VizMap Manual

Felix Atsma

Repository at github.com/felixatsma/vizmap.

1 Installation

Create an Anaconda environment using the `environment.yml` file. If using JupyterLab, you need to install the JupyterLab extension:

```
jupyter labextension install @jupyter-widgets/jupyterlab-manager
jupyter labextension install jupyter-leaflet
```

2 Importing the package

All you need to visualize your data is the VizMap class. If you want to use basemaps other than the default, you can import `basemaps` along with it. A list of included basemaps can be found here [1].

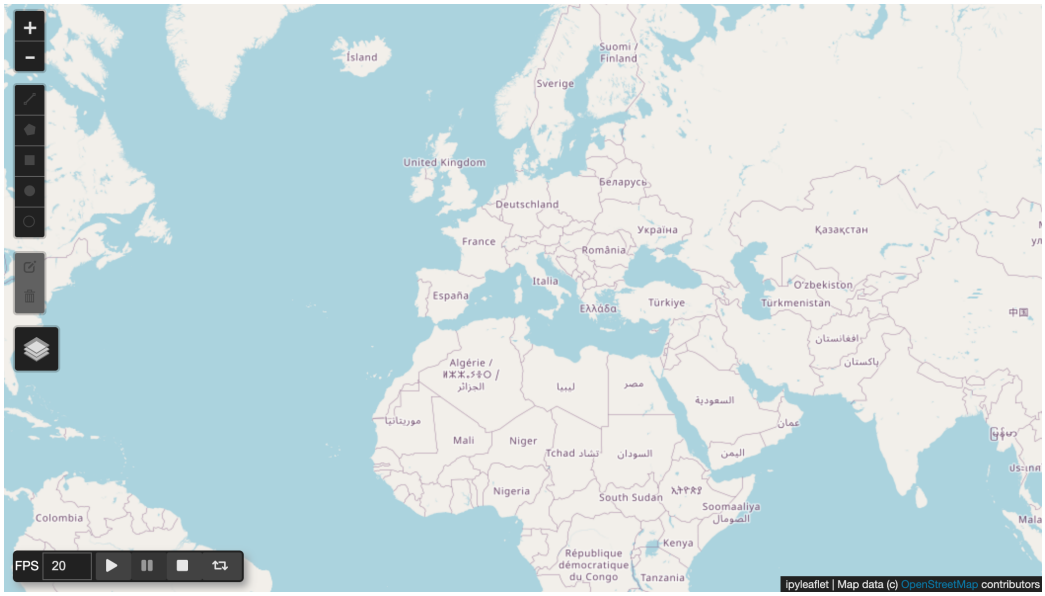
```
from vizmap import VizMap, basemaps
```

3 Creating a map

Creating a map is very simple. First create a VizMap object. The attributes of the map you can specify can be found here [2]. The layout properties of the map can be accessed with `m.map.layout`, with `m` being the map object. To display the map, you can use the map's `m.display()` method or Jupyter's `display(m)` function.

```
m = VizMap(
    basemap=basemaps.OpenStreetMap.Mapnik,
    center=(40,10),
    zoom=3
)
m.map.layout.height = '600px'
m.display()
```

[1] https://ipyleaflet.readthedocs.io/en/latest/api_reference/basemaps.html
[2] https://ipyleaflet.readthedocs.io/en/latest/api_reference/map.html

