# TissUUmaps

***Release 3.0***

**Nicolas Pielawski** **Axel Andersson**
**Christophe Avenel** **Andrea Behanova**
**Eduard Chelebian** **Anna Klemm** **Fredrik Nysjö**
**Leslie Solorzano** **Carolina Wählby**

**Apr 15, 2022**

# CONTENTS:

This page hosts the documentation for TissUUmaps 3.0. You can find a pdf version of thie documentation here.

For more information on the TissUUmaps project, including video tutorials and demos, visit our website: https://tissuumaps.github.io.

**Work in progress!**

This page is mostly empty for now. We are working actively on writing this documentation, more content will be available soon!

CHAPTER

ONE

# INTRODUCTION

## 1.1 About TissUUmaps

TissUUmaps is a free and open source browser-based tool for GPU-accelerated visualization and interactive exploration of tens of millions of datapoints overlaying tissue samples. Users can visualize markers and regions, explore spatial statistics and quantitative analyses of tissue morphology, and assess the quality of decoding in situ transcriptomics data. TissUUmaps provides instant multi-resolution image viewing, can be customized, shared, and also integrated in Jupyter Notebooks. We envision TissUUmaps to contribute to broader dissemination and flexible sharing of large-scale spatial omics data.

Currently, microscopy data can be cumbersome to share: physically transferring the images is often necessary and dedicated software must be installed. Instead, researchers can now share their findings with a simple link to a website running TissUUmaps. The images are loaded in real time, together with annotations, markers, and masks that may also be modified by the user. We also provide tools for quality control and image processing. The software is designed to display and interact with images at multiple resolutions and large numbers of markers, especially data from spatially resolved omics techniques and tissue atlases. TissUUmaps is compatible with many different bioimage informatics tools, and provides new ways to develop insights when exploring and sharing data.

You can access the TissUUmaps project gallery with interactive examples to explore data from in situ sequencing and spatial transcriptomics experiments and view localized quantification of cell and tissue morphology, including links to publications. For seeing examples of TissUUmaps compatibility with other platforms you can access the tutorials page.

## 1.2 Installation

TissUUmaps is a browser-based tool for fast visualization and exploration of millions of data points overlaying a tissue sample. TissUUmaps can be used as a web service or locally in your computer, and allows users to share regions of interest and local statistics.

### 1.2.1 Windows installation

1. Download the Windows Installer from the last release and install it. Note that the installer is not signed yet and may trigger warnings from the browser and from the firewall. You can safely pass these warnings.

### 1.2.2 PIP installation (for Linux and Mac)

1. Install `libvips` for your system: https://www.libvips.org/install.html

   An easy way to install `libvips` is to use an Anaconda environment with `libvips`:

```
conda create -y -n tissuumaps_env -c conda-forge python=3.9 libvips
conda activate tissuumaps_env
```

2. Install the TissUUmaps library using `pip`:

```
pip install "TissUUmaps[full]"
```

3. Start the TissUUmaps user interface:

```
tissuumaps
```

4. Or start TissUUmaps as a local server:

```
tissuumaps_server path_to_your_images
```

   And open http://127.0.0.1:5000/ in your favorite browser.

## 1.3 Citing TissUUmaps

Please cite our preprint on bioRxiv if using TissUUmaps in your work:

**TissUUmaps 3: Interactive visualization and quality assessment of large-scale spatial omics data.** *Nicolas Pielawski, Axel Andersson, Christophe Avenel, Andrea Behanova, Eduard Chelebian, Anna Klemm, Fredrik Nysjö, Leslie Solorzano, Carolina Wählby,* bioRxiv 2022.01.28.478131; doi: https://doi.org/10.1101/2022.01.28.478131.

## 1.4 Changelog

### 1.4.1 3.0.8.5

- Minor fixes.

### 1.4.2 3.0.8.4

- Add tiling to viewport capture for higher resolution output
- Increase resolution of markers on high resolution devices
- Fix jumps on pan with mouse gesture (mobile)
- Add fix for bright image canvas on Safari
- Add an option to remove markers' outlines.

### 1.4.3 3.0.8.3

- Fix png artifact in Firefox, by generating jpg tiles.

