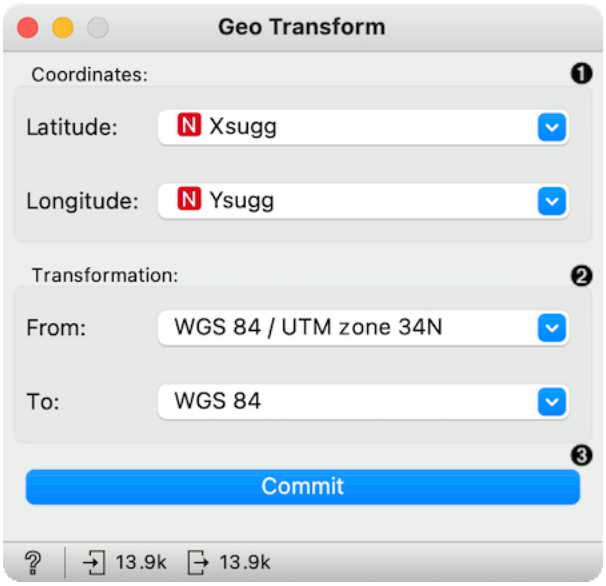
Inputs

- Data: input dataset

Outputs

- Data: data with transformed coordinates.

Geo Transform widget converts latitude and longitude data from one geodesic system to another. It uses pyproj library for the conversion.



1. Set latitude and longitude attributes, if the widget didn't recognize them automatically.
2. Geodesic systems used for transformation. By default, the output system is set to the latest revision of the World Geodesic System, WGS 84.
3. Press *Commit* to apply the transformation.

1.4.1 Example

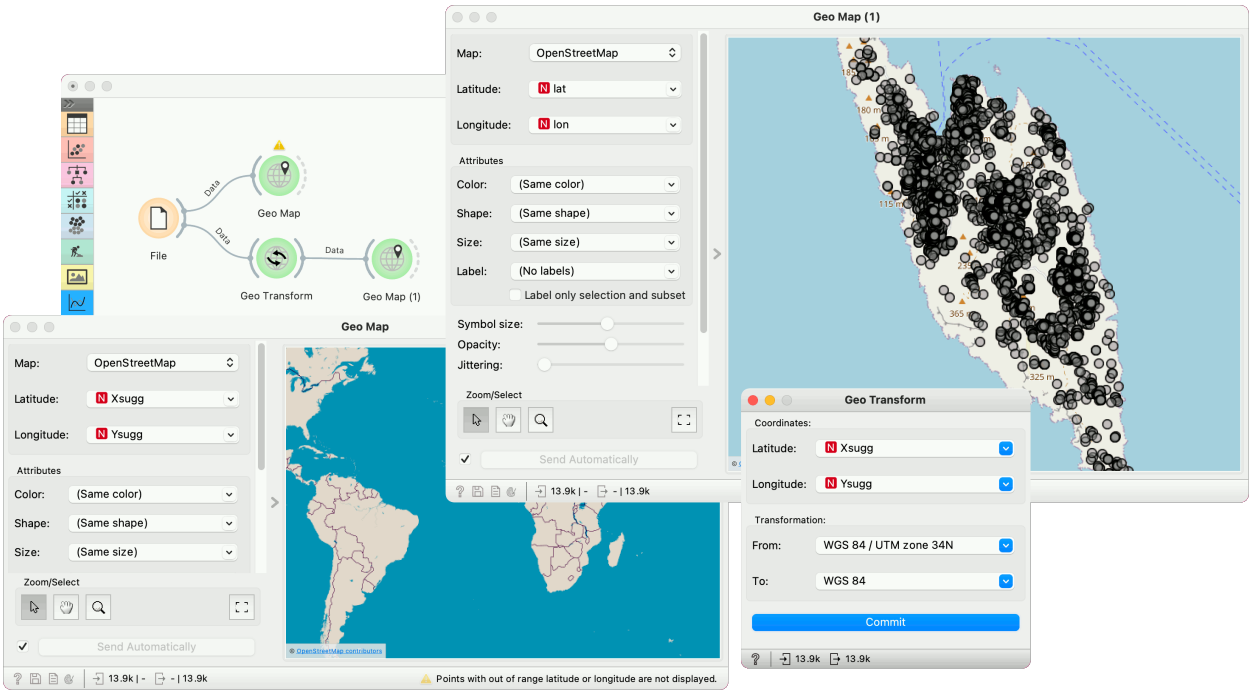
For the example, we will use the Antikythera data, an archaeological dataset describing the shards found on the Greek island Antikythera. The dataset can be found here. We will use the easy way and simply copy-paste the URL of the data into the URL line of the File widget.

https://archaeologydataservice.ac.uk/catalogue/adsdata/arch-1115-2/dissemination/csv/pottery/pottery.csv&hs=true

Latitude and longitude are encoded in the *Xsugg* and *Ysugg* variables. But if we plot these variables, we cannot see anything in the Geo Map. The widget raises a warning stating the points are outside the specified range for the map.

This is because the data is encoded in the WGS 84 / UTM zone 34N system (EPSG:32634). Geo Transform can convert the data from the original system to the standard WGS 84, which Geo Map is using. After we set the correct system for transformation, we press *Commit* to output the data.

In the second Geo Map, we can see the data is now plotted correctly. All the found shards are placed on the Antikythera island.


File
Data
Geo Map
Data
Geo Transform
Data
Geo Map (1)
Geo Map
Map: OpenStreetMap
Latitude: Xsugg
Longitude: Ysugg
Attributes
Color: (Same color)
Shape: (Same shape)
Size: (Same size)
Zoom/Select
Send Automatically
13.9k | -
- | 13.9k
Points with out of range latitude or longitude are not displayed.
Geo Map (1)
Map: OpenStreetMap
Latitude: lat
Longitude: lon
Attributes
Color: (Same color)
Shape: (Same shape)
Size: (Same size)
Label: (No labels)
Label only selection and subset
Symbol size:
Opacity:
Jittering:
Zoom/Select
Send Automatically
13.9k | -
- | 13.9k
Geo Transform
Coordinates:
Latitude: Xsugg
Longitude: Ysugg
Transformation:
From: WGS 84 / UTM zone 34N
To: WGS 84
Commit
13.9k
13.9k

CHAPTER 2

Indices and tables

- genindex