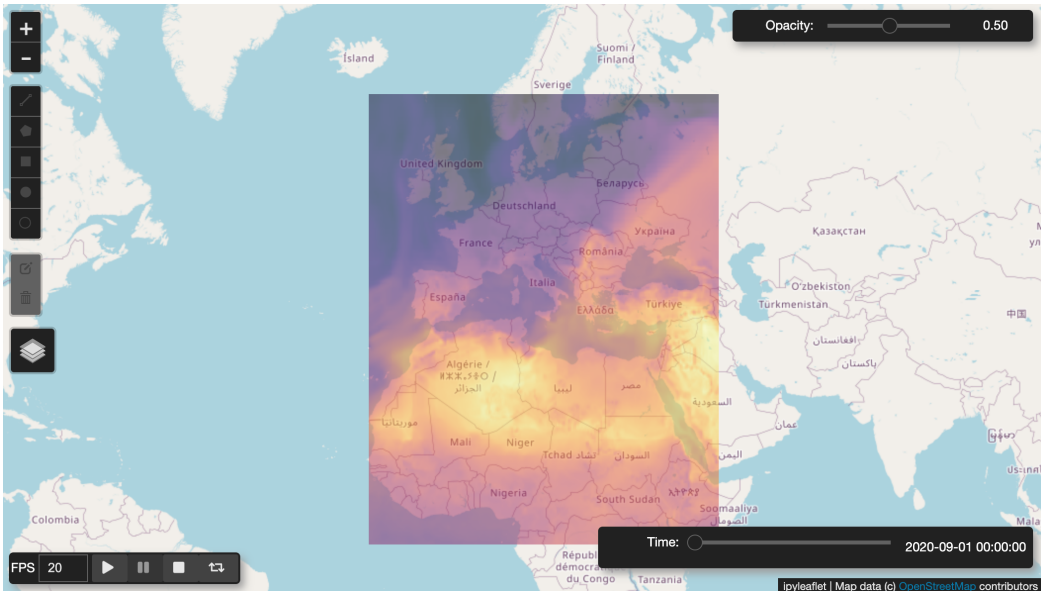


4 Adding a visualization layer

Next, to visualize your data you add a visualization layer. This is done using the `add_raster()` and `add_wind()` methods. `add_raster()` for regular single-variable raster data and `add_wind()` for two-variable windspeed data. When adding a layer, you need to provide the filename of your dataset and specify the variable and dimension names within the dataset. Only NetCDF files are supported at this moment. The raster visualization has an argument for the colormap used in the visualization. The colormaps from matplotlib are used. These can be found here [3].

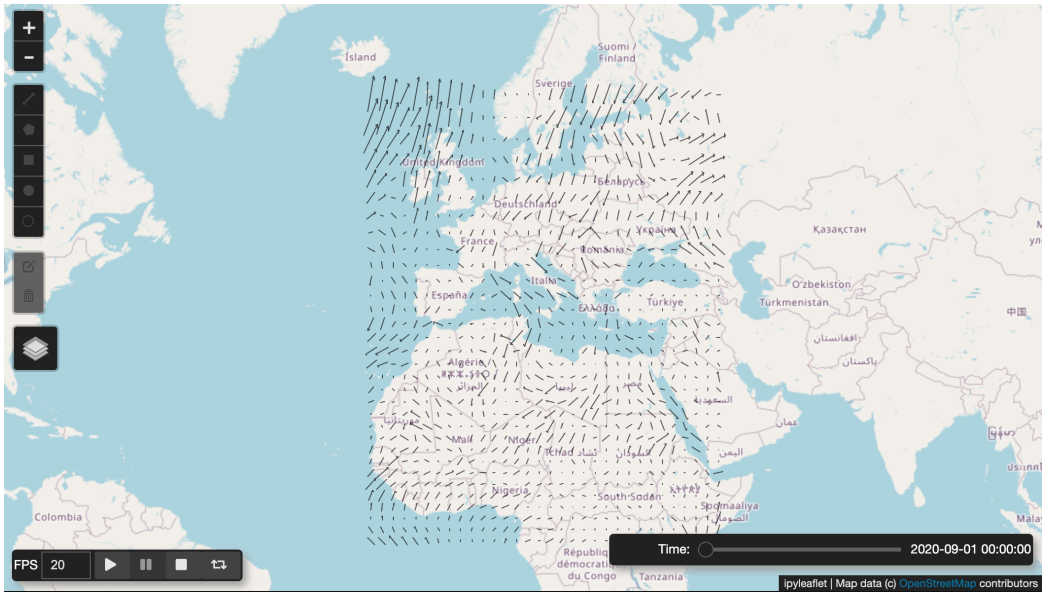
```
m.add_raster(
    'data/temperature.nc',
    data='t',
    time='time',
    lat='latitude', long='longitude',
    cmap='inferno'
)
```