### 1.4.4 3.0.8.2

- Add high resolution capture of viewport, up to 4096x4096 pixels.

### 1.4.5 3.0.8.1

- Fix multiple dataset alignment when no background image

### 1.4.6 3.0.8

- Fix black images generated by VIPS
- Fix Linux and Mac open of captures
- Auto save datasets as buttons when saving tmap projects
- Add `mpp` (microns per pixel) option in tmap files, to add scale bar to viewer
- Make region line thickness depend on zoom level
- Add compatibility with JupyterLab
- Add opacity per marker option

### 1.4.7 3.0.7

- Add menu to load plugins through an update-site

### 1.4.8 3.0.6

- Fix multiple plugins opening always last plugin
- Move to OpenSeadragon 3.0.0
- Add tooltip format in Advanced Settings
- Add drag and drop to open CSV files and images
- Add “Add layer” button for flask version
- Add viewport capture

### 1.4.9 3.0.5

- Move csv loading to Papa Parse streaming, to allow better memory management

### 1.4.10 3.0.4

- Add filtering of markers

### 1.4.11 3.0

- Add tissuumaps.jupyter module

CHAPTER

TWO

# GETTING STARTED

## 2.1 Images

### 2.1.1 Supported image formats

TissUUmaps can read whole slide images in any format recognized by the OpenSlide library:

- Aperio (.svs, .tif)
- Hamamatsu (.ndpi, .vms, .vmu)
- Leica (.scn)
- MIRAX (.mrxs)
- Philips (.tiff)
- Sakura (.svslide)
- Trestle (.tif)
- Ventana (.bif, .tif)
- Generic tiled TIFF (.tif)

TissUUmaps will automatically convert any other format into a pyramidal tiff (in a temporary `.tissuumaps` folder created in the original image folder) using vips.

If your image fails to open, try converting it to `tif` format using an external tool.

### 2.1.2 Load images

### 2.1.3 Apply filters

## 2.2 Markers

### 2.2.1 Supported marker format

TissUUmaps can read CSV (Comma Separated Values) files with a header row, and at least spatial coordinate columns (X and Y). CSV files are not limited in the number of columns or number of rows. Other columns can contain information for displaying markers (key to group markers, color, size, shape, piecharts, etc.)

CSV files can be exported from any spreadsheet program, or any programming language (Python, R, etc.)

### 2.2.2 Load markers

### 2.2.3 Markers settings

#### File and coordinates

#### Render options

#### Advanced options

#### Table of markers

## 2.3 Regions

### 2.3.1 Supported region formats

TissUUmaps can read and write region files in the GeoJSON format.

Only a subset of the GeoJSON format is supported, as TissUUmaps uses only polygonal regions:

**Main types**:

- Feature
- FeatureCollection
- GeometryCollection

**Geometries**:

- Polygon
- Multipolygon

The coordinate system must be the same as the image and marker coordinate systems.

### 2.3.2 Draw Regions

### 2.3.3 Analyze Regions

### 2.3.4 Load Regions

### 2.3.5 Export Regions

## 2.4 Projects

### 2.4.1 Saving and loading projects

### 2.4.2 The tmap file format

The tmap format contains image layers, saved markers, regions, and settings. It is highly recommended to create tmap files by saving projects from TissUUmaps applications, but you can also edit the files manually to add or change project's settings.

The tmap format uses json, with the following specifications:

### Tmap specifications

| | | | |
|---|---|---|---|
| TODO. Required properties are shown in **bold** text | | | |
| type | *object* | | |
| properties | | | |
| • **filename** | Name of the project | | |
| | type | *string* | |
| • layers | type | *array* | |
| | default | [] | |
| | items | | |
| | • | *Layer* | |
| • layerOpacities | type | *object* | |
| | patternProperties | | |
| | • ^[0-9]+$ | type | *integer* |
| • layerVisibilities | type | *object* | |
| | patternProperties | | |
| | • ^[0-9]+$ | type | *boolean* |
| • layerFilters | type | *object* | |
| | patternProperties | | |
| | • ^[0-9]+$ | *LayerFilter* | |
| • filters | type | *array* | |
| | default | [] | |
| | items | | |
| | • | *Filter* | |
| • compositeMode | type | *string* | |
| • markerFiles | type | *array* | |
| | default | [] | |
| | items | | |
| | • | *MarkerFile* | |
| • regions | GeoJSON object, see *Regions section*. | | |
| | type | *object* | |
| | default | | |
| • regionFile | type | *string* | |
| | default | | |
| • regionFiles | type | *array* | |

