TissUUmaps

Release 3.0

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

Apr 13, 2022

CONTENTS:

This page hosts the documentation for TissUUmaps 3.0.

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/10.1101/2022.01.28.47813.

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

type	*object*		
properties			
• filename	Name of the project		
	type	*string*	
• layers	type	*array*	
	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*	
	items		
	•	*Filter*	
• compositeMode	type	*string*	
• markerFiles	type	*array*	
	items		
	•	*MarkerFile*	
• regions	GeoJSON object, see *Regions section*.		
	type	*object*	
• regionFile	type	*string*	
• regionFiles	type	*array*	
	items		
• plugins	type	*array*	
	items		

continues on next page

Table 1 – continued from previous page

	•	type	*string*
• hideTabs	Hide tabs of markers dataset. Only use when you have a unique marker tab.		
	type	*boolean*	
• settings	type	*array*	
	items		
	•	*Setting*	

Layer

type	*object*	
properties		
• **name**	type	*string*
• **tileSource**	type	*string*

LayerFilter

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

MarkerFile

A button linked to a marker dataset		
type	*object*	
properties		
• **title**	type	*string*
• comment	type	*string*
• **name**	type	*string*
• autoLoad	type	*boolean*
• **uid**	type	*string*
• **expectedHeader**	List of input text options	
	type	*object*
• **expectedRadios**	List of radio options	
	type	*object*
• **path**	type	*string*
• settings	type	*array*
	items	
	•	*Setting*

Setting

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": [
        "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",
```


(continues on next page)

(continued from previous page)


```
                "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
            },
            "name": " markers",
            "path": "./istdeco_codes_n.csv",
            "title": "Download markers",
            "uid": "uniquetab"
        }
    ],
    "regions": {},
    "plugins": [
        "Spot_Inspector"
    ],
    "hideTabs": true,
    "settings": []
}
```


2.5 Exporting screenshots

2.6 Plugins

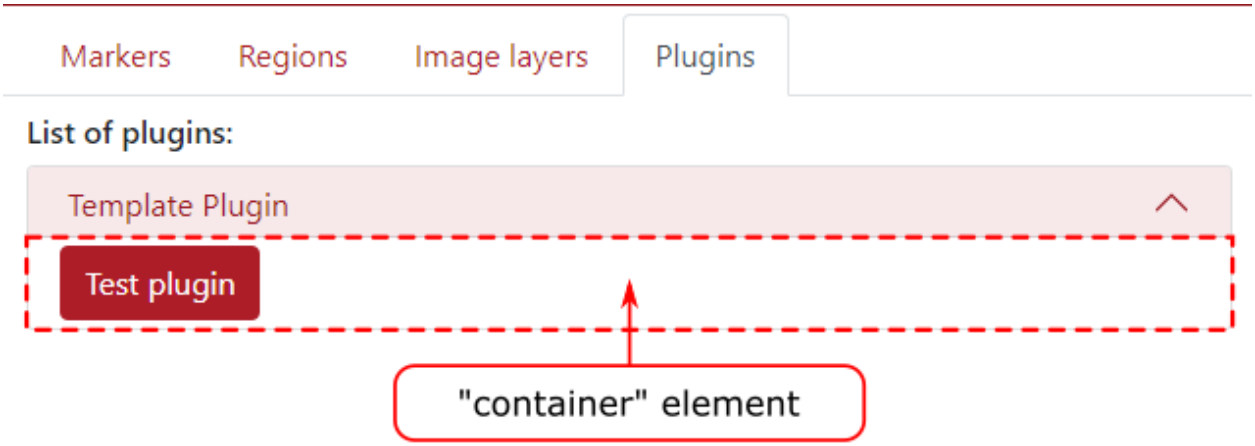
2.6.1 Load plugins

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"
}

/**
 * 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