[3] https://matplotlib.org/tutorials/colors/colormaps.html



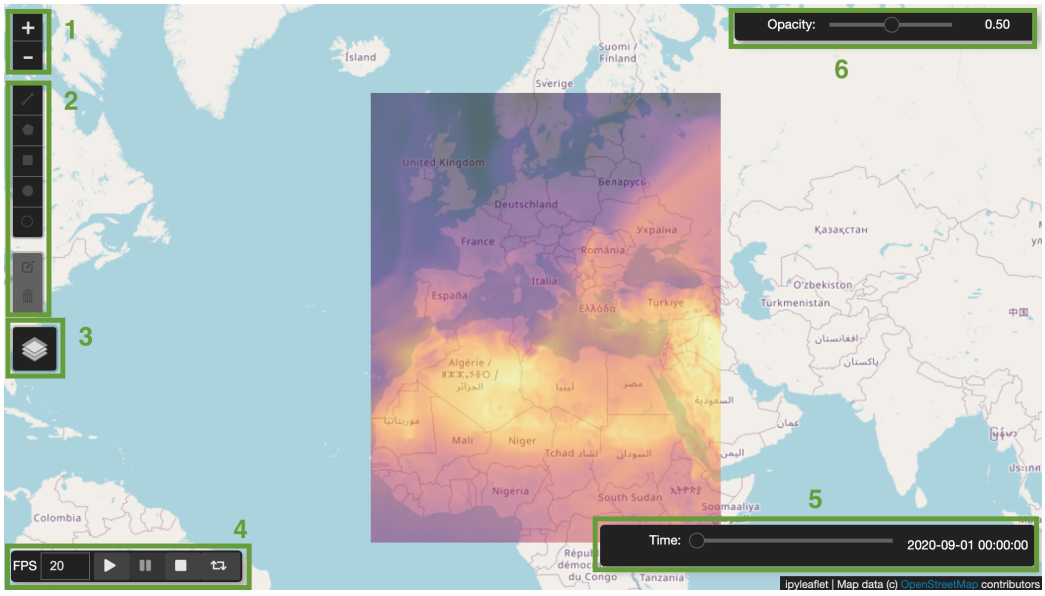
There are a few more optional arguments when adding a wind visualization. First, you need to specify the variable names of the eastward (`u`) and northward (`v`) components of the wind. There is a `stride` argument. Using a stride of n means reading every nth value from the array. Higher stride results in less arrows and higher performance. `autoscale` specifies if the arrows are sized according to their velocity, and `scale_value` specifies the amount the arrow is scaled by relative to it's velocity.

```
m.add_wind(
    'data/windspeed.nc',
    u='u', v='v',
    time='time',
    lat='latitude', long='longitude',
    stride=8,
    autoscale=True,
    scale_value=0.3,
)
```



5 Widgets

Several widgets can be used to control the visualization. As seen in the figure below, widget 1 are used to zoom in and out. 2 is the tool for creating selections. The black bar contains the selection shapes: polygon, rectangle and circles. Note the first option, a line, can not be used to create a selection. The grey bar has options to edit or delete selections. 3 can be used to hide and show layers. 4 is the animation control. The FPS of the animation can be specified. Note that the animation speed is only updated after the text box loses focus. 5 is the frame slider. This displays the time of the current frame and allows you to manually scrub through frames. 6 is used to change the opacity of a layer.



6 Selections

A selection is created by drawing a shape using the selection widget. Once a selection is drawn, the data contained can be retrieved using the `get_selection` method.

```
m.get_selection(selection, layer)
```

`selection` is the index, or indices, of the selection(s) you want to retrieve. The index of a selection is determined by the order it was created. `layer` is the index of the layer you want to get the selection from, default is 0. The index of a layer can be seen in the layer visibility widget. The `get_selection` method returns a Numpy array containing the frame with all non-selected values masked.