continues on next page

Table 1 – continued from previous page

| | | | |
|---|---|---|---|
| | default | [] | |
| | items | | |
| • plugins | type | *array* | |
| | default | [] | |
| | items | | |
| | • | type | *string* |
| • hideTabs | Hide tabs of markers dataset. Only use when you have a unique marker tab. | | |
| | type | *boolean* | |
| | default | false | |
| • settings | type | *array* | |
| | default | [] | |
| | items | | |
| | • | *Setting* | |

### Layer

| | | |
|---|---|---|
| TODO. Required properties are shown in **bold** text | | |
| type | *object* | |
| properties | | |
| • **name** | type | *string* |
| • **tileSource** | type | *string* |

### LayerFilter

| | | | |
|---|---|---|---|
| TODO. Required properties are shown in **bold** text | | | |
| type | *array* | | |
| items | | | |
| • | type | *object* | |
| | properties | | |
| | • **name** | *Filter* | |
| | • **value** | type | *string* |

### Filter

| | |
|---|---|
| enum | Color, Brightness, Exposure, Hue, Contrast, Vibrance, Noise, Saturation, Gamma, Invert, Greyscale, Threshold, Erosion, Dilation |

### ColorScale

| TissUUmaps supports most of the color scales available in the D3.js library. See https://github.com/d3/d3-scale-chromatic for reference. Note: the colors for 'interpolateRainbow' is currently overridden by a custom Turbo-like color scale in version 3.0.x of TissUUmaps. | |
|---|---|
| enum | interpolateCubehelixDefault, interpolateRainbow, interpolateWarm, interpolateCool, interpolateViridis, interpolateMagma, interpolateInferno, interpolatePlasma, interpolateBlues, interpolateBrBG, interpolateBuGn, interpolateBuPu, interpolateCividis, interpolateGnBu, interpolateGreens, interpolateGreys, interpolateOrRd, interpolateOranges, interpolatePRGn, interpolatePiYG, interpolatePuBu, interpolatePuBuGn, interpolatePuOr, interpolatePuRd, interpolatePurples, interpolateRdBu, interpolateRdGy, interpolateRdPu, interpolateRdYlBu, interpolateRdYlGn, interpolateReds, interpolateSinebow, interpolateSpectral, interpolateTurbo, interpolateYlGn, interpolateYlGnBu, interpolateYlOrBr, interpolateYlOrRd |

### Shape

| TissUUmaps supports most of the marker shapes that are also used by the Napari software, https://napari.org. In addition to the name strings listed below, shape can also be specified by a corresponding index in range 0-13. | |
|---|---|
| enum | cross, diamond, square, triangle up, star, clobber, disc, hbar, vbar, tailed arrow, triangle down, ring, x, arrow |

### MarkerFile

| | | |
|---|---|---|
| Description of settings and GUI objects for a marker dataset loaded from CSV file. Required properties are shown in **bold** text. | | |
| type | *object* | |
| properties | | |
| • **title** | Name of marker button | |
| | type | *string* |
| • comment | Optional description text shown next to marker button | |
| | type | *string* |
| | default | |
| • **name** | Name of marker tab | |
| | type | *string* |
| • autoLoad | If the CSV file for the marker dataset should be automatically loaded when the TMAP project is opened. If this is false, the user instead has to click on the marker button in the GUI to load the dataset. | |
| | type | *boolean* |
| | default | false |
| • **uid** | A unique identifier used internally by TissUUmaps to reference the marker dataset | |
| | type | *string* |
| • **expectedHeader** | *ExpectedHeader* | |
| • **expectedRadios** | *ExpectedRadios* | |
| • **path** | Relative file path to CSV file in which marker data is stored | |
| | type | *string* |
| • settings | type | *array* |
| | default | [] |
| | items | |
| | • | *Setting* |

### ExpectedHeader

| | | |
|---|---|---|
| Input field values for settings in a marker tab. Required properties are shown in **bold** text. | | |
| type | *object* | |
| properties | | |
| • **X** | Name of CSV column to use as X-coordinate | |
| | type | *string* |
| • **Y** | Name of CSV column to use as Y-coordinate | |
| | type | *string* |
| • gb_col | Name of CSV column to use as key to group markers by | |
| | type | *string* |
| | default | null |
| • gb_name | Name of CSV column to display for groups instead of group key value | |
| | type | *string* |
| | default | null |

continues on next page

Table 2 – continued from previous page

| | | |
|---|---|---|
| • cb_cmap | Name of D3 color scale to be used for color mapping. See *ColorScale* for valid string values. | |
| | type | *string* |
| | default | |
| • cb_col | Name of CSV column containing scalar values for color mapping or hexadecimal RGB colors in format '#ff0000' | |
| | type | *string* |
| | default | null |
| • cb_gr_dict | JSON string specifying a custom dictionary for mapping group keys to group colors. Example: '{"key1": "#ff0000", "key2": "#00ff00", "key3": "#0000ff"}' | |
| | type | *string* |
| | default | |
| • scale_col | Name of CSV column containing scalar values for changing the size of markers | |
| | type | *string* |
| | default | null |
| • scale_factor | Numerical value for a fixed scale factor to be applied to markers | |
| | type | *string* |
| | default | 1 |
| • pie_col | Name of CSV column containing data for pie chart sectors | |
| | type | *string* |
| | default | null |
| • pie_dict | TODO | |
| | type | *string* |
| | default | |
| • shape_col | Name of CSV column containing a name or an index for marker shape. See also *Shape*. | |
| | type | *string* |
| | default | null |
| • shape_fixed | Name or index of a single fixed shape to be used for all markers. See *Shape* for valid string values. | |
| | type | *string* |
| | default | cross |
| • shape_gr_dict | JSON string specifying a custom dictionary for mapping group keys to group shapes. Example: '{"key1": "square", "key2": "diamond", "key3": "triangle up"}'. See also *Shape*. | |
| | type | *string* |
| | default | |
| • opacity_col | Name of CSV column containing scalar values for opacities | |
| | type | *string* |
| | default | null |
| • opacity | Numerical value for a fixed opacity factor to be applied to markers | |
| | type | *string* |
| | default | 1 |
| • tooltip_fmt | Custom formatting string used for overlay displayed over selected markers; see (TODO). | |
| | type | *string* |
| | default | |

### ExpectedRadios

| Radio button state and checkbox state for settings in a marker tab. Required properties are shown in **bold** text. | | |
|---|---|---|
| type | *object* | |
| properties | | |
| • cb_col | If markers should be colored by data in CSV column | |
| | type | *boolean* |
| | default | false |
| • cb_gr | If markers should be colored by group | |
| | type | *boolean* |
| | default | true |
| • cb_gr_rand | If group color should be generated randomly | |
| | type | *boolean* |
| | default | false |
| • cb_gr_dict | If group color should be read from custom dictionary | |
| | type | *boolean* |
| | default | false |
| • cb_gr_key | If group color should be generated from group key | |
| | type | *boolean* |
| | default | true |
| • pie_check | If markers should be rendered as pie charts | |
| | type | *boolean* |
| | default | false |
| • scale_check | If markers should be scaled by data in CSV column | |
| | type | *boolean* |
| | default | false |
| • shape_col | If markers should get their shape from data in CSV column | |
| | type | *boolean* |
| | default | false |
| • shape_gr | If markers should get their shape from group | |
| | type | *boolean* |
| | default | true |
| • shape_gr_rand | If group shape should be generated randomly | |
| | type | *boolean* |
| | default | true |
| • shape_gr_dict | If group shape should be read from custom dictionary | |
| | type | *boolean* |
| | default | false |
| • shape_fixed | If a single fixed shape should be used for all markers | |
| | type | *boolean* |
| | default | false |
| • opacity_check | If markers should get their opacities from data in CSV column | |
| | type | *boolean* |
| | default | false |
| • _no_outline | If marker shapes should be rendered without outline | |
| | type | *boolean* |
| | default | false |

### Setting

| TODO. Required properties are shown in **bold** text. | | |
|---|---|---|
| type | *object* | |
| properties | | |
| • **function** | type | *string* |
| • **module** | type | *string* |
| • **value** | type | *number* |

### Example of tmap file

```
{
    "filename": "TissUUmaps_Example.tmap",
    "layers": [
        {
            "name": "Round1_A.tif",
            "tileSource": "images/Round1_A.tif.dzi"
        },
        {
            "name": "Round1_C.tif",
            "tileSource": "images/Round1_C.tif.dzi"
        }
    ],
    "layerOpacities": {
        "0": "1",
        "1": "1"
    },
    "layerVisibilities": {
        "0": true,
        "1": false,
    },
    "layerFilters": {
        "0": [
            {
                "name": "Color",
                "value": "0,100,0"
            }
        ],
        "1": [
            {
                "name": "Color",
                "value": "0,100,0"
            }
        ]
    },
    "filters": [
```

(continues on next page)

(continued from previous page)


```
        "Color"
    ],
    "compositeMode": "lighter",
    "markerFiles": [
        {
            "autoLoad": false,
            "comment": "",
            "expectedHeader": {
                "X": "global_x",
                "Y": "global_y",
                "cb_cmap": "",
                "cb_col": "null",
                "cb_gr_dict": "",
                "gb_col": "Gene",
                "gb_name": "",
                "opacity": "1",
                "opacity_col": "null",
                "pie_col": "null",
                "pie_dict": "",
                "scale_col": "null",
                "scale_factor": "0.5",
                "shape_col": "null",
                "shape_fixed": "cross",
                "shape_gr_dict": "",
                "tooltip_fmt": ""
            },
            "expectedRadios": {
                "cb_col": false,
                "cb_gr": true,
                "cb_gr_dict": false,
                "cb_gr_key": true,
                "cb_gr_rand": false,
                "pie_check": false,
                "scale_check": false,
                "shape_col": false,
                "shape_fixed": false,
                "shape_gr": true,
                "shape_gr_dict": false,
                "shape_gr_rand": true,
                "opacity_check": false
            },
            "name": " markers",
            "path": "./istdeco_codes_n.csv",
            "title": "Download markers",
            "uid": "uniquetab"
        }
    ],
    "regions": {},
    "plugins": [
        "Spot_Inspector"
    ],
    "hideTabs": true,
```


(continues on next page)

(continued from previous page)

```
    "settings": []
}
```

## 2.5 Exporting screenshots

TissUUmaps allows high resolution capture of the image viewport. Go to `Menu > File > Capture viewport` and chose a zoom factor for export (1 = screen resolution).

The screen capture will contain all filtered layers, markers, and regions. Note that legends will not be part of the export and must be added manually.

## 2.6 Plugins

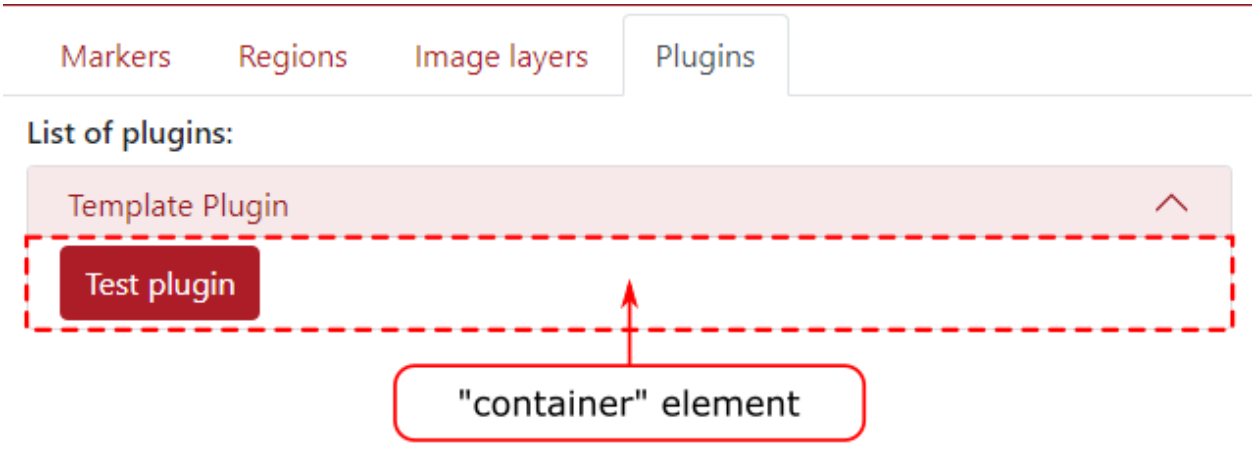
### 2.6.1 Load plugins

### 2.6.2 Make your own plugin

Download the Plugin Template python and javascript files from the Plugin Update Site and put both files in your local folder `$USER_PATH/.tissuumaps/plugins/`. You can then change the plugin name and add your own options and functions.

#### Javascript file

When loading a plugin, the function `PluginName.init(container)` will be called. The `container` is an html Element that will be added to the plugin menu. Use this element to add options and texts related to your plugin.




Here is a minimal example of plugin:

```
var Plugin_template;
Plugin_template = {
    name:"Template Plugin"
}
```

(continues on next page)

(continued from previous page)

```
/**
 * This method is called when the document is loaded.
 * The container element is a div where the plugin options will be displayed. */
Plugin_template.init = function (container) {
    container.innerHTML = "Hello world";
}
```

You can access the TissUUmaps javascript API here.

### Python file

You only need to use the Python file if your plugin needs to do processing on the server side. For pure javascript plugins, you can leave this file empty.

The python file should implement the class `Plugin`:

```
class Plugin ():
    def __init__(self, app):
        self.app = app
```

The `app` object being the flask application running the TissUUmaps server.

You can call a Python method inside the `Plugin` class from Javascript using Ajax and the Python API. The endpoint for a method `methodName` of the plugin `PluginName` will be: `/plugins/methodName/functionName`. Data can be transmitted through Ajax as stringified JSON, and will be available as a parameter inside the method.

See the Plugin Template for a working example of Javascript / Python communication.

CHAPTER

THREE

# SHARING PROJECTS

## 3.1 Apache server

TissUUmaps projects can be exported into static webpages, that can be uploaded to any Apache server.

1. Save your project from TissUUmaps (`menu > File > Save project`)
2. Export to static page (`menu > File > Export to static webpage`)
3. Copy the exported folder on your Apache server

## 3.2 Docker container

1. Start the docker container `cavenel/tissuumaps:latest` from Docker Hub:

```
docker run -it -p 56733:80 --name=tissuumaps -v /path/to/local/images:/mnt/data cavenel/
↪tissuumaps:latest
```

1. Place your images in the local folder `/path/to/local/images/share`.
2. Open http://127.0.0.1:56733/ in your favorite browser.

CHAPTER

FOUR

# ADVANCED USAGE

## 4.1 Jupyter notebooks

TissUUmaps can easily be used inside a Jupyter Notebook or Jupyter Lab.

Simple example to load an image in TissUUmaps:

```
import tissuumaps.jupyter as tj
viewer = tj.loaddata(["image.png"])

viewer.screenshot()
```

### 4.1.1 tissuumaps.jupyter

Module used to run TissUUmaps from a Jupyter Notebook or from Jupyter Lab.

tissuumaps.jupyter.**opentmap**(*path*, *port=5100*, *host='localhost'*, *height=700*)

Open a tmap project

**Parameters**

- **path** (*str*) – The path to a tmap file
- **port** (*int*) – The port to run the TissUUmaps server
- **host** (*str*) – The host to run the TissUUmaps server
- **height** (*int*) – The height of the jupyter iframe

**Returns** The TissUUmaps viewer

**Return type** *TissUUmapsViewer*

tissuumaps.jupyter.**loaddata**(*images=[]*, *csvFiles=[]*, *xSelector='x'*, *ySelector='y'*, *keySelector=None*, *nameSelector=None*, *colorSelector=None*, *piechartSelector=None*, *shapeSelector=None*, *scaleSelector=None*, *fixedShape=None*, *scaleFactor=1*, *colormap=None*, *compositeMode='source-over'*, *boundingBox=None*, *port=5100*, *host='localhost'*, *height=700*, *tmapFilename='_project'*, *plugins=[]*)

Load data in TissUUmaps

**Parameters**

- **images** (*list | str*) – List of images or single image to display
- **csvFiles** (list |str) – List of csv files or single csv file to display

- **xSelector** (*str*) – Name of the csv column defining the X coordinates
- **ySelector** (*str*) – Name of the csv column defining the Y coordinates
- **keySelector** (*str*) – Name of the csv column defining the grouping key
- **nameSelector** (*str*) – Name of the csv column defining the group name
- **colorSelector** (*str*) – Name of the csv column defining the group color
- **piechartSelector** (*str*) – Name of the csv column defining pie-charts
- **shapeSelector** (*str*) – Name of the csv column defining markers' shape
- **scaleSelector** (*str*) – Name of the csv column defining markers' scale
- **fixedShape** (*int*) – Name of the markers' shape
- **scaleFactor** (*int*) – Global scale of markers
- **colormap** (*str*) – Name of the colormap used if colorSelector is set
- **compositeMode** – (str): Composite mode used for images
- **boundingBox** (*list*) – [X,Y,W,H] of the bounding box to display
- **port** (*int*) – The port to run the TissUUmaps server
- **host** (*str*) – The host to run the TissUUmaps server
- **height** (*int*) – The height of the jupyter iframe
- **tmapFilename** (*str*) – Name of the project file that will be created
- **plugins** (*list*) – List of plugins to add to the tmap project

**Returns** The TissUUmaps viewer

**Return type** *TissUUmapsViewer*

**class** tissuumaps.jupyter.**TissUUmapsViewer**(*server*, *image*, *height=700*)

Class representing a TissUUmaps viewer instance

**screenshot**()

Capture the TissUUmaps viewport and display image in the Notebook.

**class** tissuumaps.jupyter.**TissUUmapsServer**(*slideDir*, *port=5000*, *host='0.0.0.0'*)

Class representing a TissUUmaps server instance

## 4.2 Napari

Napari features an important hub containing 118 plugins at the time of writing, many of them expanding further the capabilities of Napari when dealing with biomedical imaging. We thus created our own plugin to allow users to work in Napari, benefit from the tools, scripting and existing plugins, and easily visualize and share the output of their research through TissUUmaps.

The Napari-TissUUmaps plugin is available on Napari Hub which makes the installation trivial: from the Napari install/uninstall plugins menu, the `napari-tissuumaps` appears in the list and can be installed with a single click. Alternatively, the plugin can be installed with the Python package manager: `pip install napari-tissuumaps`.

The plugin can export all standard Napari layers, such as images, labels, points, and shapes and preserves the metadata (opacity, visibility), but also the objects parameters (e.g.: label colors, marker colors and symbols, etc…). To export a TissUUmaps project, care must be taken to save all layers of interest and type in a name with the extension `.tmap`, e.g.: `myProject.tmap`. This is important for Napari to delegate the saving of the files to the plugin. A folder is created

and contains all the necessary files and can be loaded in the TissUUmaps server, software, Jupyter Notebook, or shared with the community.

The project folders generated by the plugin contain the metadata in a `main.tmap` file, along with folders for each Napari layer types: images, labels, points and regions. Images and labels are saved as plain tif images, points are saved as CSV files, and shapes are saved as GeoJSON. We hope that the use of a simple structure and widespread file formats can simplify the modifying and updating of the TissUUmaps project when prototyping with e.g. Jupyter Notebooks. The source code is available at https://github.com/TissUUmaps/napari-tissuumaps under the permissive MIT license. A demonstration of the Cellpose plugin of Napari being exported to the TissUUmaps web viewer is available at: https://tissuumaps.github.io/tutorials/#napari.

## 4.3 AnnData

Work in progress

# INDEX